BUSINESS PROCESS MODELING AND SIMULATION WITH DPMN: PROCESSING ACTIVITIES

Gerd Wagner

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

ABSTRACT

The Business Process Modeling Notation (BPMN) has been established as a modeling standard in Business Process (BP) Management. However, BPMN lacks several important elements needed for BP simulation and is not well-aligned with the Queueing Network paradigm of Operations Research and the related BP simulation paradigm pioneered by the Discrete Event Simulation (DES) languages/tools GPSS and SIMAN/Arena. The Discrete Event Process Modeling Notation (DPMN) proposed by Wagner (2018) is based on Event Graphs (Schruben 1983), which capture the DES paradigm of Event-Based Simulation. By allowing to make flowchart models of queueing/processing networks with a precise semantics, DPMN reconciles (the flowchart approach of) BPMN with DES. DPMN is the first visual modeling language that supports all important DES approaches: event-based simulation, activity-based DES and Processing Network models, providing a foundation for harmonizing and unifying the many different terminologies/concepts and diagram languages of established DES tools.

1 INTRODUCTION

The term *Discrete Event Simulation (DES)* has been established as an umbrella term subsuming various kinds of computer simulation approaches, all based on the general idea of modeling the dynamics of a system as a series of (explicit or implicit) events that change the system's state over time.

In the DES literature, it is often stated that DES is based on "entities flowing through the system". While this narrative applies to a DES paradigm called "Process Modeling" by Pegden (2010), it characterizes a special (yet important) kind of DES only, and it does not apply to all discrete event systems. As we explain below, the "Process Modeling" paradigm should be better called *Processing Network (PN)* paradigm, since it is not about process modeling in general, but only about modeling a particular kind of discrete processes that happen in PNs (which generalize queuing networks). It has been pioneered by GPSS (Gordon 1961) and SIMAN/Arena (Pegden and Davis 1992) and is implemented in various forms by all modern off-the-shelf simulation software products.

Today, after a history of more than 50 years, the field of DES is fragmented into many different paradigms and formalisms, based on different concepts and terminologies. Unlike other scientific fields, it didn't achieve much conceptual unity regarding its foundations, which would be crucial for facilitating scientific progress.

The lack of a scientifically established conceptual foundation of DES and an accompanying standard terminology is witnessed in the diversity of terminologies used in DES software packages. Typically, DES practitioners are locked into the terminology (and implementation idiosyncrasies) of the software package they use, and often not aware of the general, technology-independent concepts.

Pegden (2010) explains that the history of DES has been shaped by three fundamental paradigms: Markowitz, Hausner, and Karr (1962) pioneered *Event-Based Simulation (ES)* with *SIMSCRIPT*, Gordon (1961) pioneered *Processing Network Simulation (PNS)* with *GPSS*, and Dahl and Nygaard (1967) pioneered *Object-Orientation (OO)* and the computational concept of *co-routines* for asynchronous programming with their simulation language *Simula*.

Notice, however, that OO does not represent a DES paradigm, but rather an information/data modeling paradigm for conceptual modeling and software design modeling. In fact, the Simula paradigm is characterized by a combination of OO modeling and using co-routines for implementing the dynamics of a discrete system (to be simulated) without an explicit computational concept of events in a way that is sometimes called 'process interaction' approach.

According to Pegden (2010), ES has been widely used during the first 20 years of simulation, due to its great flexibility allowing to efficiently model a wide range of complex systems. Later, however, the PNS paradigm, implemented by tools like Arena, Simul8, FlexSim, Simio and AnyLogic, became the dominant approach in practical applications of simulation because it is based on the higher-level concept of *processing activities* and allows no-code (or low-code) simulation engineering with graphical user interfaces and appealing visualizations.

After OO had been established as the predominant paradigm in software engineering in the 1990s, it was also adopted by many simulation tools, which weaved it into their PNS approach. Pegden (2010) remarks that "Many process and object based simulation tools maintain an event capability as a 'backdoor' for flexibility". Consequently, modern DES tools allow combining PN models with OO modeling and event scheduling.

In addition to the two important DES modeling paradigms of ES and PNS (typically combined with OO), several DES formalisms have been proposed, notably the Petri-Net-based *Activity Cycle Diagrams* by Tocher (1960), the *Discrete Event System Specification (DEVS)* by Zeigler (1976) and *Event Graphs* by Schruben (1983). Also, using the classical mathematical/computational formalisms of Petri Nets and Markov Chains for DES modeling has been proposed.

While the abstract formalism of DEVS defines its own complex systems simulation paradigm, based on the theoretical computer science concepts of *finite state machines* and *transition systems*, Event Graphs define a diagram language for specifying ES models and their formal semantics provides a formal semantics of ES.

As argued by Pegden (2010), ES is the most fundamental DES paradigm since the other paradigms also use events, at least implicitly. However, Pegden does not explain in which way the other paradigms are built upon ES. We show how to extend ES by adding concepts like objects and activities resulting in *Object Event (OE) Modeling and Simulation (M&S)* proposed by Wagner (2018) as a new general DES paradigm based on the two most important ontological categories: *objects* and *events* (Guizzardi and Wagner 2010).

OEM&S combines OO modeling with the event scheduling approach of ES. The relevant *object types* and *event types* are described in an information model, which is the basis for making a process model. A modeling approach that follows the OE Modeling (OEM) paradigm is called an *OEM approach*. Such an approach needs to choose, or define, an information modeling language (such as *Entity Relationship Diagrams* or *UML Class Diagrams*) and a process modeling language (such as *UML Activity Diagrams* or *BPMN Process Diagrams*).

OE Simulation (OES) is a conservative extension of ES. While ES defines the system state structure in the form of a set of *global variables*, OES defines it in the form of a set of *objects* (or *object states*), such that their attributes take the role of state variables.

