Proceedings of the 2011 Winter Simulation Conference
S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.

CROSS-PARADIGM SIMULATION MODELING: CHALLENGES AND SUCCESSES

Susan K. Heath

Graduate School of Business and Public Policy
Naval Postgraduate School
555 Dyer Road
Monterey, CA 93943 USA

Sally C. Brailsford

School of Management
University of Southampton
Southampton SO17 1BJ, UNITED KINGDOM

Arnold Buss

MOVES Institute
Naval Postgraduate School
700 Dyer Road
Monterey, CA 93943 USA

Charles M. Macal

Argonne National Laboratory
Center for Complex Adaptive Agent
Systems Simulation (CAS^2)
9700 S. Cass Ave.
Argonne, IL 60439, USA

ABSTRACT

This paper addresses the broad topic area of cross-paradigm simulation modeling with a focus on the discrete-event, system dynamics and agent-based paradigms. It incorporates contributions from four panel members with diverse perspectives and areas of expertise. First, each paradigm is described and definitions are presented. The difference between the process-oriented worldview and the event-oriented worldview within discrete-event simulation modeling, and the importance of this difference for cross-paradigm modeling, are discussed. Following the definitions, discussion of cross-paradigm modeling is given for each pair of these paradigms, highlighting current challenges and early successes in these areas. The basic time-advance mechanisms used in simulation modeling are also discussed, and the implications of these mechanisms for each paradigm is explored.

1 INTRODUCTION

There has been increasing interest in the relationships between various simulation paradigms and, especially, in the challenges associated with modeling problems with different aspects best represented by different paradigms. To address this broad topic area, we have assembled a panel that represents a variety of perspectives on this topic. We will approach this topic in two ways. The first way is to take a pair-wise approach toward discussing three common modeling paradigms: system dynamics (SD), discrete-event simulation (DES), and agent-based simulation (ABS). The second way is to look at the underlying simulation time-advance mechanisms, compare the Time Step to the Next Event mechanisms, and discuss their implications for simulation modeling.

In the next section we will present a basic description of each of the three modeling paradigms we will be addressing. Following this will be three sections taking a pair-wise approach to these three paradigms: Section 3 will address DES and SD, Section 4 will address SD and ABS, and Section 5 will address DES and ABS. Section 6 will cover the time-stepping issues, and Section 7 will provide a conclusion. However, before we will proceed, we need to state one caveat: although this paper has been consolidated into one common flow, it consists of contributions from four separate panel members who have diverse areas of expertise and perspectives, so it should not be assumed that all assertions made in this paper are being made by all panel members.

2 PARADIGM DESCRIPTIONS AND DEFINITIONS

In this section we present a brief description of DES, SD, and ABS. The important distinction between two main DES worldviews is also highlighted.

2.1 Discrete-Event Simulation

Discrete-event simulation (DES) is a modeling method for stochastic, dynamic models where simulation state variables change at discrete points in time. The discrete point in time when one or more state variables change is termed and *event*. Strictly speaking, therefore, DES does not include variables that change continuously with respect to time.

DES is a highly flexible approach in which almost anything can be coded; models can be incredibly detailed. Most commercial DES software has a graphical interface which allows the user to see the system operating on the screen, which in some cases is almost like watching a movie. This can be a very powerful communication aid for non-technically minded clients. Because of stochasticity, multiple replications of simulation runs are required to obtain statistically significant results.

Although the above definition is the commonly accepted definition of DES, there are two different worldviews that dominate DES modeling today: a process-oriented worldview and an event-oriented worldview. It turns out that these worldviews matter when it comes to addressing the challenges of cross-paradigm simulation modeling, so we will present a basic description of each. Although the event-oriented worldview appears to have come first, the process/resource worldview seems much more common today, so it will be presented first.

2.1.1 Discrete-Event Simulation – Process-Oriented Worldview

The process-oriented worldview for DES generally consist of describing *entities* which move through various *processes*, where each process requires one or more *resources* and takes a certain (usually stochastic) amount of time. Figure 1 shows a typical depiction showing an entity *e* moving through process *A*. The first event would be the arrival of *e* to *A*'s queue, followed by the event of the resource(s) required by *A* beginning processing *e*, and ending with the event of the completion of the processing of *e*.

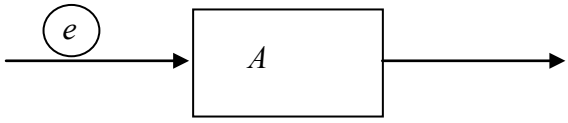

Figure 1: Process/Resource Representation

Most commercial DES software uses the process-oriented worldview. Entities in this approach are typically purely data containers and do not have any behaviors associated with them. The flow of an entity through the system is governed by rules assigned to aspects of the system, which can include probabilistic and condition-based decision, but are not governed by any decision processes internal to the entities.

2.1.2 Discrete-Event Simulation – Event-Oriented Worldview

There is another approach to DES that works at a more basic and fundamental level, that is, with the events themselves rather than with entities and resources. A parsimonious approach to represent DES models is given by Schruben's Event Graph methodology (Schruben 1983). With this methodology, the events themselves are the primary modeling element.

Buss and Sanchez proposed the LEGO framework for DES modeling, which is a method for designing simulation components based on Schruben's Event Graph methodology (Buss and Sanchez 2002). In the LEGO framework, a DES component consists of a set of parameters (variables that do not change within a given replication), state variables (which can change within a given replication) and an Event

Graph, which defines the state transitions (events) and the scheduling relationships between them. The basic Event Graph element is shown in Figure 2.

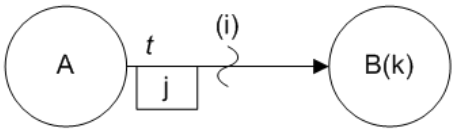

Figure 2: Fundamental Event Graph construct (after Schruben 1983)

The interpretation is as follows: when Event A occurs (i.e. the state transition associated with Event A is executed), then if Boolean condition (i) is true, then Event B is scheduled to occur t time units in the future. When Event B occurs, its argument k is set to the value of expression j when it had originally been scheduled.

