Towards an Executable Semantics for Activities Using Discrete Event Simulation

Oana Nicolae, Gerd Wagner, and Jens Werner

Department of Internet Technology
Institute of Informatics
Brandenburg Technical University at Cottbus, Germany
{nicolae,G.Wagner,wernejen}@tu-cottbus.de

Abstract. The paper aims at answering to the challenge of defining an executable semantics for the *activity* concept, in the context of Business Process Modeling and Simulation. The main purpose for introducing an activity concept on top of the basic Discrete Event Simulation concepts of objects and events is to define an activity as consisting of a start event and an end event. This idea is well-known from the business process modeling literature, e.g. from BPDM. We also expect the adoption of this concept view for the BPMN activity in the future BPMN 2.0 Specification. A case study is used throughout the paper to illustrate the concepts and to present our results.

Keywords: Discrete Event Simulation, Business Process Modeling and Simulation, AORSL, BPMN, BPDM, Activity Semantics.

1 Introduction

Each existing simulation methodology focuses on some particular aspects of the system under consideration, by highlighting its specific aspects. Business Process Simulations (i.e. BPS) or Simulation Modelling usually deal with coordinating business processes made up of a workflow queue of activities undertaken by the human and/or computer resources of an organisation ([7], [8]). The construct that is specific here is the activity seen as an executable step in the business process enactment.

In order to use a Discrete Event Simulation (i.e. DES) approach (e.g. the open source Agent-Object-Relationship (i.e. AOR) Simulation Language) for simulating business processes it is necessary to relate the elements of a business system to the entities and events used in a DES system. To achieve this, the workflow steps should be conceptualised as a series of tasks and events that occur through time. Tasks occur in response to events and are performed by the actors operating in the system, or by the system itself. Notice that our discussion envision the representation of the simulation as a graphical diagram, where we make use of the concepts specific to business processes modeling such as tasks and events.

S. Rinderle-Ma et al. (Eds.): BPM 2009 Workshops, LNBIP 43, pp. 369–380, 2010.

Therefore, one important aspect when developing business simulations models is the ability of designing them using an interactive, graphical interface. Business Process Modeling (i.e. BPM) offers this possibility by using expressive and complex artifacts, joined in a graphical notation language accepted by both industry and academia e.g. BPMN [2] (now at version 1.2). However, unlike a BPS language, OMG[1]'s BPMN is still not executable, but expected to be in the future 2.0 version.

We plead for open-source and standardises technologies that enable interoperability. On the actual business market the majority of the vendors are offering open source products that concern only business processes modeling and enactment (e.g. using BPEL). On the other side, proprietary BPS tools and languages offers expensive solutions that make them to be seldom used in the research. Usually, their model simulations are based on assigned parameters of tasks, performers, gateways, and events. Standardization of those parameters was omitted from BPMN 1.x, and is expected to be included in the 2.0 version.

With this in mind, we envision BPMN as a standardised graphical notation for simulation modeling. We made use of BPMN as a graphical notation in order to represent our case study simulation scenarios. Our ongoing research work is double focused: studying BPMN with the purpose of enriching AORSL by adding appropriate business concepts that allow it to be a business process simulation language and, in the same time, the conceptualization of different simulation scenarios in AORSL (i.e. use-cases) will obtain a useful feedback for the BPMN essential structure of its metamodel. In doing so, it will be examined how discrete-event simulation systems specific constructs elements can be represented in BPMN and in which way BPMN has to be extended for this purpose.

In this paper we present a solution for modeling activities, and business processes, on top of the basic discrete event simulation (i.e. DES) concepts of objects (or entities) and events. Our solution is obtained as an extension of the AOR simulation framework, which is an ontologically well-founded agent-based DES framework with a high-level rule-based simulation language and an abstract simulator architecture and execution model available from `http://www.AOR-Simulation.org`.

The paper is structured as follows: Section 1 motivates our topic and provides an overview of the AORS concepts, Section 2 presents a show case example and discusses its BPMN graphical representation and the AORSL implementation. Section 3 focuses on the activity concept introduced as an extension of AORSL and also its conceptual relation with the BPMN /BPDM task concept. We also discuss the show case enriched with the activity concept. In Section 4 we discuss the related works in domain. We conclude with Section 5.

