Hybrid Modeling and Simulation Methodology based on DEVS formalism

*Seong Yong Lim and **Tag Gon Kim

*Converged LAN Team, Router Technology Department, ETRI, KOREA
** Department of Electrical Engineering and Computer Science, KAIST, KOREA
E-mail: *seylim@etri.re.kr, **tkim@ee.kaist.ac.kr

Keywords: Hybrid System, Interoperable Simulation, DEVS formalism, HLA/RTI, E/A Converter, A/E Converter

Abstract

We consider a hybrid system as a mixture of continuous systems and discrete event systems. This paper proposes a framework for hybrid systems modeling and simulation. For modeling a hybrid Discrete Event Systems Specification (HDEVS) formalism is proposed, which is based on the hierarchical, modular DEVS (Discrete Event Systems Specification) formalism. For simulation, an interoperable simulation environment is developed, in which an existing continuous system simulator and a discrete event simulator are interoperated on HLA/RTI. Effectiveness of the proposed hybrid systems modeling/simulation framework is demonstrated through simulation of a hybrid system for mobile robot control. The developed hybrid simulation environment interoperated with MATLAB, for continuous system components, and DEVSim++, for discrete event system components, through HLA/RTI and produced correct simulation results.

1. Introduction

As a complex system contains both continuous and discrete event models new modeling/simulation methodologies for such systems are required. A hybrid system is defined as a mixture of continuous systems and discrete event systems[1]. Some modeling methods for such systems have been proposed, but a little on simulation environment has been developed. Modeling methods based on a combined formalism, which has both discrete event and continuous features, and associated simulation frameworks are being proposed [2][3]. A main limitation of the proposed methods/frameworks is that they cannot use existing simulators in different types, which are interoperated during simulation. The objective of this paper is to develop a modeling and simulation methodology for hybrid systems, which overcome the limitation.

This paper proposes a formal modeling and simulation framework for hybrid systems as shown in Figure 1. For modeling a hybrid Discrete Event Systems Specification (HDEVS) formalism is proposed, which is based on the hierarchical, modular DEVS (Discrete Event Systems Specification) formalism. For simulation, an interoperable simulation environment is developed, in which an existing continuous system simulator, MATLAB, and a discrete event simulator, DEVSim++, are interoperated on HLA/RTI.

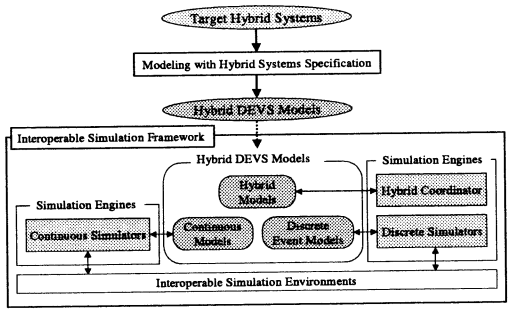

Figure 1. Overview of the proposed Framework

2. Hybrid Systems Specification

Zeigler's DEVS formalism [2] supports hierarchical modular descriptions of discrete event systems. The Hybrid Discrete Event Systems Specification (HDEVS) proposed in this paper is based on the DEVS formalism. More specially, discrete event modeling employs the DEVS formalism; continuous system modeling uses differential equations formalism. Hybrid system modeling supports communication between two formalisms based on the proposed concept of A/E(Analog-to-Event) conversion and E/A(Event-to-Analog) conversion. To be consistent with the DEVS formalism, the HDEVS formalism defines two model types, atomic and coupled, for both continuous and discrete event modeling.

2.1 Atomic Models

A basic model, called an atomic model, has specifications for the dynamics of the models. Atomic modeling for discrete

event systems is based on atomic model specification in the DEVS formalism; that for continuous systems employs differential equations in a set theoretic form. The HDEVS formalism defines two types of atomic models, DEVS-AM and CAM for discrete event systems and continuous systems, respectively.

