Conceptual Modeling in Simulation: A Representation that Assimilates Events

Sabah Al-Fedaghi
Computer Engineering Department
Kuwait University
Kuwait

***Abstract*—Simulation is often based on some type of model of the evolved portion of the world being studied. The underlying model is a static description; the simulation itself is executed by generating events or dynamic aspects into the system. In this context, this paper focuses on *conceptual* modeling in simulation. It is considered the most important aspect in simulation modelling, at the same time it is thought the least understood The paper proposes a new diagrammatic language as a modeling representation in simulation and as a basis for a theoretical framework for associated notions such as events and flows. Specifically, operational semantics using *events* to define fine-grained activities are assimilated into the representation, resulting in an integration of the static domain description and the dynamic chronology of events (so-called *process* level). The resulting unified specification facilitates understanding of the simulation procedure and enhances understanding of basic notions such as things (entities), events, and flows (activities).**

Keywords—events; flow; conceptual modeling; simulation; diagrammatic language

I. INTRODUCTION

Simulation, as employed in this paper, is a technique used to imitate a system or process, as in the case of studying concrete phenomena and their causal relations. It is often based on some type of *model* of the evolved portion of the world under study (the domain – the subject of the simulation study) [1]. Such underlying model is a static description of the domain; the simulation itself is conducted by generating *events* or dynamic aspects into the system. Accordingly, the execution of a model or part of it to reach certain conclusions about the system is known as simulation.

This paper focuses on a *conceptual* model (in contrast to, say, a mathematical or computational model) as an abstract representation of a system intended to replicate some of its properties [2] and to use as a tool for communication between stakeholders. Methods for such representation include texts, diagrams, and logic flow charts [3]. According to Wagner [4], in simulation engineering, a system model consists of both an information model and a process model. "Conceptual modelling, one of the first stages in a simulation study, is about understanding the situation under study and deciding what and how to model" [5]. This phase is considered the most important aspect of simulation modeling [6]; at the same time it is thought to be the least understood [7].

This paper seeks to utilize a new diagrammatic language for conceptual modeling of simulations. Diagrams seem "the best approach to enhancing *communication* among a wide variety of specification audiences" [8; italics added]. According to Zeigler [9], communication is one of the most important and also least appreciated aspects of modeling.

Here the purpose is not to introduce a complete technical solution, which would be outside the limited scope of a conference paper; rather, the aim is to develop an appreciation for the informality of the proposed approach, starting from defining flows and events and ending with a unified description of the system and its environment (simulation). A more ambitious aim is to propose a new approach to conceptual modeling in simulation and to develop a theoretical framework for its associated notions such as events and flows.

A. Focus of study: Diagrammatic representation

Over years of simulation research, many diagrammatic methodologies have been utilized in building models in simulation, e.g., activity cycle diagrams (state diagrams), event graphs, Petri nets, control flow graphs (see [10]), UML, and BPMN. To give an idea of the type of study examined in this paper, we consider a diagrammatic representation introduced in the form of activity cycle diagrams (ACDs) [11].

In the ACD-based approach, a simulation model is viewed as a collection of interacting entities. An active state usually involves the cooperation of different entities. To specify a model using ACDs, an activity cycle composed of queues and activities must be given for each class of entity in the model [10]. ACDs provide a means to recognize and prioritize simultaneous events at the specification level in simulation. "The strongest advantage of these diagrams is that they are simple to work with. However, to fully support the simulation process, these diagrams must be augmented" [10].

An ACD example is the English Pub, where the man entity either drinks or waits to drink. The barmaid either pours a drink or is idle. The glass is either used to drink from, empty, poured into by the barmaid, or full, waiting to be consumed (see Fig. 1) [10].

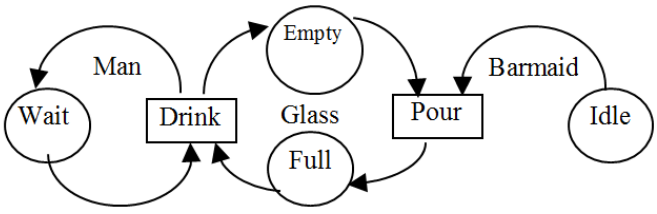

Fig. 1. Activity Cycle Diagram (redrawn, partial from [10])

B. *Contribution: Alternative Diagrammatic representation in simulation*

This paper presents an alternative to the ACD representation and applies the completed diagram to simulation. Fig. 1 of the English Pub will be recast in the proposed new language of diagramming. The claim is that this methodology provides a more complete specification suitable for a static domain model and its dynamic aspects needed in simulation.

For the sake of a self-contained paper, the next section briefly reviews the diagrammatic language, called the Flowthing Model (FM) that forms the foundation of the theoretical development of the paper, with the *deer dominance* example as a new contribution. With understanding of FM gained from this example, section 3 substantiates the claim that FM offers a more complete specification than the ACD graph of the English Pub discussed in the introduction.

After FM is illustrated with an example and used to support some claims in the introduction, section 4 introduces our research topic of conceptual modeling in simulation by contrasting some notions in the simulation literature against their representation in FM.

Section 5 contains the main contribution of the paper, an exploration of the FM representation in a wider organizational setting. Here we demonstrate that the FM representation embeds operational semantics that simplify the process of modeling a chronology of events.

Section 6 suggests that FM can provide a unifying language for modeling in simulation. Section 7 discusses a specific case study of simulation and applies the FM representation to the case.

II. A Diagrammatic Language

The approach utilized in the paper is called the Flowthing Model (FM), which has been used in a variety of applications (e.g., [12-16]). FM is a uniform method for representing things that "flow"; i.e., things that are *created*, *processed*, *released*, *transferred*, and *received* while retaining their individuality throughout the flow. They may queue in any of the flow stages. They flow within *spheres*, i.e., the relevant environment that encompasses the flow. A sphere may have subspheres.