In (Wagner 2018), we have introduced a variant of the *Business Process Modeling Notation (BPMN)*, called *Discrete Event Process Modeling Notation (DPMN)*, and have shown how an OEM approach based on *UML Class Diagrams* and *DPMN Process Diagrams* for making basic OE models allows defining a set of object types OT, a set of event types ET, and a set of event rules R. In (Wagner 2017), we have shown that (a) these three sets define a state transition system, where the state space is defined by OT and ET, and the transitions are defined by R, and (b) such a transition system represents an *Abstract State Machine* (Gurevich 1985). This fundamental characterization of an OE simulation model provides a formal semantics for OES by defining an *OES formalism* that any OE simulator has to implement.

Since OEM&S accommodates the concepts of *resource-constrained activities* and *processing activities*, as shown in Section 3, it integrates important DES concepts and supports modeling general forms of (BPMN-style) *Activity Networks* and (GPSS-style) *Processing Networks*, which extend Activity Networks by adding *processing objects* flowing through the network.

2   FROM EVENT GRAPHS TO ACTIVITY NETWORKS

In this section, summarizing (Wagner 2020), we show how to incrementally extend Event Graphs by adding increasingly high-level modeling concepts: in the first step, we add the concept of objects, resulting in basic OEM/DPMN diagrams, and in the second step, we add the concept of (resource-constrained) activities, resulting in the *Activity Network* modeling language OEM/DPMN-A.

2.1   Event-Based Simulation and Object Event Simulation

Event-Based Simulation (ES) uses *state variables* for modeling a system's state and *event scheduling* with a *Future Events List* for modeling its dynamics. A technology-independent definition of ES is provided by Event Graphs, which define graphically how an event triggers state changes (in the form of possibly conditional variable value assignments) and follow-up events.

The ES model shown in Figure 1 defines (a) two state variables: *L* for the length of an arrival queue and *B* for a performer being busy or not, as well as (b) three event variables representing *Arrival*, *ProcessingStart* and *ProcessingEnd* events, in the form of circles. In addition, it defines the sequencing of events of those types, together with caused state changes, in the form of possibly conditional variable assignments (underneath event circle names).
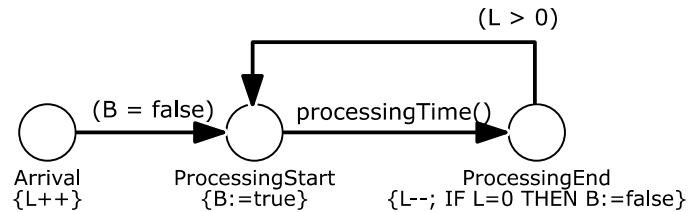

Figure 1: An Event Graph defining an ES model with two state variables and three event types.

Event Graphs provide a visual modeling language with a precise semantics that captures the fundamental event scheduling paradigm. However, Event Graphs are a rather low-level DES modeling language: they lack a visual notation for (conditional and parallel) branching, do not support OO state structure modeling (with attributes of objects taking the role of state variables) and do not support the concept of activities.

In the proposed OEM&S approach, object types and event types are modeled as special categories of classes in a special kind of UML Class Diagram, called *OE Class Model*. *Random variables* are modeled as a special category of class-level operations constrained to comply with a specific probability distribution such that they can be implemented as static methods of a class. *Queues* are not modeled as objects, but rather as ordered association ends, which can be implemented as collection-valued reference properties. Finally, *event rules*, which include *event routines*, are modeled in DPMN process diagrams (and possibly also in pseudo-code), such that they can be implemented in the form of special *onEvent* methods of event classes.

Notice that in the OE class model shown in Figure 2, events of both types, arrivals and departures, are associated with a service desk (as their only participant). This is the service desk where these events happen. While the *ServiceDesk* object class defines a random variable *serviceTime*, the Arrival event class defines a random variable *recurrence*.
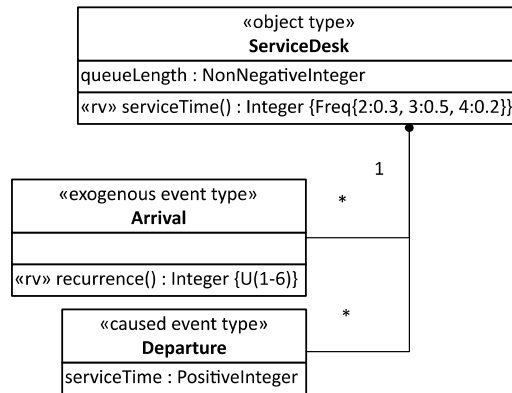
Figure 2: An OE class model defining an object type and two event types.

A process model is based on an underlying information model defining the types of its objects and events. The process model shown in Figure 3 is based on the OE class model shown in Figure 2.
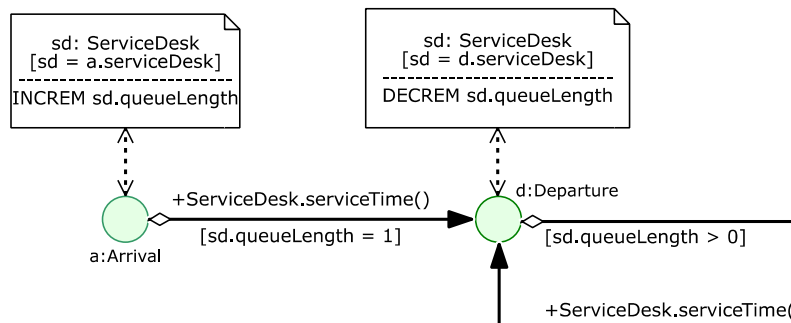

Figure 3: A basic DPMN process model based on the OE class model of Figure 2.

Basic DPMN Process Diagrams (without activities) are a conservative extension of Event Graphs. This means that a basic DPMN process diagram can be transformed to an event graph preserving its dynamics by replacing its objects with corresponding sets of state variables.

Like Petri Nets and DEVS, OES has a formal semantics. But while Petri Nets and DEVS are abstract computational formalisms without an ontological foundation, OES is based on the ontological categories of objects, events and causal regularities.

