*Proceedings of the 2017 Winter Simulation Conference*
*W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.*

# PROPOSED UNIFIED DISCRETE EVENT SIMULATION CONTENT ROADMAP FOR M&S CURRICULA

James F. Leathrum, Jr.
Roland R. Mielke
Andrew J. Collins
Michel A. Audette

Department of Modeling, Simulation and Visualization Engineering
Old Dominion University
1300 Engineering and Computational Sciences Building
Norfolk, VA 23529, USA

## ABSTRACT

This paper presents a new educational roadmap for teaching discrete event system (DES) simulation software design. This roadmap represents the hierarchical structure and inter-relationships characterizing the worldviews of DES simulation, namely, event scheduling and process interaction. The roadmap was developed from the authors' experience while teaching DES simulation to both undergraduate and graduate students, spanning several years. The roadmap's development was motivated by the need to strive for greater completeness as well as fewer inconsistencies in the material curriculum. The commonality between the worldviews is highlighted in striving for a uniform approach. The intent of this paper is to provide other educators a foundation for their own DES simulation course development. A simple example is used to illustrate the worldviews within the roadmap.

## 1 INTRODUCTION

In 2010, Old Dominion University established the Department of Modeling, Simulation and Visualization Engineering. The department was established to administer the Modeling and Simulation (M&S) masters and doctoral programs and to host a newly created undergraduate program in Modeling and Simulation Engineering (M&SE). Mielke, Leathrum, and McKenzie (2011) present an overview of these programs and Leathrum and Mielke (2012) provide a more detailed view of the undergraduate program. Two student constituencies that must be served by an undergraduate curriculum are the simulation *user* constituency and the simulation *developer* constituency. Simulation users are students who wish to utilize M&S as a tool to investigate another discipline, students who generally wish to minor in M&S. They need to know enough about M&S to select the best methodologies for their specific application and then to apply these methodologies in an appropriate way. Simulation developers are students who wish to study M&S as a discipline; they are our students who wish to major in M&S. Their focus is to learn about the technical intricacies of M&S and then to develop new M&S methodologies and enhanced M&S technologies. It is vital that they possess the knowledge and skills to design, implement, and utilize simulation-based solutions to a wide variety of problems. This requires going beyond simply using simulation tools and simulation languages, namely that students be prepared to develop simulations in general-purpose software languages.

Most of the introductory textbooks for discrete event simulation, appropriate for use at the freshman, sophomore, and junior levels, were developed to address primarily the needs of the simulation user constituency. They are excellent for teaching students how to use a simulation package but do not present sufficient breadth to make students aware of the many design alternatives available when developing

simulation software. A new core course *Simulation Software Design* was especially difficult to create because of the absence of appropriate educational materials. Our initial course content design for our *Discrete Event Simulation* course followed the usual textbook approach of introducing a simple queuing system example/model and using an event scheduling approach to event management, while our simulation software design course focused on implementing those same queuing models in software.

Experience teaching these topics over the past several years has expanded both the depth and breadth of content covered in the Discrete Event Simulation and Simulation Software Design courses. The added content is selected to expand the perspective that students have of discrete event simulation, and to make it possible to appreciate the design alternatives that are available when conceiving and implementing simulation software. However, rather than simply adding content to an already crowded curriculum, an attempt was made to construct a logical and consistent approach to discrete event simulation design. The result of our efforts is a discrete event simulation content *roadmap*. For educators, the roadmap assists in curriculum and course design. It facilitates the identification of content that lectures will cover in depth as opposed to content that only needs to be introduced to provide necessary understanding. For students, the roadmap provides a mental model that helps them organize their understanding of the role of each of the roadmap components and that identifies the design and development alternatives as one proceeds from conceptual simulation model to simulation software implementation.

The purpose of this paper is to introduce and explain the discrete event simulation content roadmap and to present an example of how the roadmap is implemented in an undergraduate modeling and simulation curriculum. In Section 2, we present additional motivation supporting the need to develop a curriculum roadmap. In Section 3, we introduce the roadmap. The various layers of the roadmap are defined, and we discuss design alternatives for realizing a discrete event simulation. The implementation of the content roadmap in our undergraduate Modeling and Simulation Engineering (M&SE) program is described in Section 4. The resulting changes that have been made in an introductory discrete event simulation course and a follow-on course in simulation software design are presented. Finally, we present conclusions and future work in Section 5. It is our hope that this paper will serve as a helpful example for those interested in developing an M&S curriculum. We also hope that it serves as a catalyst encouraging additional research on enhancements to M&S curricula.