Def.1 DEVS-AM [2]

$$DEVS-AM =< X_{disc}, Y_{disc}, S_{disc}, \delta_{ext}, \delta_{int}, \lambda, ta >$$

with the following constraints:

X_{disc} : Set of discrete event inputs

Y_{disc} : Set of discrete event outputs

S_{disc} : Set of discrete event states

$\delta_{ext} : Q \times X_{disc} \rightarrow S_{disc}$: External transition function

$\delta_{int} : Q \rightarrow S_{disc}$: Internal transition function

$\lambda : Q \rightarrow Y_{disc}$: Output function

$ta : S \rightarrow R^{+}_{0,\infty}$: Time advance function

Def.2 Continuous Atomic Model (CAM)

$$CAM =< X_{cont}, Y_{cont}, S_{cont}, \delta_{cont}, \lambda_{cont} >$$

with the following constraints:

X_{cont} : Set of continuous inputs

Y_{cont} : Set of continuous outputs

S_{cont} : Set of continuous states

$\delta_{cont} : \frac{d}{dt} S_{cont}(t) = \delta_{cont}(S_{cont}(t), X_{cont}(t), t)$: State transition function

$\lambda_{cont} : Y_{cont}(t) = \lambda_{cont}(S_{cont}(t), X_{cont}(t), t)$: Output function

Detial explaination of DEVS-AM and CAM can be found in [2],[4].

2.2 Coupled Models

A coupled model provides the method of assembly of several atomic and/or coupled models to build complex systems hierarchically. As with atomic models specification, the HDEVS formalism defines two types of coupled models, DEVS-CM and CCM for discrete event systems and continuous systems, respectively.

Def.3 DEVS-CM [2]

$$DEVS-CM =< X_{disc}, Y_{disc}, \{DM_d\}, EIC, EOC, IC, Select >$$

with the following constraints:

X_{disc} : Set of discrete event inputs

Y_{disc} : Set of discrete event outputs

$\{DM_d\}$: Set of DEVS component models

$EIC \subseteq X_{disc} \times \underset{i}{Y} X_i$: External input coupling relation

$EOC \subseteq \underset{i}{Y} Y_i \times Y_{disc}$: External output coupling relation

$IC \subseteq \underset{i}{Y} Y_i \times \underset{j}{Y} X_j$: Internal coupling relation

$Select : 2^{\{DM_d\}} - \phi \rightarrow DM_d$: Tie-breaking function

Def.4 Continuous Coupled Model (CCM)

$$CCM =< X_{cont}, Y_{cont}, \{CM_d\}, EIC, EOC, IC >$$

with the following constraints:

X_{cont} : Set of continuous inputs

Y_{cont} : Set of continuous outputs

$\{CM_d\}$: Set of CAM component models

$EIC \subseteq X_{cont} \times \underset{i}{Y} X_i$: External input coupling relation

$EOC \subseteq \underset{i}{Y} Y_i \times Y_{cont}$: External output coupling relation

$IC \subseteq \underset{i}{Y} Y_i \times \underset{j}{Y} X_j$: Internal coupling relation

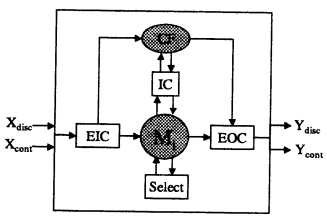

Figure 2. Structure of Hybrid Model

Hybrid Model(HM), is a class of coupled models which has CCM, CAM, DEVS-CM, DEVS-AM and other HM models as component models. A HM shown in Figure 2 is defined as follows:

Def.5 Hybrid Model (HM) [4]

$$HM =< X, Y, \{M_d\}, EIC, EOC, IC, Select, CF >$$

with the following constraints:

$X = X_{disc} \cup X_{cont}$: Set of hybrid inputs

$Y = Y_{disc} \cup Y_{cont}$: Set of hybrid outputs

$\{M_d\} = \{DM_d\} \cup \{CM_d\} \cup \{HM_d\}$: Set of hybrid components

$EIC \subseteq X \times \underset{i}{Y} X_i$: External input coupling relation

$EOC \subseteq \underset{i}{Y} Y_i \times Y$: External output coupling relation

$IC \subseteq \underset{i}{Y} Y_i \times \underset{j}{Y} X_j$: Internal coupling relation

$Select : 2^{\{M_d\}} - \phi \rightarrow M_d$: Tie-breaking function

$CF =< \{f_{AEi}\}, \{g_{EAi}\} >$: Conversion Functions

Detial explaination of DEVS-CM,CCM and HM can be found in [2],[4].