DPMN consists of three layers. The first layer, basic DPMN, corresponds to an extension of Event Graphs by adding the concept of *Objects*. The second layer, DPMN-A, adds the concept of *Resource-Constrained Activities*, while the third layer, DPMN-PN, adds the concepts of *Processing Activities* and *Processing Networks*.

2.2 Activity-Based Discrete Event Simulation

In (Wagner 2020), we have shown how to extend basic OEM/DPMN by adding support for resource-constrained activities, resulting in an extension, OEM/DPMN-A, comprised of three new information modeling elements (Activity Type, Resource Role, Resource Pool) and two new process modeling elements (Activity and Resource-Dependent Activity Scheduling Arrow). DPMN-A diagrams allow modeling *Activity Networks*.

Conceptually, an activity is a composite event with a non-zero duration that is composed of, and temporally framed by, a pair of instantaneous start and end events. In Figure 4 below, the pair of *ProcessingStart* and *ProcessingEnd* events is replaced with a corresponding *Processing* activity. This replacement pattern constitutes the semantics of activities in DPMN by reduction to a pair of corresponding start and end events.
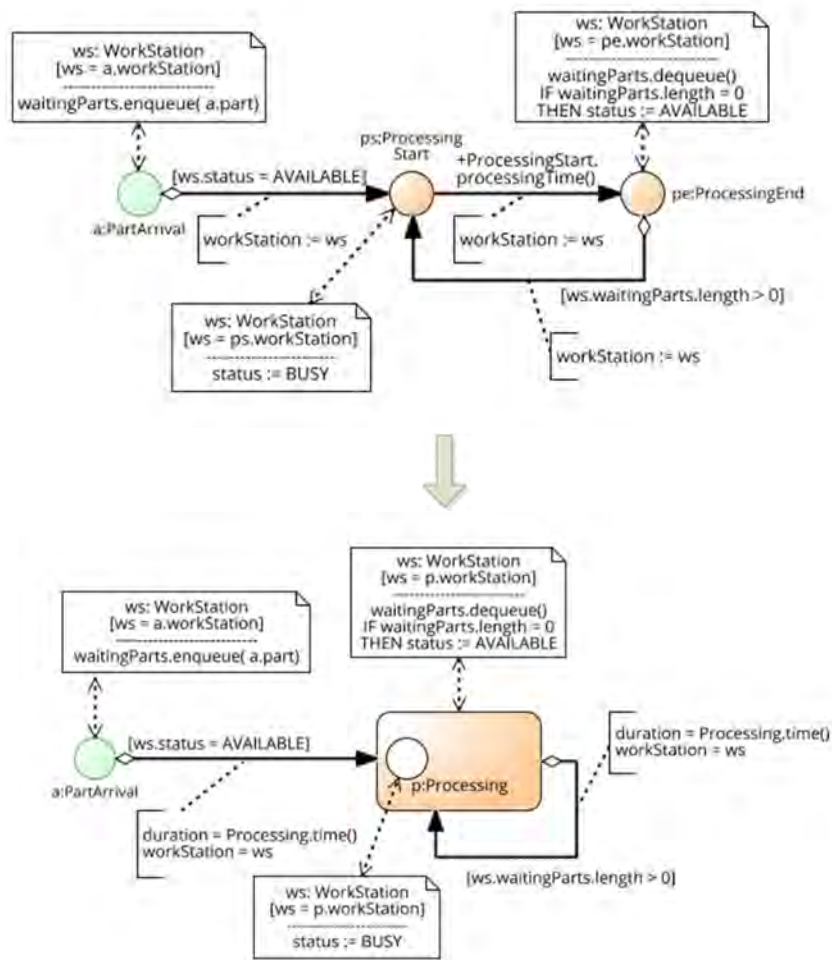
Figure 4: Introducing a simple Activity for replacing a pair of corresponding start and end events.

Activities may require certain resource objects for being performable. At any point in time, a resource required for performing an activity may be *available* or not. A resource is not available, for instance, when it is *busy* or when it is *out of order*.

For instance, a loading activity may require two resources: a truck to be loaded and a wheel loader as the ***performer*** of the loading activity. Such resource requirements are defined in an OE class model where we would express two ***resource role*** associations for the activity type *Loading*: one with the object type *Truck* and another one with the object type *WheelLoader*, the former one with a ***resource cardinality constraint*** of 1, stating that exactly one truck is required, and the latter one with a resource cardinality constraint of 1..2, stating that at least one wheel loader is required, and at most two wheel loaders can be used, for loading a truck.

Typically, a resource-constrained activity is a component of a ***business process*** that happens in the context of an organization or organizational unit, which is associated with the activity as its *process owner*.

Modeling resource-constrained activities has been a major issue in DES since its inception in the nineteen-sixties, while it has been neglected and is still considered an advanced topic in the field of Business Process Modeling. For instance, while BPMN allows assigning resources to activities, it does not allow modeling resource pools, and does neither allow specifying resource cardinality constraints.

In the PNS paradigm, Gordon (1961) has introduced the resource management operations *Seize* and *Release* in the simulation language GPSS for allocating and de-allocating (or releasing) resources. Thus, GPSS has established a standard modeling pattern for resource-constrained activities, which has become popular under the name of *Seize-Delay-Release* indicating that for simulating a resource-constrained activity, its resources are first allocated, and then, after some delay (representing the duration of the simulated activity), they are de-allocated (released).

For all types of resource-constrained activities, a simulator can automatically collect the following statistics:

1. *Queue throughput* statistics: the numbers of enqueued and de-queued tasks, and the numbers of started and completed activities.
2. *Queue length* statistics (average, maximum, etc.) of their task queues.
3. *Cycle time* statistics (average, maximum, etc.), where cycle time is the sum of the waiting time and the activity duration.
4. *Resource utilization* statistics: the percentage of time each resource object involved is busy with an activity of that type.

In addition, a simulator can automatically collect the percentage of time each resource object involved is idle or out-of-order.

