COMBINING DEVS AND MODEL-CHECKING: USING SYSTEMS MORPHISMS FOR INTEGRATING SIMULATION AND ANALYSIS IN MODEL ENGINEERING

Bernard P. Zeigler[(a)], James J. Nutaro[(b)]

[(a)]RTSync Corp. and Arizona Center for Integrative Modeling and Simulation, AZ
[(b] OakRidge National Laboratory, TN.

[(a)]zeigler@rtsync.com, [(b)] nutarojj@ornl.gov,

ABSTRACT

Our objectives here are to discuss the development of a formal framework that exploits the advantages of the Discrete Event System Specification (DEVS) formalism and builds upon recent extensive work on verification combining DEVS and model checking for hybrid systems. The mathematical concepts within the DEVS formalism encompass a broad class of systems that includes multi-agent discrete event components combined with continuous components such as timed automata, hybrid automata, and systems described by constrained differential equations. Moreover, DEVS offers the ability, via mathematical transformations called system morphisms, to map a system expressed in a formalism suitable for analysis (e.g., timed automata or hybrid automata) into the DEVS formalism for the purpose of simulation. Conversely, it is also possible to go from DEVS to formalism suitable for analysis for the purposes of model checking, symbolic extraction of test cases, reachability, among other analysis tasks. We give an example of application of these concepts and discuss the open opportunities for research in model engineering in this direction.

1. INTRODUCTION

Model checking, a well-known formal verification method, systematically explores the state space of a system model to check that states satisfy specified behavioral properties (Baier, and Joost-Pieter, 2008) ,Model checking methods encounter state space explosion in analyzing *autonomous* systems that require complex logical processes to perform complex decision making tasks. Moreover, because they are limited in their expressive capability to restricted logics, such methods must typically make stringent assumptions about physical components and environments. These assumptions and idealizations greatly reduce the methods' applicability to *cyber-physical systems* where the interplay of physical and computational elements is paramount. Finally, *cooperative* multi-agent systems raise the state space explosion exponentially through the cross product of their individual state spaces. In the absence of workable simulation approaches to enable virtual testing, the only recourse for V&V of Cyber-physical Autonomous Cooperative (CACSoS) is to brute-force methods which are severely limited in the range of conditions they can test.

A key root cause of limitations in current V&V approaches to CACSoS is that they are not based on a general dynamic systems modeling and simulation framework. Such a framework should be capable of expressing the interaction of decision logic, discrete events, and continuous dynamics that are the hallmarks of such systems. We therefore propose that the Discrete Event System Specification (DEVS) formalism, as the computational basis for a general dynamic systems theory (Zeigler, Praehofer, and Kim 2000), provides a sound and practical foundation for enhancing existing V&V methods to address their limitations in addressing in CACSoS.

The value of Modeling and Simulation in defense and other applications is well-known (Shaffer 2012) A DEVS model is a system-theoretic concept specifying inputs, states, outputs, similar to a state machine, (Mittal and Risco 2012). Critically different however, is that it includes a time-advance function that enables it to represent discrete event systems, as well as hybrids with continuous components (Nutaro 2011) in a straightforward platform-neutral manner (Zeigler and Sarjoughian 2012). A recent thesis (Denil 2013) presents a multi-paradigm model-driven approach to design, verification and deployment of software intensive systems, another formulation of cyber-physical systems. It shows that DEVS provides excellent features for modeling such systems. The thesis provides a list of properties of DEVS and their mapping to properties of automotive software and systems – here viewed as instances of CACSoS:

- *Concurrency*: Multiple processors and communication links are concurrent in a CACSoS system. The semantics of DEVS coupled models supports concurrency by appropriate interleaving of the discrete-event behavior of individual sub-models.
- *Time*: Real-time performance is a crucial property of CACSoS embedded software. End-to-end latencies are part of the requirements for these

applications. The time advance function of an atomic DEVS model can be used to model latency.

- *Events*: Event-triggered and time-triggered architectures use triggers in the form of either external events or timing events to start certain pieces of functionality. DEVS implements reaction to events using the external transition functions.
- *Priorities*: Some real-time communication channels use priority-based and other mechanisms for arbitration. DEVS supports such arbitration by means of explicit specification of executable events from the set of simultaneous events.
- *Simulation of the physical parts of the system*: DEVS is a very general formalism and is able to include different other formalisms. This generality stems from the infinite possible states that DEVS allows to model and the (continuous) time elapse between the different state transitions. The hierarchical coupling techniques are used to integrate the different formalisms using DEVS as a common denominator.