In this worldview, a DES component encapsulates both its parameters and its state variables. Specifically, its state at any simulated time is strictly a function of the initial state values together with the sequence of events in that component which have occurred by that time.

2.2 System Dynamics

System dynamics (SD) is a modeling approach whose foundations were laid in the 1950's by Jay Forrester in his pioneering work on "industrial dynamics" (Forrester 1961). However, SD was not widely adopted by the mainstream simulation modeling community until the 1990's, and the terms *simulation* and *discrete-event simulation* were, for much of the 70's and 80's, synonymous. The fundamental principle underlying SD is that the structure of any real-world system determines its behavior over time (Forrester 1961; Sterman 2000). In other words, the way that the separate components of any system relate to and affect each other determines the emergent behavior of the system as a whole. This behavior can be surprising and counterintuitive, and it is only by analysis of the component parts that the reasons for this unexpected behavior can be understood and explained. SD has two distinct aspects, one qualitative and one quantitative. The qualitative aspect involves the construction of *causal loop* or *influence* diagrams through a process of discussion and elicitation with problem owners and other stakeholders. The relevant system elements are identified, and relationships between them graphically depicted by a system of arcs and nodes, where the polarity of an arc indicates the direction of influence, positive or negative. Given that X affects Y, a positive influence means that as X increases, so does Y, whereas a negative influence means that as X increases, Y decreases. The aim is to identify feedback loops, which can be of two kinds: negative or *balancing loops* which retain a steady-state, and positive or *vicious circles* leading to uncontrolled growth. The understanding and insights that this approach can bring are very useful and can be an end in their own right; it may not be necessary to do any further modeling.

However the overall net effect of all the feedback loops in a very complex system cannot be determined merely by inspecting the diagram. The same system element can belong to several feedback loops, some negative and some positive, and it may not be at all obvious which loop dominates and drives system behavior. To determine this it is necessary to quantify the variables, and this is not always straightforward if some variables (e.g. "anxiety") are qualitative. Quantitative SD is implemented through the use of *stock-flow models* in software such as iThink *(*known as Stella to the academic community) and Vensim. Stock-flow models can be conceptualized as a system of water tanks connected by pipes. The rate of flow is governed by taps or valves on the pipes. The "water" which flows around such a system is a continuous quantity. Models can incorporate delays in the feedback effects, and the rates of flow are influenced not only by the quantity of "water" in the stocks, but also by external or *auxiliary* variables which may be taken from the original causal loop diagram. Mathematically, stock-flow SD models are a discretization of a set of ordinary differential equations representing the rates of change of the level of each stock. Clearly, SD models are deterministic and are thus very fast to run as they do not require multiple iterations. For a fuller description see Brailsford (2008).

SD models, and differential equation-based models in general, consists of entirely of system-level state variables; the model equations represent how the state variables depend on each other and change over time. These state variables usually represent aggregated information about the population(s) being modeled. The state variables in System Dynamics models cannot, in general, easily be attributed to individual entities if the level of aggregation of the state variables is too great.

2.3 Agent-Based Simulation

Agent-based simulation (ABS) modeling is a relatively recent modeling technique that is widely used to model complex systems composed of interacting, autonomous "agents" (Epstein and Axtell 1996; Bonabeau 2001; Macal and North 2009). By representing agents with their individual characteristics and behaviors across an entire population, we can explicitly consider and understand how agent diversity affects emergent behaviors of the system as a whole. There is no universal agreement on the definition of an agent partly because it depends on the intended use of a model. However, from a practical modeling standpoint, an entity would seem to need some essential characteristics to differentiate it from an ordinary object and qualify it as an agent, as described below (Macal and North 2010).

An agent is a *modular*, self-contained, and uniquely identifiable individual. The modularity requirement implies that an agent has a boundary. Whether something is part of an agent or not part of an agent can be easily determined. In addition, an agent is *autonomous* and self-directed. An agent can function independently in its interactions with other agents and with (in) its environment, at least over a limited range of situations that an agent encounters within a model. In line with this, agents have *behaviors* that are described by algorithms of varying complexity and abstraction. Behaviors are represented from simple deterministic if-then rules and decision trees, to stochastic learning algorithms and abstract representations of stimulus-response mappings. An agent's behaviors allow it to act autonomously because the behaviors relate information sensed by the agent to its decisions and actions.

An agent has an internal *state*. Just as any dynamic system has a state specified by its state variables, an agent's state represents the essential attributes associated with its current situation within the model. An agent's state is dynamic, for example, as an agent's experiences accumulate and are recorded in its memory. The state of an ABS model is the collective states of all the agents combined with the state of the environment. State is a critical element of an agent model because an agent's behavior is based on, and only on, its state.

An agent is *social* and has dynamic interactions with other agents. Interactions are often represented by networks. Interaction networks specify which agents exchange information and compete for resources. For example, agents may move over a geography and contend for space; the network consists of competing agents.

Agents may also have other interesting characteristics that are perhaps not essential to qualify as agents in the sense that a quite interesting and useful model can be developed with agents that do not have these characteristics. Models with agents that do not have these characteristics can even produce emergent behaviors. These characteristics include whether an agent's behavior is *adaptive* and whether and agent is *goal-directed*, or merely reactive.

A typical ABS *model* has three elements: (1) a set of agents, including their attributes and behaviors, (2) a set of agent relationships and methods of interaction (an underlying topology of connectedness, such as a network, defines with whom and how agents interact), and (3) the agents' environment. A model developer must identify, model, and program these elements to create an ABS model. A computational engine for simulating agent behaviors and agent interactions is then needed to make the model run. An ABS modeling *toolkit*, programming language or other implementation provides this capability. To run an ABS model is to have agents repeatedly execute their behaviors and interactions over time.