While a follow-up event can be scheduled to start immediately, a follow-up activity can only be scheduled by adding a new planned activity, or task, to a task queue, and started as soon as all required resources are available. This consideration is the basis for introducing *Resource-Dependent Activity Scheduling (RDAS)* arrows allowing to simplify the activity scheduling pattern as shown in Figure 5.
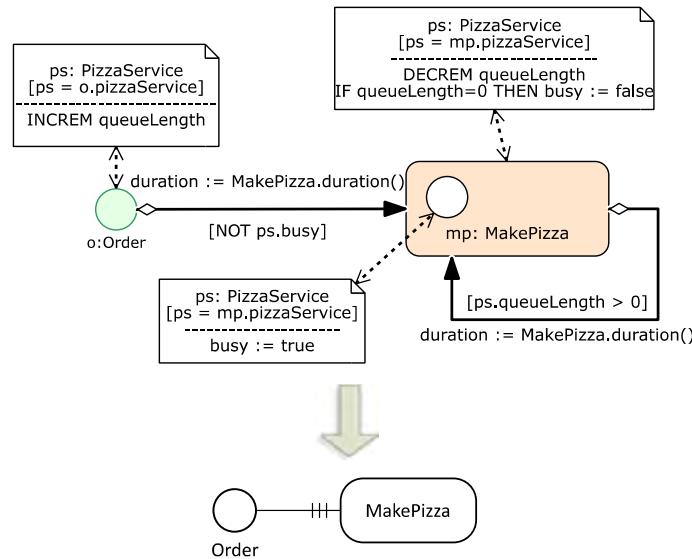

Figure 5: Introducing a Resource-Dependent Activity Scheduling arrow between *Order* and *MakePizza*.

Notice that while the queuing logic for MakePizza tasks is expressed explicitly (with the help of the *queueLength* and the *busy* variables) in the upper diagram of Figure 5, it is implied by the RDAS arrow in the lower diagram according to the resource roles and resource cardinality constraints defined in the underlying OE class diagram.

2.3   BPMN-Based Simulation is Severely Limited

In the Information Systems area of *Business Process Management*, the *Business Process Modeling Notation (BPMN)* has been established as a modeling standard that is widely used by many kinds of

organizations for documenting existing, and for designing new, business processes. It is also a goal, to simulate the business processes described by a BPMN process model. However, since BPMN has not been designed as a simulation language, several important elements needed for simulation are not included in a BPMN process model and therefore have to be defined on top of it.

A common approach taken by several academic simulators (QBP 2021, Pufahl et al 2018), DES vendors (Simul8, Lanner L-Sim) and BPMN vendors (e.g., SAP/Signavio) is to ask for the following additional items: the arrival rate for new cases, capacities of the activity performer resource pools represented by lanes (and possibly schedules for the pools' resources), activity durations, and conditional branching probabilities for gateways.

The last item points to the first limitation. Conditional branching is not modeled on the basis of state variables, but only in the abstract way of using (hard-to-guess) branching probabilities. The BPMN-based simulation approach has even more severe limitations with respect to resource management:

- It allows only one performer resource per activity while many real-world activities require several other resources, in addition to the performer (e.g., a medical examination activity may require a room, a doctor, and a nurse).
- It does not support important resource allocation methods such as alternative resource pools or pre-emption.

Consequently, BPMN-based business process simulation is inferior to DPMN-A-based Activity Network simulation (and PN-based business process simulation), especially if the simulator is based on BPMN's artificial "token flow" semantics instead of ES semantics.

2.4   Conceptual Foundations of Discrete Event Simulation

There is a lot of conceptual confusion in the field of DES. First of all, the term "discrete event simulation" is not well-defined. While many DES textbooks avoid defining it at all, others only define it in a very rudimentary way.

For instance, Banks et al (2010) define that DES is "the modeling of systems in which the state variable changes only at a discrete set of points in time". Remarkably, this definition does not even mention the concept of events, possibly for accommodating DES approaches that do not have explicit events, such as Petri Nets, Activity Cycle Diagrams and DEVS.

While some authors, such as Pegden (2010), distinguish between three fundamental DES approaches: ES, PNS and OO, others, such as Banks (1998), distinguish between four approaches: ES, PNS (called "process-interaction"), Activity Scanning and the Three-Phase Method. This disagreement about fundamental concepts, and the different terms used by different authors for the same concepts (e.g., the PNS paradigm has the following names in the DES literature: "process-oriented", "process-based", "process-interaction", "process-centric"), must be very confusing for students of and beginners in DES.

We follow the view of Pegden (2010) that ES is the most fundamental DES paradigm, although in many DES textbooks, e.g., in (Banks et al 2010), this is not explained. Adopting ES as the most fundamental DES paradigm implies that other paradigms should extend it in a conservative manner such that its basic concepts (and their semantics) are preserved. The basic concepts of ES models and Event Graphs, with hints about their use in diagrams enclosed in brackets, are :

1. (typed) *state variables* [used in the state change annotations of event circles and in condition annotations of event scheduling arrows],
2. (typed) *event variables* representing events of some type [visually expressed by event circles, with the event type name shown underneath the circle],
3. *state changes* in the form of state variable value assignments [shown in the state change annotations enclosed in braces underneath event circles],
4. *event rules* as transition functions (defining the state changes and follow-up events caused by an occurrence of an event of a certain type) [visually expressed by event scheduling arrows].
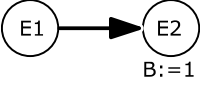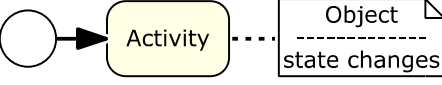
Basic OEM/DPMN extends this set of basic concepts of ES by adding the following concepts:

5. ***object types*** [visually expressed as stereotyped class rectangles in an OE class diagram],
6. ***event types*** [visually expressed as stereotyped class rectangles in an OE class diagram],
7. a categorization of event types as either *exogenous* (with a ***recurrence*** function that is typically a random variable) or *endogenous/caused* [marked with corresponding stereotypes in an OE class diagram],
8. (typed) ***object variables*** representing objects of some type participating in events [object variables are declared in object rectangles attached to event circles in a DPMN diagram],
9. object-oriented ***state change statements*** [shown in object rectangles in a DPMN diagram].

