Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

FORTY YEARS OF EVENT GRAPHS IN RESEARCH AND EDUCATION

Murat M. Gunal

Faculty of Engineering and Architecture
Fenerbahce University
Atatürk Mah. Ataşehir Bulvarı
34758, Ataşehir, Istanbul, TÜRKIYE

Yahya Ismail Osais

Computer Engineering Department
Center for Intelligent Secure Systems
King Fahd University of Petroleum and Minerals
P.O. Box 2244
Dhahran, 31261, SAUDI ARABIA

Lee Schruben

Industrial Engineering and Operations Research
University of California, Berkeley
Berkeley, CA, 94720 USA

Gerd Wagner

Department of Informatics
Brandenburg University of Technology
Konrad-Wachsmann-Allee 5
Cottbus, 03046, GERMANY

Enver Yücesan

Technology and Operations Management Area
INSEAD
Ayer Rajah Avenue
138676 SINGAPORE

ABSTRACT

Forty years ago, in 1983, Lee Schruben proposed the Event Graph formalism and modeling language, subsequently defining the paradigm of Event-Based Simulation, in a precise way, which had been pioneered 20 years before by SIMSCRIPT. The purpose of this panel is for a group of Event Graph researchers both from Operations Research and Computer Science, including the inventor of Event Graphs and one of his former PhD students who has made essential contributions to their theory, to discuss their views on the history and potential of Event Graph modeling and simulation. In particular, the adoption of Event Graphs as a discrete process modeling language in Discrete Event Simulation and in Computer Science, and their potential as a foundation for the entire field of Discrete Event Simulation and for the fields of process modeling and AI in Computer Science is debated.

1 INTRODUCTION

The world is moving towards interdisciplinary research and innovation. For the simulation community, it is the right time for them to examine their ideas that lie at the intersections with other disciplines. One such idea is the modeling language of Event Graphs, which connects the discipline of Discrete Event Simulation with the disciplines of Information Systems and Computer Science.

As we have been working on this paper, a story has unfolded itself. This story is about dedication and sacrifice to nurture an idea and push it forward. Event Graphs are not meant for building discrete event

models only. Event Graphs have a higher purpose. Hence, in this paper, the goal of the panel discussion is to reflect on the history of Event Graphs and link it with some important ideas in other fields like process modeling, computing, and artificial intelligence. The importance of reviving Event Graphs will become evident as you read through the paper. This panel discussion will chart an unexplored territory that will hopefully spark new ideas for research.

Five leading researchers, including the inventor of Event Graphs, will give their points of view on multiple issues concerning Event Graphs: Lee Schruben (University of California, Berkeley), Enver Yucesan (INSEAD), Murat Gunal (Fenerbahce University), Yahya Osais (KFUPM), and Gerd Wagner (Brandenburg University of Technology). Each was asked to answer six questions. Next, their answers and perspectives are presented.

2 QUESTION 1

Despite the facts that (1) Event Graphs define the paradigm of event-based simulation (with "event scheduling"), which is considered as the foundation of Discrete Event Simulation (DES), e.g., by Denis Pegden, and (2) diagrams help to teach ideas/concepts, they have not been used much in simulation textbooks. Also other diagram languages, such as Activity Cycle Diagrams, have not been used much.

Why have Event Graphs not been used more widely in DES textbooks for introducing event-based simulation? Is this due to the fact that most DES textbooks have been authored by people from Industrial Engineering, Operations Research and Management Science (IE/OR/MS) who tend not to appreciate the value of visual simulation models?

Enver: Given the fact that most engineering curricula offer a single course on system simulation, instructors of these courses always face a dilemma in teaching simulation modeling and analysis: should they base the course on a generic programming language (often reinforced by a few simulation-specific libraries) or should they anchor the course on a commercially available simulation language?

The former approach provides a more flexible framework as it exposes students to the fundamentals of modeling through a diverse set of modeling perspectives. In contrast, the latter approach enables the students to immediately experience the "magic of simulation" by rapidly building relatively sophisticated models, bringing them to life with animation, and displaying the performance indicators of interest through sophisticated graphics. I therefore associate the limited coverage of Event Graphs in DES textbooks not with the background or the preferences of the authors, but with their effort to make simulation accessible to a broader audience in industry.

Traditionally, Event Graphs were part of the former approach: a fully general graphical framework to describe the event-scheduling perspective for DES modeling. Some widely used simulation textbooks dedicate several sections to Event Graphs in their discussion on modeling (e.g., in the book of Averill Law).

Murat: In Computer Science (CS), a model and a simulation are seen as separate, but in IE/OR/MS, a model is a part of simulation process and cannot be seen as separate. Besides CS, in other disciplines such as engineering and social sciences, simulation is taught in many courses, but their perception of simulation resembles the one of CS, that is there is a model to mimic the system under consideration and there is a simulation which executes the model.

To create a representation of a system, authors (and teachers) tend to use their domain knowledge and prefer non-event-oriented representation methods, such as mathematical formulations. Even if there is a time dimension in a system, they tend to ignore events but use time as a simple index in equations. Visual methods, including Event Graphs (EGs), are therefore not preferred because a model does not mean anything visual but rather means a set of equations which has "regular" time steps.

We teach how a system is abstracted and represented on a computer, and as a first step we deal with conceptual modelling. Robinson et al. (2010) review this area thoroughly and present conceptual model representations including EGs.