2. BACKGROUND

The *Modeling and Simulation Framework* (MSF) (Zeigler 1976) presents entities and relationships of a model and its simulation as background for the proposed work (Figure 1.)

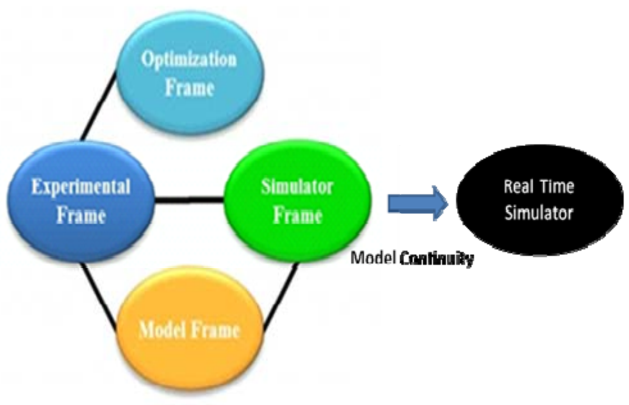

Figure 1 Model and Simulation Framework showing Model Continuity for SoS V&V

The MSF separates models from simulators as entities that can be conceptually manipulated independently and then combined in a relation which defines correct simulation. The Experimental Frame defines a particular experimentation process, e.g., Latin hypercube sampling for yielding model outcome measurements in accordance with specific analysis objectives. Figure 1 depicts the notion of an *Optimization Frame* to supplement the MSF Experimental Frame, where the Optimization Frame directs search among the possible models for one or more that satisfy design space criteria, including those that minimize uncertainty about how well the implemented design will work. Figure 1 also emphasizes the ability enabled by model continuity to transfer a simulation model from a logical-time simulator to a real-time simulator. In particular, DEVS models for autonomous control in CACSoS can be shifted without alteration (avoiding error-prone and tedious reprogramming) to interact with real environments in cooperative configurations after being verified in virtual environments (Zeigler and Sarjoughian 2012, Hu and Zeigler 2008.) The MSF underlies the DEVS Simulation Protocol which provides provably correct simulation execution of DEVS models thereby obviating time and state conflicts arising in simulation of multi-formalism models. There are numerous implementations of DEVS simulators (Nutaro 2011, Mittal and Risco 2012).

3. LIMITATIONS OF V&V METHODS APPLIED TO CACSoS

Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) which are used for expressing desired behavior and model checking have been applied to the development of vehicle routing and road monitoring in multi-UAV systems (Karaman and Frazzoli, 2008, Sirigineedi, et al/, 2010). Humphrey (2013) explored the use of LTL, the SPIN model checker, and the modeling language PROMELA (Gerth 1997, Baier. and Joost-Pieter, 2008, Holzmann 2004), for high-level design and verification in UAV related applications, reporting some success while suggesting limitations and needed extensions. Table 1 shows three UAV related cases she discussed.

Table 1. Example Applications of model checking to CACSoS

Model A centralized UAV cooperator controller that coordinates the actions of multiple UAVs performing a monitoring task
Focus of Model Checking: Assuring that All sensors are eventually visited.
Sample Simplifying Assumptions Communication between UAVs and sensors can only occur when in the same location and is error free.
Model A leader election protocol for a decentralized system of unattended ground sensors sending estimates of an intruder's position to a UAV
Focus of Model Checking At least one leader exists at every time step.
Sample Simplifying Assumptions The sensors all use sampling epochs of the same length enabling a single time step for time advance.
Model Verification of high level UAV mission plans for a scenario in which multiple UAVs must be used to safely escort an asset across a

road network.
Focus of Model Checking The path travelled by the asset is safe, i.e., all road segments in the path have been scanned by UAV.
Sample Simplifying Assumptions UAVs and VIP were assumed to travel at the same speed.

In each case, the focus of model is shown along with a simplifying assumption. Because they are oriented to verification, model checking tools tend to lack many functions that exist in DEVS environments and require abstractions that fit the tools' operation. This forces an abstraction of the real system that on the one hand enables the modeler to better understand the model, and on the other hand entails numerous assumptions to enable the model checker to verify the focal requirement. Despite these drastic simplifications, state space explosion prevents employing more than a handful of UAVs and sensors.

