

Validating Auction Business Processes using Agent-based Simulations

Emilian Pascalau¹, Adrian Giurca² and Gerd Wagner²

¹Hasso Plattner Institute, University of Potsdam

{emilian.pascalau}@hpi.uni-potsdam.de

²Brandenburg University of Technology

{giurca, wagner}@tu-cottbus.de

Abstract: Business Process Modeling and particularly modeling business processes as collaborations is one of the challenges of today enterprise software development. The industry of software development becomes more and more expensive making crucial a correct translation from business idea to implementation to allow for a complete understanding and complete exchange of information between development participants. Particularly, large established software infrastructures are critical with respect to the integration of new components. In this paper, we describe an automated mapping from Single Item English Auction BPMN model to an Agent-Object Relationship simulation towards validation of this business process. The mapping underlines the capabilities of agent-based simulations to execute business processes as well as a number of open questions with respect of BPMN collaboration models.

1 Motivation and Related Work

The Business Process Management Initiative (BPMI) has developed a standard Business Process Modeling Notation (BPMN) [OMG08]. The primary goal of BPMN is to provide a notation that is readily understandable by all business users such that business processes can be illustrated using a standard notation understandable also to business analysts that create the initial drafts of the processes, continuing towards technical developers responsible for implementing the technology that will perform those processes, and finally going on, towards the business people who will manage and monitor those processes. A complete introduction to business processes is given in [Wes07]. This work tries to narrow the gap between two different points of view underlined by two different communities: researchers with a background in formal methods interested in investigating structural properties of processes and the second group consisting of software community interested in providing robust and scalable software systems.

However, building complex business processes on already existent enterprise infrastructure it might be an expensive task. Usually, corporations spent millions integrating heterogeneous applications (otherwise known as Enterprise Application Integration or EAI). Why? Because one of the best ways to generate more profits out of a company is to reduce the costs of doing business.

BPMN helps us to design a business process, but is it able to guarantee the efficiency of the implementation?

As stated in [WG08] the BPMN notation does not have a formal behavioral semantics, which is important in behavioral specification and behavior verification. There are several approaches that try to overcome this inconvenience. The traditional approaches are based upon Petri Nets (e.g. [vdAtHW03, vdAdMW06]), Event-driven Process Chains (e.g. [DvdAtH05]) and Workflow Nets (e.g. [HTvdAW01]).

We propose a new approach for defining behavior semantics for business processes as Agent-based simulations.

How can we validate a BPMN model with respect of its execution? In [Wes07] (page 7) was argued that "business process models are the main artefacts for implementing business processes". In addition in [Wes07] was defined a business process management system as "*a generic software system that is driven by explicit process representations to coordinate the enactment of business processes.*" On the other side in [Woo02] was argued that "*an agent is a computer system that is capable of independent action on behalf of its user or owner.*". In this context at least semantically it is obvious that the generic software system in the form of a business process management system can be in fact a multi agent system.

As far as we know the literature concerning this approach is in a starting point, [EKHA07, EHKA07]. The first paper addresses a similar task as this work, while the second one abandon this approach and uses a different one by trying to *normalize* BPMN towards a better mapping to Petri Nets. However, the work in [EKHA07] is in a beginning stage and is not really related to the BPMN validation but towards "to use an intuitive graphical process notation for designing multi-agent systems" (page 104). By contrast our work proposes an use case investigation i.e. a BPMN mapping to an established agent-based platform, Agent-Object Relationship (AOR) introduced in [Wag03, Wag04], with the goal of validation of BPMN based business process models. The main research behind this approach is to investigate if and to what extent BPMN can be executed in an agent-based simulation environment.

There is at least one more reason, why this approach is important and adequate. According to [Wes07] "*a business process consists of a set of activities that are performed in coordination in an organizational and technical environment*" (page 5). Moreover, Service Oriented Architectures (SOA) and business process modeling go hand in hand. Very often the activities modeled within BPMN models are typically implemented as services. The relationship between agent-based simulation environments and services is a critical one since may help understanding organizational models as services.

2 Single Item English Auction

Automated negotiations including here electronic auctions are very well suited to be modeled with BPMN and also with rules, and research focused on defining and on development of protocols and strategies to be used in multi agent systems that are to perform negotiations [BPJ02], [BPJ03], [JFL⁺01],[Kra97]. The distinction between the negotiation mech-

anism and negotiation strategies is their access modifier. The access modifier refers to the fact that rules describing the negotiation mechanism are *public* and rules defining behavior or strategies are *private*, they belong to agents. Auctions are a form of negotiation mechanism for electronic commerce discussed also in many papers such as [RE05], [WWW01], [WWW02].

