

# Discrete Event Process Modeling Notation: an analysis

CRISTAUDO GIUSEPPE

University of Bologna, Dipartimento di Informatica – Scienza e Ingegneria

October 20, 2022

## Abstract

*Business process management has developed into a mature discipline over the past ten years, with a well-established set of principles, methods, and tools that combine knowledge from industrial engineering, management sciences, and information technology with the aim of enhancing business processes. Discrete Event Process Modeling Notation overcomes standard tools used in BPM by blending the BPMN's intuitive flowchart modeling approach with the strict semantics offered by the Event Graphs' event scheduling arrows and the Object-Event Modeling and Simulation paradigm's event rules.*

## 1 Introduction

Since the first industrial revolution, productivity has risen as a result of technological advancements, changes in how labor is organized, and the use of information technology. Machines (such as those powered by steam and water) entered the workplace during the first industrial revolution (1784–1870). The division of labor, mass manufacturing, and the utilization of electrical energy were the cornerstones of the second industrial revolution (1870–1969). The accessibility of computers, networks, and other IT systems propelled the third industrial revolution (1969–2015). Today, "Industry 4.0" is referred to as the fourth industrial revolution. Using a mix of embedded systems, sensors, networks, service orientation, big data, and analytics, the objective is to develop "smart" manufacturing systems.

The aforementioned four industrial revolutions apply to *administrative procedures and services* even though they are frequently connected with factories and physical production systems.

## 2 Business Process Management (BPM)

Business process management (BPM), as defined by Gartner [1], employs methods to discover, model, analyze, measure, improve and optimize business strategy and processes. While it is sometimes confused with task and project management, its scope is broader than these adjacent topics. Task management focuses on individual tasks whereas BPM observes the whole end-to-end process. Project management refers to a one-time scope of work while BPM focuses specifically

on processes that are repeatable. Through continuous process reengineering, organizations can streamline their overall workflows, leading to increased efficiencies and cost-savings.

Indeed, BPM aims to shift a company's emphasis from functional to process orientation, aligning business processes with strategic objectives and customer demands. Although the literature on BPM is not extensive, it is possible to extrapolate the essence of BPM in:

- structured,
- analytical,
- cross-functional,
- a continuous improvement of processes.

With the intention of apply business process management in real cases, company relies on business process management software (BPMS). Sometimes referred to as BPM systems or suites, BPMS is a collection of different types of technologies that include the following [2]:

- process mining tools for the discovery, representation and analysis of the tasks that drive business processes,
- BPMN tools for diagramming business processes,
- workflow engines to automate the flow of tasks that complete a business process and support,
- business rules engines (BREs) to enable end users to change business rules without having to ask a programmer for help,
- simulation and testing tools for observing how processes behave without having to code first.



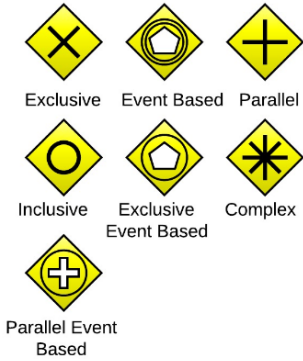
### **3 Business Process Modeling Notation (BPMN)**

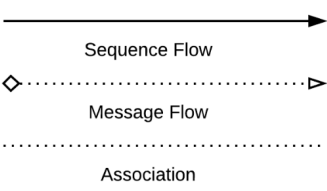

The purpose of this study is to describe what the BPMN is and its robust semantics. To do that, we must first discuss the limitations of the current business process diagramming standard, Business Process Modeling Notation.

Business Process Modeling Notation (BPMN), also called Business Process Model and Notation, is an open standard to diagram a business process [3]. Similar to a flowchart, it uses standardized visuals to illustrate the participants, alternatives, and process flow. Although the diagrams are complex, readers without a background in technology will find them easy to understand. Executives, analysts, and technical implementation staff can all use the same visual in this way to encourage group communication and comprehension. There is no direct correlation between a BPMN diagram and any specific implementation.

BPMN can be thought of as a graphic programming language. In the diagram, each icon has a clear purpose and significance. Each of these icons denotes a stage or activity in the workflow.

The following table includes a collection of the most crucial components of the BPMN process model [4]:

Visual symbol(s)	Name of element	Meaning
 <p>Start   Intermediate   End</p>	Event	"Something that 'happens' during the course of a process", affecting the process flow. "There are three types of Events, based on when they affect Event the flow": a Start Event is drawn as a circle with a thin border line, while an Intermediate Event has a double border line and an End Event has a thick border line.
 <p>Task   Sub Process   Transaction   Call Activity</p>	Activity	"Work that is performed within a Business Process". A Task is an atomic Activity, while Activity a Sub-Process is a composite Activity. A Sub-Process can be either in a collapsed or in an expanded view.
 <p>Exclusive   Event Based   Parallel</p> <p>Inclusive   Exclusive Event Based   Complex</p> <p>Parallel Event Based</p>	Gateway	A Gateway is a node for branching or merging control flows. A Gateway with an "X" symbol denotes an Exclusive OR-Split for conditional branching, if there are 2 or more output flows, or an Exclusive OR-Join, if there are 2 or more Gateway input flows. A Gateway with a plus symbol denotes an AND-Split for parallel branching, if there are 2 or more output flows, or an AND-Join, if there are 2 or more input flows. A Gateway can have both input and output flows.