Things that flow in a flow system are referred to as *flowthings* or, as here, simply as *things*. The life cycle of a thing is defined in terms of six mutually exclusive *stages*: creation, release, transfer, arrival, acceptance, and process (in which its *form may be changed*, but no new thing is generated), as shown in Fig. 2. The flow system shown in the figure is a generalization of the input-process-output model. The reflexive arrow in the figure indicates flow to the *Transfer* stage of another flow system. For simplicity, the stages Arrive and Accept can be combined and termed *Receive*.

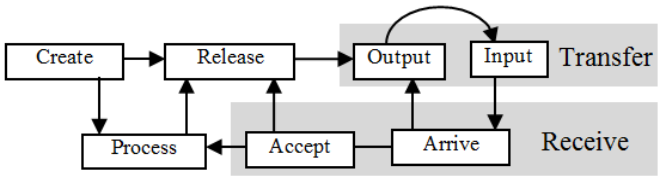

Fig. 2. Flow system

The *stages* in the life cycle of a thing are mutually exclusive (i.e., a flowthing can be in one and only one stage at a time). All other states or conditions of flowthings are not mutually exclusive stages. For example, we can have *stored* created things, *stored* processed things, *stored* received things, etc.; thus *stored* is not an exclusive stage. Things can be released but not transferred (e.g., a channel is down), or arrive but not be accepted (wrong destination), and so on.

The following example illustrates the FM diagrammatic language and its expressive power.

Example: As shown in Fig. 3, Geva [17] provides a text description of behavior of red deer during breeding season, followed by a diagram showing the causal connections described in the text, as produced by a typical student who "has in his repertoire skill for identifying basic meaningful units" [17].

> *It is frequently necessary to distinguish between dominance—one individual intimidating or threatening another—and leadership—one individual directing the group. Among red deer the dominance shown by males during the breeding season is vastly different from the leadership by the older females who are out ahead of the group and determine its direction without the use of force.*

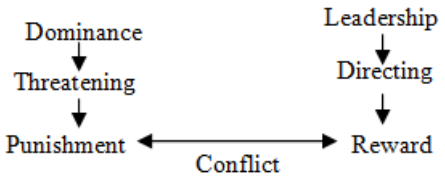

Fig. 3. Diagramming of descriptive text to identify causal relationships (partial, from [17])

Fig. 4 shows the FM representation of *male dominance among deer during breeding season* as interpreted from Fig. 3. The female flows to the male domain (circle 1 in the figure), where she is physically "processed" (2) in different ways. In one scenario she is under threat by the male (3). In case she disobeys (4), she is punished (5). (For simplicity, the flow of the female (1 and 2) is not enclosed in a box.)

The figure can be contrasted with the left side of Fig. 3 where Dominance → Threatening → Punishment.

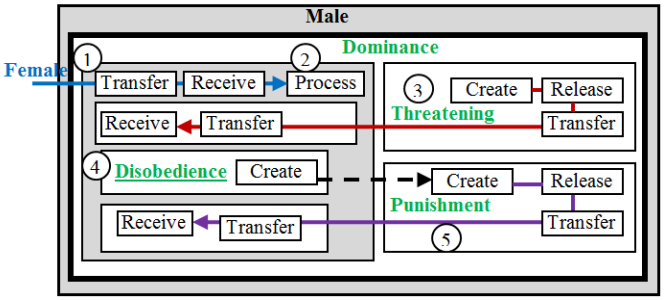

Fig. 4. Deer dominance by males during the breeding season

Similarly, Fig. 5 shows the FM representation of *leadership by the older females who are out ahead of the group and determine its direction without the use of force*. The process Leadership → Directing → Reward of Fig. 3 is also taken into account. The older female (circle 1) issues a direction (2) that flows (3) to each (4) member of the group (5). Execution of the direction (6) triggers (7) the generation (creation, 8) of a reward that flows to the member.

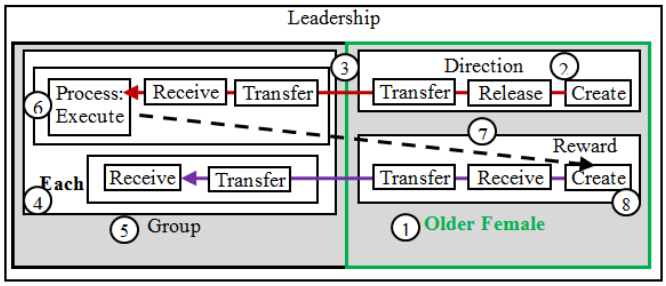

Fig. 5. Deer leadership by the older females who are out ahead of the group and determine its direction

III. The English Pub in FM

Now, with this understanding of FM, we can substantiate our claim that the specification offered by FM, suitable for static domain modeling along with dynamic aspects needed in simulation, is more complete than the ACD graph of the English Pub discussed previously. Specifically, this section focuses on the static representation, leaving the event-oriented simulation to a later section.

Returning to the example of an English pub discussed in the introduction, from the representational point of view, the diagram of Fig. 1 seems incomplete as a picture of the domain of the pub. The diagram is formed from three separate scenes:

- The state of a Man changes between drinking and waiting
- The state of a Glass changes between empty and full
- The state of a Barmaid alternates between idle and pouring

The resulting conceptual picture is formed by jumping from one state to the next. Thus we find, for example, that the glass is empty during the entire time of pouring, the state specified as (Man: Waiting, Glass: Empty?, Pouring: ON). Modeling of a domain is selective in its details; nevertheless, in the opinions of the present authors, the model should not "contradict" reality in the domain, e.g., the glass is not empty during the pouring process.