The *Activity Network* modeling language OEM/DPMN-A extends this set of basic concepts by adding the following concepts, as illustrated by Table 1:

10. ***activity types*** [visually expressed as stereotyped activity class rectangles in an OE class diagram],
11. ***resource roles*** [visually expressed in the form of stereotyped association ends in an OE class diagram],
12. ***resource cardinality constraints*** [visually expressed in the form of association end multiplicities in an OE class diagram],
13. ***resource pools*** [visually expressed in the form of stereotyped association ends in an OE class diagram],
14. (typed) ***activity variables*** representing activities of some type [visually expressed by activity rectangles with rounded corners in a DPMN diagram],
15. ***activity flows*** [visually expressed by resource-dependent activity scheduling arrows in a DPMN diagram].

Table 1: From Event Graphs to Activity Networks.

	Layer	Elements/Concepts	Diagrams
Event-Based Simul.	Event Graphs (Schruben 1983)	Event Circles, Event Scheduling Arrows, Variable Value Assignments	E1 → E2 B:=1
Object Event Simulation (OES)	Basic DPMN	+ Objects w/ State Changes	○ → ○ ┈ Object state changes
	DPMN with Simple Activities	+ Activities	○ → Activity ┈ Object state changes
	DPMN with Resource-Constrained Activities (DPMN-A) ***Activity Networks***	+ Resource Roles + Resource Cardinality Constraints + Resource Pools + Resource-Dependent Activity Scheduling Arrows	E1 ⊢⊢⊢ Activity → E2

The *Processing Network* modeling language OEM/DPMN-PN, presented in the next section, extends the set of basic concepts of Activity Networks by adding the following concepts:

16. ***processing activity types*** [visually expressed as stereotyped activity class rectangles in an OE class diagram],
17. ***entry nodes*** consisting of an ***entry station object*** and a (typed) ***arrival event variable*** [visually expressed by a special object shape in a DPMN diagram],
18. ***processing nodes*** consisting of a ***processing station object*** and a (typed) ***processing activity variable*** [visually expressed by a special object shape in a DPMN diagram],
19. ***exit nodes*** consisting of an ***exit station object*** and a (typed) ***departure event variable*** [visually expressed by a special object shape in a DPMN diagram],
20. ***processing flows*** combining an activity flow and an object flow [visually expressed by processing flow arrows in a DPMN diagram].

3   PROCESSING ACTIVITIES AND PROCESSING NETWORKS

The concept of discrete *Processing Networks* is a generalization of the Operations Research concept of Queueing Networks. A *Processing Object* enters a processing network via an *Arrival* event at an *Entry Station*, is subsequently routed along a chain of *Processing Stations* where it is subject to *Processing Activities*, and finally exits the network via a *Departure* event at an *Exit Station*. In typical definitions of a queueing network, the processing station is the only required resource of the processing activities performed at that station, while in a processing network, processing activities can have many required (and optional) resources of various types being allocated with various methods.

In general, there are also continuous processing processes (with continuous processing flows) happening in continuous processing networks (to be modeled with differential equations). Since we are only interested in discrete systems in this article, we simply say *Processing Network (PN)* and *processing process* and omit the qualifying adjective "discrete".

PNs have been investigated in *operations management* (Loch 1998) and in the mathematical theory of queuing (Williams 2016), and have been the application focus of most industrial simulation software products, historically starting with GPSS (Gordon 1961) and SIMAN/Arena (Pegden and Davis 1992). They allow modeling many forms of *discrete processing processes* as can be found, for instance, in the manufacturing and services industries.

In the field of DES, PNs have often been characterized by the narrative of "entities flowing through a system". In fact, while in Activity Networks (DPMN-A), there is only a flow of events (including activities), in Processing Networks (DPMN-PN), this flow of events is over-laid with a flow of (processing) objects.

It is remarkable that the PN paradigm has dominated the DES software market since the 1990s and still flourishes today, often with object-oriented and "agent-based" extensions. Its dominance has led many simulation experts to view it as a synonym of DES, which is a conceptual flaw because the concept of DES, even if not precisely defined, is clearly more general than the PN paradigm.

The PN paradigm has often been called a "process-oriented" DES approach. But unlike the business process modeling language BPMN, it is not concerned with a general concept of *business process models*, but rather with the more special concept of *processing process models*. A processing process includes the simultaneous handling of several "cases" (processing objects) that may compete for resources or have other interdependencies, while a "business process" in Business Process Management has traditionally been considered as a case-based process that is isolated from other cases.

3.1   Processing Activities

A *Processing Activity* is a resource-constrained activity that is performed at a processing station and takes one or more objects as inputs and processes them in some way (possibly transforming them from one type of object to another type), creating one or more objects as outputs. The processed objects have been called "transactions" in GPSS and "entities" in SIMAN/Arena, while they are called *Processing Objects* in DPMN.

Ontologically, there are one or more objects participating in a processing activity, as shown in Figure 6. Some of them may represent resources, while others may represent processing objects.

Processing activities typically require immobile physical resources, like rooms or workstation machines, which define the inner nodes of a PN.
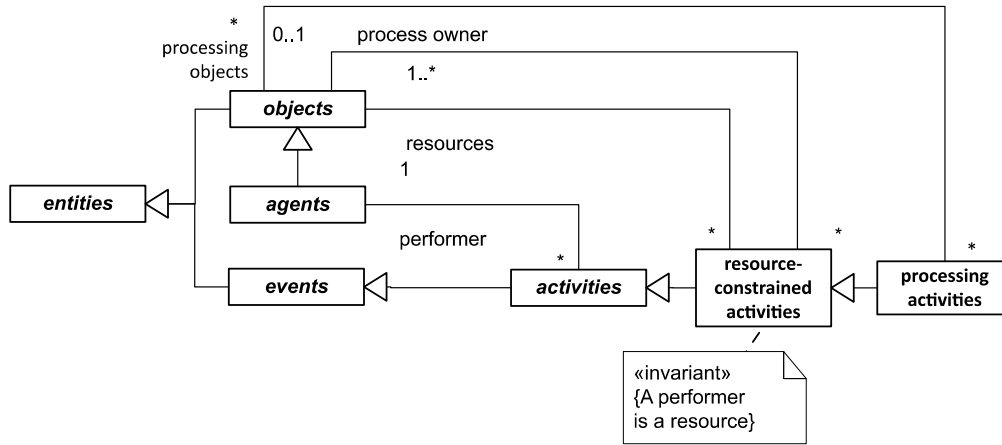

Figure 6: The concept of Processing Activities.