 <p>Sequence Flow</p> <p>Message Flow</p> <p>Association</p>	Sequence Flow	An arrow expressing the temporal order of Sequence Events, Activities, and Gateways. A Conditional Sequence Flow arrow starts with a diamond and is annotated with a condition (in brackets).
 <p>Data</p>	Data Object	Data Objects may be associated with Events or Activities, providing a context for reading/writing data. A unidirectional dashed arrow denotes reading, while a bidirectional dashed arrow denotes reading/writing.

**Table 1:** *The most important elements of a BPMN process model*

BPMN process diagrams can be used for making:

- conceptual process models (e.g. for both describing existing business processes and creating new ones),
- process automation models for certain process automation systems by including platform-specific technical information in the form of model annotation.

However the BPMN process diagram language has several semantic issues and is not expressive enough for making platform independent process design models that can be used for designing discrete event simulation models [4].

Wagner cites the following issues for BPMN as a business process diagram [4]:

1. A limited concept of "business processes" as isolated "cases", which does not allow to account for any dependency between business processes (e.g., competing for resources).
2. Overloading/ambiguity of sequence flow arrows, which represent various kinds of connections, including resource-independent event flows and resource-dependent activity scheduling.
3. Insufficient integration of the objects that participate in a process.
4. Insufficient support of resource management. In particular, no other resources except (human) performers can be modeled, and the important concepts of resource cardinality

constraints, parallel participation constraints, resource pools, alternative resource types and task priorities are not supported.

5. No support of processing activities in processing networks, which are generalized queueing networks where processing objects enter a system via arrival events at an entry station and then "flow through the system" (being subject to processing activities at processing stations) before they leave the system via departure events at an exit station.
6. No convincing formal semantics. BPMN's official execution semantics is defined in terms of an abstract Petri-Net-style "token" flow (following the predominant academic paradigm), which does not match its intuitive semantics based on event flows and resource-dependent activity scheduling.

He also suggests how his Discrete Event Process Modeling Notation (DPMN) solves these issues.

## 4 Discrete Event Process Modeling Notation (DPMN)

DPMN is a BPMN-based diagram language for making (computational) process design models for discrete event simulation. It blends the BPMN's intuitive flowchart modeling approach with the strict semantics offered by the Event Graphs' event scheduling arrows and the Object-Event Modeling and Simulation paradigm's event rules. While DPMN employs UML Class Diagrams as its default (data, object, and event) type definition language, BPMN uses XML Schema as its default type definition language.

A DPMN model may be formalized using an Abstract State Machine (ASM) whose state structure is specified by an object-oriented signature and whose transitions are determined by event rules reflecting causal regularities. The Discrete Event Process Modeling Notation (DPMN) extends and modifies the language of BPMN Process Diagrams for the purpose of making event rule design models and process design models, which are computationally complete process specifications that can be used for Discrete Event Simulation DES modeling [4]. Finally object types, event types, event rules, and activity kinds familiar to Object-Event Modeling and Simulation paradigm serve as the foundation for the development of DPMN.

The following list is a summary of the key concepts that underlie the development of DPMN [4]:

1. Diagrams in the flowchart style offer a simple visual syntax for process modeling,
2. Events, actions, and (conditional and parallel) branching must all be included in a flowchart language to be considered adequately expressive,
3. Process design models must accurately and completely express the process specifications in terms of computation,

4. There is no explicit computational (control flow) semantics for BPMN's sequence flows. The computational semantics for Sequence Flows leading to event circles is provided by the event scheduling semantics of Schruben's Event Graphs,
5. The event rules of OEMS, which enable capturing causal regularities for describing the dynamics of a system, provide an operational (transition system) semantics for generalized forms of Event Graphs because they provide transition functions.

#### 4.1 DPMN and Event Graphs

The Event Graph diagrams allow for the definition of computationally comprehensive process models for DES, and DPMN diagrams can be seen as a generalization of these diagrams. In these diagrams, circles stand in for different sorts of events. These event types may be annotated with variable assignments that reflect state changes and (potentially conditional) event scheduling arrows that connect them to other event types.

Wagner, then, proposes a list of the main elements of DPMN [4]:

1. the three control flow node types:
  - (a) Event types (circles),
  - (b) Activity types (rounded-corner rectangles), and
  - (c) Gateway types (diamonds)
2. Data Objects (rectangles),
3. Text Annotations left brackets that are attached, via dashed connection lines, to control flow node types, Data Objects or Sequence Flows),  
and the following diagram arc types:
  - (a) Sequence Flows (solid arrows) between control flow node types,
  - (b) Data Object attachments (dashed arrows) between Event/Activity types and Data Objects.