1.1 Introduction to AOR Simulation

The AOR Simulation framework was proposed in [12]. It supports both basic discrete event simulations without agents and complex agent-based simulations

[1] OMG - http://www.omg.org

with (possibly distorted) perceptions and (possibly false) beliefs. A simulation scenario is expressed with the help of the XML-based AOR Simulation Language (AORSL).

The scenario is then translated to Java source code, compiled to Java byte code and finally executed. A simulation scenario consists of a simulation model, an initial state definition and zero or more view definitions.

A simulation model consists of: (1) an optional space model (needed for physical objects/agents); (2) a set of entity type definitions, including different categories of event, message, object and agent types; and (3) a set of environment rules, which define causality laws governing the environmental state changes.

An entity type is defined by means of a set of properties and a set of functions. There are two kinds of properties: attributes and reference properties. Attributes are properties whose range is a data type; reference properties are properties whose range is another entity type.

The upper level ontological categories of AOR Simulation are objects (including agents, physical objects and physical agents), messages and events. Notice that according to this upper-level ontology of AOR Simulation, agents are special objects. For simplicity it is common, though, to say just *object* instead of using the unambiguous but clumsy term *non-agentive object*. Both the behavior of the environment (i.e. its causality laws) and the behavior of agents are modeled with the help of rules, which support high-level declarative behavior modeling.

1.2 Basic Discrete Event Simulation with AORSL

In basic DES, we deal with two basic categories of entities: objects and events see Figure 1. A simulation model defines a number of object types and event types, each of them with one or more properties and zero or more functions (to be used for all kinds of computations such as for computing pseudo-random numbers following an empirical distribution). Among the event types, we distinguish those that define *exogenous events* (typically with some random periodicity) and those that define *caused events* that follow from the occurrence of other events.

The state of the environment (i.e. the system state) is given by the combination of the states of all objects. Environment rules define how the state of objects is changed by the occurrence of events. An environment rule is a 5-tuple $\langle$ *EvtT, Var, Cond, UpdExpr, ResEvtExpr* $\rangle$ where: (1) *EvtT* denotes the type of event that triggers the rule; (2) *Var* is a set of variable declarations, such that each variable is bound either to a specific object or to a set of objects; (3) *Cond* is a logical formula allowing for variables; ex-pressing a state condition, (4) *UpdExpr* specifies an update of the environment state; and (5) *ResEvtExpr* specifies a list of resulting events, which will be created when the rule is fired.

In each simulation step, all those rules are fired whose triggering event types are matched by one of the current events and whose conditions hold. The firing of rules may lead to updates of the states of certain objects and it may create new future events to be added to the future events list. After this, the simulation time is incremented to the occurrence time of the next future event, and the evaluation and application of rules starts over.

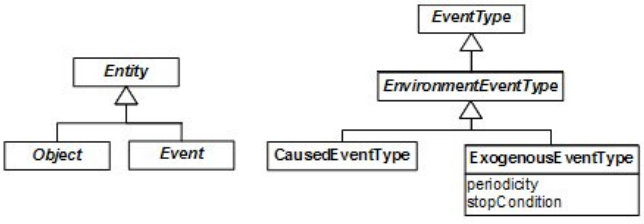

Fig. 1. a) AORSL Entities and b) The ontology of event types as required for modeling discrete-event simulation

2 Simulating a Double Queue System Using DES

In this section we show how to model and simulate a double queue system purely with events, without using activities. We use the BPMN-like graphical notation to describe our simple simulation scenario i.e. the general case - a double queue system where the entities are waiting in line for some resources with restricted capacity to be available. The purpose of the simulation is to estimate the load and scale resource utilization statistics (i.e. the percentage of time they are busy in the system).

2.1 Simulation Showcase

Our particular case is the Dump-Truck problem: trucks are arriving periodically at the Loading Service consisting of two loaders. If there is already a waiting line and the Loading Service is busy, the truck gets in line and waits for its turn. As soon as a serviced truck departs from the Loading Service, the next truck is started to be loaded and a new departure (i.e. end of service) event for this truck is scheduled. If there are no more trucks in line, the Loading Service becomes available. After being loaded, trucks immediately move to the Scale Service (a resource with capacity equal to 1) to be weighed as soon as possible. The Scale Service also has a FIFO waiting line for the trucks. The travel time from the Loading Service to the Scale Service is considered negligible. After being weighed, the truck begins a travel (during which it unloads) and then returns to the Loading Service queue.