Each node in a PN model represents both an object and a typed event variable. An *Entry Node* represents both an *Entry Station* (e.g., a reception area or an entrance to an inventory) and an *Arrival* event variable. A *Processing Node* represents both a *Processing Station* (e.g., a workstation or a room) and a processing activity variable. An *Exit Node* represents both an *Exit Station* and a *Departure* event variable. A *Processing Flow* arrow connecting two processing nodes represents both an event flow and an object flow. Thus, the node types and the flow arrows of a PN are high-level modeling concepts that are overloaded with two meanings.

The (entry, processing and exit) stations of a PN define locations in a network space, which may be based on a two- or three-dimensional Euclidean space. Consequently, OEM-PN models are spatial simulation models, while basic OEM and OEM-A allow to abstract away from space. When a processing object is routed to a follow-up processing station, it moves to the location of that station. The underlying space model allows visualizing a PN simulation in a natural way with processing objects as moving objects.

A PN modeling language should have elements for modeling each of the three types of nodes. Consequently, DPMN-A has to be extended by adding new visual modeling elements for entry, processing and exit nodes, and for connecting them with processing flow arrows.

The simulation modeling concepts of the PN paradigm have been adopted by most DES software products, including Simul8, Simio and AnyLogic. However, each of these products uses its own variants of the PN concepts, together with their own proprietary terminology and proprietary diagram language, as illustrated by Table 2.

Table 2: Different terminologies used for the same PN modeling concepts.

OEM/DPMN	Arena	Simul8	Simio	AnyLogic
Processing Object	Entity	Work Item	Token	Agent
Entry Node	Create	Start Point	Source	Source
Processing Node	Process	Queue+Activity	Server	Service or Seize+Delay+Release
Exit Node	Dispose	End Point	Sink	Sink

The use of such proprietary terminologies and diagram languages makes it hard for simulation beginners to learn how to use a tool and for expert users of a tool to switch, or interchange models, from their tool to another one. Notice especially the strange term "Agent" used by AnyLogic, instead of the Arena term "Entity", for processing objects like manufacturing parts in production systems or patients

in hospitals. It is confusing to call a manufacturing part, such as a wheel in the production of a car, an "agent".

For PNs, a simulator can automatically collect the following statistics, in addition to the Activity Network statistics described in Section 2.2:

1. The number of processing objects that arrived at, and departed from, the system.
2. The number of processing objects in process (that is, either waiting in a queue/buffer or being processed)
3. The average time a processing object spends in the system (also called *throughput time*).

During a simulation run, it must hold that the number of processing objects that arrived at the system is equal to the sum of the number of processing objects in process and the number of processing objects that departed from the system, symbolically:

$$arrived = in\text{-}process + departed$$

3.2   Conceptual Modeling of Processing Networks

For accommodating PN modeling, OEM-A is extended by adding pre-defined types for processing objects, entry nodes, arrival events, processing nodes, processing activities, exit nodes and departure events, resulting in *OEM-PN*. These built-in types, which are described in Figure 7, allow making PN models simply by making a process model (with DPMN) without the need of making a separate OE class model as its foundation, as shown in Figure 8.
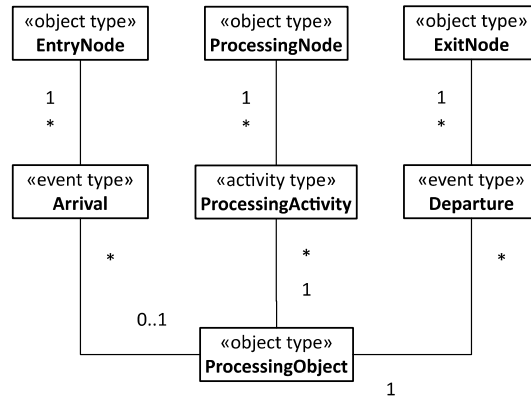

Figure 7: Built-in types for PN modeling in OEM-PN class diagrams.

DPMN is extended by adding the new modeling elements of *Entry Node*, *Processing Node* and *Exit Node* rectangles as well as *Processing Flow* arrows, representing combined object-event flows. Node rectangles take the form of stereotyped UML object rectangles, while Processing Flow arrows have a special arrow head consisting of a circle and three bars, as shown in Figure 8. Notice however, that this is just a preliminary visual syntax that may be improved in future work.

As a simple example of a PN model we consider a Department of Motor Vehicles (DMV) with two consecutive service desks: a reception desk and a case handling desk. When a customer arrives at the DMV, she first has to enqueue at the reception desk where data for her case is recorded. The customer then goes to the waiting area and waits for being called by the case handling desk where her case will be processed. After completing the case handling, the customer leaves the DMV via the exit.
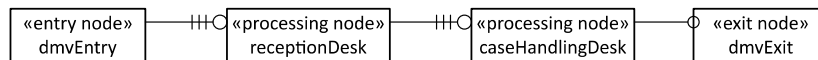

Figure 8: A PN model of the DMV with node rectangles and Processing Flow arrows.

Customer arrivals are modeled with an «entry node» element (with name "DMV entry"), the two consecutive service desks are modeled with two «processing node» elements, and the departure of customers is modeled with an «exit node» element (with name "DMV exit").