I agree with the statement that many DES textbooks authored by OR people do not appreciate EGs, but disagree that they ignore visual models. Some authors of DES textbooks are in fact using some sort of visualization for model representation, but these methods are not formal, like EGs, and sometimes they are inventing their own diagramming techniques, as, e.g., in (Elizandro and Taha 2007) or using common-sense shapes to picturize entity flows in a system.

Gerd: Enver has argued that EGs may not have been used in some textbooks for making the topic of simulation accessible to a broader audience. But since almost all books not covering EGs still discuss event-based simulation (with event scheduling), using EGs would help them to teach the basic DES concepts of events and causation, as well as the mechanics/meaning of event-based simulation.

I agree with Murat that in IE/OR/MS, it has not yet been widely understood that there are several types of models in simulation: (1) executable simulation models expressed in code, (2) implementation-agnostic simulation design models expressed with modeling languages like EGs, and (3) conceptional models of the problem domain expressed in English and with the help of informal diagrams (such as flowcharts).

I also understand that OR authors, as trained mathematicians, may prefer classical mathematical representations instead of diagram languages such as EGs. But I doubt that this is a didactically good choice.

Enver: Murat's broader perspective on modeling triggered a few other observations regarding on continuous versus discrete-event systems. For the former, we have a well-established universal modeling framework: difference or differential equations. To support such a modeling exercise, we also have modeling and simulation languages like Vensim. For DES, such a universal modeling framework does not exist. This is where EGs, along with the EG development and execution environment SIGMA (Schruben 1995a, Schruben and Schruben 2006), can have the most significant contribution: providing a simple and elegant, yet powerful, modeling framework for DES.

Yahya: Computer scientists and engineers have always felt the need for different books on simulation. This need is very clear in the area of computer networks. The most popular simulation tools for computer networks are NS-3 [https://www.nsnam.org] and Omnet++ [https://omnetpp.org]. Both are discrete-event simulators.

These two tools are not built on top of solid abstractions. They only provide hooks for developers to extend the built-in models or to write new models from scratch. Both scenarios are a nightmare. As a result, researchers in the area of Computer Systems and Networks (CSNs) decided to develop their own abstractions and tools for simulation, including the use of (possibly concurrent) components for achieving modularity. Remarkably, Lee predicted the importance of modularity in EGs in (Schruben 1995b), followed by a proposal by Buss and Blais (2007). Unfortunately, these efforts have not been picked up by researchers in the computing field. There are not many books on DES authored by researchers from the computing field, and EGs are typically not covered in these books. An exception is (Osais 2017), which models CSNs as queuing systems in the form of EGs.

3 QUESTION 2

Like Gantt and PERT/GERT charts, Event Graphs have been developed in the fields of IE/OR/MS, while Petri Nets have been developed in the field of Computer Science. Unlike Petri Nets, Event Graphs have received little attention in computer science, information systems, and business process modeling, despite the fact that they would have been more natural and more expressive for discrete process modeling.

Why have Event Graphs, instead of Petri Nets, not become more widely used in Computer Science as a discrete process modeling language?

Gerd: To me, the popularity of Petri Nets (PNs) and the disregard of EGs in Computer Science is hard to understand. Both of them are similar in that they fire transitions/events that may change their state. But EGs

support the normal Computer Science concept of a system state (in terms of the values of state variables) while the concept of a PN marking as a state is very peculiar and limited.

While it has been shown in (Schruben and Yucesan 1994) that there is a simple mapping of Stochastic PNs to EGs, where places become conditional edges, a mapping the other way around is not known due to the difficulty to map an EG state to a PN marking.

One possible reason for the popularity of PNs in Computer Science is the fact that checking formal properties, such as reachability and deadlocks, is easier for PNs due to their simple state concept. I'm not aware of any efforts to define a formal verification approach for EGs, but I think this can be done, even if it may be more difficult since the states of EGs are more expressive.

Enver: Even though Schruben and Yücesan (1994) proposed a mapping between PNs and EGs, the former modeling formalism maintained its popularity in Computer Science while the latter modeling formalism carved a niche for itself in the fields of IE/OR/MS. Although breakthroughs typically come from "technology transfers" across disciplines (e.g., using principles from physics and chemistry to model the evolution of stock prices or fluid approximation to highly congested queueing systems), PNs and EGs have evolved along parallel paths, which is quite common in our siloed technical education. I should also underline the facts that PNs have a much longer history than EGs and that the Computer Science community is much larger than the DES community dispersed across IE/OR/MS communities.

This lack of cross fertilization is unfortunately a handicap for students of discrete dynamic systems as either formalisms has its strengths and drawbacks. For instance, while it is easier to conduct formal verification in PNs (e.g., reachability or deadlock detection), the explicit representation of the system state through places, edges, tokens and transitions results in rapidly growing bipartite graphs with the increase in complexity of the system state. Within the discrete-event simulation perspective, however, Jacobson and Yücesan (1999) show that checking such structural properties (e.g., reachability) is an NP-complete problem.

Perhaps it is time to bring together all the work associated with EGs from many diverse fields in a book that is as rigorous and as complete as the book by Peter Haas on PNs. The two books would then provide us with a solid starting point for patching the weaknesses of one formalism with the innovative solution adopted in the other.