## 2 CURRENT STATE OF DISCRETE EVENT SIMULATION SOFTWARE

Since this paper addresses the educational process for developing student expertise in discrete event simulation software, it is important to reflect on the current state of modern simulation software. Rashidi (2016) provides an assessment of current simulation software packages, categorizing them based on six taxonomies. He bases the majority of his taxonomies on the simulation development environment or higher level modeling perspectives. The focus of this paper is the definition, representation, and management of events, which falls under Rashidi's first taxonomy: worldviews. Worldviews are defined as some subset of four approaches: event scheduling, activity scanning, three-phase, and process interaction (Balci 1988). The approaches differ in the representation of events and the underlying management of events. The two most prevalent approaches used in commercial software are event scheduling and process interaction (Rashidi 2016). In event scheduling, the system is viewed from the perspective of system state. Time-stamped events are ordered by time of occurrence. As the events occur over time, each event may change the system state and may schedule new future events. This worldview requires the modeler to express the system model in terms of events and states, which for many systems is known to be difficult (Pegden 2010). In process interaction, the system is viewed from the perspective of entities moving through one or more system processes. Since the late 1980's, process interaction has replaced event scheduling (Pegden 2010) as the most frequently used approach. Rashidi's survey of 62 simulation packages supports this claim, finding 32 software packages using process interaction, 26 using event scheduling, and a total of 20 using forms of the other two worldviews. However, it is interesting to note that the overwhelming majority of available general simulation educational materials still focus on

the event scheduling approach. This educational focus may be due to historical reasons (event scheduling was developed earlier), a consequence of the simulation name (it is Discrete Event Simulation, not Discrete Process Simulation), or the fact that process interaction representations can be converted to an event scheduling representation (Overstreet and Nance 2004).

The use of commercially available simulation packages or environments has become very popular because even an M&S novice is able to develop and execute a simulation model. Detailed knowledge of simulation software design and development is no longer a prerequisite. This approach to M&S is encouraged by our simulation user approach to introductory discrete event simulation education. However, recent publications (Pidd 2004; Schriber, Brunner, and Smith 2014) have called attention to the dangers of using a simulation package without understanding how it works. These dangers provide additional motivation to modify the current approach to discrete event simulation education, not only for the simulation developer student constituency, but also for the serious simulation user student constituency.

## 3 DISCRETE EVENT SIMULATION CONTENT ROADMAP

In this section, we present a new discrete event simulation content roadmap. The roadmap addresses the modeling activities that begin with the conceptual modeling of the simuland (the thing to be simulated) and end with a design model for the simulation software implementation. The modeling steps are captured in the roadmap as modeling layers. Each modeling layer identifies the design alternatives available at that layer. When we implement the roadmap in a course sequence, we provide additional instruction, problem assignments, and laboratory exercises so that students immediately apply their theoretical knowledge to specific discrete event simulation design problems. The discrete event simulation content roadmap is shown in Figure 1.

### 3.1 Purpose

The roadmap gives structure to a sophomore level Discrete Event Simulation course and a junior level Simulation Software Design course, leading into a junior level Model Engineering course. The roadmap is not intended to reflect industry standards, but rather to provide an educational framework for students to understand the full spectrum of topics. To enable effective student learning, it is critical to have a precise partitioning of the material covered, even if in practice, the dividing lines are blurred. Our current roadmap supports only the two introductory courses indicated and does not yet reflect the complete curriculum. For example, formalisms such as DEVS are not included as students are not introduced to formalisms until the junior level Model Engineering course with graduate courses providing a much more in-depth discussion on formalisms. The roadmap also does not yet include the breadth necessary to include the junior level course Continuous Simulation. A more complete discussion of the curriculum is provided in Section 4 and in (Mielke, Leathrum, and McKenzie 2011; Leathrum and Mielke 2012).

### 3.2 Roadmap Overview

The approach that was taken in developing a simulation content roadmap was to identify a consistent educational model for addressing the full spectrum of worldviews. The first step was to find the commonality between the various models, and then to highlight the differences. We found the commonalities by using a layered model approach. The layers, illustrated in Figure 1, present a process for developing an appropriate simulation software model to handle discrete events starting from a conceptual model of the simuland. In addition, several layers include a mapping to a software implementation, a topic not covered in this paper. The roadmap layers include:

- Simuland Model Layer – This layer represents the simuland conceptual model expressed in a recognized modeling paradigm.