The FM representation provides a "continuous" portrait of the system, as shown in Fig. 6. First, the man orders a drink (circle 1 in the figure) and the order flows to the barmaid (2) to trigger taking an empty glass (3) from storage that flows to the barmaid (4). This triggers release of the liquor (5) to the barmaid, thus creating a filled glass (6) that flows to the man (7) to drink (8).

As we see, inherent characteristics of the FM representation, based on the notion of flow, force the modeler to "connect the dots" and paint a complete chronology of events in the system.

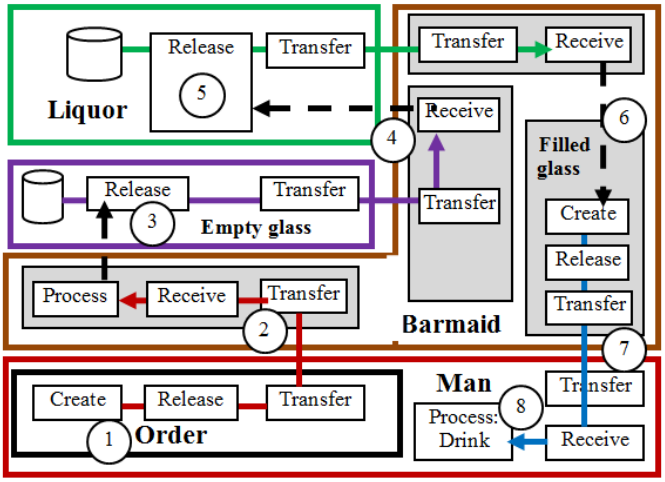

Fig. 6. FM representation

IV. Contrasting Some Notions

This section contrasts some notions used in the simulation literature with their representation in FM. The purpose is to highlight how the FM approach is different from current simulation methods, as an introduction to applying FM in the rest of the paper. Additionally, the discussion in this section shows how FM provides a *unifying* view of many notions in simulation, using the same diagrammatic language introduced in section 2.

In simulation literature, an *object* is typically defined as anything with attributes [18]. Objects and attributes in FM are *things* as defined previously. The two concepts of *time* and *state*, fundamental in the context of simulation, are also things. Similarly, an event is a thing. According to Page [10],

An *event* is a change in an object state, occurring at an instant, and initiates an activity precluded prior to that instant. A process is the succession of states of an object over a span (or the contiguous succession of one or more activities)." [Italics added]

Page [10] illustrates the notions of event, activity, and process in simulation as shown in Fig. 7.

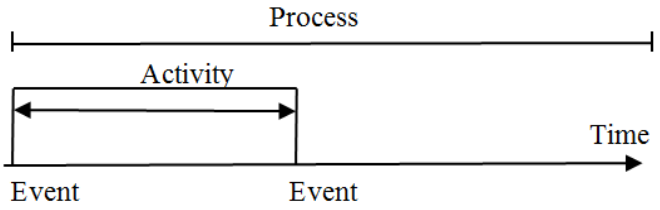

Fig. 7. An illustration of event, activity, and process (redrawn, partial from [10])

An event in FM is a thing since it can be created, processed, released, transferred, and received. It triggers the creation, release, processing, transfer and receipt ("activity" in the above quote) of other things. *Process* in the quote is the *flow* in FM, e.g., a thing is created, released, and transferred. Fig. 8 illustrates the flow, event, and "activities", i.e., states in a flow system.

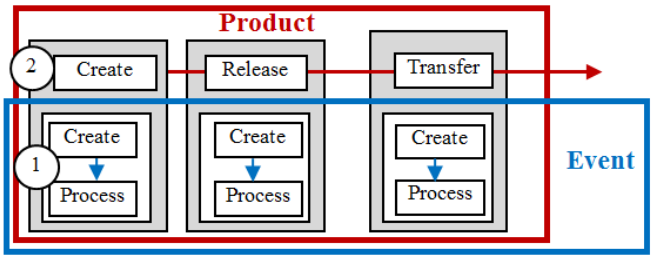

Fig. 8. Illustration of event flow and "activities"

In the figure, the product is modeled from a static description as a thing that is created, released, and transferred; however, such creation, releasing, and transferring occurs only as a result of events. The event is the trigger that activates each stage of flow. Such a distinction between a static description and dynamic activation of a system is a very important aspect in simulation; nevertheless, the two levels of specification (static and dynamic) can be modeled by the same diagrammatic language.

For example, the event at circle 1 activates (causes creation of a product thing: instance). Process (1) in the figure refers to the event (creation) taking its course (e.g., duration). Event scheduling in the diagram could be accomplished through clock "jumps"; e.g., every hour, with each clock tick creating the next period or next new product, etc.

V. Applying FM to Events

This section explores the FM representation in a wider organizational setting. It demonstrates that the FM representation embeds *operational semantics* that simplify simulating the chronology of events in comparison with other diagrammatic representations such as UML sequence diagrams. The resultant operational semantics are conceived along the lines of UM causality model specification in which *events* are used to define fine-grained behaviors. First, let us explore the notion of event in various fields.

According to McKee [19] in the context of English literature, "STRUCTURE is *a selection of events* … that is composed into a strategic sequence" [italics added]. Such a definition seems to imply that the structure of a literary piece is the structure of events.