DPMN adopts and adapts the syntax and semantics of BPMN in the following way:

- (a) A DPMN diagram has an underlying UML class diagram defining its (object and event) types.
- (b) DPMN Sequence Flow arrows pointing to an event circle denote event scheduling control flows (adopted from Event Graphs). They must be annotated by event attribute assignments and an occurrence time assignment for creating/scheduling a new event.

4. DPMN has three special forms of Text Annotation:
  - (a) Text Annotations attached to Event circles for declaring event rule variables,
  - (b) Text Annotations attached to Sequence Flow arrows for state change statements,
  - (c) Text Annotations attached to Sequence Flow arrows pointing to Event circles for event attribute assignments.
5. DPMN has an extended form of Data Object visually rendered as rectangles with two compartments:
  - (a) a first compartment showing an object variable name and an object type name separated by a colon, together with a binding of the object variable to a specific object;
  - (b) a second compartment containing a block of state change statements (such as attribute value assignments).
6. BPMN's temporal semantics and visual syntax distinction between Start, Intermediate and End Events is dropped. A DPMN Event circle implicitly represents a start (or end) Event when it has no incoming (or outgoing) Sequence Flow arrows. It represents an intermediate Event if it has both incoming and outgoing Sequence Flow arrows.
7. In a DPMN event rule design diagram, there is exactly one start Event circle followed by zero or more end Event circles, but there is no intermediate Event circle.

## 5 Case Study: Online Sneaker Store

A fresh new online shop has finally opened its gates for limited edition sneaker releases. HTTP requests arrive with a random recurrence rates during a sneaker release and the current site is able to handle only a limited number of requests at the same time. Moreover, if the site has reached its maximum capacity of visitors per time, new clients enter a queue. Clients in a queue may grow impatient and hang up the queue without placing an order.

Then customers, in order to buy their dream pair of shoes, have to select the size and add them to the cart. If the requested size isn't available, customers are able to queue one more time and try a different size. Finally, if the selected sneaker is available, customers are able to complete the order. Since sneaker drops are always used as a stress test for a site, the online shop manager asked to keep track of the following key parameter indicator:

- Number of lost orders,
- Number of total checkouts,
- AVG. queue length,
- Number of lost orders for size unavailable.

## 5.1 Conceptual Model

An online sneakers shop has resource for initial requests, size requests, size channels, checkout requests and checkout channels, where channels are the number of requests for both size and checkout that the site can handle in one time. While, take request activities are performed without any limitation, size request and checkout request activities requires respectively a size channel and a checkout channel.

## 5.2 Conceptual Information Model

The potentially relevant object types are:

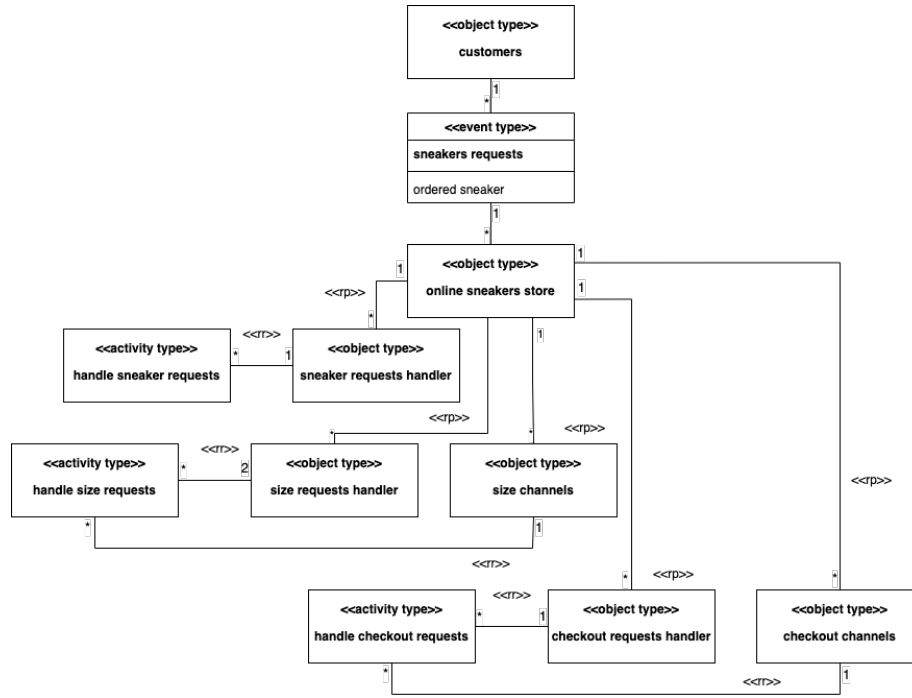
1. online sneaker store,
2. customers,
3. orders,
4. sneakers,
5. requests,
6. size requests,
7. size channels,
8. checkout requests,
9. checkout channels.