Yahya: EGs were used for modeling sequential computation. Also, EGs were extended for the purpose of modeling parallel and distributed computation in the domain of DES. However, to the best of my knowledge, there have been no attempts to extend or use EGs for the purpose of formal verification. One reason could be the lack of constructs in EGs for expressing concurrency. Further, while the concept of a system state in PNs is captured by the notion of a marking by tokens, a system state in EGs is not as easy to define. Hence, the need for tools to do formal verification was the main driving force for the popularity of PNs in Computer Science.

Gerd: Not only PNs, but also EGs support concurrency by allowing parallel branching. In PNs, parallel branching is modeled by a transition having two or more output places. In EGs, parallel branching is modeled by an event circle having two or more (unconditional) outgoing arrows to follow-up event circles. Therefore, EGs could have been used in Theoretical Computer Science for modeling concurrency, but, unfortunately, they have been ignored.

Yahya: I agree on the fact that parallel events (branching) can be modeled using EGs. And, I agree that this capability is essential to modeling concurrency with EGs.

Lee and Enver (1994) used restricted Stochastic PNs (without concurrency) in their mapping to EGs. Unless it is shown how a general Stochastic PN can be mapped to an EG, I will not feel comfortable to say "EGs can model concurrency in a natural way". I still feel that additional constructs need to be added to EGs to facilitate the modeling of concurrency.

Gerd: I don't think that the fact that Lee and Enver have restricted their mapping to PNs without concurrency has any implications on EGs supporting concurrency or not. In fact, I think that there are also other meaningful mappings from certain classes of PNs to EGs, in addition to that of Lee and Enver.

Murat: A use case for concurrency is the parallel treatment behavior of physicians in emergency rooms, which I discovered during my PhD research. In emergency rooms, physicians perform multitasking, so it is not possible to model their processes "sequentially". Using DES software, we can only approximately model this behavior. However, if we can find ways to model concurrency using EGs, we will be able to realistically model more real processes. I am sure that there are many use cases for such concurrent processes in industry.

Coming back to the reasons why PNs have become popular in CS, but EGs not, I would like to point out a huge software sector, Enterprise Resource Planning (ERP) software, and its overlap with PN. The market leader in ERP is SAP and we see many SAP implementers using PN along with BPMN and Event-driven Process Chains. When modelling industrial and business processes and implementing them in ERP software, PN is a preferred specification and EG is not mentioned in the modelling. This is sad to me as a proponent of EGs, but I believe further research on EGs has potential for broader use.

Gerd: The fact that PNs have been chosen as the preferred semantics of Business Process models in the Business Process Management community is just an unfortunate historic development. EGs would have been a much more natural choice since, unlike PNs, they are based on the same ontological foundations as Business Process modeling: events, state changes and causal regularities.

4 QUESTION 3

In early discrete process modeling languages, such as Gantt and PERT/GERT charts, as well as in early simulation languages, such as Tocher's GSP and Gordon's GPSS, the concept of activities has played a prominent role. An activity can be decomposed into an activity start event and an activity end event, as in Tocher's Three-Phase approach.

Question to Lee and Enver: *At the time when you worked on Event Graphs, have there been any considerations to extend them by adding a concept of activities?*

Enver: When I had the great opportunity to work with Lee at Cornell in 1985, he had already introduced the innovative framework of EGs. At that point, he proposed further exploration of EGs as a potential dissertation topic for me, which resulted in four great years at graduate school under his supervision and mentoring.

During that exploration, we carefully studied a number of modeling paradigms. The big (pedagogical) debate within the simulation community at the time was whether to adopt a particular simulation language in a simulation course or opt for a concise modeling formalism (and ask the students to carry out the model implementation in a general-purpose programming language). In evaluating the pros and cons of each approach, we were inspired by Zeigler's DEVS formalism as well as Overstreet and Nance's Canonical Methodology, which facilitated the translation of a model specification into a model implementation.

When we focused on simulation-language-independent graphical representations (i.e., steering clear from GPSS or SLAM), we discovered an extremely rich literature in simulation modeling. As early as 1963, Tocher was proposing wheel charts. Evans et al. (1967) were depicting the interactions among event occurrences by drawing schematic chains of occurrences. Hutchinson (1981) was using activity life cycle diagrams. Törn (1981) was specializing PNs to simulation modeling by incorporating such extensions as inhibitor arcs and queues. Ultimately, we arrived at the conclusion that EGs provided the most parsimonious representation of the event-scheduling world view with an immediate connection between model specification and model implementation. We documented this exploration in a 1993 ORL paper.

Lee: I will try to make the case that Event Graphs are a natural way to model activities. Without always formally recognizing it, activity modeling has long been my preferred approach to describing the dynamics of a discrete event system. Activities are also easier for me to communicate to other people than are isolated, sometimes abstract, events or complex entity process flow interactions. I will give some historical background, as old men do, that might help my case.

My first Event Graph resulted from me trying to organize the notes I was taking during interviews with some of Cornell Professor Bill Maxwell's consulting clients. This was for a logistics project at a major greeting card company. It was the mid-1970s and I had just started on the faculty at Cornell University. Bill had brought me in because his client had heard of simulation and thought it might be useful. It was not clear that simulation would even be possible. At that time, a run of even a modest simulation could take several days (to find your next error).