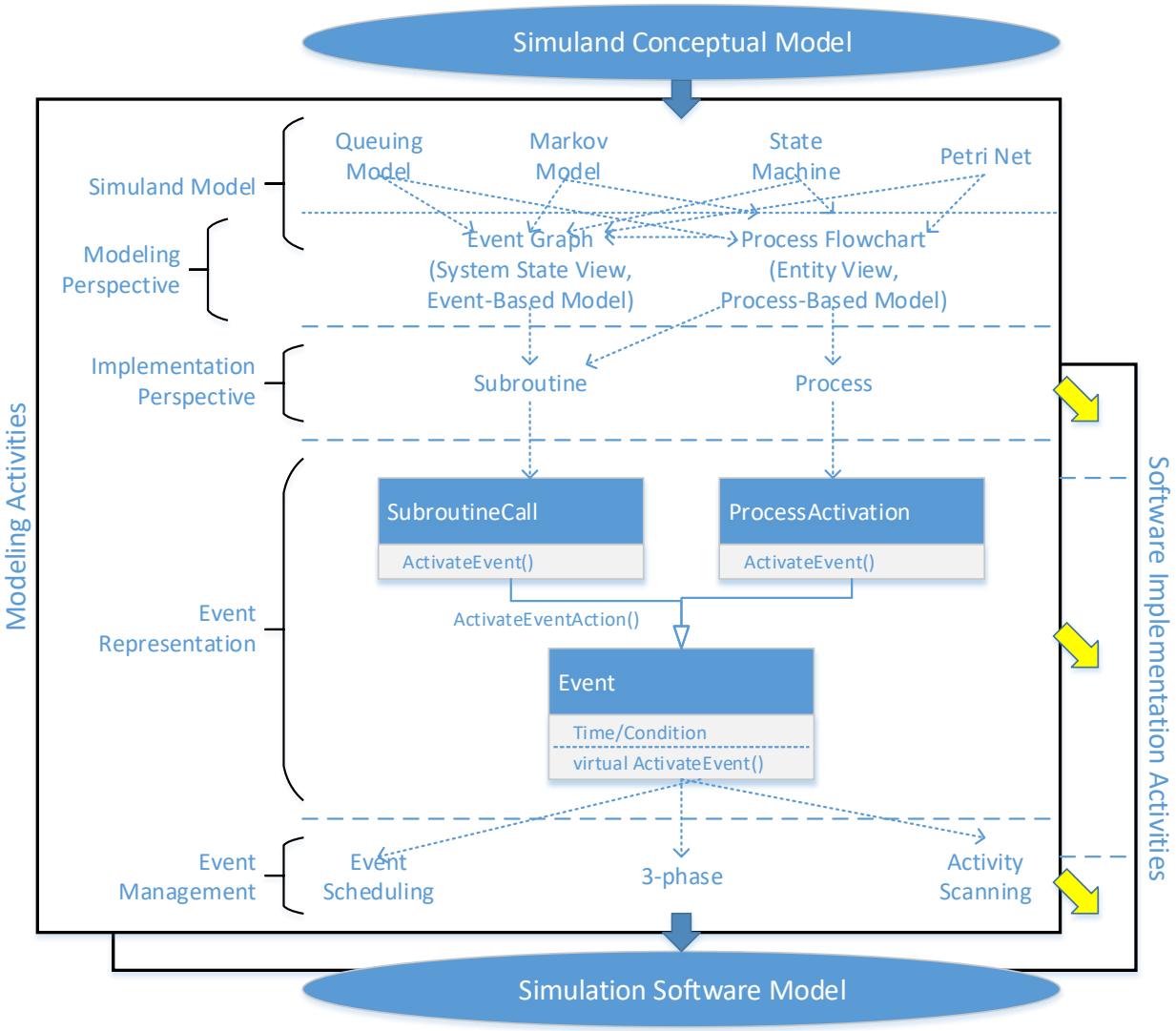


Figure 1: Discrete event simulation content roadmap.

- Model Perspective Layer – This layer identifies the intent of the simulation model, but in a manner that easily identifies events. This approach normally involves defining the model perspective, a system state perspective or an entity process perspective.
- Implementation Perspective Layer – This layer identifies the software approach for implementing the selected model perspective. A subroutine approach normally is used to implement a system state perspective while a process flow approach (threads or coroutines) normally is used to implement the entity process perspective.
- Event Representation Layer – This layer encapsulates the conditions for event or process execution and the logic associated with event or process implementation. Execution of an event is different under the two implementation perspectives, so this layer also presents an abstraction leading to a common event execution representation.
- Event Management Layer – This layer selects the core logic for time management as well as the selection and execution of events. This layer supports various strategies, but all are defined to work with the abstraction of an event present in the event representation layer.

The roadmap guides the development process and provides a consistent view of this process for all alternatives of simuland modeling paradigm, model perspective, or worldview.

We attempted to deviate slightly from the present interpretation of worldviews in the roadmap. Originally, the literature treated worldviews at the model perspective layer. Differentiation was made between a system state perspective and an entity process perspective. However, over time worldviews came to encompass event management strategies including event scheduling, activity scanning, and three-phase scheduling. In the roadmap, the event management strategies are present in the event management layer. This separation of modeling perspective choices and event management choices seemed more appropriate to the development of an educational approach.

The roadmap presented in Figure 1 shows students the steps taken and the design alternatives available in transforming a simulation conceptual model into a simulation software model. It also allows those steps to be mapped to a software architecture. In the remainder of this section, the roadmap layers and their relationships are discussed in more detail.

### 3.2.1 Simuland Model Layer

The simuland model layer captures the simuland conceptual model in a modeling paradigm appropriate to the simuland domain and the questions which are to be addressed by the simulation study. The model can utilize any of the available modeling paradigms, to include, but not exclusive to, those shown in Figure 1. It should be noted that the event graph and process flow models also are valid models for the simuland, but are presented in a separate layer because they are the models used to identify modeling perspective.