We abstract away from the individual truck objects and also from the composition of the queues, since for calculating the resource utilization statistics we do not need any information about them.

2.2 Using BPMN for Simulation Modeling

We decided to build a BPMN representation of our simulation model that is different from the classical view of BPMN diagrams with which business people and academia members are used to.

Our motivation involves the impossibility of classical BPMN diagrams to represent some basic simulation models components such as: the queues. In simulation languages we define and use queues where entities wait for available resources or messages. BPMN has no explicit, corresponding equivalent to this concept of queues of entities, as the entire workflow is viewed from the perspective of that entity. This matter of facts does not exclude the existence of some *transparent queueing process for tokens* that, for example, arises in routing patterns of splitting and joining token flow through gateways and also in the synchronization of the message flow between tokens. Figure 2 provides a BPMN diagram that describes the dynamics of the Double Queue System. In our BPMN representation we can actually model the queues that form in front of the resources i.e. when an entity departs from a service, the following step is a decision logic where if there still are entities in the queue, the next entity will move forward and benefit from the resource and therefore an *EndService* event will be scheduled.

We use the BPMN *task* concept to represent the logic steps that update the state of the system involving the manipulation of state variables such as `loadersQL/freeLoaders/scaleBusy` and to schedule appropriate events. The state of the system is also affected by decision logic (i.e. BPMN exclusive data-based gateway) e.g. the decision to get in line, waiting for the Loading Service

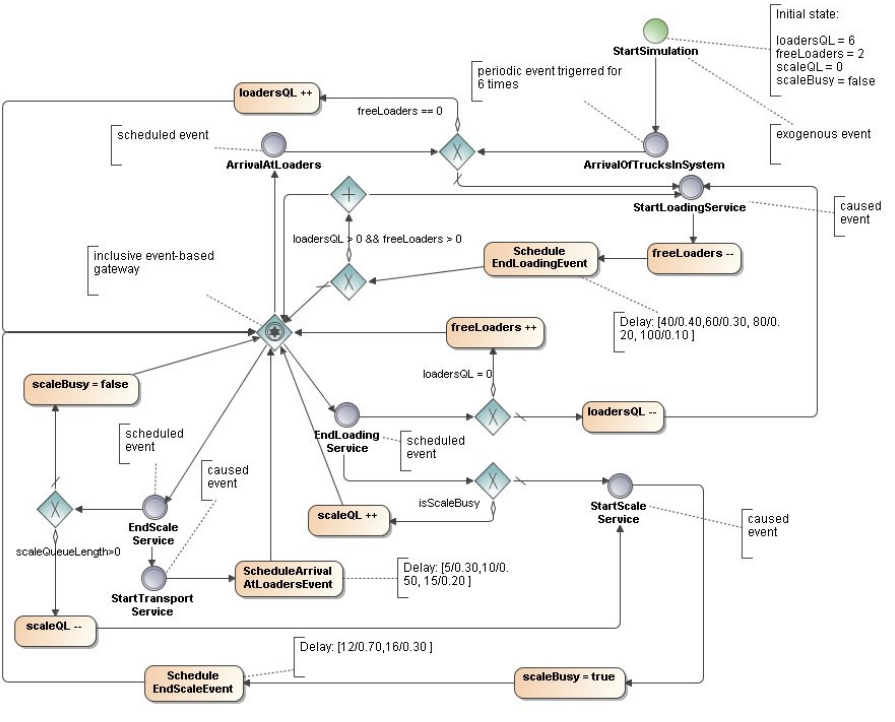

Fig. 2. Modeling a Double Queue System without activities

to be available, otherwise begin to use the Loading Service if the state variable `(freeLoaders > 0)`.

In order to differentiate between different logic steps types in the graphical representation of our simulation, we used different task to reflect the AORSL structure of rules. For example, we used the task `freeLoaders --` to denote the updating procedure of the variable `freeLoaders` and the task `ScheduleEnd LoadingEvent` to schedule the occurrence of an event in the system with a particular delay. With other words, we want to represent some logic step in the simulation where some resource is busy for a definite period of time. The time while the resource is busy varies in minutes with associated probabilities (see Figure 2).