Clear, explicit, communication was vital. Bill's clients patiently answered four questions about their activities: What is done (an activity name)? What is needed to do them (start events)? What do you get when activities are completed (end events)? And how long might they take (delay times)? For each activity they described, I sketched the start-end event vertices connected with an edge labeled with a delay function.

Once I had collected these separate 'activity' edges (there were pages of them), I could tediously, but methodically, represent the activity interactions on my own as conditional edges. This was done by connecting the end events for each activity that provides something, to the start events of all the activities that need it, conditional on everything else the activity needs to begin being available. This was my first Event Graph. It was a mess. The clutter was reduced dramatically by allowing activity edges to be both conditional and timed, and passing attribute function values among event parameters to distinguish similar events. I added a few isolated event vertices that were not naturally part of specific activities, and event cancelling edges for when activities might be interrupted.

The resulting graph fit easily on a single page. It was simple to describe (one sentence per edge), and straightforward to program (in FORTRAN on a self-assembled Heathkit desktop). I informally used these hand-drawn graphs for about a decade.

When graphics were introduced to desktop computers, I discovered I could save time with a program for developing "Event Graph" simulations. Since graphic programming was difficult, I was forced to embrace minimalism: using only circles and lines (getting arrowheads to point in the right direction was a major breakthrough.)

With the arrival of the computer mouse, software suitable for the classroom could be developed: first on an Apple, then in DOS followed by Windows. This software, called SIGMA (https://sigmawiki.com), allows activity start-end edges to be grouped into "activity" vertices. Most simulation models I create in my consulting practice use activity vertices. Nevertheless, it wasn't until Gerd posed this question that I ever referred to such grouped vertices as "activity vertices".

SIGMA is primarily used to introduce commercial software to students. Entities were also added to show students how "process modeling" languages work. Demonstrations of GPSS, Arena, and Simio are provided. This simulation teaching approach has worked well. A student who from the University of Arizona won first place in the Arena Modeling contest said "Our simulation class was taught in SIGMA, not Arena, but our instructor did an excellent job in teaching us how to construct simulations so that we could pick up any package and use it" (Stiles 2002). Students taught with SIGMA have won first place in several other modeling contests in various fields.

In Event Graph education, an emerging tool is Tao (initially called "tau", following "sigma") that runs in any browser (with any operating system) since it is implemented with JavaScript. I encourage anyone looking for an Event Graph sandbox to download this open-source software.

After reading this question, I realized that I have always developed my own Event Graph models by focusing on activities, adding isolated events when helpful or necessary. Thinking of grouped start-end event vertices as activity vertices feels natural. My answer to the question is that the concept of activities has always been included in Event Graphs, even if I wasn't fully aware of it.

Yahya: Based on Lee's narrative, the concept of a canceling edge was introduced to provide a mechanism for interrupting activities before they end. Therefore, it is the notion of activity while reasoning about the model that led to the necessity of a canceling edge in the set of modeling constructs for EGs. Interestingly enough, it is also the notion of activity that led to the discovery that canceling edges in EGs are redundant (by introducing additional artificial activities into the model). These additional activities may not exist in the actual system (i.e., artificial activities) and they are conveniently made-up by the modeler. Hence, the canceling edge was deemed redundant while the artificial activity became essential.

Since the modeling power of the notion of activity was shown by eliminating an element from the initial EG formalism, was this decision evaluated empirically in terms of complexity of the newly resulting models and their execution times?

Clearly, the concept of activity is implicitly implied in the EG formalism. Furthermore, a modeler has to utilize the notion of activity while analyzing the system under study and then map the activities onto the possible events that occur in the system. Eventually, a proper EG-based model of the system should emerge. However, in introductory simulation courses using EGs as the main modeling formalism, students will experience challenges since the notion of activity is not explicitly included in the EG formalism. In addition, some activities and events are artificial (i.e., they are not naturally observed in the system). These artificial activities and events are introduced to replace canceling edges in EG-based models. As a result, modeling with EGs becomes more challenging.

From your experience, how can we help students to think in terms of activities when using EGs for developing DES simulation models?

Lee: I have informally taught the activity framework using Start-End event pairs through answering the same four questions as in my first Event Graph. This seems to work well once I go through a few examples.

Murat: I do not have much to add to this discussion, but my position on "activity" in EGs is very similar to what has already been said. According to the principle of parsimony, an activity can be modelled with only two events: a start event and an end event. This is in fact what we have used in many real-world modelling projects, such as in Varol and Gunal (2015). In this model, an activity represents naval operations that require one or more naval units, such as frigates and helicopters, in operations against piracy at sea. We have discovered in this particular modelling that we do not need a new concept for an activity.

However, I should also note that if you have a complicated model with many events, it becomes difficult to track them, so we used the concept of "components." It is as simple as using input and output pins to connect EGs together. A similar architecture is used by Buss (2002) in the military domain.

Gerd: As emphasized by Enver, EGs provide the most parsimonious representation of event-based simulation (with event-scheduling). They define this modeling paradigm in an implementation-agnostic way. Since event-based simulation is the foundation of DES, EGs are fundamental for the understanding of DES.

However, being the most parsimonious modeling language for DES implies that EGs are low-level and therefore not very user-friendly. In particular, they do not provide a modeling element for expressing activities. Rather, activities have to be expressed in the form of pairs of start and end events. Adding a new modeling element for expressing activities to EGs (such as a rectangle with rounded corners) is straightforward and does not change the semantics of EGs, as these activity rectangles can be easily eliminated by replacing them with a corresponding pair of start and end events.