3 DISCRETE-EVENT SIMULATION AND SYSTEM DYNAMICS

Traditionally SD has been used at a higher, more aggregated and strategic level than DES. Forrester (1961) believed strongly that SD models were "learning laboratories" and were definitely not optimization tools. The data requirements of an SD model are generally much less than for DES. While SD models are useful for clarifying the complexities of organizational behavior, their simplified representation of systems, the necessity to aggregate entities, and the use of average flow rates are some of their significant limitations. In contrast, DES models are much more flexible, capturing interactions between entities and detailed characteristics of the system being modeled. However, a major disadvantage of DES is that the data requirements, and the need to maintain a next-events list and to carry out long runs or multiple replications to get reliable results, mean that the models are relatively time-consuming to develop and run.

Philosophically, there is a profound difference between the two approaches which over the years has led to the development of two entirely separate research communities, with their own journals and conferences (Lane 2000). Basically, SD modelers and DES modelers see the world in different ways; this affects how they conceptualize problems, in a way which is far more subtle than merely the distinction between discrete and continuous variables. Lane (2000) talks about dynamic as opposed to detail complexity. An SD modeler sees the world as a holistic synthesis of system elements which are dynamically connected, whereas DES modelers look at the world in detail, paying attention to the variability between individual components. To say that SD is strategic and DES is operational is too crude an approximation, but it does give a flavor of one of the key conceptual differences between them.

The idea of combining discrete-event simulation and system dynamics has been a topic of debate in the OR community for over a decade (Brailsford and Hilton 2001; Brailsford, Churilov and Liew 2003), prompted by a recognition that the division between strategic, tactical and operational decision-making was becoming increasingly blurred. During the 1990's there was a growing realization that decisions made at one end of the operations-strategy spectrum could rapidly impact outcomes at the other end. Moreover, it is often difficult to draw clearly-defined boundaries around any part of a large system or organization and say that the resulting subsystem can be studied in isolation. There have been several examples in the literature. Martin and Raffo (2001), for example, developed a hybrid simulation by modifying Extend™, a package designed for both discrete event and continuous simulations. The two case studies described in Brailsford (2010) both contain linked models where a "whole system" SD model contains a smaller, DES sub-system.

It is often very difficult to define the boundaries of a model in a system which appears self-contained but is actually very connected into the wider environment. Of course in practice, the art of modeling involves a pragmatic decision regarding where to draw the line between what is inside and what is outside the model, but this could mean that the model excludes some key elements or drivers of system performance. Suppose we are modeling a production line which is part of a large manufacturing plant owned by a multinational company with a global supply chain. It may well be that the performance of this production line is affected by events occurring upstream or downstream in the supply chain. A devotee of DES would say that it is possible to model the whole supply chain, and indeed it is technically possible. But why ever would you want to do it? Such a model would be massive, cumbersome, slow to run and data intensive. We are only really interested in the performance of a tiny part of the system; we don't even want to model the whole factory in detail, let alone the whole supply chain. All we want to do (as in any model) is to be sure we have captured the important features of the real system, i.e. those key aspects of the "outside world" that impact on the system of interest. Therefore, a broad-brush model which captures these critical features and within which our detailed production line model sits would suit our purpose perfectly. It is those very characteristics of SD which would fit this purpose; the whole-system, big picture approach which slices through all this unnecessary detail and provides an elegant structure which illuminates the key relationships and captures those external factors which we cannot exclude from our DES model.

Of course, there are technical challenges in moving backwards and forwards between continuous and discrete models which raise some very interesting research questions. How much information loss can be

tolerated in switching from DES to SD and back again? How can essential information be retained? These technical questions do not appear to have been addressed in much detail in the mainstream OR literature, where papers have been at more of a paradigm or conceptual level. It would seem relatively easy to merge a group of individual entities into a continuous mass, but how much of the "population profile" would one need to retain and how would one do it? And coming back the other way, when one starts with a continuous population and wants to split this up into individual entities (which feels much harder to do with any degree of rigor), one would need to define algorithms which transformed the original population (the last time it was discrete) into a new population, based on what had happened to it during the time it was continuous.

The software undoubtedly exists to combine both DES and SD. Several DES software packages, such as Witness (www.lanner.co.uk], can model continuous as well as discrete phenomena and can therefore be adapted to provide the underlying structures of SD models. AnyLogic (XJ Technologies 2011) is another powerful package which can provide both DES and SD models (and also agent-based models) with nearly all the main features available in individual software packages such as Vensim, Powersim and iThink (SD tools) and Arena, Simul8 and ProModel (DES tools). However, these packages remain essentially either a DES environment with some continuous features, or an SD environment with some discrete or stochastic features. Moreover, merely including both continuous and discrete variables in the same model is only half the story. There is no genuinely hybrid modeling methodology that combines the characteristic features of both DES and SD. The "holy grail" is a methodology which combines the benefits and virtues of each approach, allowing a truly holistic systems view yet at the same time capturing the detailed individual variability within parts of that system which is the strength of DES models. A genuinely integrated approach would be advantageous because, at a macro level, it could describe the movement of individual entities as a homogeneous flow, which would be relatively fast and data-efficient. In addition, at a micro level, where there are detailed interactions that affected the overall behavior of the system, it would be possible to incorporate important individual characteristics. The real challenge therefore, is not to develop the software, but to develop both a conceptual philosophy and a practical methodology for combining SD and DES.

4 DISCRETE-EVENT SIMULATION AND AGENT BASED SIMULATION

There are many real-world situations that are desirable to model which have some aspects best represented by the DES paradigm and others best represented by an ABS paradigm. This can be any situation which includes resources that must perform activities as well as human interactions where individual behaviors alter how these activities proceed. One example is mass-casualty disaster response situations (Heath et al 2009; Heath 2011). It is clear to see that a wide variety of health care situations could be included here as well. Management and production-related situations can also fall in this arena, as shown in Seibers and Aickelin (2011) and Hao and Shen (2008).

When considering integrating DES and ABS modeling, it turns out that the DES worldview matters. We will first look at the topic using a process-oriented DES worldview, and then address it using an event-oriented DES worldview.