English auction also called *Open-outcry auction* is an ascending type of auction. It is an important type of auction discussed in a wide range of papers such as [DASK02], [DRS⁺05], [BGW06], and we consider that the subject is far from being finished. In Single Item English auction only one item is sold at a time. Bidding is open; all participants bid against each other openly. Each successive bid must be higher than the old one. The seller begins the auction by asking for bids at a low price. Buyers bid against each other by raising the price, until only one willing buyer remains.

2.1 The BPMN Model

The model presented in Figure 1 is based on the protocol defined in [BPJ02], [BPJ03] for multi agent systems and underlines that agent models can be modeled with BPMN. The model consists of three pools: `Seller`, `AuctionHost` and `Bidder`. Usually in an auction participate more bidders, but for simplicity the BPMN diagram (1) shows us only one.

An auction is started by the `Seller` with the `AuctionRequest` activity. This activity creates a process starting message to be consumed by the `AuctionHost`. The message transports both the `sellerID` and `startPrice` of the auction. After that the seller's process waits for the `AuctionHost`'s `AuctionCreationRequestResponse` message.

Bidders have to be *admitted* to the auction i.e. they must do admission requests and wait for confirmations.

So following the same pattern (as in the `AuctionRequest` activity), bidders send `RequestAdmissionToAuction` and receive an *admission confirmation* (`AllowToAuctionMessage`) or a *denial* in the form of a `DenyAccessToAuction` message.

Periodically, bidders post bids (`Bid`) to the `AuctionHost` until they receive `AuctionEndNotificationB` messages.

Inside of the `AuctionHost` pool a similar activity is taking place (`processBid`). It consists of receiving bids and processing them. A bid processing ends up by sending a `BidStatus` message containing the `currentHighestPrice` of the auction, so the agents could raise the bid, in the next cycle.

An auction is finished when the auction time expires and all the participants are notified. After the auction end, the `AuctionHost` has no other tasks to perform. The auction notification messages are different from seller and bidders. A seller auction end notification message (`AuctionEndNotificationS`) contains the `auctionStatus`,