2.3 Conversion Functions

In simulation of hybrid systems a discrete output from a discrete event model can be transmitted as a continuous input to a continuous model or the other way around. CF, in definition of HM, is proposed as interface for information conversion in such a case. The interface is defined in two types of functions: E/A(Event-to-Analog) converter for converting an discrete event output to a continuous input, and A/E(Analogy-to-Event) converter for converting a continuous output to a discrete event input. Formally, the two converters as shown in Figure 3 are defined as follows.

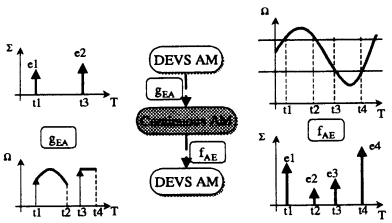

Figure 3. Conversion Function
E/A Converter (left) and A/E Converter (right)

Def. 6 Conversion Functions

$$CF = <\{f_{AEi}\},\{g_{EAi}\}>$$

with the following constraints:

$f_{AEi} : \Omega \rightarrow 2^{\Sigma}$: A/E converter

$g_{EAi} : \Sigma \rightarrow \Omega$: E/A converter

where Σ: Set of events
Ω: Set of piecewise continuous segments.

Note that an analogy segment may produce multiple output events simultaneously in A/E conversion. The A/E converter generates an event based on predefined rules. An example of such rules is: when the segment changes its region or crosses a predefined point such as zero.

3. Hybrid Simulation Algorithm: Interoperable Simulation

3.1 Concept of Pre-Simulation

Hybrid simulation interoperates a discrete event simulator with a continuous simulator as simulation proceeds. Such interoperation requires data exchange as well as simulation time synchronization between the two simulators. Data exchange can be done by E/A converter and/or A/E converter explained earlier in section 2.3. Simulation time synchronization requires a chronological order of logical times between continuous and discrete event simulations. To synchronize the two simulation times with different time advance mechanisms the concept of pre-simulation for continuous simulation is introduced. Pre-simulation is continuous simulation only within a predefine time window to unify the time advance mechanism between two simulations. Such window is generated by the hybrid simulation algorithm to be explained later.

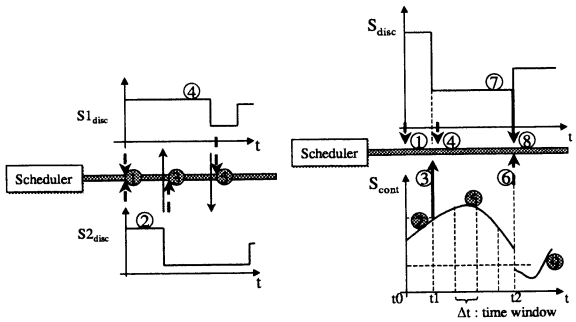

Figure 4. Concept of Pre-Simulation

Time advance mechanism of the event scheduling concept in discrete event simulation is shown in the left side of Figure 4. The number shown in the Figure is a chronological sequence of the simulation process. The pre-simulation concept is shown in the right side of Figure 4, where a discrete event model(top) and a continuous model(bottom) are interoperated. We briefly explain simulations interoperation based on the pre-simulation concept. The scheduler first computes a next event time of the discrete event model. It then creates a time window between the current simulation time and the next event time, and sends the window to the continuous simulator. The continuous simulator proceeds simulation either until an output event is found within the time window or until the time window is completed. If an output event is found, it would be sent to the discrete event simulator as an input event. When received the input event, the discrete event simulator executes its transition at the current time and schedules another next event time. If, however, no event is found, the continuous simulator sends a null output to the discrete event simulator. When received the null input, the discrete event simulator executes its state transition at the scheduled next event time and schedules another next event time. Simulation interoperation is achieved by iterating the sequence explained above until no next event time is found. Note that pre-simulation guarantees an output event of continuous simulator within a time window.

[Theorem 1] Simulators interoperation using pre-simulation is deadlock free.

[proof] Informal outline

Interoperable communication between discrete event and continuous simulators is done by discrete events. Discrete event simulator can receive an event from continuous simulator within a pre-defined time window, which is the next event time of itself. Continuous simulator proceeds its simulation either by events received from discrete event simulator or by pre-simulation.