The same consideration applies to the use of event canceling edges, which allow expressing the interruption (or preemption) of ongoing activities. As pointed out by Yahya, it's of theoretical interest to know if they are redundant, but for the sake of the usability of EGs, they should be kept in the language.

There is a third basic modeling element that can be added to EGs for enhancing their usability without changing their semantics: *objects*. Adding objects to EGs allows expressing state variables in the form of object attributes. Since a discrete dynamic system in the real world consists of objects that are affected by

events, it is very natural to add objects to EGs as a language for modeling such systems. It also follows the fundamental insights of the simulation language Simula about the value of objects as a modeling and programming concept, and it harmonizes EGs with Object-Oriented modeling and programming languages.

Interestingly, when adding activities to EGs, another opportunity for reducing model complexity arises. Since the scheduling of activity start events requires to check if the required resources for starting the activity are available, activity start event scheduling arrows are always conditional with a condition that follows a certain pattern. This recurrent pattern, and the clutter that comes with it, especially when the model includes many activities, can be eliminated by adding a new type of arrow, which I call "Resource-Dependent Activity Scheduling" arrow.

I have described these extensions of the EG modeling language by adding objects, activities and Resource-Dependent Activity Scheduling arrows in (Wagner 2022).

5 QUESTION 4

There have been several efforts where other discrete simulation formalisms, like Petri Nets and DEVS, have been mapped to Event Graphs as a target formalism. It seems that Event Graphs do have the potential for becoming the assembly language for DES. Following the analogy of assembly instructions in computer architecture, constructs in Event Graphs should be extended and formally defined to enable the transformation of high-level concepts from other formalisms to Event Graphs.

What do you think is needed to accomplish the goal of having Event Graphs as the target formalism for DES?

Yahya: An operational semantics for EGs based on Abstract State Machines (ASM) was proposed by Wagner (2017). This semantics was used in (DeBuhr and Sarjoughian 2021) for transforming DEVS models into EGs. Therefore, ASMs provide an executable specification of EG-based simulation models. This is a major step in the right direction. Furthermore, this step will enable the automatic verification of EGs before they are translated into executable code.

On another front, several attempts were made to extend EGs by adding new elements (Wagner 2019; Osais 2018). These proposals range from using new visual syntax elements, such as mini diamonds at the source side of conditional scheduling edges and using a diamond with an internal plus sign for expressing parallel branching from event nodes, to new concepts such as event annotations (e.g., for redundant and terminal events). Further work still needs to be done. For example, event priorities should be explicitly captured in EGs (possibly with a new type of event node annotation). Event annotation will enable the modeler to communicate her intention to the ASM module in the toolchain (EG -> ASM -> Executable Specification -> Verification -> Actual Runnable Code). In this way, ASM is the abstract machine and EG is its assembly language. This abstract machine then generates code runnable on actual hardware.

Finally, canceling edges should not be discarded from the formalism. This construct will enrich the EG formalism and it will enable building larger EGs. As a matter of fact, a more general replacement for canceling edges (referred to as the execution condition) was already proposed in (Ingalls et al. 2003). This construct can be easily incorporated into the ASM semantics of Wagner (2017).

Gerd: I agree with Yahya's characterization of EGs as a kind of assembly language into which other DES languages can be compiled. This view confirms the fundamental role played by EGs for DES modeling. We have already discussed above that activities can be compiled away by replacing them with a pair of start and end events together with suitable conditions for scheduling their start event only when the required resources are available.

In principle, objects (used, e.g., for representing individual resources) can be compiled away into a set of state variables corresponding to the object attributes. However, since this would lead to a large and unstructured set of state variables, it seems preferable to have objects in EGs (attached to events), which may then be called "Object Event Graphs".

An important goal in this project should be to show how the "process models" of today's predominant GPSS/SIMAN/Arena-style simulation tools can be mapped to EGs. Notice that the processing nodes of such "process models" (confusingly called "Process"/"Server"/"Service" in Arena/Simio/AnyLogic) are overloaded with two meanings: they represent both a processing station (resource) object and a processing activity.

6 QUESTION 5

SIGMA, a simulation tool developed by Lee's group, represents a major milestone in the history of Event Graphs. SIGMA was successfully used in the classroom. It contains a syntax checker and a code synthesizer. SIGMA produces a verbalization of an Event Graph, which can be used by subject matter experts for validating a model. However, it does not support automated model checking based on correctness properties. Later, in 2017, there was an attempt to replace SIGMA with a web-based tool called *Tao*.

Do you think tools like SIGMA and Tao would have pushed Event Graphs more to the forefront if they would have supported forms of automated model checking?

Enver: I believe that the ultimate success of any academic work is reflected in how widely it is adopted in practice, i.e., in how it provides a better decision support tool to engineers and managers. The adoption is typically facilitated in two (not necessarily mutually exclusive) ways: the methodology can be incorporated in a commercial simulation package (e.g., optimization and ranking and selection techniques embedded in Arena or Simio). Alternatively, the author of the work might set up a commercial infrastructure or platform where he/she can market the methodology to a broader audience while providing support services (e.g., a hotline or user groups) to the users. For SIGMA, the first option, adoption by a commercial package, was never viable. As for the commercialization of SIGMA or Tao, Professor Schruben has always been an academic. While he used the software in his own academic and consulting work, demonstrating innovative ways of combining modeling and experimental design, he never invested much effort to fully commercialize the software. As a result, SIGMA and Tao remain among the best kept secrets in simulation modeling and analysis.