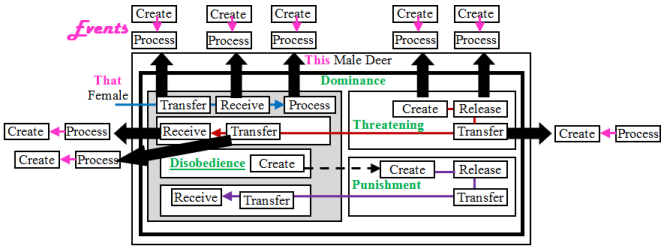

Fig. 9. Example of flows of events

In this paper, structure is the conceptual description of where the flow "takes place." It is the FM diagram. An *event* is a "happening" or "occurrence" in a point of the diagram (e.g., domain). It may be connected to earlier events (e.g., causal connections). Philosophically, in the context of a definition given by the French philosopher Jean-Luc Marion [20-21],

An event is that which … changes the given order of things, and then disappears, leaving its mark without return. An event shows itself as much as it gives itself to be seen. An event, properly so-called, poses a problem for thought to think it in ways other than according to the measure of its visible appearance. [22]

Additionally, an event *inaugurates* something new from within a given situation and a break or rupture in the current state of a situation. It is both situated in and supplementary to what the present authors call a structure. (Some of these expression are taken from Butchart [22] describing Badiou's [23] philosophy, but reinterpreted differently).

This interpretation views the FM diagram as the structure: a static map of things and their flows. Events are *things* of happenings (occurrences), as shown in Fig. 9 for dominance by males described in Fig. 4. This latter figure of dominance by males is a static scenario of the dominance, while Fig. 9 shows actual events (occurrences) as things that can be created, processed, released, transferred, and received.

All things flow in their streams in the basin of flow (the total FM diagram) which forms the *structure* of events, where

1) The event in this structure of a specific (that) female transferred to the dominance of a specific (this) male.
2) The event in which the female is received.
3) The event in which the female is processed, etc.

The scenario is like a scene in a movie being actualized, shot by shot (event by event). Each event "shows itself" and disappears, as in Butchart's [22] quote: "event, event, event, … assimilated in a structure."

As an illustration of assimilation of events in the FM representation, consider the following computer program taken from Podgurski and Clarke [24].

input (X, Y>;

1. if X > Y then
2. Max := X;

else

3. Max := Y;
4. endif;

Fig. 10 shows a partial view of its FM representation (see [25]). The figure also shows different events in this execution.

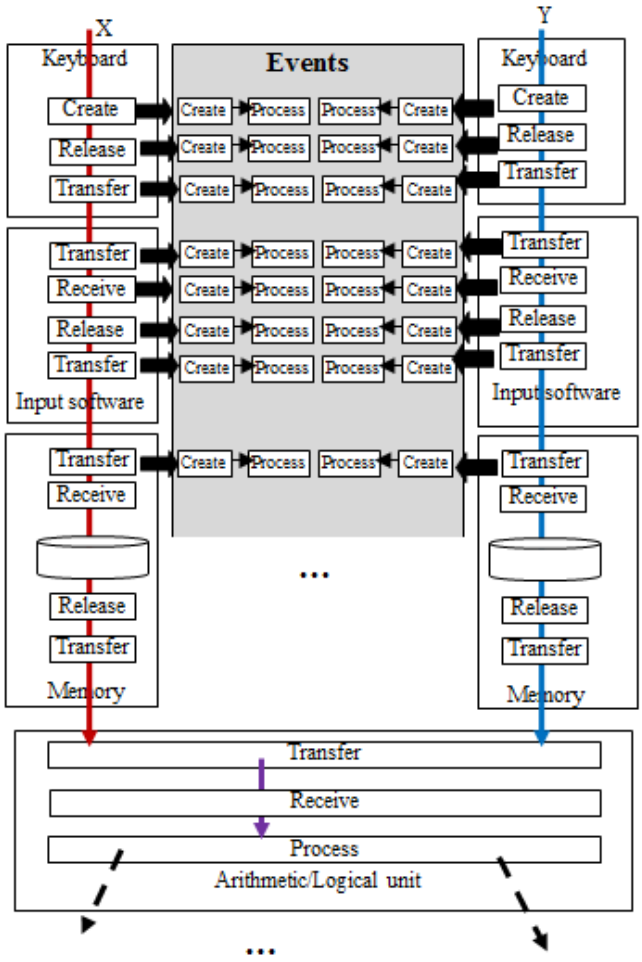

Fig. 10. Events when executing a program

First, the event in which X and Y are created (generated) in the keyboard. The values are release and transfer to the input software (e.g., *cin* in C++). Then the events continue one after another. Note that we have two levels of representation: (a) the structure of the situation, a static FM diagram of the program, and (b) the events, an FM diagram of events at the dynamic level where the event "*inaugurates* something new from within a given situation" [22].

Accordingly, an operational semantics can be defined to describe how to schedule events in the FM representation, therefore specifying the thread of control of execution. Operational semantics can also uncover concurrency of threads of events that offer several options for executions or occurrences, which is required in such systems as distributed systems.

For instance, consider the deer dominance scenario represented in Fig. 4. Two possible sequences of events (behavior – possible execution) are shown in Fig. 11,

1) A female arrives and immediately disobeys; hence, it is punished (1, 2, 3, 4B–10B)

2) A female arrives and immediately faces threats; hence, she immediately submits to the male without disobedience (1, 2, 3, 4A–9A)

Here, the male is assumed to exist prior to the arrival of the female. If we want to express this explicitly, we can add *Create* in the Male sphere to precede these two sequences of events.

Note that the flow and triggering force a partial order on some events; thus, *release* is preceded either by *create*, *process*, or *receive*; a threat is preceded by being communicated by the male (*released* and *transferred*). By the nature of this triggering, punishment is caused by disobedience, hence the sequence 4B, 5B in Fig. 11, etc. Accordingly, alternative sequences of events are modeled at certain points such as circle 3 in Fig. 11 where the A and B threads of events are separated according to circumstances, e.g., actual occurrences or execution.