Potentially relevant types of events and activities are:

1. sneakers requests coming in from customers,
2. sneaker request handling (an activity performed by the site without limitation),
3. size request handling (performed via size channels),
4. checkout request handling (performed via checkout channels).

A particular kind of UML class diagram called Object Event (OE) class diagram can be used to visually represent object, event, and activity types together with their participation associations in a conceptual information model, as shown below.





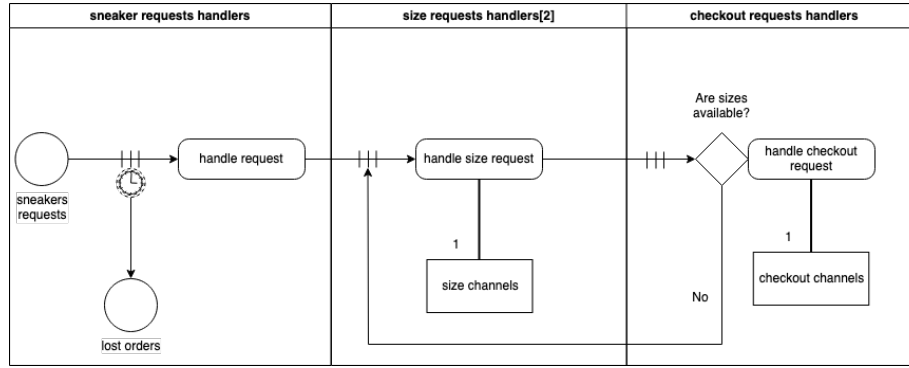
**Figure 1:** Object Event Class Diagram for Online Sneaker Shop

The association and annotations «rr» and «rp» denote *resource roles* and *resource pools*. For instance, the activity type handle size request has two resource roles, size request handlers and size channel. The online sneaker shop has resource pools for requests, size requests handlers, size channels, checkout requests handlers and checkout channels.

Resource roles have resource cardinality constraints. For instance, an handle size request activity requires two size request handlers (one for selecting the size and one for adding it to the cart) and a size channel.

### 5.3 Conceptual Process Model

The following DPMN diagram shows a conceptual model of the Online Sneaker Shop business process, with three swimlanes for the three activities of the process. Since, at the moment of writing there is no DPMN-specific modeling tool, Microsoft Visio or draw.io can be used to draw basic or new DPMN modeling elements.



**Figure 2:** Conceptual Model for the Online Sneaker Shop

Notice how the timeout event circle (with a clock icon) is attached to the three bars of the resource-dependent activity scheduling (RDAS) arrow, representing the queue of planned taking sneakers activities waiting for the availability of a sneaker requests. This implies that the timeout applies to the waiting phase only, and not to the entire taking sneaker request activity.

A resource pool element is typically absent from a notional DPMN process diagram. On the other hand, it might show the performance roles of various activity categories, such as the sneaker request handler and size request handler in the mentioned diagram. Any organizational role defined in the mentioned OE class model is expected to have a matching resource pool within the organization under examination.

## 5.4 Simulation Design

The following simplifications are made in our simulation design.

We only take into account one specific online sneaker drop, which is not required to be described as an explicit object, and only three possible sizes instead of a full size run. Also, we abstract away from individual customers, orders and sneakers.

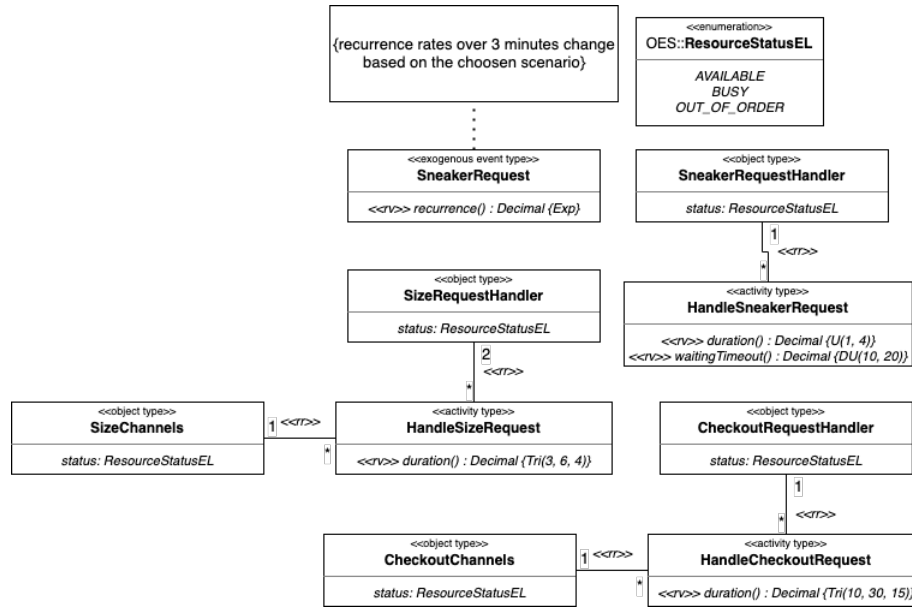
We consider a scenario with the online site being able to handle 4 sneaker request, 6 size request, 10 size channels and 15 checkout channels. In a future work, the default scenario may be chosen by the user.