We use in our BPMN representation the *event-based gateway* as a central coordinator of the entire system. We also used the following text annotation: *inclusive event-based gateway*. This means that our gateway does not comply with the deferred choice pattern behavior that provides the ability to defer the moment of choice in a process, i.e. the moment as to which only one of several possible courses of action should be chosen is delayed to the last possible time and is based on factors external to the process instance. An *inclusive event-based gateway* has an OR-semantics, meaning that the interaction with the environment may cause one or more branches (i.e. events to be triggered) to represent possible future courses of execution. Moreover, we impose a restriction on the event-based inclusive gateway: it must be followed only by intermediate events types (i.e. we do not use in our BPMN representation receive tasks (see BPMN 1.2 Specification - Section 9.5.2.4 pp. 77) to represent possible branches, as an actor i.e agent in the simulation can perceive only events).

We believe that the understanding of the above BPMN diagram representing a Double Queue System may be time-consuming. Too many events represent possible branches for execution at a certain point in time. Therefore, even a simplified view of the real-world system can be quite complex, especially when a lot of independent components and events play together.

2.3 AORSL Entities

Our simulation comprises a primary entity type that represents the system on which the simulation runs i.e. the Double Queue System. The unique instance of this entity provides the environment of the simulation. It is active in the system for a definite number of steps, which represents the stopping condition for our simulation. Its attributes represent the state variables of the system i.e. the queue lengths at any moment in time, the Boolean variable `scaleBusy` and the variable `freeLoaders` that represents the number of free loaders.

2.4 AORSL Events

Events happen at a point in time and cause changes of the system state variables. They represent the pivot point in all DES-based simulation systems as the entire

simulation model comprises a sequence of scheduled discrete events which are triggered when some logic steps are completed.

The real computation time does not influence the time of the simulation. The modeled scenarios imagine situations that must be simulated much faster that their occurrences in the real life. The time of the simulation skip among events that are scheduled to occur at different moments in time, facilitating this way the simulation of complex systems. We exemplify the use of the event-based inclusive gateway in our diagram when, more than one event can occur: a truck is finished to be loaded (i.e. an `EndLoadingService` event), a truck starts to be weighted (i.e. an `StartScaleService` event) or a truck arrives at loaders resource (i.e. an `ArrivalAtLoaders` event).

When reading the text annotation of the events we can observe the distinction between *caused event* and *scheduled event*. The scheduled event is also a caused event, but a scheduled event always ends a service after some definite delay of time.

3 Extending AORSL by Adding an Activity Concept

The Activity concept is introduced into the AORSL meta-model by making activities a special case of complex events having a start event and an end event components. In our simulation language the entities have types, therefore, an Activity has an `ActivityType` represented by an abstract class that refines the top-level class `EntityType` with optional start and end event correlation properties (i.e. `UML::Property`). The values of these correlation properties act just like identifiers for the start event and end event types that belong to an activity, placed in some flow of activities.

It is a way to define the semantic of the next step in a chain of activities that follow each other in time, and to correlate the activity end event with the triggering (i.e. start event) of the next activity in sequence. This behavior is closely related with the BPDM way to define sequence of activities. We agree with their perspective and simply consider AORSL activities as steps in the simulation process that can be ordered in time. As we already mentioned, BPDM ([1]) uses the concept of *successions* to express time ordering of the end event of the *predecessor activity* and the start event of the *successor activity*. In this view, events enable the time ordering of triggered events in order to identify exactly which attached event they are referring to, at each end (see BPMN 2.0 Proposal pp.205).

BPMN adds its notation to the above explained semantic i.e. the BPMN Sequence Flow. For simplification, BPMN uses the same notation to denote: (1) an activity that immediately succeeds another activity, (2) an activity that succeeds another activity in a flow of a process, but have other intermediate activities in between, (3) an activity that succeeds a sub-process, (4) an activity that succeeds an event, and (5) an activity that succeeds a gateway. Using the same notation, the different semantics that a succession may have collapse in the unique, classical notation of BPMN sequence flow.