Similarly, in the execution of the computer program of Fig. 10, the left and right sequences of inputting X and Y can be executed in parallel along the two threads of flow; their arrival at the ALU would then have to be synchronized.

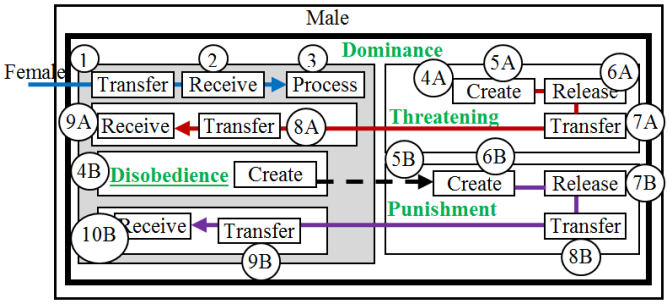

Fig. 11. Two possible sequences of events following the arrival of a female

VI. Application to Simulation

This section suggests that FM can provide a unifying language that supports modeling in simulation.

According to Wagner [4],

Due to their great expressivity and their wide adoption as modeling standards, UML Class Diagrams and BPMN seem to be the best choices for making information and process models in a model-driven simulation engineering approach. [4]

Wagner [4] illustrates how to develop a simulation of the functions of *a service desk* with the help of UML class diagrams and BPMN diagrams.

In the DES [Discrete Event Simulation] literature, it is often stated that DES is based on the concept of "*entities flowing through the system*". However, the loose metaphor of a "flow" only applies to entities of certain types: events, messages, and certain material objects may, in some sense, flow, *while many entities of other types, such as buildings or*

*organizations, do not flow in any sen*se. Also, subsuming these three different kinds of flows under one common term "entity flow" obscures their real meanings. It is therefore highly questionable to associate DES with a "flow of entities". Rather, one may say that DES is based on a flow of events. [4; italics added]

This view is different from the concept of flow in FM, where the conceptual flow is defined in terms of states of Create, Process, Release, Transfer, and Receive. In FM, *buildings* can flow conceptually. Suppose that a city zoning board decided to change the zoning for an area containing building 1234, where each zone has different regulations. The building in effect will "flow" to the sphere of a different zone, as shown in Fig. 12. In FM, events, entities, buildings, signals are all things that can flow.

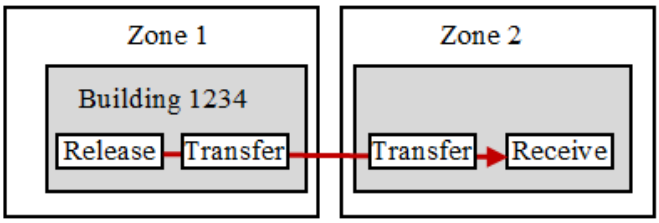

Fig. 12. The flow of a building

According to Wagner [4], "Unfortunately, this [concept of DES as a flow of events] is often obscured by the standard definitions of DES that are repeatedly presented in simulation textbooks and tutorials." Wagner [4] then describes Pegden's [26] simulation modeling worldview which provides "a framework for defining a system in sufficient detail that it can be executed to simulate the behavior of the system."

Over the 50 year history of simulation there have been three distinct world views in use: event, process, and objects.

Event worldview: The system is viewed as a series of instantaneous events that change the state of the system over time. This is the basis of the events flow systems in FM.

Process worldview: The system is described as a process flow in which sets of passive entities flow through the system and are subject to a series of process steps (e.g. seize, delay, release) that model the state changes that take place in the system.

Object worldview: The system is modeled by describing the objects that make up the system. The system behavior emerges from the interaction of these objects. [4]

According to Pegden [26], agent based modeling is typically implemented with the object worldview.

So, today's DES landscape is largely based on the process worldview and object worldview, and the fundamental concept of events is hardly considered anywhere. All three worldviews, and especially the process and object worldviews, which dominate today's simulation landscape, lack important conceptual elements. The event worldview does not support modeling objects with their (categorical and dispositional) properties. The process worldview does neither support modeling events nor objects. And the object worldview, while it supports modeling objects with their categorical properties, does not support modeling events. None of the three worldviews does support modeling the dispositional properties of objects with a full-fledged explicit concept of transition rules. [4]

Clearly, a Process worldview (the stages of the flow system) and an Object worldview are integrated into the FM diagram, which also includes events as flowthings. The FM description unifies these worldviews through describing them in one language. The FM representation models the system (e.g., deer dominance by males) and models a particular realization of that system using events as things (e.g., arrival of a female with no disobedience), similar to a computer program and testing a path of execution in the program.

Accordingly, we propose that FM can provide a unifying language that supports modeling in simulation.

VII. Example: Application of FM to Simulation

This section discusses a specific case study of simulation and applies the FM representation to such a case.

Wagner [4] views simulation engineering as a special case of software engineering, hence, applies model-driven development that includes conceptual, design, and implementation models. Wagner [4] uses an example that includes the entity types *Person*, *Customer*, and *ServiceQueue*, as classes in UM. UML stereotypes are utilized to distinguish between object types and event types as two different categories of entity types and use them for categorizing classes.

In the study case, an *event* involves exactly one customer who gets in line at exactly one service queue. The customers wait in a queue; when the queue is empty and the service desk is not busy, they are served. Four entity types can be extracted by analyzing the noun phrases of the description: ServiceDesk, ServiceQueue, Service clerk, and Customer. The last two are subclasses of Person.