Zervoudakis et al (2013) write that "Research in model checking has focused on enhancing its efficiency and scalability thereby enabling model builders to verify larger, more elaborate models. Popular model checkers tend to support low-level modeling languages that require intricate models to represent even the simplest systems. For example, PROMELA, the language of the model checker SPIN, is essentially a dialect of the low-level programming language C. Another example is the modeling language used by the probabilistic model checker PRISM, whose lack of control structures forces model builders to pollute model components with counter variables that explicitly encode the components' state transitions." These authors show that mapping of domain knowledge, assuming it exists in the right form, can be used to reduce the manual and error-prone encoding of state transitions at relatively low levels of abstraction. Here we propose to develop such mappings using the domain knowledge contained within simulation models and their ontological representations within the DEVS-based Modeling and Simulation Framework.

4. DEVS SUPPORT FOR CACSoS

Cyber-physical systems are real-time hybrid systems, i.e., include both discrete and continuous dynamics, which, as earlier indicated, are well represented within the DEVS-based MSF. Typically such a system is described by a state consisting of both discrete control phases and continuous variables (Nutaro 2008., Saadawi, Wainer, and Moallemi 2012). developed a methodology that combines DEVS and Timed Automata (TA) (Bengtsson and Yi 2004. Henzinger,. 1997, Alur 1995, Courcoubetis et a;l. 1995) to allow the designer to model, simulate, verify, and deploy real-time hybrid systems. This is achieved by guaranteeing the correctness of the model with a methodology that verifies DEVS models with TA model-checking techniques and tools. Under model continuity (Figure 1) the verified DEVS models are then made executable on the target platform, thus eliminating the risk of introducing errors in the final system implementation. TA provides a solid theory and algorithms for model checking, and many existing tools implement these algorithms (e.g., UPPAAL). The combined DEVS/TA methodology deals with RTA-DEVS - a restriction of DEVS to Rational (a subset of real) Time Advance values. For this subclass the methodology provides automated mappings to TA's abstract formal system specification that is verifiable by *decidable* model checking. In this methodology, if UPPAAL (or other model checker) faces a problem of state explosion, and no answers can be obtained in reasonable time, the user can use model checking on an abstraction of the system while employing DEVS-based simulations to empirically check out properties not included in the formal analysis. In particular, the methodology applies to hybrid systems whose continuous components are expressed using differential equations solved using Quantized State System (QSS) integration. Concurrently, a burgeoning literature is developing on the use of QSS, a class of DEVS models, to efficiently and accurately model such systems, for example using multicore processors.
While multi-agent-based simulation (ABS) is well established using DEVS (e.g. (Perez et al. 2010), time-step scheduling is still used in classic ABS models (e.g., Repast. 2009) However, Zhang et. al. (2014) developed a DEVS simulation model which is significantly more efficient than the Repast ABS model 350 times faster for 10000 agents) while keeping high model spatial fidelity and the same agent cognitive capability, collision avoidance, and low agent-to-agent communication cost.

Work on non-DEVS model checking for hybrid systems includes that based on timed automata (Henzinger 1997, Alur 1995, Courcoubetis et a;l. 1995), abstraction and simplification of systems (Chauhan et al. 2002, Clarke, et al. 2003, Long 1993), and statistical model verification (Younes et al. 2006).

5. INTEGRATING DEVS AND NON-DEVS VERIFICATION METHODS

As discussed earlier, several DEVS methodologies have been developed which incorporate non-DEVS verification methods (Zeigler and Sarjoughian 2012, Hu X. and Zeigler 2008.) These methodologies attempt to employ DEVS to enable loosening the simplifying assumptions typically made by non-simulation models. For example, the Sample Simplifying Assumptions in Table 1 suggest that such simplifications include those concerning perfect communication among component systems occurring when they are in exactly specified locations, that time advances for all components using the same fixed time step, and that speeds of vehicles are constant or change instantaneously neglecting realistic accelerations. However, so far, such methodologies have not provided a general approach to combining simulation and verification using available system theoretic concepts.