Since each node of a DPMN-PN process diagram implicitly defines both a station object and a typed event/activity variable, the diagram shown in Figure 8 is equivalent the modeling elements explicitly shown Figure 9. The station objects and the event/activity variables are elements of a running simulation.
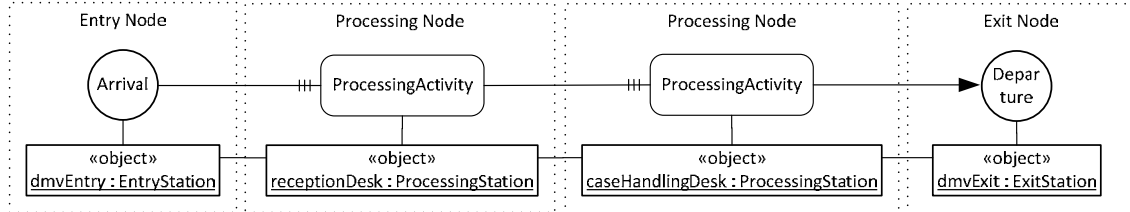

Figure 9: The elements defined implicitly by the DMV model of Figure 8.

While a Processing Flow arrow between two nodes implies both a flow of the processing object to the successor node and the resource-dependent scheduling of the next processing activity, an Event Scheduling arrow from a processing node to an Event circle represents an event flow where a processing activity end event causes/schedules another event, as illustrated in the example of Figure 10.
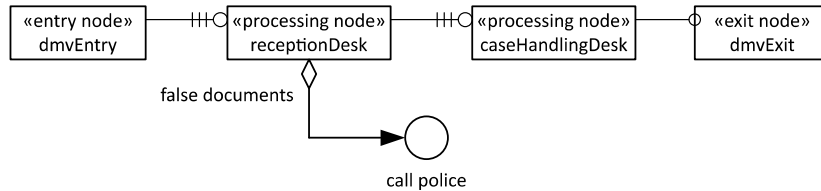

Figure 10: A DPMN-PN process diagram with an Event Scheduling arrow.

3.3   Processing Network Design Models

For accommodating PN modeling, OEM-A is extended by adding pre-defined types for processing objects, entry node objects, arrival events, processing node objects, processing activities, exit objects and departure events, as described in Figure 11, resulting in *OEM-PN*.
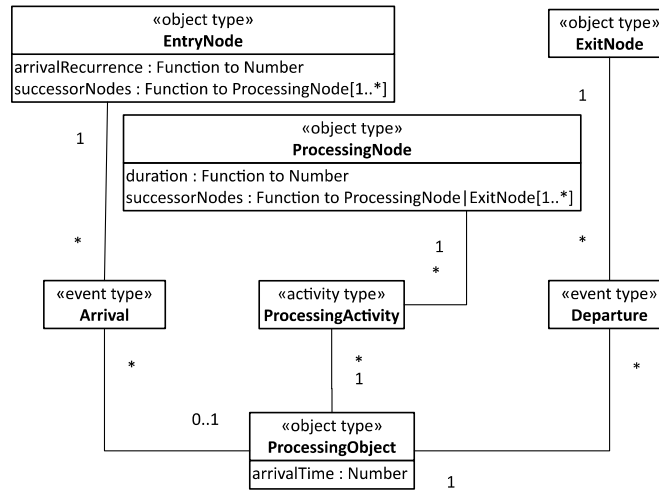

Figure 11: Built-in types for making PN design models.

In simple cases, these built-in types allow making PN models based on them without first defining custom types of processing objects, arrival events and processing activities, simply by making a process model with DPMN without the need of making an OE class model as its foundation, as shown in Figure 12.

Notice that the range of the properties *arrivalRecurrence*, *successorNodes* and *duration* of the built-in object types *EntryNode* and *ProcessingNode* is Function, which means that the value of such a property for a specific node is a specific function. While the standard UML semantics does not support such an extension of the semantics of properties in the spirit of the *Functional Programming* paradigm, its implementation in a functional OO programming language like JavaScript, where objects can have instance-level functions/methods, is straightforward.

The property *successorNodes* allows to express a function that provides, for any given entry or processing node, a (possibly singleton) set of processing nodes or exit nodes. Such a function can express several cases of routing a processing object from a node to one or more successor nodes:

1. a fixed unique successor node for modeling a series of processing nodes connected by (possibly conditional) processing flow arrows, as in Figure 13;
2. a conditional unique successor node for modeling an Exclusive (XOR) Gateway leading to one of several possible successor nodes;
3. a variable subset of a set of potential successor nodes for modeling an Inclusive (OR) Gateway;
4. a fixed set of successor nodes for modeling a Parallel (AND) Gateway;

In a DPMN diagram, the set of successor nodes of a node is defined by Flow Arrows, possibly in combination with Gateways.

3.3.1 PN example 1: a Single Workstation

Part arrivals are modeled with an «entry node» element (with name "partEntry"), the workstation is modeled with a «processing node» element, and the departure of parts is modeled with an «exit node» element (with name "partExit").

DPMN is extended by adding the new modeling elements of PN Node rectangles, representing node objects with associated event types, and Processing Flow arrows, representing combined object-event flows. PN Node rectangles take the form of stereotyped UML object rectangles, while PN Flow arrows have a special arrow head, as shown in Figure 12.
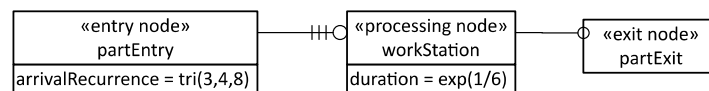

Figure 12: A PN model of a workstation system using node rectangles and Processing Flow arrows.

This model defines:

1. An entry station object named "partEntry".
2. A *recurrence* random variable (with a triagonal distribution with parameters 3,4,8) for the inter-occurrence time of *Arrival* events occurring at the "partEntry" station.
3. A processing station object named "workStation".
4. A *duration* random variable (with an exponential distribution with parameter 1/6) for the duration of activities of type *WorkStationActivity* performed at the "workStation".
5. A *successor* function for the "partEntry" station, providing the "workStation".
6. An exit station object named "partExit".
7. A *successor* function for the "workStation", providing the "partExit" station.