### 3.2.2 Modeling Perspective Layer

The modeling perspective layer transforms a model of the simuland to one that better identifies the perspective to be utilized. The roadmap presently supports two perspectives, an event-based model to represent the system state perspective and a process-based model to represent the entity process perspective. These models capture the perspectives provided by the event scheduling worldview and the process interaction worldview. Event graphs (Schruben 1995; Buss 1996) are utilized to represent the event-based modeling perspective, while process flowcharts are utilized to represent the process-based modeling perspective. An alternate representation for the process-based perspective is the control flow graph (Sargent 2004), but this representation was considered less intuitive for a first encounter. An event graph example for modeling a flexible manufacturing system is found in Sargent (1988) and a process flowchart example for modeling a seaport system is found in Wanke (2011).

To illustrate the different representations, a simple M/M/n queuing model is considered where Q is the number of entities in the queue, I is the number of idle servers, $t_{IA}$ is the interarrival time process, and $t_S$ is the service time process. Figure 2 demonstrates the representation of an M/M/n queuing model as an event graph. Each node in the graph represents an event; state changes associated with the event are identified directly below the node. Edges represent the scheduling of a future event; the time increment for scheduling the future event is labeled on the edge and event conditions are labeled in parentheses on the edge. Figure 3 demonstrates the representation of the queuing system as a process flowchart. White boxes denote logic associated with the execution of an event. The flowchart component subsets corresponding to events in the event graph are circled and labeled in red and lead to a method splitting implementation of the process flow (Jacobs and Verbraeck 2004). Gray boxes denote the passage of time, either waiting for a time delay or for a condition to be met. Each gray box presents the opportunity for a context switch where a new process can be activated based on scheduling or a condition being met. Dotted lines indicate one process triggering a condition associated with another process.

Stepping from a simuland model to a model representing perspective requires formal mappings from each modeling paradigm to the appropriate event model. Little work has been done in this area and is a

topic of future research. A method for transforming Petri Nets into event graphs is found in (Schruben and Yucesan 1994).

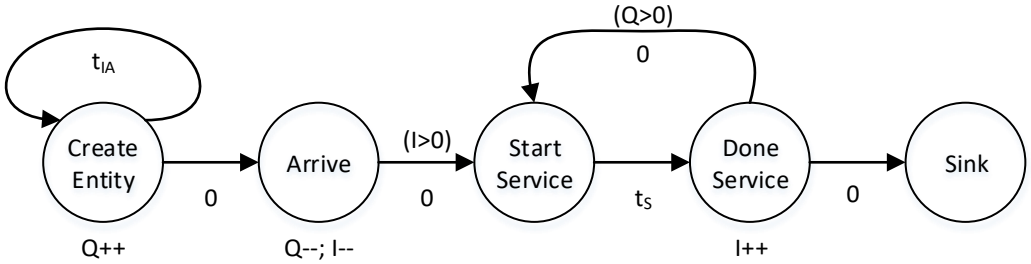


Figure 2: Event graph representation of M/M/n queuing system.