3.2 Simulators Algorithm

Abstract simulator concept associated with the DEVS formalism characterizes what has to be done to execute atomic or coupled models with hierarchical structure[5]. Interoperable simulators based on such abstract simulator concept imports simulation capability of continuous models simulation. DEVS simulator can simulate discrete event models, and any differential equation solver can simulate basic behavior of continuous models.

Hybrid Coordinator (HC) is an abstract simulator, or simulation algorithm, associated with hybrid models. It manages a discrete event simulator and a continuous simulator by means of message passing based on the concepts of A/E- E/A converters and pre-simulation. Figures 5 and 6 show a state diagram and pseudo code of the HC algorithm, respectively. Figures 7 and 8 show a state diagram and pseudo code of a continuous simulator algorithm, respectively. Note that state diagrams show all input/output messages of the HC and the continuous simulator, and pseudo code explains transitions between states in such diagrams. Simulation algorithms for discrete event simulators are found in [5].

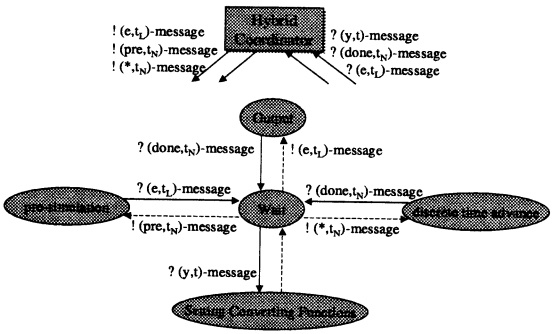

Figure 5. State Diagram of HC Algorithm

4. Hybrid Simulation Environment: Interoperation of MATLAB and DEVSim++

4.1 Continuous Simulator: MATLAB

MATLAB integrates mathematical computing, visualization, and a powerful language to provide a flexible environment for continuous systems modeling and simulation. The open architecture makes it easy to use MATLAB and its companion products to explore data, create algorithms, and create custom tools that provide early insights and competitive advantages [6].

```
Hybrid Coordinator:
: t_C -> current simulation time
: t_N -> next event time
main loop
        if t_C = t_N
                send (*,t_N) to its discrete child
        else if t_C < t_N
                send (pre,t_N) to its continuous child
        endif
until end of loop

When receive (done,t_N') from its discrete child
        t_N = t_N'

When receive (e,t_L') from its continuous child through f_AE
        t_C = t_L'
        if e != NULL
                send (e,t_L) to its discrete child
        endif
When receive (y,t)
        g_EA(y,t) converting event to analog
End Hybrid Coordinator
```

Figure 6. Pseudo Code of HC Algorithm

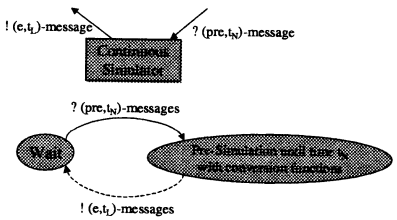

Figure 7. State Diagram of CS Algorithm

```
Continuous Simulator:
: t_L -> last simulation end time

When receive (pre,t_N) from parent
        simulate from time t_L
                if an event e found at time t_L'
                        t_L = t_L'
                        send (e, t_L) to parent
                        return
                endif
        until time t_N
        t_L = t_N
        send (NULL, t_L) to parent
End Continuous Simulator
```

Figure 8. Pseudo Code of CS Algorithm

4.2 Discrete Event Simulator: DEVSim++

DEVSim++ realizes the DEVS formalism for modeling and associating abstract simulator concepts for simulation, all in C++. For modeling, DEVSim++ provides the modeler with facilities for specification of atomic models and coupled models in a diagraph form within the DEVS framework. For simulation, DEVSim++ implements hierarchical scheduling

algorithms in abstract simulators of atomic and coupled models [7].

4.3 Interoperation of CS/DES Simulators

This paper develops two types of interoperable simulation environment; centralized and distributed. HDEVSim++, the centralized environment, is an extension of DEVSim++. As shown in Figure 9 HDEVSim++ adds a continuous simulator, MATLAB, to DEVSim++ in which the two simulators communicates through A/E and E/A converters in C++. On the other hand, the distributed environment of HDEVSimHLA exploits the concept of conservative time synchronous mechanism in High Level Architecture (HLA) proposed by Defense Modeling and Simulation Office (DMSO) [8] (Figure 10). HDEVSimHLA has a common time management arbiter, communicating bus called Run-Time Infrastructure (RTI) on which two existing simulators, DEVSim+ and MATLAB, are interoperable.