Our objective here is to develop and employ system morphisms and model transformations to integrate the various types of models to be included in a general DEVS-based framework for verification in model engineering. Model transformations are a key means of converting between different model types and must preserve desired aspects of structure and behavior to qualify as system morphisms. Our approach is to define system morphisms for such transformations and to prove these morphisms are mathematically correct using existing theory of system morphisms (Zeigler, B. P.. , 1976]. This will enable us eventually to automate verification of properties for complex simulation models as opposed to simplified models developed specifically for model checking. Then we will explore algorithmic approaches to automate the construction of these types of systems mappings. Such automation is necessary to create a practical tool for engineering systems of systems.

Figure 2 compares the methodology of (Saadawi, Wainer, Moallemi 2012.) which provides automated transformations from RTA-DEVS to TA enabling tractable model-checking using UPPAAL (Behrman 2004) with the approach we discuss here. The RTA-DEVS approach seeks to verify DEVS models by transforming them into a subset of TA that can be verified using UPPAAL. To do this, it must

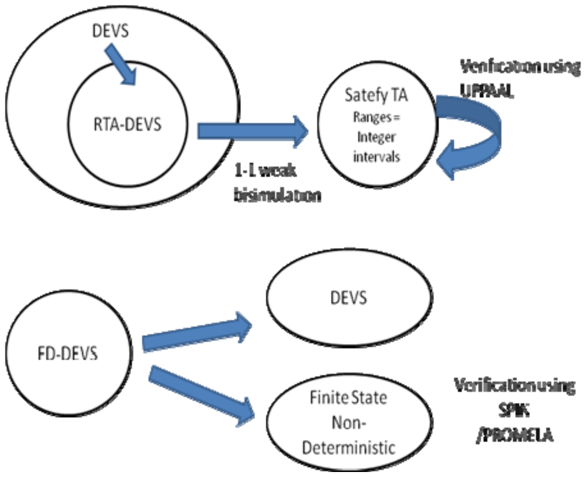
DEVS
RTA-DEVS
1-1 weak bisimulation
Satefy TA Ranges= Integer intervals
Verification using UPPAAL
FD-DEVS
DEVS
Finite State Non-Deterministic
Verification using SPIN /PROMELA

Figure 2 Comparing DEVS-based Verification Approaches

appropriately limit the class of DEVS models to a subclass that can be mapped to the input class of TA for UPPAAL via 1-1 weak bi-simulation with the Safety TA subclass of TA. Weak bi-simulation will be shown to be a system morphism. The mappings are not automated.

In contrast, as shown in Figure 2, our approach is to start with the FD-DEVS (Zeigler and Sarjoughian 2012) subclass of DEVS models having finite sets of states, inputs, and outputs and whose construction is supported by MS4 Me. Then two mappings are defined:

1. Map FD-DEVS models into non-deterministic automata that are subject to model checking using SPIN/PROMELA.
2. Elaborate FD-DEVS models into full-fledged DEVS models that can be simulated to obtain behavior of interest and possibly to discover unexpected or emergent behaviors via simulation.

To illustrate the general idea, we will construct and illustrative, mathematically verifiable transformations from DEVS models to model checking models and vice versa.

5.1. Example of System Morphism between PROMELA and DEVS

For an example of this approach, consider the alternating bit protocol introduced by (Bartlett, Scantlebury and Wilkinson, 1969) for implementing full-duplex communications over half-duplex communication lines. This protocol is illustrated in Figure 3. It has been used to illustrate fundamental elements and analysis capabilities of the PROMELA language by proving that the protocol operates correctly (see the SPIN manual, Gerth 1997); that is, that ``Every message fetched by A is received error-free at least once and accepted at most once by B". It is apparent from the figure that this PROMELA model is a finite state automaton, and all finite state automata are are instances of DEVS models that have a fixed time advance (see (Zeigler, Praehofer, & Kim. 2000). Hence,

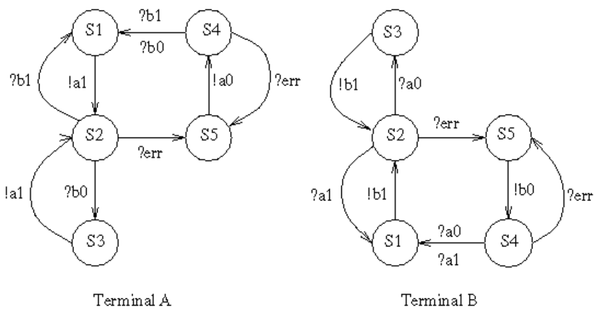

Figure 3. Model for automatic verification of the

it is a DEVS model of this protocol can be built in a simple way. We do this by adding two pieces of information to the PROMELA description: the time T to transmit a bit and the probability p of an error. One important use of this DEVS model is to answer questions about performance of the protocol. Conversely, we may map any instance of this DEVS model with parameters p and T onto a PROMELA model by abstracting away the specific probability distribution and stating only that a bit may arrive or not arrive.