4.1 DES and ABS with Process-Oriented DES Worldview

DES and ABS are both very useful simulation paradigms with some commonalities, but they also have significant differences that make cross-paradigm modeling challenging. Similarities include the fact that both are used to model stochastic elements and both are generally used at a low level of aggregation, focusing on individual entities or agents. One significant difference is that DES uses system-level rules to govern the movement and behavior of entities, and these entities do not have rules within them that alter movement or behavior based on entity-entity or entity-environment interactions. A fundamental idea of ABS, however, is that agents do have rules within them that alter movement or behavior based on agent-agent or agent-environment interactions.

The fact that ABS has rules within agents and DES includes only system level rules makes some things that are easy to model in software designed for one paradigm difficult to model in software designed for the other paradigm. For example, it is relatively easy to use ABS to model an agent who prematurely completes an activity based on environmental conditions. In DES, however, once an entity begins processing, it is difficult to interrupt that processing based on changes in the environment that occur after the processing has begun. In the other direction, it is easy in DES software to model queueing behavior, as well as the processing of entities that require multiple resources. In some ABS toolkits, however, it is more difficult to model this type of behavior (Heath et al 2009). The ease with which discrete-event type behavior can be modeled using an ABS toolkit may depend on how much the toolkit has been specialized through the development of different constructs or features which are designed to make some more common agent behavior easier to model. (The development of these features may have the consequence of making some more traditional discrete-event type behavior more difficult to model.)

Another possible approach to take, besides trying to construct a single model with both ABS and DES attributes, is to use a two-model approach. In this approach, the same situation is modeled in two different software packages, one designed for DES modeling and the other designed for ABS modeling, and use the models to inform each other. In this way it may be possible to draw on strength of both modeling paradigms. For example, a DES model of a situation could be used to determine expected queueing times as one type of entities/agents wait for limited resources. These waiting time distributions could then be incorporated into an ABS model as additional delays. On the other hand, an ABS model could be used to see how agents are redirected to move toward a different goal or perform a different functions or call for additional resources over the course of the scenario. This information could then be incorporated into a DES model using timed triggers or probabilities to simulate this emergent behavior. There are significant drawbacks to this approach, however. One is that it needs to be an iterative approach, with a couple cycles of using the results of one model to improve the other model; and making any significant changes to the situation modeled would require updating both models and repeating this iterative improvement of the models. In addition, the information in one model derived from the other model may not be properly related to, or correlated with, the behavior of the rest of the model.

Clearly, a way to model ABS and DES model characteristics in a single integrated model is the best possible option, but software that allows complete integration of these paradigms is not yet available. One example of software that has made a significant step in this direction is AnyLogic. This software has utilities to construct ABS and DES models and link them together allowing the behavior of one model to influence the other over the simulation run, but it is not yet a fully integrated solution. Another software that has promise in this area is Repast, which has generalized time-advance scheduling mechanisms (Repast 2011).

4.2 DES and ABS with Event-Oriented DES Worldview

In order to discuss how ABS models could potentially be incorporated with DES models, constructed with an event-oriented worldview, we build upon the initial description of Buss and Sanchez' LEGO framework begun in Section 2.1.2. Recall that, using this framework, a DES component encapsulates both its parameters and its state variables. Specifically, its state at any simulated time is strictly a function of the initial state values together with the sequence of events in that component which have occurred by that time. As can be seen, this is a step towards using DES to model agents, since encapsulation of state within distinct components is a necessary feature for ABS modeling. Furthermore, it is straightforward to create many instances (copies) of a given DES component.

Multiple DES components can be connected allowing one component to essentially 'listen' to another component using the SimEventListener pattern, shown in Figure 3. Both the Source and the Listener are DES components, although in general they are not the same type. The SimEventListener pattern works as follows: whenever an event occurs in the Source component, it is "heard" by the Listener component (after first executing its state transition and scheduling any events within the Source component). When the Listener component "hears" an event, it checks to see whether it has an event that is identical in name and

signature. If one is found, then it is executed as if it had been scheduled by the Listener component. The one exception is that a "heard" event is not dispatched to any listeners. There is no restriction on the number of SimEventListeners a component may have, nor on the number of other components to which a component is listening.

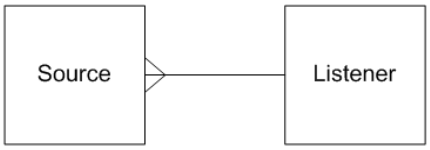

Figure 3. The SimEventListener pattern

The connector in Figure 3 is shaped like a stethoscope to suggest "listening." Alternatively, the triangular part on the left can be thought of as the direction of the flow of events.

A form of SimEventListening has proved to be so useful it has been incorporated into the framework as its own connector. The situation occurs when the modeler desires an event of one name in a component to trigger an event of another name (but identical signature) in another. This is done using the Adapter pattern, shown in Figure 4.

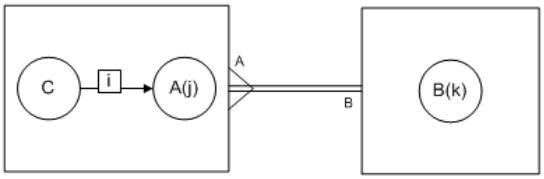

Figure 4: Adapter: When event A occurs, Event B also occurs.

In Figure 4, when the event A occurs in the left component, the event B is triggered in the right one. Unlike SimEventListener, the Adapter only applies to one event. If more are desired, then an adapter must be created for each one.

4.2.1 DES Approach to Modeling Agent Behavior

The first insight to utilizing DES for Agent-Based Modeling (ABM) is to utilize the basic approach to DES based on Events and the primary elements, as described above and exemplified by the Event Graph methodology, rather than the commercially popular entity/process/resource approach. The component framework described in the previous section forms the basis of what could be a fruitful and robust methodology to modeling agent behavior.

Perhaps the most useful feature of a DES approach is how it serves to more closely represent how agents (and what they represent) behave compared with a time-step approach. Recall that with a time-step approach, first the clock is advanced by a fixed amount Δt, and then all agents update their state. In addition to the inaccuracies described previously, closer examination reveals a disconnect between how platforms, such as humans, vehicles, aircraft, etc., operate and how they are modeled in this manner. It is easiest to see by considering human behavior. Simply put, people do not re-evaluate their entire state at small time increments. Rather, they engage in an activity for a period of time, and then move to another activity. The move to the second activity could be motivated by either the completion of the original one or by some interruption that caused the change in the activity.