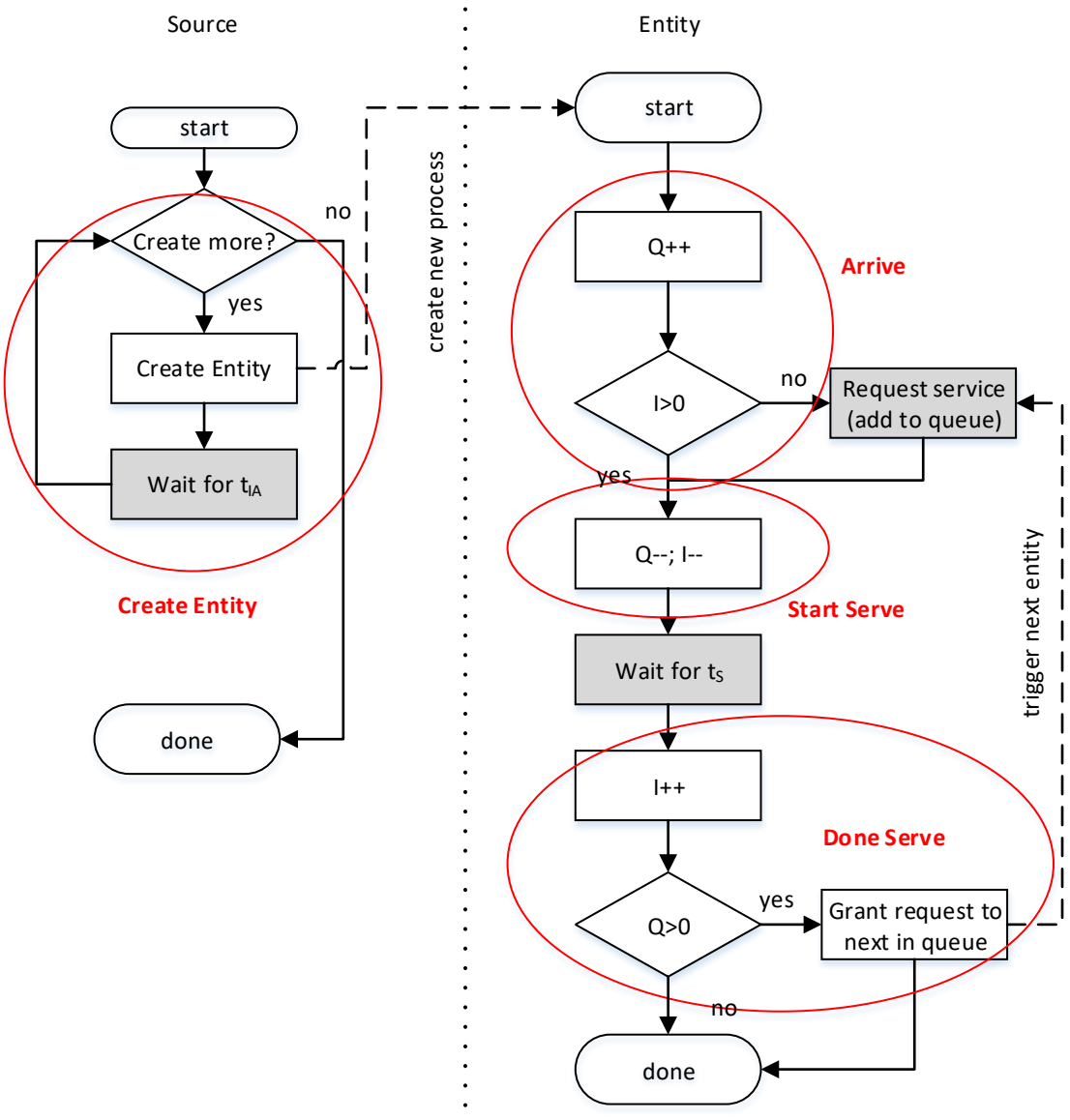


Figure 3: Process flowchart representation of M/M/n queuing system.

### 3.2.3 Implementation Perspective Layer

Once a model is developed for the modeling perspective layer, an appropriate implementation perspective is presented. A separate approach is presented for the different modeling perspectives. For an event-based model, subroutines efficiently capture the logic necessary to execute system state changes. A simulation executive implementing event management calls the subroutine to execute the state changes and control is returned to the executive on completion of the subroutine. Note that there is an accepted mapping from a process-based model to a subroutine implementation through the technique of method splitting (Jacobs and Verbraeck 200).

Alternatively, processes are commonly implemented by coroutines or threads, with multiple events being captured within a single process implemented within a coroutine or thread. Calls are made from the process to the simulation executive to indicate the passage of time, allowing the simulation executive to activate a new process for execution through a context switch. Coroutines are acknowledged as more efficient than multithreading for implementing process interaction (Weatherly and Page 2004; Xu and Li 2012) since discrete event simulations only utilize one thread of execution at a time. However, to reduce reliance on external libraries that often are required to implement coroutines, whichever approach is directly supported by the implementation language is utilized. At this point in the educational process, achieving high performance is not of primary concern. It is discussed later that this level of implementation is hidden from the students as it is beyond their current educational experience level.

### 3.2.4 Event Representation Layer

The event representation layer is important in that this layer abstracts away the implementation details of the implementation perspective, thus presenting a single representation for the event management layer. An event is classified as either a scheduled event, executed based on an application-supplied schedule time, or a conditional event, executed when an application-supplied condition is met. Then an event can be defined by the tuple {schedule time/condition, event action}. An event is represented as the virtual class Event, holding the information for event execution and a virtual method (ActivateEvent()) providing access to either call or activate the event action.

Implementations of the virtual class Event are defined for both implementation perspectives. For subroutine-based models, a *command pattern* (Gamma et. al. 1995) approach is presented for scheduled execution of subroutine-based events and implemented as the SubroutineCall class. The implementation of the virtual method ActivateEvent() provides the functionality to execute the proper subroutine. The ProcessActivation class implements the Event class for process-based models. The implementation of the virtual method ActivateEvent() provides the functionality to activate the proper thread of execution. This class relationship is shown by the UML class diagram in Figure 1. Students are exposed to the implementation of the SubroutineCall class, but the specifics of the ProcessActivation class are hidden from the students as currently they are not taught coroutine-based or multithreaded programming.

It should be noted that the current relationship between the event representation and the modeling perspective has a flaw. The selection of the event graph for representing event-based modeling does not provide a means to represent conditional events. In Figure 2, the Start Service event is conditionally scheduled by other events. It could alternatively be defined as a conditional event, execution of which is not based on being scheduled by other events, but rather on a condition predicated on state variables being met, in this case ((I > 0) and (Q > 0)). A proposed modification to event graphs is being explored to address this shortfall.