Table 2 shows the results of a simulation study, using the DEVS model sketched above, to discover how the bit rate of the protocol varies as a function of *T* and *p*. This table was constructed with the DEVS simulation model by sweeping over a range of values for *T* and *p*, and for each combination recording the bits per second that could be exchanged by the system assuming 100% utilization of the communication channel. These simulations provide important, systems level performance metrics that cannot be obtained via a query of the PROMELLA model. At the same time, we may be certain that the simulation model preserves the formal properties that we have proven about the protocol by use of the PROMELA model. The capability to construct performance studies, like this bit rate study, using a simulation model derived directly from the formal verification model illustrates the power of the proposed approach for engineering complex systems.

Table 2. Performance study of the alternating bit protocol

T (seconds)	*P*	Bits per second	*T* (seconds)	*p*	Bits per second
1E-9	0	5.0E8	1E-6	0	5.0E5
	0.25	3.8E8		0.25	3.8E5
	0.5	2.5E8		0.5	2.5E5

5.2. Extending the Range of Verification Models with Simulation Models

Transformations between simulation models and verification models can also facilitate the reuse of model components throughout the lifecycle of an M&S

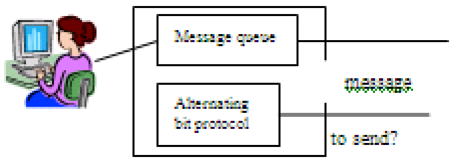

Figure 4 Message Layer addition to ABP

system, and this reuse can have the important effect of revealing implicit assumptions in proofs constructed for the verification model. For instance, it is natural to use the alternating bit protocol as a media access control layer within a more comprehensive network simulation. This more comprehensive model could have a sender and receiver each with two components. This is illustrated in the Figure 4. The first component is our DEVS model of the alternating bit protocol. The second component sends and receives messages, rather than just bits, and queues messages that are pending transmission. This second component appends to each message the bit that it receives from the MAC layer, and sends to the MAC layer the first bit in each message received from the network or an error indicator, as appropriate. Transmissions in the upper layer occur at the instant that a bit is received from the MAC layer below. Central to understanding the behavior of this model is its queue capacity, rate of requests to send messages, and bit rate.

5.2.1. Alternating Bit Protocol with Infinite Queue

Let us first consider a combination of these that may be reduced to a slightly more complex version of the verification model for the alternating bit protocol. To obtain this model we reduce the queue to two states: empty and occupied. The former indicates no messages waiting for transmission and the latter indicates a message waiting to be transmitted. Adding these states to the transmitter increases the size of the verification model from 25 states to 50 states, and we may prove for this larger model that every message transmitted is received once and only once. This proof is significant because it reflects the intended, but idealized, behavior

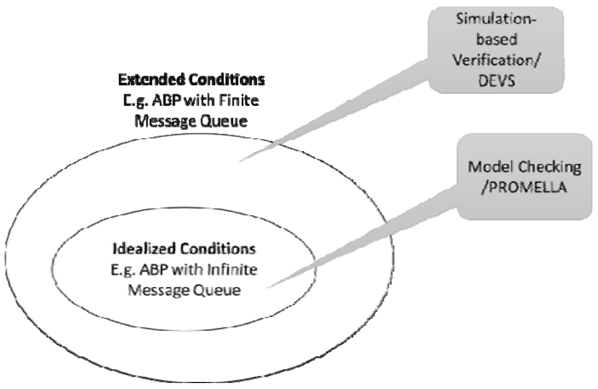
Simulation-based Verification/ DEVS
Extended Conditions
E.g. ABP with Finite Message Queue
Model Checking /PROMELLA
Idealized Conditions
E.g. ABP with Infinite Message Queue

Figure 5 Relation between Verification and Simulation

of the system. If, for instance, this assertion could not be proved then the design of the system should be reconsidered before moving to other forms of testing.