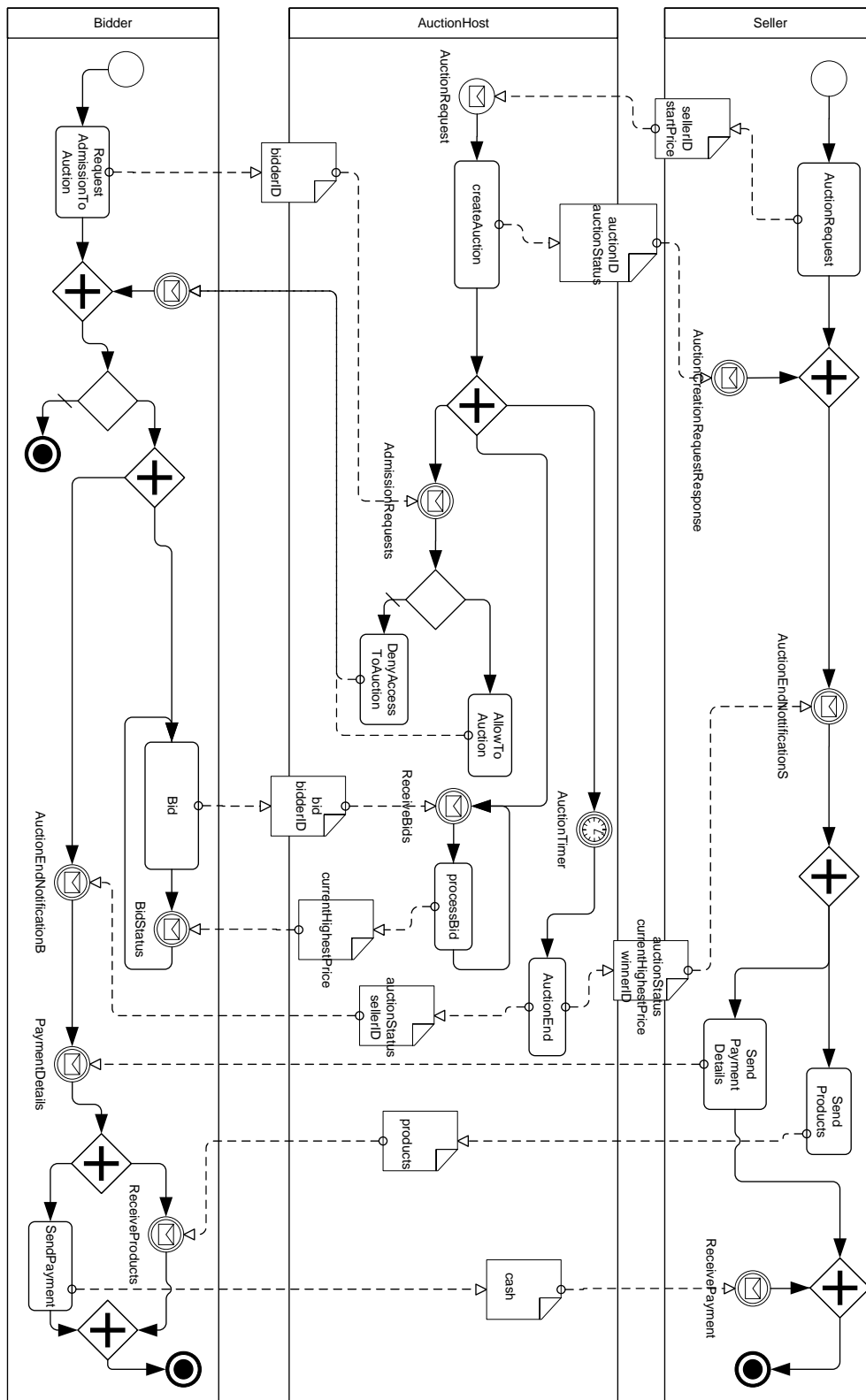


Figure 1: BPMN model of Single Item English Auction

currentHighestPrice and winnerID. The bidders' auction end notification message (AuctionEndNotificationB) content is slightly different since it transports only the auctionStatus and sellerID.

In the seller pool the auction end notification message precedes 2 tasks, namely the SendPaymentDetails and SendProducts. The whole process ends up after the winning bidder receives its products and performs the payment, and the sellers receives the money.

The reader may notice that for modeling English auctions was enough to use the service interaction patterns as they are described in [Wes07] (page 249) i.e. *Send pattern, Receive pattern, Send and Receive pattern, One-to-many send pattern, One-from-many receive pattern.*

2.2 Agent-Object-Relationship Model

In [Wag03] was argued that "*semantics of business processes may be more adequately captured if the specific business agents associated with the involved events and actions are explicitly represented in the information systems in addition to passive business objects*" (page 1). In addition, Agent-Object Relationship offers high-level abstraction that facilitate modeling, expressing and actually simulating concepts and interaction between agents. According to [Wag03] an entity is either an agent, an event, an action, a claim, a commitment, or an ordinary object. AOR models mainly reactive agents having the state represented by a knowledge base and its behavior modeled by means of actions and reaction rules. The visual language represents agents by using a modified UML package representation. Yet there is an obvious difference, emphasized by two rectangles that are below the agent name. The left rectangle comprises the self belief properties of an agent, or it's subjective properties. The right rectangle comprises the agent's objective properties, those properties characterized by physical attributes, those that follow causality rules. The body of the agent can encapsulate beliefs about other entities, events, actions and rules. The reaction rules (or Event-Condition-Action Rules) are rules of the form

```
ON <Event> IF <logical-condition> THEN DO <actions>
```

They are represented graphically as a circle with an internal label RR and a rule identifier attached to it. There are two kinds of incoming arrows (condition arrows and event arrows) and two kinds of outgoing arrows (action arrows and postcondition arrows).

2.2.1 The Seller

Single Item English Auctions have one seller as agent. The Seller (see Figure 2) is a reactive agent having the following internal properties: startPrice, endPrice, inventory, cash, winnerID, auctionID and auctionStatus. The agent behavior is modeled by reactions rules as following:

```
RULE "AR"
```