Yahya: An EG is a simulation model that has phenomenological content relating causes and effects. In his answer to question 3, Lee mentioned that he used the elements of EGs to visualize the phenomenon under study. At the end of this exercise, Lee would successfully reduce a complex phenomenon to a set of activities, events, transitions, state variables, and conditions. This verbal formulation of the phenomenon can conform to semantic and rules. Conformance is at the core of model checking.

In an interview (Sargent 2013), Lee mentioned that he had to learn many new things in order to push EGs forward (e.g., graph theory which is a standard topic in computer science). Clearly, Lee as a researcher did get out of his comfort zone to nurture his idea. On the other hand, the idea of EGs came out at a time where silos are the norm in academic environments. The first collaboration with a computer scientist to formalize EGs in (Feng, Lee and Schruben 2010) happened 27 years after the invention of EGs. Unfortunately, this collaboration was very brief although it could have led to the exploration of model checking in embedded computation.

Hence, due to the phenomenological capabilities of EGs and the importance of model verification and validation in the simulation community, EGs would have seen more adoption if SIGMA and Tao (or any other EG-based tool) supported model checking. In fact, the breakthrough to computer science and engineering would have happened early on. However, as Lee mentioned in an interview (Sargent 2013), before an entrepreneurial idea can be pursued, someone has to think about "who will feed the chickens".

Murat: I will respond to this question from two points of view. First, the challenge of developing an idea and turning it into a product. In today's "start-up" culture, there are many examples of this challenge. I am referring to EGs as an idea and SIGMA and Tao as products of that idea. Professor Schruben was one of

those who recognized the challenge many years ago and had the courage to act. EGs could not stand alone as a method for DES and event scheduling, it had to be supported by a software product. He did that very well and proved that products are the medium to "spread the message". As academics, we go down this road a lot. There are many examples, and all are great stories, like that of Taha (2017) and his TORA. I do not mean that "great stories" as becoming a unicorn start-up company, but a medium for spreading ideas.

Secondly, a clear answer to this question, in my opinion, is "Certainly yes!". The reason for my answer is that I believe in software that helps users achieve what they want. Part of this could be, in today's jargon, "user experience". Automated model checking in SIGMA and Tao means that a modeler can see if the properties defined in an EG represent the system as intended. The future of simulation software lies in supporting automated model creation and verification using verbal definitions of a system, and pictorial representations such as EGs. We still have a lot of work to do...

Gerd: As pointed out by Yahya and Murat, supporting forms of automated model checking would have made SIGMA, and EGs, more attractive for computer scientists and may have helped EGs to get the same attention as PNs. However, SIGMA, as an early visual process modeling tool, was a milestone anyway, also without automated model checking capabilities.

In fact, defining a general approach to automated model checking for DES models in the form of EGs is still an open problem. I think that Leslie Lamport's work on TLA+, the "temporal logic of actions", see (Lamport 1994) and (TLA+ Homepage 2023), and on PlusCal, a language for writing and debugging concurrent algorithms (PlusCal Tutorial 2023), is highly relevant for developing such an approach.

7 QUESTION 6

Capturing causality will play a significant role in Artificial Intelligence (AI). AI models become explainable if causal models can be extracted from them. Events and the relationships between them are the core abstractions used in constructing causal models. Event Graphs are a good candidate for such a task.

How can Event Graphs be evolved to meet the needs of causal modeling in the AI domain?

Lee: I used EGs as my fundamental tool for several years teaching our graduate course in Machine Learning and AI at Berkeley. This seemed particularly natural for solving Bayesian Networks. Over several years we built some EG tools in SIGMA for some AI methodologies for this course. It was a natural fit.

Gerd: The basic pattern of EGs, stating that whenever an event (of a certain type) occurs, this leads to certain state changes and certain follow-up events, can be read as a pattern that captures a causal regularity. This interpretation of EGs as models of causal chains also implies that the delays of scheduling edges in EGs are always strictly greater than zero, since the occurrence times of caused events are strictly greater than the occurrence time of their cause event. As a consequence, even if a scheduling edge from A to B specifies that a B event is scheduled to happen immediately after an A event, this does not allow a delay of zero time units, contrary to what Lee has allowed for EGs in SIGMA.

Yahya: With the invention of EGs by Lee in 1983, causal reasoning became a habit in simulation model development. An EG represents a causal structure over a set of events that drive a phenomenon. This explainable causal model is used as a source of sample paths (or data) that can be used for training AI algorithms. However, EGs only represent primitive atomic events that are related by only two types of relations: scheduling and canceling. We are in the right time for the next version of EGs. The next version of EGs should distinguish between atomic and non-atomic events (Kshemkalyani 1997). Non-atomic events, a group of primitive atomic events, enable new abstractions. In fact, an activity can be thought of as a non-atomic event. For example, in the single-server queueing system, combining the atomic events *ProcessingStart* and *ProcessingEnd* results in a non-atomic event *Processing*, which is essentially the *Processing* activity.