## 5.5 Information Design Model

A design model is platform-independent model in that it excludes any modeling components made specifically for a given platform, such as Java datatypes. The objective of the solution design model is to retain only those entity types that are necessary for providing answers to the research questions.

An information design model can be derived from a conceptual information model by [4]:

1. Abstracting away from that are not design-relevant.
2. Adding properties, function and methods to all resource object, event and activity classes.  
In particular, a status attribute is added to all resource object types, such as SneakerRequestHandler and SizeChannels, and a class-level duration function is added to all activity classes.



**Figure 3:** Information Design Model for the Online Sneaker Shop

Observe how the keyword «rv», which stands for "random variable," is used to indicate functions that represent random variables, such as the duration function for all activity categories. These random variable functions draw data from a probability distribution function (PDF), which is symbolically represented by expressions such as  $\text{Tri}(10, 30, 15)$ , which denotes a triangular PDF with lower and upper bounds of 10 and 30 and a median of 15, or  $\text{DU}(1, 4)$ , which denotes a discrete uniform PDF with lower and upper bounds of 1 and 4.

The random variable function recurrence selects samples for the event type SneakerRequest from an exponential PDF with three potential event rates for each of the three minutes the online shoe store is dropping.

Keep in mind that SneakerRequest events are exogenous, with a recurrence function established case-by-case for each of the Online Sneaker Store's three minutes drop durations. HandleSneakerRequest provides a class-level waitingTimeout function that implements a random variable with PDF for the waiting timeout event specified in the process model (10,20).

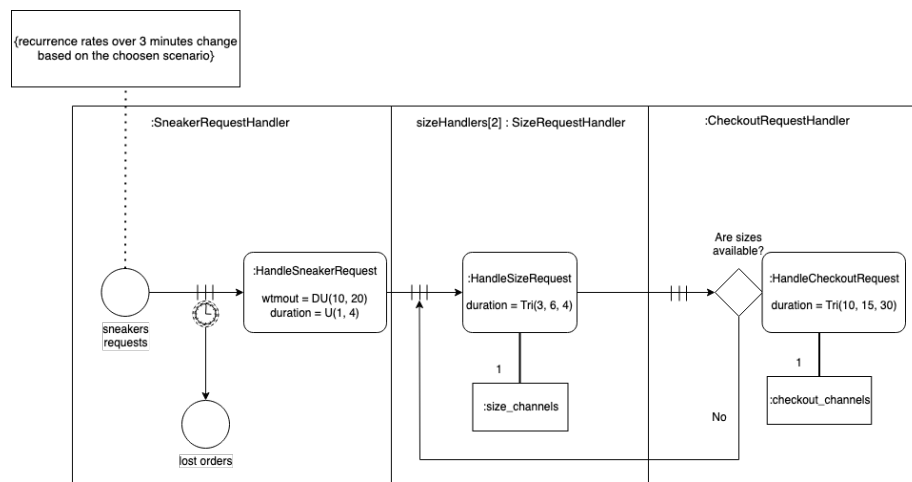
## 5.6 Process Design Model

These models can be expressed textually as event rule tables and visually as BPMN and DPMN process diagrams.

The event types listed in the conceptual information model should be included in a conceptual process model, which should also specify the possible temporal sequences for events based on conditional and parallel branching. We may achieve this by specifying the causal regularity associated with each of the event types from the conceptual information model in the form of an event rule that specifies the state changes and subsequent events brought on by that type of event. An event rule is a conceptual expression of a causal regularity, and an event rule for each relevant type of event is the goal of a conceptual process model for simulation. By abstracting away from elements that are not design-relevant and perhaps adding some computational features, a conceptual process model can be transformed into a process design model, represented as a DPMN process diagram as illustrated below.

A DPMN process design model essentially defines the admissible sequences of events and activities together with their dependencies and effects on objects, while its underlying OE class design model defines the types of objects, events and activities, together with the participation of objects in events and activities, including the resource roles of activities, as well as resource cardinality constraints, parallel participation constraints, alternative resources, and task priorities [4].

The frequency of exogenous events, the duration of an activity, and the most important resource management characteristics defined in the underlying OE class design model, such as resource roles (in particular, performer roles can be displayed in the form of Lanes) and resource cardinality constraints, can all be displayed in a DPMN process design model to improve it.