After modeling all relevant object types in the first step, Wagner [4] models the relevant event types in a second step, as shown in Fig. 13. At this point, we reach the focus of our discussion in this paper: Methods of attaching events to the system description.

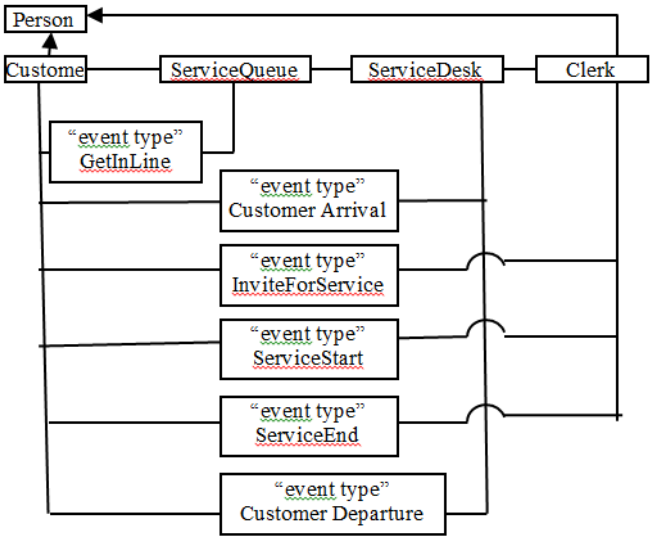

Fig. 13. The complete information model includes event types (redrawn, partial from [4])

Wagner [4] then draws a completely new diagram for the control of the Queue shown partially in Fig. 14, using yet another diagrammatic language, BPMN. Additionally, Wagner [4] models randomness of events using event types. For example, *CustomerArrival* is drawn (not shown here) as an exogenous event type with the class-level attribute *occurrenceTime* having the value (1,8), denoting a random variable with uniform distribution with lower bound 1 and upper bound 8.

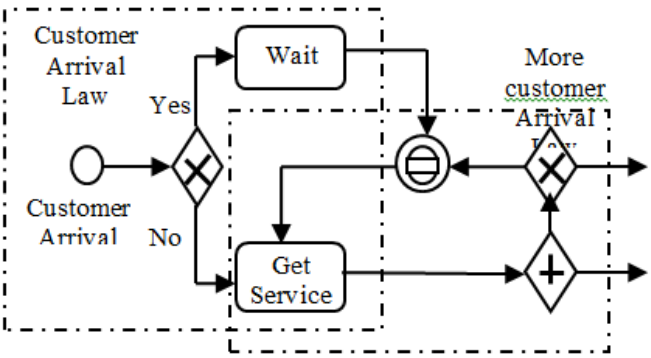

Fig. 14. Queue control (redrawn, partial from [4])

Wagner [4] then introduces a diagram (not shown here) for process design modeling using BPMN.

In the process design model, we turn the causal laws defined in the conceptual process model into corresponding transition rules described as BPMN sub-processes with

- a start/intermediate event for triggering the transition rule;
- tasks expressed as statements that contain typed variables and have a clear computational meaning;
- bindings of variables to entities. [4]

The whole approach exemplified by the case study discussed in this and the previous section seems to lack a unified foundation. First, it is based on "entities flowing through the system," as mentioned previously. Then it states that "flow" applies only to entities of certain types: events, messages, and certain material objects may, in some sense, flow, while many entities of other types, such as buildings or organizations, do not flow "in any sense" [4]. Finally, it is suggested that one could say that DES is based on a *flow of events*. Accordingly, as shown in the example, heterogeneous types of diagrammatic representations are used, ranging from UML and BPMN either in a mix of hierarchal structures with flowing events (Fig. 13) or as refinements of higher-level diagrams (Fig. 14).

The next section introduces the FM representation of the same case study.

VIII. Unified Schemata: Entities, Events, and Flows

To make this example more interesting, we assume two service clerks. Fig. 15 shows the FM representation of this case. Such an addition to the problem will allow us to distinguish between the system (the FM diagram) and its environment: in this case, simulation with events is performed on only one service desk while excluding the other service desk. In the figure, for simplicity, details of the second queue and service clerk are omitted.

Accordingly, in the figure, the customer arrives (circle 1) and proceeds to either Service desk 1 (circle 2) or Service desk 2 (circle 3). At desk 1, the customer is received into its queue (4) then proceeds to Clerk 1 (5) and leaves when finished (6).

We assume that our aim is to simulate the system with only one service desk; thus events are activated only along Service desk 1: circles 1, 2, 4, 5, and 6. Fig. 16 shows the simulated part of the system, with bold boxes (red lines in the online version) indicating activity events (instead of creation and processing of an event).