Notice that the above description is very closely aligned with a DES world view, and specifically the component approach described above. That is, an agent will be in a particular state for a certain period of time, and then change to another state for another period of time based on an event occurring. The new state will of course depend on which event is the one that causes the change. It is therefore natural to

model each agent as an instance of an Event Graph component. Different agents may be based on identical components but with different parameter values. Other agents may have entirely different internal states and Event Graphs governing their behaviors.

Since often the agents in a model are required to move about, a DES approach to modeling movement is necessary for a comprehensive approach. It turns out there is such an approach to modeling movement in a pure DES manner (see Buss and Sanchez 2005). Briefly, that approach begins by not making an agent's location part of state, since that would necessitate a continuously changing state variable which is inadmissible in a DES model. Rather, the starting point is an equation of motion that describes the agent's movement. The DES state therefore includes the initial conditions for that equation of motion, and these initial conditions do stay constant throughout the agent's maneuver. See Buss and Sanchez (2005) for further discussion of this approach.

4.2.2 Interactions with Environment and Other Agents

The SimEventListener pattern provides a clean and flexible way of modeling the interaction of agents with their environment as well as other agents. This is illustrated in Figure 5.

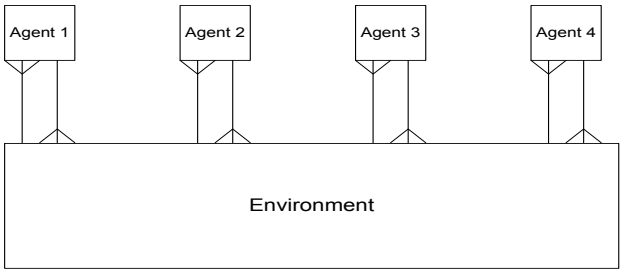

Figure 5: Agents Interact with Environment by Listening.

Note that each agent both listens to and is listened to by the Environment, which itself consists of a number of different components. A simple example of this type of interaction is illustrated by DES component model for sensing (see Buss and Sanchez 2005). In that, the components of the Environment include a Referee for determining and scheduling interactions between sensors (components of the agents) and targets (the movement components of the agents). Other Mediator components adjudicate and schedule the Detection events which are heard by the appropriate sensor components. Other components of the Environment could include obstacles, which would affect the movement of agents, communications, and weapons effects between hostile agents.

5 AGENT BASED SIMULATION AND SYSTEM DYNAMICS

To begin a useful discussion on SD and ABS, let us consider what we mean by ABS models and ABS modeling. Do we mean the theory of ABS modeling, the background and motivations for doing ABS modeling, or the algorithms that are commonly employed in ABS models? Do we mean the available software and the toolkits, or the steps that people use to build ABS models? It has been my experience that the people who want to do ABS modeling have questions stemming from all these perspectives. It is instructive to compare ABS modeling and SD along several of these dimensions. Each technique takes a different approach to analyzing the system being modeled, designing the model, and implementing the design in software.

Some people come to ABS modeling from SD because they find they cannot include the kinds of features they would like to into their models. ABS modeling and SD take fundamentally different perspectives when modeling a system. The ABS modeling approach has been described as bottom-up, i.e., modeling a system by modeling the individual entities that compose the system and their interactions. The SD approach is considered top-down, i.e., modeling a system by breaking it into its major components and

then modeling the component interactions. But generally this process of disaggregation does not go to their individual entity level. In addition to the differences in perspectives, ABS modeling and SD modeling have largely independent histories with different approaches to model building. SD modeling came out of control theory and was influenced by cybernetics, whereas ABS modeling was motivated by Artificial Life (Macal 2009), social interactions (Axelrod 1997), and natural evolution (Holland 1995).

Suppose we would like to build an ABS model of a system. Where do we begin? Good modeling practice dictates that we begin by analyzing the system and developing a model design – implementation is separate and comes later, based on the modern software development philosophy of the clean separation of design and implementation. For ABS modeling, the natural starting place is to use the techniques of object-oriented (OO) design, due to the close relationship between OO and ABS modeling. Another way to begin the process is with the techniques for developing SD models. SD has a very mature thought process for analyzing systems and designing models, and has been well documented (e.g. Sterman 2000; Morecroft 2007). This initial top-down design can then be used as the basis for the bottom-up development process characteristic of ABS modeling.

Many, if not all, systems can be modeled using either an ABS or an SD approach. Generally, the respective models of the same real-world system are different and produce different quantitative results. However, there are cases of SD models in which the exact numerically equivalent results can be obtained (North and Macal 2009). Several classic models, such as the Kermack-McKendrick model of epidemic dynamics, the Lotka-Volterra equations for modeling predator-prey relationships, the Bass model for innovation diffusion, chemical kinetics models, and many others are formulated as systems of ordinary differential equations and have corresponding SD representations. It is well-known that the Kermack-McKendrick model in particular can be formulated as an ABS also, and produces similar results to an SD formulation (Bagni et al 2002; Borshchev and Filippov 2004; Wakeland et al 2004; Epstein et al 2008; Rahmandad and Sterman 2008). Macal (2010) elaborates on this similarity and describes the specifics of how to transform an SD model into an "equivalent" ABS model; these equivalent SD and ABS models are different in form and do not necessarily produce the same numerical results given identical assumptions. The issue of equivalent representations is whether the state as represented in a SD model is cleanly, and unambiguously, decomposable into agent representations. However, most SD models do not have state definitions that are cleanly decomposable into state representations at the individual entity level. The relationship between SD and ABS is an area of interest and research (Scholl 2001; Parunack 1998; Marin et al 2006; Norlin 2007).

An ABS model, as any model, is a model of a system embedded in a larger system. Good modeling practice is to include system details beyond the boundary of the system of interest as aggregate representations. Thus, a natural hybrid approach is to combine elements of SD, which represent aggregate state variables that influence agent behavior, into an ABS model. For example, economic agents in a market model make pricing and production decisions base on macroeconomic variables, such as economic growth and interest rates, which are modeled at an aggregate level using SD. Borshchev and Filippov (2004) address this ABS/SD hybrid approach, which has been incorporated in the AnyLogic simulation system. Repast also has the capability to combine ABS and SD approaches in a single model.