### 3.2.5 Event Management Layer

The abstraction presented in the event representation layer means that the event management layer is not affected by the implementation perspective that is chosen. It does not matter if the execution of an event

is subroutine or thread based; in either case an event management strategy must be present to select the next event to execute/activate and to manage time.

In the roadmap, process interaction is not considered within the event management layer. The worldviews event scheduling, activity scanning, and three-phase each identify specific event management approaches and all are presented to students. The only event classification required is to note the difference between scheduled and conditional events as they are managed differently.

The interface to the event management layer is the only evidence of a difference between the implementation perspectives. The activation of events differs for subroutine and thread-based implementations, and the related scheduling/context switching also differs, requiring a different interface for each. However, this can be captured at the event representation layer and, once represented as an event, there is no difference in implementation. In fact, it has been shown that a hybrid simulation can be created where both implementation perspectives are present in a single simulation.

### 3.3 Simulation Architecture

It is desirable in teaching simulation software design and development to present a software architecture to delineate functionality clearly. A basic architecture that separates the application from a reusable simulation executive (Pidd 2004) is utilized. The simulation executive includes the event management layer and the abstraction of the event representation layer. All of these components are independent of the application. By providing an interface to support both subroutines and thread-based event representations, the simulation executive can support the full spectrum of the event models.

The remainder of roadmap components are considered as being part of the application. The application developer constructs subroutine representations for each event in an event graph, or constructs a process thread representation for each process in a process flowchart. In the case of process interaction, a further architectural level is provided to implement the ThreadEventAction. This allows the specifics of handling threads/coroutines to be hidden from the student because these topics presently are beyond the scope of an undergraduate M&S engineering curriculum. Students are provided an interface similar to JAPROSIM (Belattar and Bourouis 2013).

### 3.4 Educational Coverage

It is not practical to provide complete coverage of the simulation software implementation roadmap in Figure 1 in an undergraduate curriculum. Therefore compromises must be made, identifying where a student is expected to develop an understanding of concepts and where a student is expected to develop a level of mastery. Currently, the coverage at each roadmap layer is:

- Simuland Model – fairly complete coverage is provided.
- Modeling Perspective – complete coverage is provided, but only a subset of the mappings from a simuland model to the modeling perspective is presented.
- Implementation Perspective – complete coverage is provided. Students experience developing pseudocode from both perspectives.
- Event Representation – the SubroutineCall representation is covered in detail and students experience significant implementations of this representation. The ProcessActivation representation is presented in concept, but implementation requires significant background in either coroutines or multithreading. Therefore students are provided library support to execute implementations of their process-based implementation perspective.
- Event Management - Event scheduling is covered in detail and students develop a complete simulation executive to support their SubroutineCall based exercises. Three-phase and activity scanning are covered as concepts; on occasion, three-phase is included as an exercise.

## 4 CURRICULAR IMPLEMENTATION

In this section, we discuss the integration of the DES simulation roadmap into our undergraduate program in Modeling and Simulation Engineering. The undergraduate curriculum is organized around a set of core courses required of all students. The subset of core courses that address aspects of discrete event simulation are shown in Figure 4.a. Introduction to M&S Engineering is the first required core course. It is designed to introduce students to the department and to the engineering profession. In addition, it provides an introduction to M&S terminology and concepts, describes system representation and classification, and defines the primary simulation paradigms. Students are then asked to develop simple spreadsheet simulations representative of Monte Carlo simulation, continuous simulation, and discrete event simulation. The Discrete Event Simulation course started out as a very standard DES simulation course and was taught from (Banks et al. 2010). An accompanying laboratory introduced the Arena and Simio simulation packages and covered a set of increasingly more complex case studies. The Simulation Software Design course was, and continues to be, taught from faculty course notes because we were unable to identify an appropriate textbook. Analysis for M&SE, Engineering Models, and Computer Graphics & Visualization courses provide the theoretical underpinnings for aspects of M&S that the students have already experienced in their earlier simulation courses. And finally, the M&SE Capstone Design courses require the students, working as a project team, to utilize their core knowledge and skills to conduct a substantial M&S project for a local company or organization. The project results in the development of a detailed engineering design and implementation of a prototype system.

Integration of the DES simulation roadmap into our curriculum has resulted so far in changes primarily to the Discrete Event Simulation course and the Simulation Software Design course. The motivation for making changes occurred as a result of our initial experience in these two courses. It was observed that students exiting the Discrete Event Simulation course did not possess the background knowledge desired for the Simulation Software Design course. They did not appreciate the division of DES simulation software into an application component and a simulation executive component, and the required dialog that must occur between these two components. They did not understand the design alternatives presented by the event scheduling perspective and the process interaction perspective. Similarly, their knowledge of event management strategies was limited to event scheduling; they had not been introduced to the activity scanning strategy or the three-phase event scheduling strategy. These limitations made it difficult to adequately cover the knowledge and skills required to design and implement simulation software in a single course.