Furthermore, the next version of EGs should support new types of causal relations. The current scheduling relation supported in EGs is referred to in AI as a one-shot causal relation, e.g., an arrival causes a departure (Rieger and Grinberg 1977). Other types of causal relations, such as continuous causation, can also be defined (Terenziani and Torasso 1995). An example of continuous causation is that the non-atomic event *Processing* causes energy consumption. In this case, the effect persists as long as the cause (i.e., the activity) is present. These new additions to the EG formalism, in addition to other ones, will generalize EGs to model phenomena and applications of interest to the AI community.

Murat: AI tries to mimic what neurons do in human brains, by examining the patterns in data, and create some sort of causality with patterns. I believe that EGs can help improve AI applications, by its nature in causality. EGs are naturally "pattern-friendly" structures. I call that "chain of events" to represent "patterns", as a representation of patterns of events in real life. When modelling a real system, we track "observable" events and try to sort them in time to create patterns, and then say that this pattern creates this outcome, just like what AI does. Furthermore, EGs handle "time" and "stochasticity" better than any other methodology. For this reason, I see great potential of EGs in AI. I am sure our discussion will open up new dimensions in AI research.

8 CONCLUSIONS

This panel discussion has provided various views on the meaning and potential of Event Graphs for Discrete Event Simulation and Computer Science. Panelists have reflected on the history of Event Graphs and have linked it with some important ideas in other fields like Process Modeling, Computing, and Artificial Intelligence. The paper underlines the importance of reviving Event Graphs for sparking new ideas for research in these fields.

ACKNOWLEDGMENTS

Yahya Osais would like to acknowledge KFUPM for financial support.

REFERENCES

Bus, A., and C. Blais. 2007. "Composability and Component-based Discrete Event Simulation". In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. Henderson, B. Biller, M.-H Hsieh, J. Shortle, J. Tew, R. Barton, 694–702. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Buss, A. 2002. "Component-Based Simulation Modeling with Simkit". In *Proceedings of the 2002 Winter Simulation Conference*, edited by E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 243–249. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

DeBuhr, N., and H.S. Sarjoughian. 2021. "Model Transformation across DEVS and Event Graph Formalisms". In *Proceedings of the 2021 Winter Simulation Conference*, edited by S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo and M. Loper, 1–12. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Elizandro, D., and H. Taha. 2007. *Simulation of Industrial Systems: Discrete Event Simulation Using Excel/VBA*. Auerbach Publications. https://doi.org/10.1201/9781420067453.

Evans, G.W., G.F. Wallace, and G.L Sutherland. 1967. *Simulation Using Digital Computers*. Prentice Hall. Englewood Cliffs, New Jersey.

Feng, T.H., E.A. Lee, and L.W. Schruben. 2010. "Ptera: An Event-Oriented Model of Computation for Heterogeneous Systems". In *Proceedings of the 10th ACM International Conference on Embedded Software*, October 24 - 29, Arizona, USA, 219–228.

Haas, P. 2002. *Stochastic Petri Nets: Modelling, Stability, Simulation*. Springer Series in Operations Research and Financial Engineering. New York, NY: Springer.

Hutchinson, G.K. 1981. "The Automation of Simulation". In *Proceedings of the 1981 Winter Simulation Conference*, edited by T.I. Ören, C.M. Delfosse, and C.M. Shub, 489–495. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Ingalls R., D.J. Morrice, E. Yücesan, and A.B. Whinston. 2003. "Execution Conditions: A Formalization of Event Cancellation in Simulation Graphs", *INFORMS Journal on Computing* 15(4):397–411.

Jacobson, S.H., and E. Yücesan. 1999. "On the Intractability of Verifying Structural Properties of Discrete Event Simulation Models." *Operations Research* 47(3):476–481.

Kshemkalyani, A.D. 1997. "Reasoning about Causality between Distributed Nonatomic Events", *Artificial Intelligence* 92(1-2):301–315.

Lamport, L. 1994. "Introduction to TLA", *ACM Transactions on Programming Languages and Systems (TOPLAS)* 16(3):872–923.

Osais, Y. 2017. *Computer Simulation – A Foundational Approach Using Python*, New York: CRC Press.

Osais, Y. 2018. "Towards a Framework to Detect Modeling and Semantic Errors in Event Graphs", In *Proceedings International Symposium on Performance Evaluation of Computer and Telecommunication Systems* (SPECTS), July 9-12, Bordeaux, France, 1–10.

PlusCal Tutorial. https://lamport.azurewebsites.net/tla/tutorial/intro.html. Accessed 6th May 2023.

Rieger, C., and M. Grinberg. 1977. "The Declarative Representation and Simulation of Causality in Physical Mechanisms". In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, August 22-25, Massachusetts, Cambridge, USA, 250–256.

Robinson, S., R. Brooks, K. Kotiadis, and D.-J. Van Der Zee, (Eds.). 2010. *Conceptual Modeling for Discrete-Event Simulation*. CRC Press.

Sargent, R.G. 2013. Interview with Lee W. Schruben. December 10th, 2013. NC State University Libraries, Computer Simulation Archive. https://d.lib.ncsu.edu/computer-simulation/videos/lee-w-schruben-interviewed-by-robert-g-sargent-schruben/. Accessed 6th May 2023.