An AORSL `ActivityType` has a mandatory `StartEventType` that triggers the activity instance, and one or more `EndEventTypes` (i.e. specialization subclasses of `EnvironmentEventType`) that express different possible completion horizons of the activity (see Figure 3). As the `Activity` class extends the `Event` class, we can also specify the duration of the work performed by using the optional `duration` attribute. In this case, the activity is triggered by the mandatory `StartEvent` and has a random established duration. When the time expires, we do not have to specify the `EndEvent`, as it is automatically generated in a form of a default `ActivityEndEvent`. An activity may have associated an actor that deals with the activity enactment. The same concept, but under different terminology we can find it in BPDM ([3]) and BPMN ([4]) Specifications (i.e. the *Performer* concept).

In our simulation model, when an activity is performed by an actor, it is usually triggered by an `ActionEventType`, therefore the activity's `StartEvent` should be explicitly defined. The association of an actor with an activity may be done dynamically by the activity triggering event. This is the case when the actor of the triggering event becomes also the actor for the activity.

But, there are situations when the activity is not performed by an actor of the environment i.e. an agent type in the simulation system, but by the environment itself. In this case, we consider that the activity has no defined actor. A consequence is that the activity will be triggered by a default `ActivityStartEvent`. Our simulation model also allows us the possibility not to mention the actor for an activity in certain cases i.e. the case of our example where the unique entity in the simulation model is the Double Queue System agent itself, therefore all activities and environment events have it as a default actor.

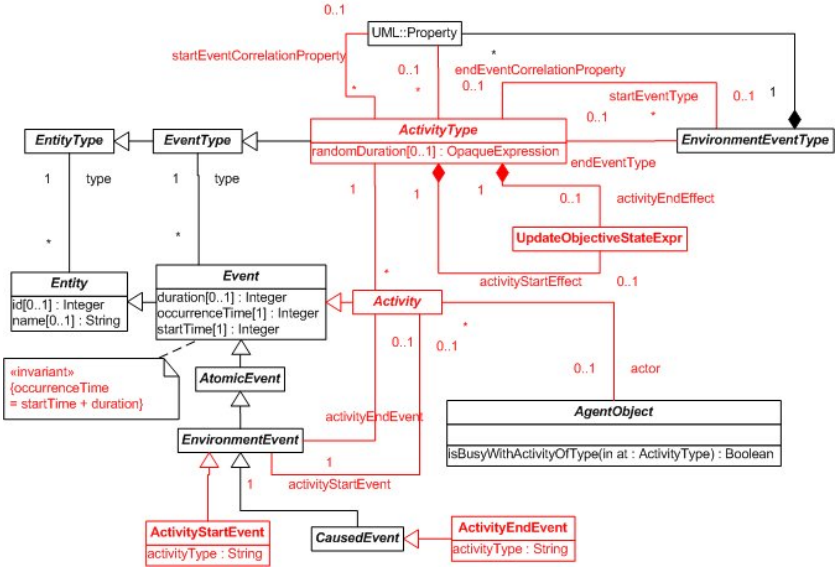

Fig. 3. AORS Activity - Metamodel

Real situations in life show us that the start and the end of an activity usually have some consequences or event effects. The start event of the `PerformScale` activity will cause the service to became busy in the system i.e. `(scaleBusy == true)`, and the end event effect of the same activity is to update the same state variable i.e. `(scaleBusy == false)`.

3.1 Simulating a Double Queue System Using Activities

Further on, we show how to model and simulate the Double Queue System by using the activity concept introduced in the previous Section. This example includes three activities: (1) *Loading at the Loading Service*, (2) *Weighing at the Scale Service* and (3) the *Transportation Service*. In our scenario there are described the entity types used in our simulation. We considered the Double Queue System entity as an agent enriched with attributes that allow us to manipulate the characteristics of the two queues: the length and the state when they are busy or not in the system. The Double Queue System plays the role of the environment of our simulation model and its attributes are state variables.

For each activity from the system we have to define the `ActivityType`. The activity is triggered by an implicitly activity start event, therefore we do not have to explicitly mention it in code. The activity have a duration calculated using