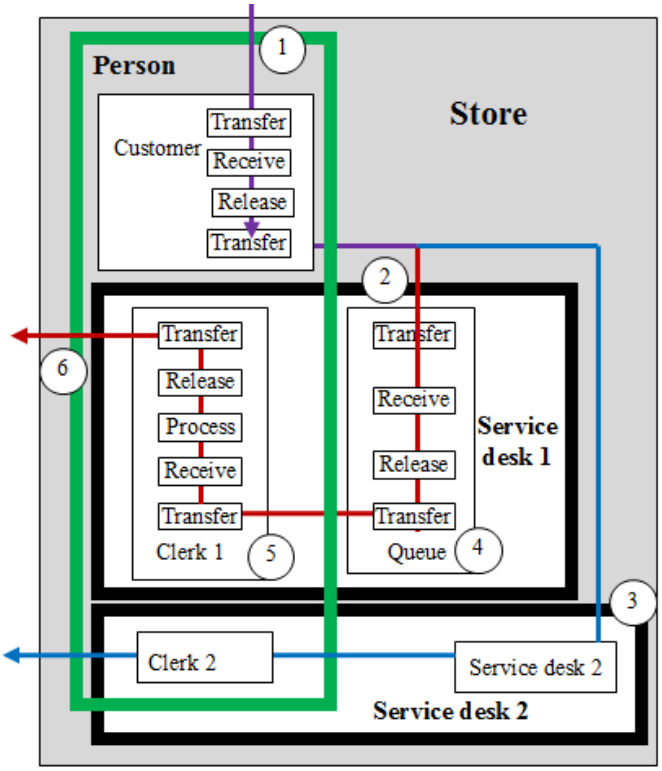

Fig. 15. FM representation

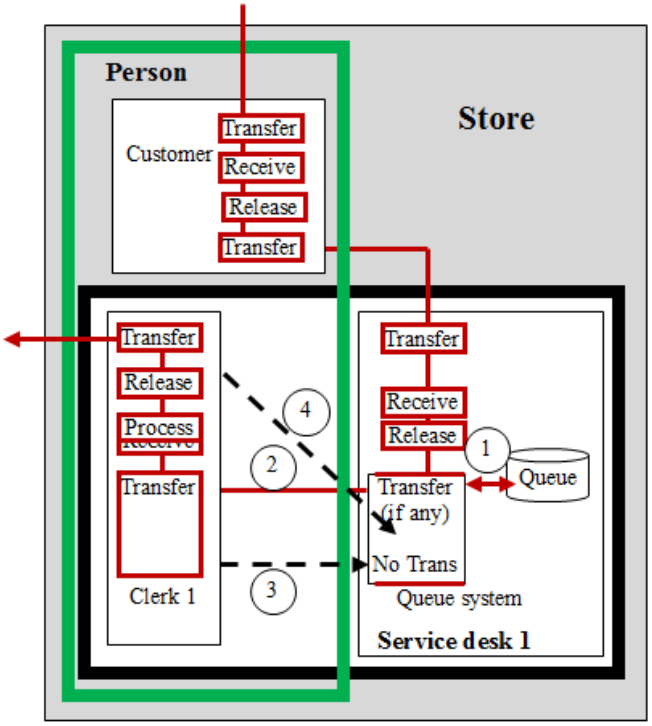

Fig. 16. Activating events of only Service Clerk 1

Active here refers to being in a state of liveliness and reactiveness, or in engineering terminology, in the ON state. An *event* is a thing that facilitates such active state by a "switching" or "firing" occurrence (Petri net terminology). A flow (create, process, …) begins with an event and may end with the beginning of another event. The event is what causes the flow of things. Note that Create is a type of flow.

Thus, in the figure, a person passes through various active stages. Here we combine the descriptive (static) level and the events level by assuming that each stage is event-active: it performs its function as soon as the thing flows through it. In the queue, the person proceeds to the clerk (2) and this transfer "blocks" any additional flow (3). When the customer leaves the clerk (is released and transferred), the next transfer to the clerk is permitted (4).

Fig. 17 shows the addition of generating random numbers of arrivals, between 1 and 8 (circle 1). The clerk's processing time (circle 2) can be added in similar fashion.

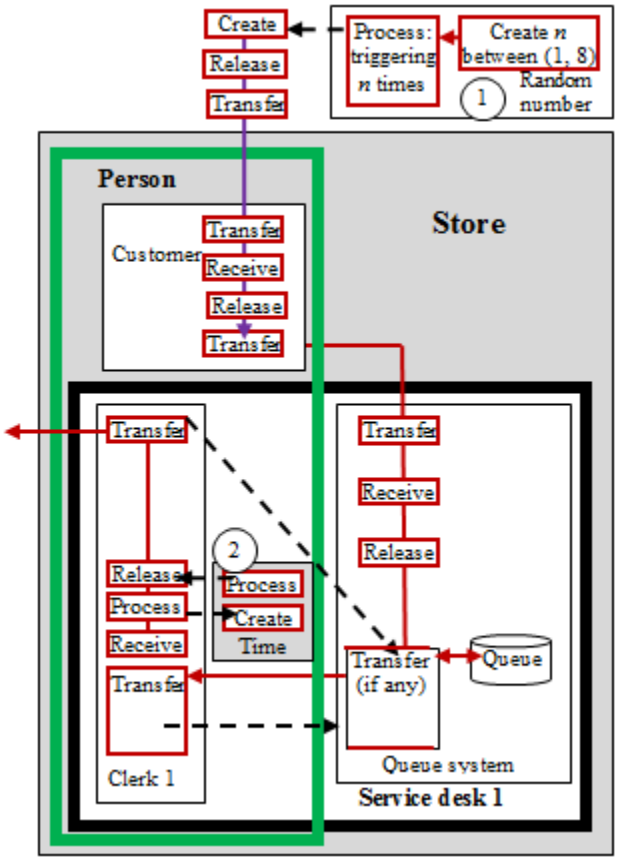

Fig. 17. Activating events of only Service Clerk 1

IX. Conclusion

As stated in the introduction, this paper proposes a new diagrammatic language for modeling representations in simulation and as a basis for a theoretical framework incorporating associated notions such as events and flows. Additionally, where events are used to define fine-grained activities, operational semantics are assimilated into the representation, resulting in an integration of the static domain description and the dynamic chronology of events.

We have shown that this unified specification facilitates such goals; nevertheless, the paper is still a theoretical proposal waiting to be realized in an actual simulation project. It is an attempt to point a direction substantiated by examples and re-

casting of work from the published literature. Furthermore, the paper seems to clarify some of the involved issues such as the two-level approach (domain description and events sequences) needed for simulation, and it also seems to enhance understanding of such notions as events and flows.