**Figure 4:** An enriched process design model for the Online Sneaker Shop

Such an enriched DPMN process design model includes all computational details needed for an implementation without a separate explicit OE class design model.

A performer role name (such as `sizeHandlers`) and an object type name (such as `SizeRequestHandler`) designating its range are separated by a colon and used to define performer roles in Lanes. A resource cardinality limitation is indicated when the performer role name is followed by a multiplicity expression enclosed in brackets, as in `sizeHandlers[2]` (stating that exactly 2 `sizeHandlers` are required).

The implicit performer role name is produced by lowercasing the performer type name when just a performer type preceded with a colon (such as: `RequestSneakerHandler`) is provided (as in `requestHandler:RequestSneakerHandler`).

The notation for defining a non-performer resource role, such as `size_channels:SizeChannel`, consists of a named object rectangle, such as the `:size_channels` rectangle, attached to an activity rectangle by means of a connection line representing a resource link, such as the line between the `HandleSizeRequest` activity rectangle and the `:size_channels` object rectangle in Figure 4.

It should be noted that there is no feature in the model of Figure 4 that represents a resource pool. It is expected that the business company under analysis has a corresponding resource pool for each organizational role represented in the underlying OE class model. In order to ensure that the resource objects in each resource pool are instances of the corresponding resource role type, each resource role of an activity type is by default linked to a resource pool with the same (yet pluralized) name.

For instance, a pool `size_channel` is by default given to the resource role `size_channel` for the `HandleSizeRequest` activity. The mechanisms in the pool `size_channels` are instances of the `size_channel` (resource) object type. The `HandleSizeRequest` resource role `sizeHandler` is also given access to a pool of `sizeHandlers`. An OE simulator handles these default pool allocations, which are typically not visible in a DPMN diagram.

## 5.7 Implementation with OESjs

DPMN is an open (non-proprietary) Discrete Event Simulation (DES) modeling language for making platform-independent simulation design models, which can be implemented with any particular DES platform, such as Arena, Simio, AnyLogic, etc. For our case of study, the JavaScript-based simulator OESjs-Core2 has been used. OESjs implements the Object Event Simulation (OES) paradigm, and, consequently, allows a straight-forward coding of OE class models and DPMN process models. The OESjs Core 2 simulator implements an architecture that extends the OESjs Core 1 simulator by adding support for activities, which may be resource-constrained. More information about Object Event Simulation and OESjs are available in [5].

### 5.7.1 Implementing the Information Design model

In order to implement the OE class design model with OESjs-Core2, all of the object types, event types, and activity types defined in the model must be coded in JavaScript classes that extend the corresponding OESjs framework classes oBJECT, eVENT, and aCTIVITY. We start with the object type SneakerRequestHandler shown in the following diagram:

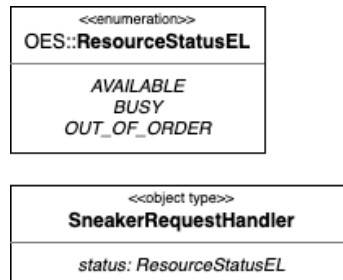


Figure 5: SneakerRequestHandler diagram

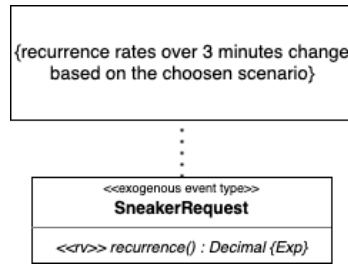
The SneakerRequestHandler object class can be coded in the following way:

```

1 class RequestHandler extends oBJECT {
2   constructor({ id, name, status }) {
3     super( id, name);
4     this.status = status;
5   }
6 }
7 RequestHandler.labels = { "status": "st", "activityState": "act" };
  
```

Listing 1: SneakerRequestHandler

All object classes inherit an id attribute and a name attribute from the pre-defined OES foundation class oBJECT. Since request handlers are resource objects, we need to define a status property having the pre-defined enumeration data type ResourceStatusEL as its range. The other object classes (SizeRequestHandler and CheckoutRequestHandler) are coded in the same way. Next, we implement the SneakerRequest event type, which is shown in the following diagram:

Figure 6: *SneakerRequest* diagram

The *SneakerRequest* event class can be coded in the following way:

```

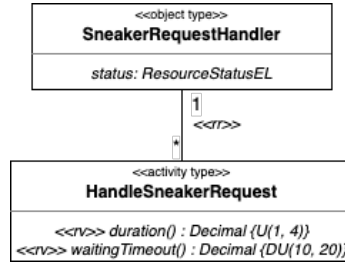
1 class SneakerRequest extends eVENT {
2   constructor({occTime, delay}={}) {
3     super({occTime, delay});
4   }
5
6   createNextEvent() {
7     return new SneakerRequest({delay: SneakerRequest.recurrence()});
8   }
9
10  static recurrence() {
11    var cent = Math.floor(sim.time / 60);
12    return rand.exponential(SneakerRequest.arrivalRates[cent]);
13  }
14 }
15
16 SneakerRequest.successorNode = "HandleSneakerRequest";

```

Listing 2: *SneakerRequestHandler*

The predefined OES foundation class eVENT provides the occTime and delay attributes, which are inherited by all event classes. Any event in OES can be produced with either a value for the attribute delay or a value for the attribute occTime, which stands for occurrence time.