A course content outline for the Simulation Software Design course is shown in Figure 4.b. Content components shown in blue indicate where changes or additions in content have been made to address the simulation roadmap. It is apparent that significant changes, mostly content additions, were made to this course. These additions were possible because of the additions to the content in the preceding course, Discrete Event Simulation. In addition, students enter this course previously having completed two programming courses and a course on object oriented programming and design from the Computer Science Department. Added content includes an introduction to the process interaction worldview and an associated discussion concerning potential software implementation strategies. Also added is content on event representation, centered around the event graph representation, and event management strategies.

A course content outline for the Discrete Event Simulation course also is shown in Figure 4.c. Enhancements to support the simulation roadmap again are shown in blue. Two enhancements are most significant. First, an entirely new course content module, called More General DES Representations, has been added. Additional system models, state automata and Petri Nets, are introduced. Then, event graphs and process flowcharts are introduced as approaches to event representation. Finally, additional event management strategies, activity scanning and three-phase event scheduling, are introduced. Second, a brief section has been added to explain that the software architecture of simulation tools typically consists of an application component and a simulation executive component.

**DES M&S CURRICULUM THREAD**
- Introduction to M&S Engineering
- Discrete Event Simulation & Lab
- Simulation Software Design & Lab
- Analysis for M&SE
- Engineering Models
- Computer Graphics & Visualization
- M&SE Capstone Design I & II

a) DES M&S core course list.

**SIMULATION SOFTWARE DESIGN**
- Structure of DES Simulations
  - World Views
    - Event Scheduling
    - Process Interaction
    - Software Concepts
  - Simulation Executive
    - Time Management
    - Event Representation
    - Event Management
- Simulation Architecture Concepts
  - Application/Simulation Executive
  - GUI/Simulation
- Simulation Tool Development (Lab)
- Object Oriented Simulation Design
- Data Structures
- Algorithms
- Software Development Practices

b) Simulation software design course.

**DISCRETE EVENT SIMULATION**
- DES Modeling and Simulation
  - Simple Queuing Systems
  - Event Scheduling
  - Spreadsheet Simulation
- DES Simulation Tools
  - Organization/User Interface
  - Application/Simulation Executive
  - Arena and Simio (Lab)
- Simulation Methodology
  - Random Number/Variate Generation
  - Input Data Modeling
  - Output Data Analysis
- More General DES Representations
  - System Models
    - Queuing Networks
    - State Automata
    - Petri Nets
  - Event Representation
    - Event Graphs
    - Process Flow Graphs
  - Event Management
    - Event Scheduling
    - Activity Scanning
    - Three Phase Scheduling
- Conducting a Simulation Study
- Case Studies (Lab)

c) Discrete event simulation course.

Figure 4: Modifications to M&S engineering curriculum.

The changes described here have only recently been implemented. Thus we still do not have all desired assessment data to determine the overall impact that these enhancements have had on our students. However, based on individual course assessment data, it appears that students have responded very favorably. Surprisingly, one of the biggest benefits of this work to our students seems to be the visualization of the simulation roadmap. It serves as a mental model that helps students organize their understanding of the role of each of the roadmap components and that identifies the design and development alternatives as one proceeds from conceptual simulation model to software implementation. Another is the introduction of event graphs as a visual representation of events, their structure, and their scheduling properties. Students more clearly see the relationship between the abstract event concepts inherent in a simple queuing system and the underlying scheduling and execution of events. A tool was created to visualize the execution and scheduling of events in the event scheduling approach, which directly connects to the student's simulation software (Collins et. al. 2017).

## 5 CONCLUSION

This paper presents a new roadmap for the instruction of DES simulation at the undergraduate level, including software design, with the intent to provide a template for course/curriculum development. The main contribution of this roadmap is to explicitly describe the hierarchical structure and inter-relationships characterizing the worldviews of DES simulation, in particular event scheduling and process interaction. Simple queuing model examples serve to illustrate the event scheduling perspective and the process interaction perspective.

There are several potential next steps for our educational roadmap, which we summarize as: improvement, deployment, and generalization. Firstly, the roadmap makes several necessary connections that are not supported by any of the existing literature. For example, how to create the links from the simuland model level to the implementation perspective level is not well-defined for all possible pairs. Secondly, the authors have initiated the deployment of the actual roadmap in the curriculum and will be able to draw lessons learn from it. Thirdly, this roadmap highlights the need for a general discussion on M&S education in our modern world.

## REFERENCES


Banks, J., J. Carson, B. Nelson, and D. Nicol. 2010. *Discrete-Event System Simulation*. Upper Saddle River, New Jersy: Pearson Education, Inc..

Belattar, B. and A. Bourouis. 2013. "A Java Based Discrete Event Simulation Library". *International Journal of Computer, Electrical, Automation, Conrol and Information Engineering*. 7(2):278-282.