5.2.2. Alternating Bit Protocol with Finite Queue

However, this proof does not indicate how the system will behave over its entire range of structural variations and realizable behaviors. In particular, we may encounter a case where the queue's capacity is finite. In this case, the bit rate and rate of requests to send messages may be such that at certain points in time the queue's capacity is exceeded as can be predicted by simple queuing theory (Jain 1991). The consequence of this will be messages that are lost: a clear violation of the above proof! In this simple example, the cause of this violation is obvious. The verification model assumes that the queue never reaches its capacity, and so the proof implicitly assumes a restricted range of values for the queue capacity, rate of transmission requests, and bit rate.

6. COMBINING SIMULATION AND FORMAL VERIFICATION

These types of implicit assumptions can be difficult to identify in a large model, and simulation offers an opportunity to explore the system's parameter space and identify boundaries beyond which any particular proof fails to hold.

As illustrated in Figure 5, the combination of simulation and formal verification gives a much more powerful capability to test designs than can be achieved with either alone. In a design process that incorporates both types of analysis, verification models can be used to obtain absolute answers concerning system behavior under idealized conditions. Failures in this verification stage clearly indicate a need to find and correct fundamental flaws in the system design. On the other hand, a successfully verified model can be formally extended into a simulation model for which the verification model is a homomorphic simplification. Hence, the simulation model retains the properties that were verified with the simpler model, and then can be used to explore scenarios that are necessarily outside the scope of formal verification. In some cases, other simplifications of a full-fledged simulation model can be applied (for example, as mentioned above, queueing theory can predict the probability of a finite capacity queue being exceeded.) However, in general, simulation models can incorporate the complexity needed to deal with real systems using the base model concept. The traceability between these two types of models, which is obtained by the application for formal system morphisms, is central to the success of this two tiered approach to testing.

7. FUTURE RESEARCH

Future research must develop robust transformation techniques that can be shown to be systems morphisms of suitable types. This will ensure consistency between the results of verification of emergent foreseen behaviors using analytical techniques and the discovery of emergent unforeseen behaviors through dynamic simulations. This will require formulating a meta-modeling approach that will lead to a multi-step verification process that can handle the dynamical complexity of CACSoS models. We recognize that the most expeditious way to develop an inclusive framework is to build on existing methods and software tools to the extent possible. The multi-step verification process will help to manage and cross-check results obtained from the various analytical and simulation methods, as well as from integrated methods that are supported by the proposed framework. This approach helps to deal with the complexities of CACSoS by enabling more robust designs and a more thorough and organized simulation and verification process. These complexities can be addressed with the help of the theory of M&S (Zeigler 1976) to develop methods for hierarchical decomposition and simplification as essential tools within the emerging field of model engineering.

REFERENCES

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, and J., Yovine, S. 1995, The algorithmic analysis of hybrid systems. Theoretical Computer Science 138(1), 3–34

Baier, C. and Joost-Pieter, K. 2008, Principles of Model Checking. The MIT Press

Bartlett, K.A., , Scantlebury, R.A., and Wilkinson, P.T. 1969`A note on reliable full-duplex transmission over half-duplex lines,' Comm. of the ACM, , Vol. 12, No. 5, 260-265,

Behrmann, G., David A., Larsen K. 2004. "A Tutorial on UPPAAL". *Proceedings of the 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems*. LNCS 3185.

Bengtsson, J. and Yi W.. 2004. "Timed Automata: Semantics, Algorithms and Tools". *Lectures on Concurrency and Petri Nets*, 3098.

Chauhan, P., Clarke, E., Kukula, J., Sapra, S., Veith, H., and Wang, D., 2002, Automated abstraction refinement for model checking large state spaces using SAT based conflict analysis. In: Aagaard, M.D., O'Leary, J.W. (eds.) FMCAD 2002. LNCS, vol. 2517, pp. 33–51.Springer, Heidelberg

Clarke, E., Grumberg, O., Jha, S., Lu, Y., and Veith, H., 2003, Counterexample-guided abstraction refinement for symbolic model checking. Journal of the ACM (JACM) 50(5), 752–794()

Denil, J, 2013 Design, Verification and Deployment of Software Intensive Systems: A Multi-Paradigm Modelling Approach, Ph. D. Dissertation, University of Antwerp.