Finally, an activity type such as HandleSneakerRequest can be implemented in the following way:

Figure 7: *HandleSneakerRequest* diagram

```

1 class HandleSneakerRequest extends aACTIVITY {
2   constructor({id, startTime, duration, node}={}) {
3     super({id, startTime, duration, node});
4   }
5   static duration() {
6     return rand.uniform( 1, 4); // durata dell'evento
7   }
8
9   static waitingTimeout() {
10    return rand.uniformInt(10, 20); // dopo quanto tempo si stancano di stare in
        coda
11  }
12 }
13 HandleSneakerRequest.resourceRoles = {
14   "requestHandler": {range: RequestHandler}
15 }
16 HandleSneakerRequest.PERFORMER = ["requestHandler"];
17 HandleSneakerRequest.successorNode = "HandleSizeRequest";

```

Listing 3: *HandleSneakerRequest*

The predefined OES foundation class `aACTIVITY` provides the attributes `id`, `startTime`, and `duration`, which are inherited by all activity classes. The `duration()` function, which is defined as a class-level ("static") function for the activity class, is called when an activity is formed as a JS object during a simulation run to determine the value of its `duration` attribute.

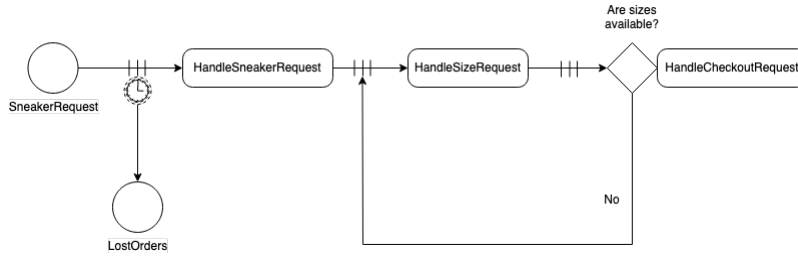
Observe how a corresponding entry in the map-valued class-level property `resourceRoles` codes the resource role association between `HandlerSneakerRequest` and `RequestHandler`, which creates the resource reference property `requestHandler::RequestHandler`.

### 5.7.2 Implementing the Process Design Model

The following process design model specifies six types of events: sneaker request events, handler sneaker request waiting timeouts, lost order events, handle sneaker request activities, handle size



request activities, handle checkout request activities:

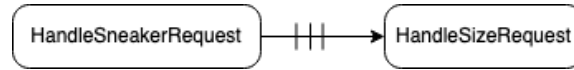


**Figure 8:** Process Design Model diagram

A DPMN process design model can be decomposed into a set of event rule design models, one for each type of event specified in the design model.

Since activities are composite events, we also have rules for them. These rules are triggered when an activity completes (corresponding activity end event).

All events rules have been designed and implemented in the same way.



**Figure 9:** Event Rule diagram

```
1 HandleSneakerRequest.successorNode = "HandleSizeRequest";
```

**Listing 4:** Event Rule

## 6 Results

Simulation have been run over three different scenarios:

1. Default scenario described above,
2. Same architecture of the default scenario but with higher request due to a more coveted pair of sneakers,
3. High request due to a more coveted pair of sneakers, but more resources given to the online shop.

For all three scenario the number of sizes available were 60 for size 1, 20 for size 2 and 30 for size 3.

**Default scenario**

Standalone simulation run with a simulation time/duration of 180 sec.

## User-defined statistics

completed_checkouts	30
size_1	49
size_2	6
size_3	18
lost_orders_do_to_sizes_unavailable	0

Activity node	enqu	LostCheckouts	start	compl	qLen	AVGLen	wTime	cTime	resource utilization
handleSneakerRequestNode	1	0	40	40	1	1	0.01/0.31	2.72/4	{"1":0.19,"2":0.17,"3":0.12,"4":0.12}
handleSizeRequestNode	11		40	37	3	2	0.47/3.19	4.76/7.08	{"11":0.29,"12":0.29,"13":0.3,"14":0.3,"15":0.28,"16":0.28,"size_channels":0.09}
handleCheckoutRequestNode	0		37	30	0	1.33	0/0	17.92/29.84	{"checkout_channels":0.2}

**Figure 10: Result from Scenario 1**

As shown in the figure above, with low number of request, the site can easily handle all of them without loosing any checkout due to impatient clients, without even going sold out and with a final average queue length of 1.33.