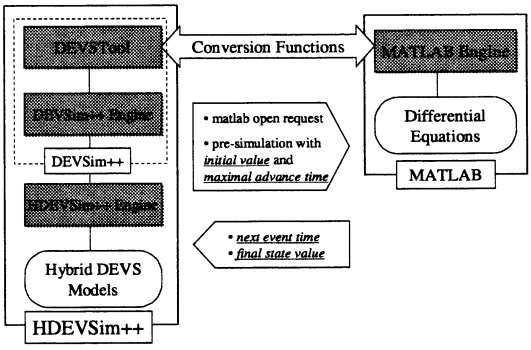

Figure 9. Centralized Hybrid Simulation Environment

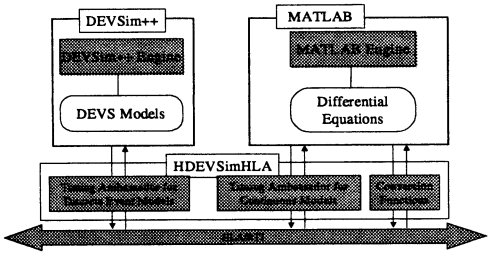

Figure 10. Distributed Hybrid Simulation Environment

5. Example: Mobile Robot Control System

A hybrid system, a mobile robot control system consists of eight continuous systems and two discrete event systems. A disabled person controls a wheelchair and moves to the destination along the street with landmarks. Figure 11 shows a hybrid model of the system which contains discrete event models, continuous models and conversion functions[9]. DES models consist of Human Operator that controls Mobile Robot (as electromotive wheel chair) with Joystick and an environment of landmarks information. Continuous models consist of Mobile Robot that outputs his/her positions and velocity and Joystick.

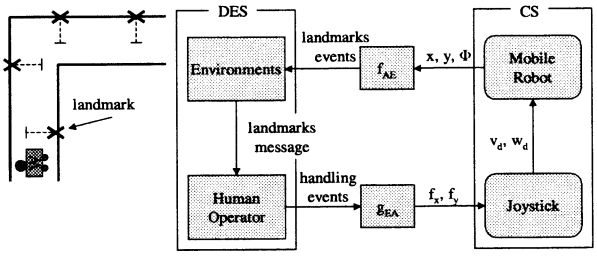

Figure 11. Mobile Robot Control System

5.1 Continuous Systems Modeling

Continuous systems are modeled in SIMULINK of the MATLAB Graphics User Interface. Figure 12 shows a CS models in which every block is modeled as differential equations [4].

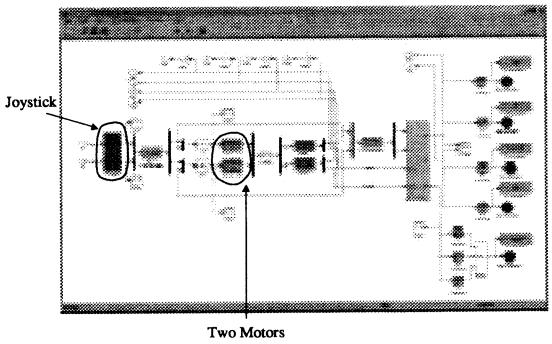

Figure 12. Continuous Model in Simulink

5.2 Discrete Event Systems Modeling

Discrete event systems are modeled in the DEVSim++ environment. In Figure 13, Human Operator receives input events as ordering direction forms from Environment and gives control signal about elaborately stick movement to Joystick. Environment receives crossing events of landmark from Mobile Robot and gives the next movement information to Human Operator.

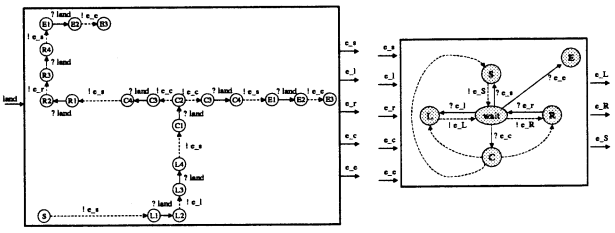

Figure 13. Discrete Event Model in DEVSim++

5.3 A/E and E/A Converters

Figure 14 shows A/E converters and E/A converters. If the position of the Mobile Robot received from a continuous model is crossed a landmarks point, an A/E converter creates an occurrence event to the Environment. On the other hand, an E/A converter generates a piecewise continuous segment as an output for Joystick mapped to an input event from Human Operator.