REFERENCES

[1] T. Grüne-Yanoff and P. Weirich, "The philosophy and epistemology of simulation: a review," Simulation & Gaming, vol. 41, no. 1, pp. 20–50, 2010.

[2] C. M. Overstreet, (1982). "Model Specification and Analysis for Discrete Event Simulation," PhD Dissertation, Department of Computer Science, Virginia Tech, Blacksburg, VA, December.

[3] W. Wang and R. Brooks, 2007. Empirical Investigations of Conceptual Modeling and the Modeling Process. In Proceedings of the 2007 Winter Simulation Conference, ed. S.G. Henderson, B. Biller, M.-H. Hsieh, et al., 762-770. Piscataway, NJ: IEEE Computer Society Press.

[4] G. Wagner, Information and Process Modeling for Simulation, Toturial, Jan 2015 · Proceedings - Winter Simulation Conference.

[5] A. A. Tako, K. Kotiadis, and C. Vasilakis, "A participative modeling framework for developing conceptual models in healthcare simulation studies," In Proceedings of Winter Simulation Conference, Baltimore (MD), USA, B. Johansson, S. Jain, J. Montoya-Torres, et al., Eds. 2010, pp. 500−512. http://www.informs-sim.org/wsc10papers/045.pdf

[6] A. M. Law, Simulation Modeling and Analysis. Boston: McGraw-Hill, 2007.

[7] S. Robinson, S. "Conceptual modeling for simulation: issues and research requirements," in Proceedings of the 2006 Winter Simulation Conference, L. F. Perrone, F. P. Wieland, J. Liuet, et al., Eds. Monterey, CA: Institute of Electrical and Electronic Engineers, 2006

[8] J. W. Brackett, "Formal specification languages: a marketplace failure; a position paper," in Proceedings of the 1988 IEEE International Conference on Computer Languages, Miami, FL, p. 161, October 9-13, 1988.

[9] B. P. Zeigler, (1976). Theory of Modelling and Simulation. New York: John Wiley and Sons, 1976.

[10] E. H. Page Jr., Simulation Modeling Methodology: Principles and Etiology of Decision Support. Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Blacksburg, September 1994.

[11] R. J. Paul, "Activity cycle diagrams and the three phase approach," in Proceedings of the 1993 Winter Simulation Conference, pp. 123–131, Los Angeles, CA, December 12-15, 1993.

[12] S. Al-Fedaghi and A. Alrashed, "Threat risk modeling," paper presented at International Conference on Communication Software and Networks (ICCSN 2010), Singapore, February 26–28, 2010.

[13] S. Al-Fedaghi, "Toward flow-based semantics of activities," Int. J. Softw. Eng. Appl., vol. 7, no. 2, pp. 171-182, 2013.

[14] S. Al-Fedaghi and B. Al-Babtain, "Modeling the forensics process," Int. J. Secur. Appl., vol. 6, no. 4, 2012.

[15] S. Al-Fedaghi, "Awareness of context and privacy," Am. Soc. Inform. Sci. Tech. (ASIS&T) Bull., vol. 38, no. 2, 2011.

[16] S. Al-Fedaghi, "Typification-based ethics for artificial agents," paper presented at Second IEEE International Conference on Digital Ecosystems and Technologies (IEEE-DEST 2008), Phitsanulok , Thailand, February 26–29, 2008.

[17] E. Geva, Facilating Reading Comprehension through Flowcharting. University of Illinois, Urbana-Champain, July 1981. https://www.google.com.kw/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjM5ea5jM7OAhUHlxQKHeyiBewQFggfMAA&url=https%3A%2F%2Fwww.ideals.illinois.edu%2Fbitstream%2Fhandle%2F2142%2F18065%2Fctrstreadtechrepv01981i00211_opt.pdf%3Fsequence%3D1&usg=AFQjCNEXx3Z9X8TVvl4G_v4Glbct3AL0Fg&bvm=bv.129759880,d.d24

[18] R. E. Nance, "The time and state relationships in simulation modeling," Comm. ACM, vol. 24, no. 4, pp.173- 179, 1981.

[19] R. McKee, Story: Substance, Structure, Style, and the Principles of Screen Writing. ReganBooks, 1997. ISBN: o-o6-o39168-5.

[20] J.-L. Marion, Being Given: Toward a Phenomenology of Givenness, Jeffrey L. Koskey (trans.). Stanford, CA: Stanford University Press, 2002.

[21] J.-L. Marion, Reduction and Givenness: Investigations of Husserl, Heidegger, and Phenomenology, Thomas A. Carlson (trans.). Evanston, IL: Northeastern University Press, 1998.

[22] G. C. Butchart, "An excess of signification: or, what is an event?" Semiotica, vol. 187, no. 1/4, pp. 291–307, 2011.

[23] A. Badiou, Ethics: An Essay on the Understanding of Evil, Peter Hallward (trans.). London: Verso, 2001.

[24] A. Podgurski and L. A. Clarke, "The implications of program dependences for software testing, debugging, and maintenance," ACM SIGSOFT Softw. Eng. Notes, vol. 14, no. 8, pp. 168-178, December 1989. DOI: 10.1145/75309.75328

[25] S. Al-Fedaghi, "Conceptual understanding of computer program execution: application to C++," Int. J. Comp. Sci., vol. 10, no. 2, March 2013.

[26] C. D. Pegden, "Advanced tutorial: overview of simulation world views," in Proceedings of Winter Simulation Conference, Baltimore (MD), USA, B. Johansson, S. Jain, J. Montoya-Torres, et al., Eds., pp. 643−651, 2010.