**Model variant: same online shop architecture, higher request**

Standalone simulation run with a simulation time/duration of 181 sec.

## User-defined statistics

completed_checkouts	105
size_1	21
size_2	0
size_3	0
lost_orders_do_to_sizes_unavailable	0

Activity node	enqu	LostCheckouts	start	compl	qLen	AVGLen	wTime	cTime	resource utilization
handleSneakerRequestNode	282	45	259	256	34	34	7.12/16.05	9.66/19.75	{"1":0.91,"2":0.9,"3":0.91,"4":0.9}
handleSizeRequestNode	253		122	119	135	84.5	39.64/83.59	43.94/88.47	{"11":0.97,"12":0.97,"13":0.96,"14":0.96,"15":0.92,"16":0.92,"size_channels":0.28}
handleCheckoutRequestNode	11		119	105	2	57	0.06/1.83	18.8/29.22	{"checkout_channels":0.73}

Script files loading time: 19 ms, simulation execution time: 28 ms. Reload the page [Ctrl-R] to start over.

**Figure 11: Result from Scenario 2**

In the second scenario we have way more requests but despite loosing 45 checkouts due to impatient customers the site still is able to have some stock left with an average final queue length of 57.

**Model variant: improved online shop architecture, higher request**

Standalone simulation run with a simulation time/duration of 181 sec.

## User-defined statistics

completed_checkouts	111
size_1	17
size_2	0
size_3	0
lost_orders_do_to_sizes_unavailable	0

Activity node	enqu	LostCheckouts	start	compl	qLen	AVGLen	wTime	cTime	resource utilization
handleSneakerRequestNode	591	92	501	494	85	85	7.7/15.72	10.18/19.17	{ "1":0.86,"2":0.85,"3":0.85,"4":0.84,"5":0.86,"6":0.87,"7":0.85,"8":0.83 }
handleSizeRequestNode	491		128	125	368	226.5	66.74/133.57	70.98/137.13	{ "11":0.98,"12":0.98,"13":0.98,"14":0.98,"15":0.98,"16":0.98,"size_channels":0.2 }
handleCheckoutRequestNode	0		125	111	0	151	0/0	18.03/28.96	{ "checkout_channels":0.56 }

**Figure 12:** Result from Scenario 3

Adding more SneakerRequestHandler didn't seem the best idea as can be seen from the Scenario 3 simulation results. In fact, customers get stuck in picking the size which badly influence the initial queue leading to more lost checkouts.

## 7 Conclusion

Although BPMN is an effective tool for illustrating business processes, its scope is constrained and additional standards must be used to implement it or to include different kinds of activities. Decision Model and Notation is advised for decision flows. A data flow diagram is advised because, despite the fact that BPMN has standards for data objects, it does not account for every stage of the data lifecycle.

By modifying the process diagram language for the goal of simulation-based modeling, where a process model must express a computationally full process specification, DPMN addresses the shortcomings of BPMN. Even though DPMN can retain a significant portion of BPMN's vocabulary, visual syntax, information semantics, and intuitive flowchart modeling style, it also needs to integrate the rigorous semantics offered by the event scheduling arrows of Event Graphs and the event rules of the Object Event Modeling and Simulation paradigm.

In conclusion, the DPMN JavaScript Framework (OESjs) has also been used to demonstrate how effective it is in displaying extensive simulation results, even in simple case studies like the one proposed in this study.

## 8 Future Work

DPMN is an open (non-proprietary) DES modeling language for creating platform-independent simulation design models that may be implemented with any specific DES platform, like AnyLogic,

as was previously stated. As a result, as part of future development, we might want to use AnyLogic to construct the Online Sneaker Shop business process model.

Since AnyLogic does not support Activity-Based Discrete Event Simulation, but only Process Network simulations with "entities flowing through the system" we need to impose PN view on the Online Sneaker Shop business process. This requires to figure out what could be used as "entities" for being able to make a PN model [4].

## References

- [1] Gartner. Business Process Management (BPM). <https://www.gartner.com/en/information-technology/glossary/business-process-management-bpm>.
- [2] Lawton, G., Tucci, L. (2022, April 27). business process management software. SearchCIO. Retrieved October 19, 2022, from <https://www.techtarget.com/searchcio/definition/Business-process-management-suite-BPMS>.
- [3] Wright, G. (2022, April 27). Business Process Modeling Notation (BPMN). SearchCIO. Retrieved October 19, 2022, from <https://www.techtarget.com/searchcio/definition/Business-Process-Modeling-Notation>.
- [4] Wagner, G. (2022, June 8). Business Process Modeling and Simulation with DPMN. <https://sim4edu.com/reading/bpms-dpmn/>.
- [5] Wagner, G. (2022b, June 8). Discrete Event Simulation Engineering: How to design discrete event simulations with DPMN and implement them with OESjs, Simio or AnyLogic (1st ed.). <https://sim4edu.com/reading/des-engineering/>.