3.3.2 PN example 2: Department of Motor Vehicles

A Department of Motor Vehicles (DMV) has two consecutive service desks: a reception desk and a case handling desk. When a customer arrives at the DMV, she first has to queue up at the reception desk where data for her case is recorded. The customer then goes to the waiting area and waits for being called by the case handling desk where her case will be processed. After completing the case handling, the customer leaves the DMV via the exit.

Customer arrivals are modeled with an «entry node» element (with name "dmvEntry"), the two consecutive service desks are modeled with two «processing node» elements, and the departure of customers is modeled with an «exit node» element (with name "dmvExit").
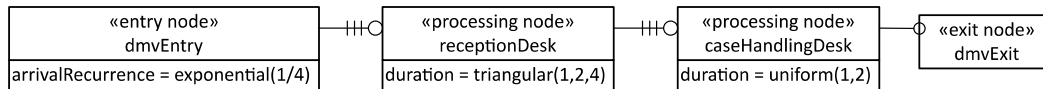

Figure 13: A PN model of a Department of Motor Vehicles.

3.4   Processing Networks versus Activity Networks

While business process models made with BPMN are essentially Activity Networks (ANs), business process models made with classical DES tools, like Simio and AnyLogic, are essentially Processing Networks (PNs).

AN models abstract away from space: the nodes of an AN do not have a location in space, while PN models are spatial models: the nodes of a PN do have a location in space.

In ANs with resource-constrained activities, there are (implicit) task queues, which (typically) have an unlimited capacity. PNs have combined object/task queues, which (typically) have a limited capacity due to limited space.

3.5   Further Case Studies

In (Wagner 2021), three further case studies of OEM/DPMN models are presented, with implementations in the JavaScript DES framework OESjs, in Simio and in AnyLogic:

1. Make and Deliver Pizza
2. Load-Haul-Dump
3. Diagnostic Clinic

4   CONCLUSIONS

We have shown how Event Graphs and Event-Based Simulation (ES) can be incrementally extended by adding the concepts of objects, (resource-constrained) activities and processing activities. The resulting modeling and simulation framework, called OEM/DPMN, supports both Activity Networks (generalizing BPMN) and Processing Networks (generalizing GPSS), and provides a foundation for harmonizing and unifying the concepts and terminologies of established DES tools.

Remarkably, the pioneering ES language/tool SIMSCRIPT abandoned the ES paradigm in a later version and adopted the PN paradigm of GPSS instead of adding the GPSS/PN concepts to ES, as we have done in the development of the layers of OEM/DPMN. In expressive DES models, both events and activities are needed.

REFERENCES

Banks, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. 2010. *Discrete-Event System Simulation*. 5th ed. Upper Saddle River, New Jersey: Prentice-Hall, Inc.

Banks, J. 1998. "Principles of Simulation". In *Handbook of Simulation*, edited by J. Banks, 3−42, New York: John Wiley & Sons, Inc.

BPMN (Version 2.0), 2011. http://www.omg.org/spec/BPMN/2.0, accessed 5th July 2021.

Gordon, G. 1961. "A general purpose systems simulation program". In *AFIPS '61: Proceedings of the Eastern Joint Computer Conference*, 87–104, New York: Association for Computing Machinery.

Guizzardi, G., and G. Wagner. 2010. "Towards an Ontological Foundation of Discrete Event Simulation". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, E. Yücesan. 652−664. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Gurevich, Y. 1985. "A New Thesis". *Abstracts, American Mathematical Society*, 6(4):317.

Loch, C.H. 1998. Operations Management and Reengineering. European Management Journal, 16, 306−317.

Markowitz, H., B. Hausner, and H. Karr. 1962. *SIMSCRIPT: A Simulation Programming Language*. Englewood Cliffs, N. J.: Prentice Hall.

Pegden, C.D. and D.A. Davis. 1992. "Arena: a SIMAN/Cinema-Based Hierarchical Modeling System". In *Proceedings of the 1992 Winter Simulation Conference*, edited by J.J. Swain, D. Goldsman, R.C. Crain, and J.R. Wilson, 390–399. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Pegden, C.D. 2010. "Advanced Tutorial: Overview of Simulation World Views". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 643−651. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Pufahl, L., T.Y. Wong, and M. Weske 2018. "Design of an Extensible BPMN Process Simulator". In *BPM 2017 Workshops*, edited by E. Teniente and M. Weidlich, LNBIP 308, 782–795. Springer International Publishing.

QBP 2021. "QBP Simulation Engine". https://bimp.cs.ut.ee/products/qbp-simulation-engine/, accessed 5th July 2021.

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

Tocher, K.D. 1960. An Integrated Project for the Design and Appraisal of Mechanized Decision-Making Control Systems. Operational Research 11(1/2) :50–65.

Wagner, G. 2021. *Discrete Event Simulation Engineering*. https://sim4edu.com/reading/des-engineering/, accessed 5th July 2021.

Wagner, G. 2020. "Business Process Modeling and Simulation with DPMN: Resource-Constrained Activities". In *Proceedings of the 2020 Winter Simulation Conference*, edited by K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing. 762–773. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Wagner, G. 2019. "Information and Process Modeling for Simulation – Part II: Activities and Processing Networks". https://dpmn.info/reading/Activities.html, accessed 5th July 2021.

Wagner, G. 2018. "Information and Process Modeling for Simulation – Part I: Objects and Events". *Journal of Simulation Engineering* 1:1–25.

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.

Williams, R.J. 2016. Stochastic Processing Networks. *Annual Review of Statistics and Its Application* 3:1, 323–345.

Zeigler, B.P., 1976. *Theory of Modeling and Simulation*. Wiley, New York.

AUTHOR BIOGRAPHIES

GERD WAGNER is Professor of Internet Technology in the Dept. of Informatics, Brandenburg University of Technology, Germany, and Adjunct Associate Professor in the Dept. of Modeling, Simulation and Visualization Engineering, Old Dominion University, Norfolk, VA, USA. 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.