One can chooses how to frame the relationship between SD and ABS modeling as either inclusive or exclusive. In the exclusive view, the issue is SD *versus* ABS modeling, making it into a debate as if it were a winner take-all competition. In this regard, SD is not the modeling technique of choice if these conditions are present (North and Macal 2007):

- When the problem has a strong spatial or geographical component, as when agents need to interact over a spatial landscape, for example in modeling the locations of consumers and retail markets,
- When the problem involves networks of agents that interact over time, especially if these networks are dynamic in the sense that they are created and transform themselves as a result of the agent interactions that go on within the model,

- When the problem has discrete decision variables, as when a state variable is required to be a whole number at all times and fractional levels don't make sense, such as in a supply chain example when items being ordered and shipped can only come in discrete units,
- When the problem has constraints on decision variables, as when we require a state variable to be within a range, such as when we need to maintain inventory within capacity limits

In contrast, ABS readily addresses these aspects of modeling a system because it was in part developed to do so.

ABS and SD can be used together in a very constructive way. From an inclusive perspective, the best capabilities from ABS and SD, whether they be philosophical or implementation-related, are incorporated into an effective systems modeling process, even to the point of combining the techniques in a single hybrid model. ABS modelers have much to learn from the SD community and probably vice-versa.

6 TIME-ADVANCE MECHANISMS

The method of time advancement used in all three simulation paradigms is not intrinsic to their methodologies. However, when implementing a model in a simulation toolkit, a time-advance mechanism must be specified and the choice of a time-advance mechanism is important; it turns out to have a greater impact on the results of a model than might appear at first glance. This section will first describe the two basic methods of time advance: Time Step and Next Event. Then, for each paradigm, we discuss the time-advance mechanism(s) currently in use, and follow that with a discussion of the implications for each approach.

Time Step begins by specifying a time increment, Δt. All simulated time is a multiple of this value. The Time Step method works by first incrementing the simulation clock by Δt, then updating the state of the model. In contrast, the Next Event method of time advance associates state transitions with events, which are scheduled to occur at some time in the future. These events may be scheduled to occur at *any* time (up to the ability of the computer to represent floating point numbers).

DES models all use the Next Event method of time advance; indeed, the Next Event approach is tightly associated with DES. For models implemented using a process-oriented worldview, as are built using most commercial software, this is all hidden from the modeler. Although the particular events, their state transitions, and the underlying Event List are present and make the model execute, the model itself is expressed in terms of process blocks, resource levels, and entity definitions. On the other hand, the Event Graph and LEGO methodologies, being based on the event-oriented worldview, work directly with the events themselves and explicitly schedule the times of future events. In the cases where a continuous variable is incorporated into a DES model, the common approach is to schedule events at fixed time intervals and evaluating the continuous variables at these events, effectively introducing Time Stepping to the model.

SD models, being systems of differential equations at their core, are typically solved using standard numerical methods, most often the Euler method and a form of the Runge-Kutta method. Each of these may be thought of as a form of Time Step, since they both begin by discretizing time, then approximating the differential equations with difference equations.

Many current implementations of ABS models use the Time Step method of time advance. After incrementing the clock by Δt, each agent updates their state, effectively performing state transitions. These transitions occur simultaneously, that is, at the exact same simulated time. To inform the discussion of the implications of this time advance mechanism we need to distinguish between discrete processes and continuous processes. We note that not all ABS models include continuous processes (e.g. Conway's Game of Life). For these models there may be no implication of the use of the Time Step method. However, in models that represent continuous processes, state changes may need to occur at times other than the discrete time-step points in time. For these types of models, using the Time Step method could have significant implications, as will be discussed below. At this time, the authors are not aware any ABS models that include continuous processes and do not use the Time Step method, although some ABS toolkits are starting to incorporate the ability to schedule events at times other than the fixed time-step points in time.

Although the Time Step approach is most often used for SD models as well as for ABS models with variables that change continuously over time, there are a number of problems that are typically overlooked by modelers. One difficulty with a Time Step implementation is the fact that $1/\Delta t$ becomes a hard upper-bound on the rate at which anything can occur over time. This has the potential of making it very difficult to ask certain questions of the model, such as "what if we could make decisions 10% faster through better training?"

Perhaps even more seriously, the introduction of a time step increment adds artifacts to the behavior of the model that are often spurious. That is, a given model can produce vastly different results purely as a function of the size of its time step Δt. This should not come as a surprise to anyone with experience in the numerical solution of differential equations. It is well known that a naïve Euler method (which is directly analogous to how a Time Step ABS operates) can have wildly different solutions depending on the coarseness of the time discretization, that is, the size of the time step Δt. Even the Runge-Kutta method, which generally has superior stability of solutions, can have difficulty with certain "stiff" equations. It is conceivable that some SD models also may produce erroneous results because of the size of Δt. When specifically modeling continuous agent actions in ABS, such as moving and sensing, the problems induced by the size of Δt can prove to be very serious indeed. Difficulties include not reaching waypoints at the appropriate times, getting "stuck" at waypoints by not being able to "find" it, and missing detections by "skipping" over the targets. These and other problems are explored in more detail in Ali, Buss and Lieberman (2011). A common way to mitigate these problems is to make Δt very small. In some cases this may be a viable approach, but in others it could increase the run-time so much that the model cannot be used effectively.

In addition to the artifacts introduced by discretizing time, the Time Step paradigm is not an ideal one for representing human behavior. Most people do not constantly update their internal state on a rapid periodic basis. Rather, they tend to engage in an activity, which itself typically consists of other activities, over a period of time, and then update their internal state and move on to another activity. For example, a person may begin reading a book. Her focus is on that activity for awhile, which has a number of substates, such as the particular words she is currently reading, which page she is on, etc. When finished reading, she would then move to another state. While reading, however, she will typically not be updating her state every Δt time units, as time-step agent models are wont to do. This description of human activity is much closer to the DES Next Event approach than the Time Step approach.