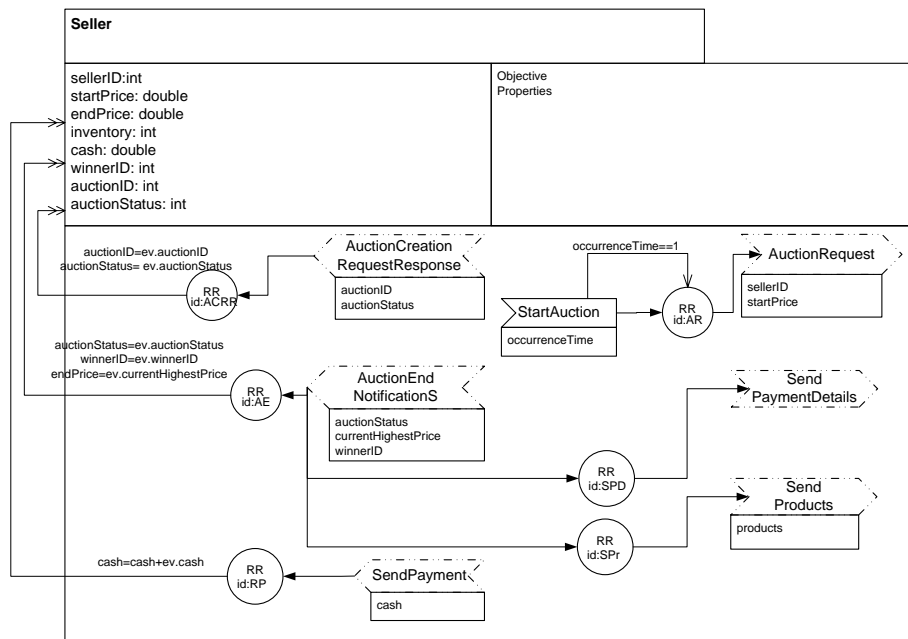


Figure 2: The Seller Agent Model

```

ON StartAuction(?ev)
IF Seller(?Seller) AND ?ev.occurrenceTime == 1
THEN
DO AuctionRequest(?Seller.sellerID, ?Seller.startPrice)

```

```

RULE "ACRR"
ON AuctionCreationRequestResponse(?ev)
IF Seller(?Seller)
THEN DO (?Seller.sellerID = ?ev.auctionID,
        ?Seller.auctionStatus = ?ev.auctionStatus
        )

```

```

RULE "AE"
ON AuctionEndNotificationS(?ev)
IF Seller(?Seller)
THEN DO (?Seller.auctionStatus = ev.auctionStatus,
        ?Seller.winnerID = ?ev.winnerID,
        ?Seller.endPrice = ?ev.currentHighestPrice
        )

```

```

RULE "SPD"
ON AuctionEndNotificationS(?ev)
IF Seller(?Seller)
THEN DO SendPaymentDetails()

```

```

RULE "SPr"
ON AuctionEndNotificationS(?ev)

```

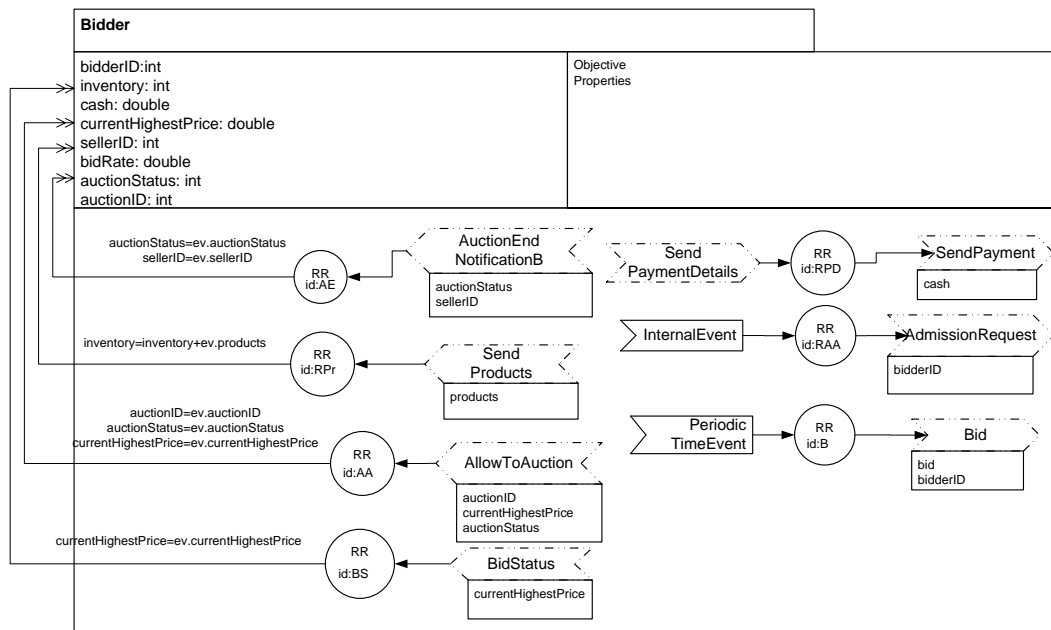


Figure 3: The Bidder Agent Model

```

IF Seller(?Seller)
THEN DO SendProducts(?Seller.inventory)

RULE "RP"
ON SendPayment(?ev)
IF Seller(?Seller)
THEN DO ?Seller.cash = ?Seller.cash + ?ev.cash