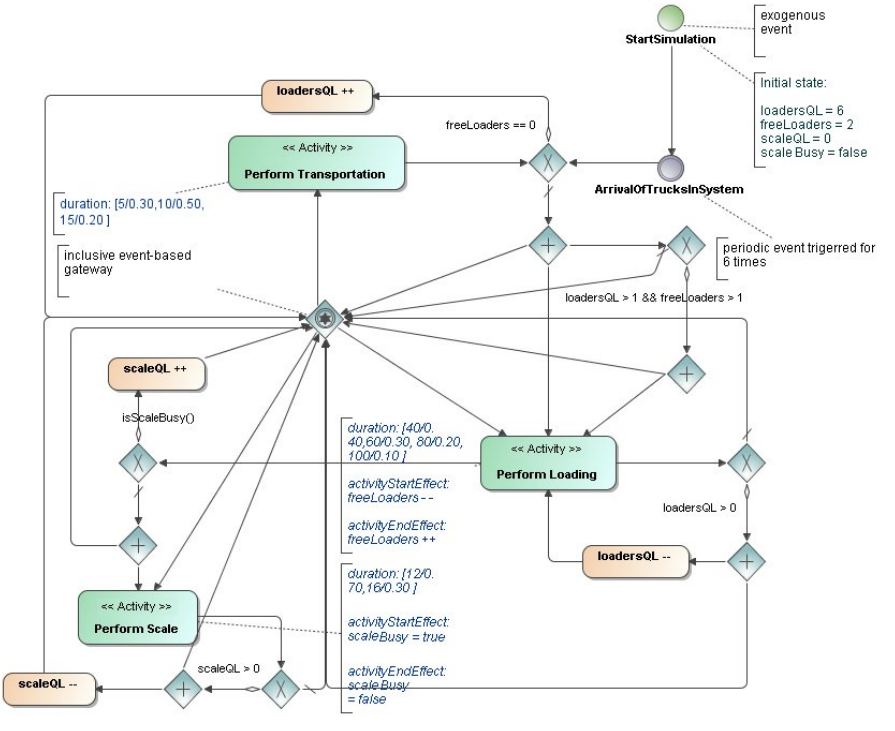

Fig. 4. Modeling a Double Queue System with Activities

the `randomScaleTime()` inner function implementation and expressed using the `Duration` child element.

One should notice that the Figure 4 includes the inclusive event-based gateway, whose semantics is not yet included into the BPMN set of core elements, but scheduled to be included in the future version[2]. Moreover, the inclusive event-gateway can be succeeded by an activity, as it triggers the activity start event. We consider in our example only activities that have delays i.e. a duration in time, implicitly expressed by a scheduled activity end event, or explicitly mentioned by the activity duration.

We can easily remark that the diagram becomes more readable in what regards the quantitative measurement of used symbols and also their semantics. Activity symbols, represented using BPMN task notation with added stereotypes `Activity` simplifies the diagram semantic by make it easier to be followed by an inexperienced user.

4 Related Works

In ([10]) is presented a study on the suitability of the available tools for BPS, by evaluating their capabilities for modeling /simulation /execution and the quality of statistical analysis. The overall conclusion reveals the existence of an ongoing effort to accomplish the goal, as the process modeling tools lack on different percents simulation capabilities, and tools that perform well on business processes simulation do not provide a modeling module.

[6] introduces in the context of Multi-Agent Systems (i.e. MAS) a formal language which have its roots in social sciences: UML-AT Activity Theory. In modeling, the AT paradigm focuses on the activity and its social context instead of agents and their interaction as in agent-oriented methods. Due to these reasons it is more used in the area of social sciences where the focus is on understanding and analyse of human-working activities.

In [11] it is described an open-source DES-based simulation framework i.e. DESMO-J, which is build on top of the JBoss jBPM workflow engine. The simulation is executed using jPDL process language, as an alternative to BPEL. This tool allows the simulation of processes modeled with jBPM in order to evaluate performance indicators, like cycle times or costs. But the focus is on the tool implementation and they miss the conceptualisation and modeling parts, that should provide a deep understanding of the used methodology. Also, they lack a graphical notation.

Ingalls, R. presents the main characteristics of a DES system. Notice that our activity concept is different (i.e. more like a sub-case) from the one described in ([9]) where a broad ontology of activities is provided. After a close inspection of the ontology, we could identify in our simulation language analogous concepts. An activity type brought into discussion is the queue: common used in simulation models as a place where entities wait in line for some resources to be available. This process supposes that entities go through a series of waiting - moving one

[2] `http://www.omg.org/issues/bpmn-ftf.html#Issue10096`

step further - steps and also involves entities state changes. Our example comprises two queues that form when waiting for loaders and scale resources to be available. The queue is more like a complex activity type i.e. the sub-process concept.