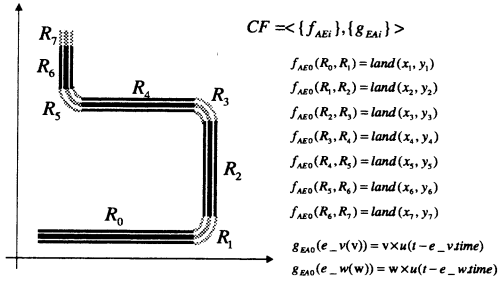

Figure 14. A/E and E/A Converters

5.4 Hybrid Simulation Results

Simulation results are the same in the two types of environments. One such result is shown in Figure 15 where a wheelchair moves along with landmarks. HDEVSim++ and HDEVSimHLA employ different number of machines for simulation. HDEVSim++ uses only one machine for both MATLAB and DEVSim++; HDEVSimHLA maps MATLAB and DEVSim++ in different machines communicating through IP network.

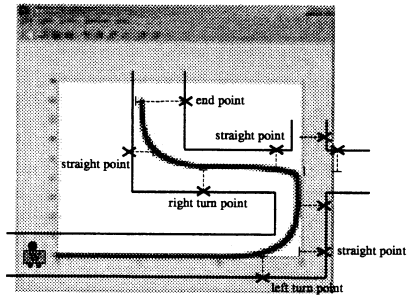

Figure 15. Simulation Result

6. Conclusions

This paper proposed the HDEVS formalism for modeling of hybrid systems, and developed associated modeling and simulation environments in two types, centralized and distributed. Both environments allows modelers to use existing simulators in different types, which can interoperate during simulation either through process communication in the centralized environment or through RTI in the distributed one. Such interoperation is achieved through two types of interfaces, an A/E converter for Analog-to-Event conversion and an E/A converter for Event-to-Analog conversion.

An example of modeling and simulation for a mobile robot hybrid system shows effectiveness of the proposed modeling methodology and power of the developed environments.

References

[1] Panos J. Antsaklis, James A. Stiver and Michael D. Lemmon, "Interface and Controller Design for Hybrid Control Systems", *Hybrid Systems II*, Springer, pp. 462-492, 1994.

[2] Bernard P. Zeigler, Herbert Praehofer and Tag Gon Kim, *Theory of Modeling and Simulation 2nd edition*, Academic Press, 2000.

[3] Thomas A. Henzinger and Pei-Hsin Ho, "HYTech: The Cornell Hybrid TECHnologyTool", *Hybrid Systems II*, Springer, pp.265-293, 1994.

[4] Seong Yong Lim, "Hybrid Systems Modeling and Simulation Methodology based on DEVS formalism", *MS Thesis, KAIST*, 2001.

[5] Zeigler B. P., *Multifaceted Modeling and Discrete Event Simulation*, Academic Press, 1984.

[6] MathWorks, *Using MATLAB Manual*, 1998, ftp://ftp.mathworks.com.

[7] Tag Gon Kim, *DEVSim++ User's Manual*, 1994, ftp://sim.kaist.ac.kr/pub/

[8] DMSO, *RTI 1.3-Next Generation Programmer's Guide Version 3.2*, 2000.

[9] Chong-Hui Kim, "Implementation of Distributed Mobile Control Robot System using COTS Systems", *MS Thesis, KAIST*, 2001.

Biography

Seong Yong Lim received the B. S. and M. S. degrees in Electrical Engineering in 1999 and 2001, respectively, from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. His research interests include hybrid systems, network router/switch systems and distributed simulation.

Tag Gon Kim is Professor of Electrical Engineering & Computer Science, KAIST. His research interests include discrete event systems, modeling methodology, and simulation environment. He is a senior member of IEEE and SCS, and a member of ACM and Eta Kappa Nu. He is a co-author (with B.P. Zeigler and H. Praehofer) of *Theory of Modeling and Simulation*(2nd Edition.), Academic Press, 2000. He is the Editor-in-Chief for the Methodology Section of *Simulation: Transactions of SCS, International.*