Balci, O. 1988. "The Implementation of Four Conceptual Frameworks for Simulation Modeling in High-Level Languages". In *Proceedings of the 1988 Winter Simulation Conference*, edited by M. Abrams, P. Haigh, and J. Comfort, 287-295. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Buss, A. 1996. "Modeling with Event Graphs". In *Proceedings of the 1996 Winter Simulation Conference*, edited by J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain, 153-160. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Collins, S., L. Dumaliang, N. Gonda, J. Leathrum, and R. Mielke. 2017. "Visualization of Event Execution in a Discrete Event System". In *Annual Simulation Symposium (ANSS'17), Spring Simulation Multi-Conference 2017*.

Gamma, E., R. Helm, R. Johnson and J. Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, Massachusetts: Addison-Wesley Longman Publishing Co., Inc.

Jacobs, P. and A. Verbraeck. 2004. "Single-Threaded Specification of Process-Interaction Formalism in Java". In *Proceedings of the 2004 Winter Simulation Conference*, edited by R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 1548-1555. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Leathrum, J. and R. Mielke. 2012. "Outcome-Based Curriculum Development for an Undergraduate M&S Program". In *AutumnSim 2012, Conference on Education and Training Modeling and Simulation (ETMS'12)*, 28-31.

Mielke, R., J. Leathrum, and F. McKenzie. 2011. "A Model for University-Level Education Modeling and Simulation". *M&S Journal*, 6(3):14-23.

Pegden, C. D. 2010. "Advanced Tutorial: Overview of Simulation World Views". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yucesan, 210-215. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Pidd, M. 2004. "Simulation Worldviews – So What?". In *Proceedings of the 2004 Winter Simulation Conference*, edited by R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 288-292. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rashidi, H. 2016. "Discrete Simulation Software: a Survey on Taxonomies". *Journal of Simulation*, 1-11.

Sargent, R. 1988. “Event Graph Modelling for Simulation with an Application to Flexible Manufacturing Systems”. *Management Science*. 34(10):1231-1251.

Sargent, R. 2004. “Some Recent Advances in the Process World View”. In *Proceedings of the 2004 Winter Simulation Conference*, edited by R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 293-299. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Schriber, T., D. Brunner, and J. Smith. (2014). “Inside Discrete-Event Simulation Software: How It Works and Why It Matters”. In *Proceedings of the 2014 Winter Simulation Conference*, edited by A. Tolk, S. Diallo, I. Ryzhov, L. Yilmaz, S. Buckley, and J. Miller, 132-146. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Schruben, L. and E. Yucesan. 1994. “Transforming Petri Nets into Event Graph Models”. In *Proceedings of the 1994 Winter Simulation Conference*, edited by J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, 560-565. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Schruben, L.. 1995. “Building Reusable Simulators using Hierarchical Event Graphs”. In *Proceedings of the 1995 Winter Simulation Conference*, edited by C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman, 472-475. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Wanke, P. 2011. “Ship-Berth Link and Demurrage Costs: Evaluating Different Allocation Policies and Queue Priorities via Simulation”. *Pesquisa Operacional*. 31(1):113-134.

Weatherly, R., and E. Page. 2004. “Efficient Process Interaction Simulation in Java: Implementing Co-Routines within a Single Java Thread”. In *Proceedings of the 2004 Winter Simulation Conference*, edited by R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 1437-1443. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Xu, X. and G. Li. 2012. “Research on Coroutine-Based Process Interaction Simulation Mechanism in C++”. In *AsiaSim 2012, Part III, CCIS 325*, edited by Xiao, T., L. Zhang, and M. Fei. 178-187. Spinger-Verlag.

## AUTHOR BIOGRAPHIES

**JAMES F. LEATHRUM, JR.** is an Associate Professor of the Department of Modeling, Simulation and Visualization Engineering at Old Dominion University. He holds a Ph.D. in electrical engineering from Duke University. His research interests include discrete-event simulation, distributed simulation, simulation architectures, and their applications. His e-mail address is jleathru@gmail.com.

**ROLAND R. MIELKE** is a University Professor in the Department of Modeling, Simulation and Visualization Engineering at Old Dominion University. He earned the Ph.D. in Electrical and Computer Engineering for the University of Wisconsin-Madison. His research interests include system theory, the theory and application of simulation, and simulation education. His e-mail address is rmielke@odu.edu.

**ANDREW COLLINS** is a Research Assistant Professor at Old Dominion University’s Virginia, Modeling, Analysis, and Simulation Center (VMASC). He has a Ph.D. in Operations Research from the University of Southampton. His research focus is agent-based simulations. His contact details are acollins@odu.edu and www.drandrewjcollins.com

**MICHEL AUDETTE** is an Associate Professor of the Department of Modeling, Simulation and Visualization Engineering at Old Dominion University. He holds a Ph.D. in Biomedical Engineering from McGill University. His research interests include medical simulation, featuring physiological state-based virtual-reality simulation, surgery planning and medical image analysis. His e-mail address is maudette@odu.edu.