Schruben, L.W. 1983. "Simulation Modeling with Event Graphs". *Communications of the ACM* 26:957–963. https://dl.acm.org/citation.cfm?id=358460.

Schruben, L.W. 1995a. *Graphical Simulation Modeling and Analysis: Using Sigma for Windows*. Course Technology Inc.

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

Schruben, D. A. and L. Schruben. 2006. Simulating Dynamic Systems with Event Relationship Graphs. Accessed 6th May 2023. https://sigmawikidotcom.files.wordpress.com/2022/11/sigma-simulation-graphical-modeling-and-analysis.pdf

Schruben, L.W. and E. Yücesan. 1993. Modeling paradigms for discrete event systems. *Operations Research Letters*, 13(5):265-275.

Schruben, L.W., and E. Yücesan. 1994. "Transforming Petri Nets into Event Graphs Models". In *Proceedings of the 1994 Winter Simulation Conference*, edited by J. Tew, S. Manivannan, D. Sadowski, A. Seila, 560–565. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Stiles, E. 2002. "UA Industrial Engineering Seniors Win Software Simulation Competition". University of Arizona News. https://news.arizona.edu/story/ua-industrial-engineering-seniors-win-software-simulation-competition. Accessed 6th May 2023.

Taha, H. A. 2017. *Operations Research: An Introduction*. 8th edition. Upper Saddle River, New Jersey. Pearson Education, Inc.

Terenziani, P., and P. Torasso. 1995. "Time, Action-Types, and Causation: An Integrated Analysis", *Computational Intelligence* 11(3): 529–552.

Törn, A.A. 1981. "Simulation Graphs: a General Tool for Modeling Simulation Designs". *Simulation* 37(6):187–194.

TLA+ Homepage. 2023. https://lamport.azurewebsites.net/tla/tla.html. Accessed 6th May 2023.

Varol, A. E. and M.M. Gunal. 2015. "Simulating prevention operations at sea against maritime piracy". *Journal of the Operational Research Society* 66(12):2037–2049.

Wagner, G. 2017. "An Abstract State Machine Semantics for Discrete Event Simulation". In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 762–773. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc. URL: https://www.informs-sim.org/wsc17papers/includes/files/056.pdf

Wagner, G. 2019. "Process Design Modeling with Extended Event Graphs", *Proceedings of the 2019 Summer Simulation Conference*, July 22-24, 2019, Berlin, Germany. Society for Computer Simulation International.

Wagner, G. 2022. *Discrete Event Simulation Engineering*. https://sim4edu.com/reading/des-engineering. Accessed 6th May 2023.

AUTHOR BIOGRAPHIES

MURAT M. GUNAL is an Associate Professor in the Department of Industrial Engineering at Fenerbahce University, Istanbul Turkiye. He is also the founder of Simarter Ltd. an analytics company specialised in modelling, simulation and optimisation. His main research area is simulation methodologies, healthcare and industrial applications. He developed many models for decision making. His models are used in real and challanging decision making problems in many sectors. His email address is murat.gunal@fbu.edu.tr.

YAHYA E. OSAIS is an Assistant Professor in the department of computer engineering at King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia. He is also a member of the interdisciplinary research center for intelligent secure systems at KFUPM. His research interests include simulation modeling and reinforcement learning. His email address is yosais@kfupm.edu.sa, and his homepage is https://faculty.kfupm.edu.sa/coe/yosais/.

LEE SCHRUBEN is Professor Emeritus of the University of California, Berkeley. He is the inventor of Event Graphs and has led the development of the Event Graph modeling and simulation tools SIGMA and Tao. Among other prizes, he is a two-time winner of the Informs ISIM Outstanding Publication Award with single-authored papers and he received the Informs Society on Simulation's highest honor, its Lifetime Professional Achievement Award in 2018. His email address is lees@berkeley.edu.

GERD WAGNER is Professor of Internet Technology in the Department of Informatics, Brandenburg University of Technology, Germany. After studying Mathematics, Philosophy and Informatics in Heidelberg, San Francisco and Berlin, he (1) investigated the semantics of negation in knowledge representation formalisms, (2) developed concepts and techniques for agent-oriented modeling and simulation, (3) participated in the development of a foundational ontology for conceptual modeling, the *Unified Foundational Ontology (UFO)*, and (4) created a new Discrete Event Simulation paradigm, *Object Event Modeling and Simulation (OEM&S)*, and a new process modeling language, the *Discrete Event Process Modeling Notation (DPMN)*. Much of his recent work on OEM&S and DPMN is available from sim4edu.com and dpmn.info. His email address is G.Wagner@b-tu.de.

ENVER YÜCESAN holds the Abu Dhabi Commercial Bank Chair in International Management in the Technology and Operations Management Area at INSEAD. He is an Industrial Engineer from Purdue University with a PhD in Operations Research from Cornell University. His research is at the interface of simulation, optimization, and statistics. He has recently been focusing on agricultural supply chains to address key challenges such as identification of robust parent seeds, farmer contracting, small holder management, and production and inventory planning under increasing volatility driven by population dynamics and climate change. He has served as the Proceedings Co-Editor for WSC'02 and Program Chair for WSC'10. He was also a member of the WSC Board of Directors between 2012 and 2020. He has received INFORMS Distinguished Service Award in 2015 and been elected an INFORMS Fellow in 2022. His email address is enver.yucesan@insead.edu.