Greasley et al. envision in [7] a DES approach for simulation of a workflow system. The workflow steps are conceptualised as a series of tasks and events that occur through time. Tasks occur in response to events and are performed by actors. But, the concept of activity, mentioned in their terminology as task, is unnecessary too complicated and comprises: an event type, which can be a seed event (T0) or a completion event (T3), a start event type (T1), an end event type (T2), the lead time before task needs to start (T1- T0 or T1- T3), the task duration (T2 - T1), and next event relation and identifier.

In the actual BPMN/BPDM 2.0 Specification Proposal ([5]) that aims at joining both BPDM ([3]) and (BEA, IBM, Oracle, SAP) team ([4]) working results, the concept of activity is envisioned as a construct that involves two events i.e. *start-event* and *end-event* and/or the *time in between*.

The activities are described *steps in a process* that can be ordered in time by *successions* (i.e. BPMN Sequence Flow), and are based on start event, end event, or other events of the activities. Successions express the time ordering of the end of their predecessor activity and the start of their successor activity, unless other events are specified (see BPMN 2.0 Specification Proposal pp.204).

5 Conclusions and Future Works

Our work involves aspects from modeling, management and simulation of DESs. We presented an ongoing work that extends the AORSL meta-model with the *activity* concept, a special case of complex events having a start event and an end event components. We showed that our approach is closely related with BPDM/BPMN (i.e. BPMN 2.0 Specification proposals) perspective of modeling activities. With this in mind, the purpose of future works is to identify an extension of the core BPMN for capturing important simulation concepts such that the extended BPMN can be used as a simulation modeling language. To clarify this, the applicability of BPMN for modeling agent-based simulations has to be investigated. Future work directions also envision a BPMN graphical notation for AORSL business process simulation.

References

1. Bock, C.: Tutorial: Introduction to the Business Process Definition Metamodel (2008), `http://www.omg.org/cgi-bin/doc?omg/08-06-32`
2. (OMG) Business Process Modeling Notation 1.2, (BPMN 1.2) (2009), `http://www.omg.org/docs/formal/09-01-03.pdf`
3. (BPDM Team) Adaptive, Axway Software, EDS, Lombardi Software, MEGA International, Troux Technologies, Unisys, BPMN 2.0 Specification Proposal (2008), `http://www.omg.org/cgi-bin/doc?bmi/08-02-03`

4. (BEA, IBM, Oracle, SAP), BPMN 2.0 Specification Proposal (2008), http://www.omg.org/cgi-bin/doc?bmi/08-02-06
5. BPMN/BPDM 2.0 Specification Proposal (2008), http://www.omg.org/docs/bmi/08-09-07.pdf
6. Fuentes-Fernandez, R., Gomez-Sanz, J.J., Pavon, J.: Model Integration in agent-oriented development. Int. Journal of Agent-Oriented Software Engineering 1(1), 2–28 (2007)
7. Greasley, A., Chaffey, D.: A Simulation of an Estate Agency Workflow System. In: Proceedings of the 2000 Summer Computer Simulation Conference, Society for Computer Simulation, San Diego, USA (2000)
8. Greasley, A.: The book: Simulation Modeling for Business. Ashgate Publisher, England (2004)
9. Ingalls, R.G.: Introduction to Simulation. In: Mason, S.J., Hill, R.R., Mönch, L., Rose, O., Jefferson, T., Fowler, J.W. (eds.) Proceedings of the 40th Conference on Winter Simulation, pp. 17–26 (2008)
10. Jansen-Vullers, M.H., Netjes, M.: Business Process Simulation - A Tool Survey. In: Jensen, K. (ed.) The Seventh Workshop on the Practical Use of Coloured Petri Nets and CPN Tools. DAIMI PB, vol. 579, pp. 77–96. University of Aarhus, Aarhus (2006)
11. Ruecker, B.: Master Thesis: Building an open source Business Process Simulation tool with JBoss jBPM (2008), http://www.camunda.com/content/publikationen/bernd-ruecker-business-process-simulation-with-jbpm.pdf
12. Wagner, G.: AOR Modeling and Simulation - Towards a General Architecture for Agent-Based Discrete Event Simulation. In: Giorgini, P., Henderson-Sellers, B., Winikoff, M. (eds.) AOIS 2003. LNCS (LNAI), vol. 3030, pp. 174–188. Springer, Heidelberg (2004)