Gerth, R., 1997, Concise PROMELA reference (), http://SPINroot.com/SPIN/Man/Quick.html

Henzinger, T.A., Ho, P.H., and Wong-Toi, H. 1997, HyTech: A model checker for hybrid systems. International Journal on Software Tools for Technology Transfer (STTT) 1(1), 110–122 ()

Holzmann, G.J., 2004,The SPIN Model Checker: Primer and Reference Manual. Addison Wesley Publishing Company ()

Hu X. and Zeigler B. P., 2005. A Simulation-Based Virtual Environment to Study Cooperative Robotic Systems, *Integrated Computer-Aided Engineering (ICAE), 12:4, pp. 353-367,*

Humphrey, L. R. 2013, Model Checking for Verification in UAV Cooperative Control Applications Recent Advances in Research on Unmanned Aerial Vehicles, Lecture Notes in Control and Information Sciences Volume 444, 2013, pp 69-117

Jain R. K. 1991, The Art of Computer Systems Performance Analysis: Techniques For Experimental Design, Measurement, Simulation, And Modeling Hardcover: 685 pages Wiley.

Karaman, S. and Frazzoli, E., 2008 ,Vehicle routing with linear temporal logic specifications: Applicationsto multi-UAV mission planning. In:

Proceedings of the AIAA Conference on Guidance, Navigation, and Control

Long, D.E. 1993, Model Checking, Abstraction, and Compositional Verification. PhD thesis, Carnegie Mellon University

Mital, S,. Risco Martin José L 2012, Netcentric System of Systems Engineering with DEVS Unified Process, CRC Press; 1 edition

Nutaro J. 2008. On constructing optimistic simulation algorithms for the discrete event system specification, ACM Transactions on Modeling and Computer Simulation, 19(1), pp. 1-21.

Nutaro J., 2011. Building Software for Simulation: Theory and Algorithms with applications in C++. Wiley.

Perez E, Ntaimo L, Bailey C and McCormack P , 2010, Modeling and simulation of nuclear medicine patient service management in DEVS. Simulation-Transactions of the Society for Modeling and Simulation International 86(8–9): 481–501.

Repast, 2009. Repast home page. Available at http://repast.source forge.net,

Saadawi, H, Wainer, G. Moallemi, M.. 2012. “Principles of Models Verification for Real-Time embedded Applications”. *Real-Time Simulation Technologies: Principles, Methodologies, and Applications*. K. Popovici, P. Mosterman Eds. Taylor and Francis. CRC Press. 2012.

Shaffer, Alan R 2012 The Value of Modeling and Simulation for the Department of Defense M&S Journal pp 2-3

Sirigineedi, G., Tsourdos, A., White, B., Zbikowski, and R. Kripke, 2010, Modelling and model checking of a multiple UAV system monitoring road network. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference.

Younes, H.L.S., Kwiatkowska, M., Norman, G., and Parker, D. 2006, Numerical vs. statistical probabilistic model checking. International Journal on Software Tools for TechnologyTransfer (STTT) 8(3), 216–228

Zeigler, B. P. and Sarjoughian, H. S. 2012, Guide to Modeling and Simulation of Systems of Systems Springer; pp. 393

Zeigler, B. P., Praehofer, H., & Kim, T. G., 2000, *Theory of Modeling and Simulation* (2nd ed.). Academic Press.

Zeigler, B. P., 1976. Theory of Modeling and Simulation (1st ed.). Academic Press.

Zervoudakis, F., Rosenblumy D. S., Elbaumz, S., and Finkelstein, A, 2013 Cascading Verification: An Integrated Method for Domain-Specific Model Checking, ESEC/FSE 2013, Saint Petersburg, Russia

Zhang, B, Chan, W K V, and S V Ukkusuri, 2014, On the modelling of transportation evacuation: an agent-based discrete-event hybrid-space approach *Journal of Simulation* advance online publication 21 February 2014

AUTHORS BIOGRAPHY

Bernard P Zeigler, Chief Scientist of RTSync Corp.,and Emeritus Professor from Arizona Ceter for Integrative Modeling and Simulation is internationally known for his seminal contributions in M&S theory, and has published several books including *Theory of Modeling and Simulation*, IEEE named him a Fellow of the IEEE for his invention of the Discrete Event System Specification (DEVS).

James Nutaro is Senior Research Staff at Oak Ridge National Laboratory. He has extensive experience in M&S and systems modeling in both defense and commercial domains. He has applied M&S techniques for design, analysis, and testing in diverse enterprises including missile systems, space systems, communications, uranium processing, electrical power, disease processes, and high performance computing technology. He is also the author of *Building Software for Simulation: Theory and Algorithms, with Applications in C++*.