7 CONCLUSIONS

The panel has explored three seemingly disparate approaches to creating simulation models: Discrete Event Simulation (DES), Agent Based Simulation (ABS) and System Dynamics (SD). In addition, the two significantly different worldviews for DES modeling were explained. Each individual modeling paradigm has a rich history and exemplar cases in which the strengths of the respective methodology make it a good choice for a particular modeling situation. And, recognizing that each paradigm has its strengths, the possibilities for combining each pair of approaches to develop hybrid models was explored.

Overall, from the discussion presented in this paper it seems that the ideal of a modeling methodology and toolkit for developing completely integrated cross-paradigm simulation models is still out of reach, although significant progress has been made in that direction. For each pair-wise hybrid possibility some software appears to have functionality within it to support such hybrid modeling, although there are currently shortfalls in defining the precise hybrid methodology itself. It is simply not enough for software to add certain features; what is needed is to develop the philosophical and methodological underpinnings that would give weight and validity to such hybrid models. Significant steps forward are being made in the research community though. One such step in the area of combined DES-ABS modeling has been included in this paper: leveraging the event-oriented worldview through Event Graph or LOGO models to effectively integrate agent behavior into a DES environment. This methodology may also prove beneficial in achieving ways to model agent behavior without the drawbacks of the Time Step time-advance me-

chanism. Another step forward in this area has also been made by in the Repast software toolkit, which incorporates DES-type event scheduling into an ABS toolkit. Yet more sophisticated time advance mechanisms may be incorporated into software toolkits to facilitate cross-paradigm simulation modeling in the future. For example, the combined continuous/discrete modeling approaches introduced by GASP IV (Pritsker 1974) and historically incorporated into other simulation packages may be a promising area for future development.

Another approach to cross-paradigm modeling is to develop a model within each paradigm and cycle between them carrying information from one model to the other. Clearly this approach has significant drawbacks but may prove more beneficial than sacrificing aspects of the model to stay within one paradigm. In addition, there is not yet well developed methodology to tackle all the issues of a dual-model approach.

As we move to develop more comprehensive simulation models, this issue of cross-paradigm simulation modeling will only increase in importance. There are many significant open research questions in this area including how to think about systems that are to be modeled through the lens of two or more divergent paradigms, ways to construct fully integrated models, and how to implement these models in a simulation software, especially considering the issues associated with the two different time-advance mechanisms.

REFERENCES

Al Rowaei, A, A. Buss, and S. Lieberman. 2011. "The Effects of Time Advance Mechanism on Simple Agent Behaviors in Combat Simulations". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

Axelrod R. 1997. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton University Press: Princeton, NJ.

Bagni, R., R. Berchi and P. Cariello. 2002. "A Comparison of Simulation Models Applied to Epidemics". *Journal of Artificial Societies and Social Simulation* 5(3). http://jasss.soc.surrey.ac.uk/5/3/5.html.

Bonabeau, E. 2001. "Agent-Based Modeling: Methods and Techniques for Simulating Human Systems." *Proceedings of National Academy of Sciences* 99(3): 7280-7287.

Borshchev, A., and A. Filippov. 2004. "From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools". *The 22nd International Conference of the System Dynamics* Society. July 25-29, 2004. Oxford, England.

Boyd, J. 1995. "The Essence of Winning and Losing". Five slide set from briefing on 28 June 1995.

Brailsford S.C., Churilov L. and Liew S-K. 2003. "Treating Ailing Emergency Departments with Simulation: An Integrated Perspective". In *Proceedings of Western Multiconference on Health Sciences Simulation*. Florida. ed. J.Anderson.

Brailsford S.C. and Hilton N.A. 2001. "A Comparison of Discrete Event Simulation and System Dynamics for Modelling Healthcare Systems". In *Proceedings from ORAHS 2000*, edited by J. Riley. 18-39.

Brailsford S.C. 2008. "System Dynamics: What's in it for Healthcare Simulation Modelers". In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. Hill, L. Moench, and O. Rose. 1478-83. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

Brailsford S.C., Desai M.S. and Viana J. 2010. "Towards the Holy Grail: Combining System Dynamics and Discrete-Event Simulation in Healthcare". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

Buss, A. and A. Al Rowaei. 2010. "A Comparison of the Accuracy of Discrete Event and Discrete Time". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

Buss, A. and P. Sanchez. 2005. "Simple Movement and Detection in Discrete Event Simulation". In *Proceedings of the 2005 Winter Simulation Conference*, edited by M. Kuhl, N. Steiger, F. Armstrong, and J. Joines. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

Buss, A. and P. J. Sanchez. 2002. "Modeling Very Large Scale Systems: Building Complex Models with Legos (Listener Event Graph Objects)." In *Proceedings of the 2002 Winter Simulation Conference,* edited by E. Yücesan, C. H. Chen, J. L. Snowdon, and J. M. Charnes, 732-737. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

Epstein, J. M., and R. Axtell. 1996. *Growing artificial societies: social science from the bottom up*. MIT Press: Cambridge, MA.

Epstein, J. M., J. Parker, et al. 2008. "Coupled contagion dynamics of fear and disease: mathematical and computational explorations". *PLoS ONE* 3(12): e3955.

Forrester J.W. 1961. *Industrial Dynamics*. MIT Press, Cambridge, MA. reprinted by Productivity Press (1994) and now available from Pegasus Communications, Waltham, MA, USA.

Hao, Q., and W.M. Shen. 2008. "Implementing a hybrid simulation model for a Kanban-based material handling system". *Robotics And Computer-Integrated Manufacturing* 24(5). 635-646.

Heath, S.K., D. Dolk, E. Lappi, B. Sheldon, and L. Yu. 2009. "Investigating the Use of Simulation Tools for Mass Casualty Disaster Response". *Scythe*. Issue 6. Workshop 18. http://harvest.nps.edu/scythe/Issue6/Scythe6-IDFW18-Team07.pdf .

Heath, S.K. 2011. "Mass Casualty Disaster Response Simulation Modeling". *Working Paper*.

Holland, J. 1995. *Hidden order: how adaptation builds complexity*. Addison-Wesley: Reading, MA.

Lane, D.C. 2000. "You just don't understand me: Modes of failure and success in the discourse between system dynamics and discrete event simulation." *LSE OR Dept Working Paper* LSEOR 00-34. London School of Economics and Political Science.

Law, A. and D. Kelton. 2000. *Simulation Modeling and Analysis*. 3rd Edition. New York: McGraw Hill.

Macal, C. M. 2009. "Agent based modeling and artificial life". In Meyers R (ed). *Encyclopedia of Complexity and Systems Science*. Springer: New York. 112–131.

Macal, C. and M. North. 2009. "Agent-based modeling and simulation". In *Proceedings of the 2009 Winter Simulation Conference*. edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. 86-98. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

Macal, C. 2010. "To Agent-based Simulation from System Dynamics". *Proceedings of the 2010 Winter Simulation Conference*. B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds. 86-98. Wiley-IEEE Press.

Macal, C.M., and M.J. North. 2010. "Tutorial on Agent-Based Modelling and Simulation". *Journal of Simulation*. Special Issue: Agent-Based Modelling 4(3): 151–162.

Marin, M., Y. Zhu, P. T. Meade, M. Sargent, and J. Warren. 2006. "System dynamics and agent-based simulations for workforce climate." In *Proceedings of the 2006 Winter Simulation Conference*, edited by L. R. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 667-671. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Martin R. and Raffo D. 2001. "Application of a hybrid process simulation model to a software development project". *The Journal of Systems and Software*. 59:237-246.

Morecroft J. 2007. *Strategic Modelling and Business Dynamics: A Feedback Systems Approach*. Wiley, London, UK.

Norling, E. 2007. "Contrasting a system dynamics model and an agent-based model of food web evolution". in *Multi-Agent-Based Simulation* VII. 57-68. Lecture Notes in Computer Science. Springer Berlin / Heidelberg. Volume 4442/2007.

North, M. J., and C. M. Macal. 2007. *Managing business complexity: discovering strategic solutions with agent-based modeling and simulation*. Oxford University Press: Oxford, U.K.

North, M. J., C. M. Macal. 2009. "Agent-based modeling and systems dynamics model reproduction". *International Journal of Simulation Process Modeling* 5(3): 256-271.

Parunak, H., R. Savit and R. Riolo. 1998. "Agent-based modeling vs. equation-based modeling: a case study and users' guide". In *Proceedings of Workshop on Modeling Agent Based Systems* (MABS98).

Pritsker, A. A. B. 1974. *The GASP IV Simulation Language*. Wiley.

Rahmandad, H., and J. Sterman. 2008. "Heterogeneity and network structure in the dynamics of diffusion: comparing agent-based and differential equation models". *Management Science*. 54(5): 998–1014.

Repast. 2011. Repast home page. http://repast.sourceforge.net/.

Scholl, H. J. 2001. "Agent based and system dynamics modeling: a call for cross study and joint research." *34th Annual Hawaii International Conference on System Sciences* (HICSS-34). vol. 3. 3003.

Schruben, L. 1983. "Simulation Modeling with Event Graphs". *Communications of ACM* 26: 957-963.

Siebers P.O. and U. Aickelin. 2011. "A First Approach on Modelling Staff Proactiveness in Retail Simulation Models". *JASSS-The Journal Of Artificial Societies And Social Simulation* 14(2).

Sterman J.D. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Irwin McGraw-Hill, Boston, MA.

Wakeland, W. W., and E. J. Gallaher, L, M. Macovsky, and C. A. Aktipis. 2004. "A comparison of system dynamics and agent-based simulation applied to the study of cellular receptor dynamics". *37th Annual Hawaii International Conference on System Sciences* (HICSS-37).

XJ Technologies. 2011. AnyLogic home page. http://www.xjtek.com/.

AUTHOR BIOGRAPHIES

SUSAN K. HEATH is an assistant professor of logistics and operations management in the Graduate School of Business and Public Policy at the Naval Postgraduate School in Monterey, CA. She received her Ph.D. in management science and information systems from the University of Texas at Austin in 2006. She also has a Master of Engineering degree in operations research and industrial engineering, as well as a B.S. in psychology, from Cornell University. From 1997 to 2000 she worked as a senior business analyst in information systems for Kraft Foods, Inc. Her research interests include disaster planning and response, production planning and scheduling, simulation and optimization. Her email address is <skheath@nps.edu>.

SALLY C. BRAILSFORD is Professor of Management Science at the University of Southampton, UK. She received a BSc in Mathematics from the University of London, and MSc and PhD in Operational Research from the University of Southampton. Her research interests include simulation modeling methodologies, system dynamics, health service research and disease modeling, and the modeling of human behavior in healthcare systems. She is chair of the European Working Group on OR Applied to Health Services (ORAHS) and is on the editorial boards of Health Care Management Science, the Journal of Modeling in Management, the Journal of Simulation and the Flexible Services & Manufacturing Journal. Her email address is: <s.c.brailsford@soton.ac.uk>.

ARNOLD BUSS is a Research Associate Professor in the MOVES Institute at the Naval Postgraduate School. His research is in the area of Discrete Event Simulation modeling, particularly in developing effective and reusable frameworks for developing models. His e-mail address is <abuss@nps.edu>.

CHARLES M. MACAL is the Director of the Center for Complex Adaptive Agent Systems Simulation, Argonne National Laboratory, Senior Fellow in the joint Computation Institute of the University of Chicago and Argonne, and Adjunct Professor in the Graham School of the University of Chicago. He is a member of the INFORMS Simulation Society, the Association for Computing Machinery, the Society for Computer Simulation International, the Systems Dynamics Society and a founding member of the Computational Social Simulation Society. Dr. Macal has a Ph.D. in Industrial Engineering & Management Sciences (Operations Research) from Northwestern University and a Master's Degree in Industrial Engineering (Computer Simulation) from Purdue. He is also a Registered Professional Engineer. His email address is <macal@anl.gov>.