```

2.2.2 The Bidder

Single Item English Auctions have one or more bidders as agents. The Bidder (see Figure 3) is a reactive agent having the following internal properties: *inventory*, *cash*, *currentHighestPrice*, *sellerID*, *bidRate*, *auctionID*, and *auctionStatus*. Its behavior is governed by the following ECA rules:

```

RULE "RAA"
ON InternalEvent(?ev)
IF Bidder(?Bidder)
THEN DO AdmissionRequest(?Bidder.bidderID)

RULE "AA"
ON AllowToAuction(?ev)
IF Bidder(?Bidder)
THEN DO (?Bidder.auctionID = ?ev.auctionID,
?Bidder.auctionStatus =?ev.auctionStatus,
?Bidder.currentHighestPrice = ?ev.currentHighestPrice

```

```

)

RULE "B"
ON PeriodicTimeEvent (?ev)
IF Bidder(?Bidder)
  AND
  ?bid = ?Bidder.currentHighestPrice + ?Bidder.bidRate
THEN DO Bid(?Bidder.bidderID, ?bid)

RULE "BS"
ON BidStatus(?ev)
IF Bidder(?Bidder)
THEN DO (?Bidder.currentHighestPrice = ?ev.currentHighestPrice)

RULE "AE"
ON AuctionEndNotificationB(?ev)
IF Bidder(?Bidder)
THEN DO (?Bidder.auctionStatus =?ev.auctionStatus,
        ?Bidder.sellerID = ?ev.sellerID
        )

RULE "RPr"
ON SendProducts(?ev)
IF Bidder(?Bidder)
THEN DO (?Bidder.inventory = ?Bidder.inventory + ?ev.products)

RULE "RPD"
ON SendPaymentDetails(?ev)
IF Bidder(?Bidder)
THEN DO SendPayment(?Bidder.cash)

```

2.2.3 The Environment

According with [Wag04] the state of an Agent-based Discrete Event Simulation system consists of:

- The simulated time,
- The environment state representing:
 - The non-agentive environment (as a collection of objects) and
 - The external states of all agents (e.g., their physical state, their geographic position etc.),
- The internal agent states (e.g., representing perceptions, beliefs, memory, goals),
- A (possibly empty) list of future events.

In Single Item English Auctions the host is responsible with the bidders and seller coordination therefore it is clear that the `AuctionHost` is suitable to be modeled as the agent environment. The environment model is depicted in Figure 4 and consists of: (i) The Seller agent (see Figure 2), (ii) The Bidder agents (see Figure 3), (iii) The Auction, (iv) The environment behavior rules:

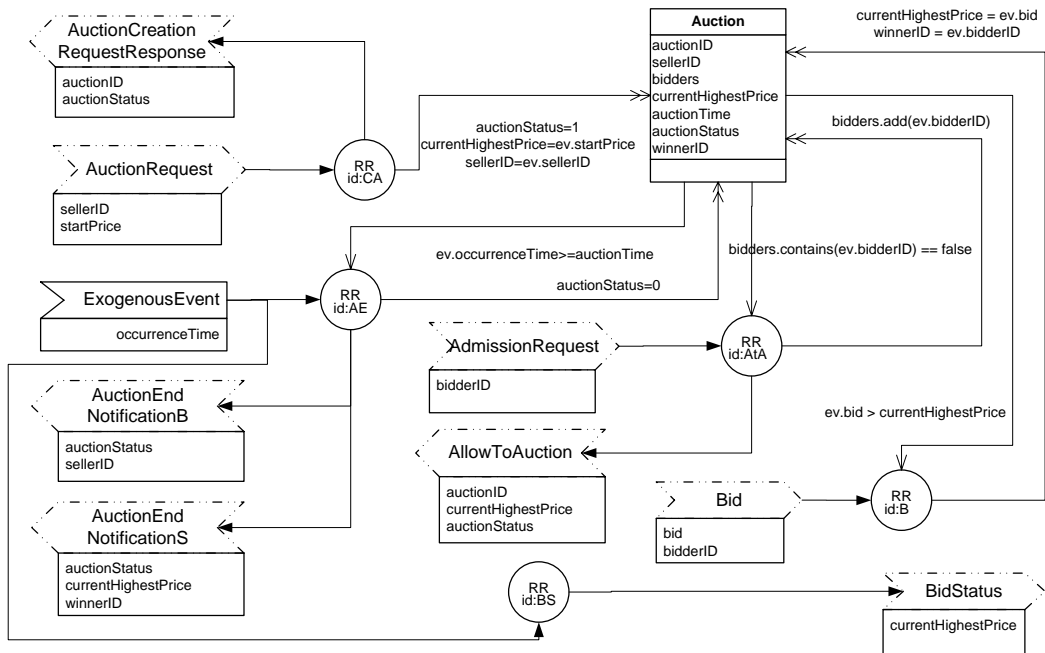


Figure 4: The Environment Model

```

RULE "CA"
ON AuctionRequest(?ev)
IF Auction(?A)
THEN DO (?A.auctionStatus = 1,
         ?A.currentHighestPrice = ?ev.startPrice,
         ?A.sellerID = ?ev.sellerID
        )
      AuctionCreationRequestResponse(?A.auctionID,
                                     ?A.auctionStatus)

RULE "AtA"
ON AdmissionRequest(?ev)
IF Auction(?A) AND NOT(?A.bidders.contains(?ev.bidderID))
THEN DO (?A.bidders.add(?ev.bidderID))
      AllowToAuction(?A.auctionID,
                    ?A.currentHighestPrice,
                    ?A.auctionStatus)

RULE "B"
ON Bid(?ev)
IF Auction(?A) AND ?ev.bid > ?A.currentHighestPrice
THEN DO (?A.currentHighestPrice = ?ev.bid
         ?A.winnerID = ?ev.bidderID
        )

RULE "BS"
ON ExogenousEvent(?ev)

```

```

IF Auction(?A)
THEN DO BidStatus(?A.currentHighestPrice)

RULE "AE"
ON ExogenousEvent(?ev)
IF Auction(?A) AND ?ev.occurrenceTime >= ?A.auctionTime
THEN DO (?A.auctionStatus = 0)
    AuctionEndNotificationB(?A.bidders,
                            ?A.auctionStatus,
                            ?A.sellerID)
    AuctionEndNotificationS(?A.auctionStatus,
                            ?A.currentHighestPrice,
                            ?A.winnerID)

```

3 Mapping Auction BPMN Model to AOR Model

The validation of a BPMN process model with the help of multi agent system simulations requires a mapping between BPMN models and multi agent systems models. Some steps towards were already done in our previous work [PGW09] where the Agent-Object Relationship simulation has been modeled as a BPMN model.

In an agent based simulation the agents are the main entities of the system. The BPMN 1.1 specification [OMG08] states that "a pool represents a participant in a process" (page 43). It may also contain lanes to partition activities. Lanes are sub-partitions of pools. For the simple case a pool should naturally be mapped to an agent. It is the case for `Seller` and `Bidder`.

The agent properties are artifacts for storing data. The BPMN models offer only `Data Object` and `Annotations` as artifacts. The mapping investigates how can be used such artifacts to model agents properties. The central point of agents behavior is declaratively modeled by means of agent reaction rules. By contrast BPMN models express flows by means of events, activities and gateways.

3.1 Mapping the Seller and the Bidders

The `Seller` properties are derived from BPMN objects available in the `Seller` lane. For example since `sellerID` and `startPrice` are sent by the `Seller` to `Auction-Host` they are suitable candidates as agent properties (see Figure 2). This is similar with `endPrice`, `inventory`, `cash`, `winnerID`, `auctionID`, and `auctionStatus`.

Much more complex issues are raised when we map the flow to agent reaction rules. Suitable candidates to identify agent rules are BPMN activities and gateways. It is obvious to assume that any BPMN activity is started by an event and ends with an event, even these events might not be explicitly modeled in the BPMN model.

For example the `AuctionRequest` activity is the origin of the `Seller` agent reaction rule "AR" (see Section 2.2.1). The Figure 5 show us the intuitive mapping.

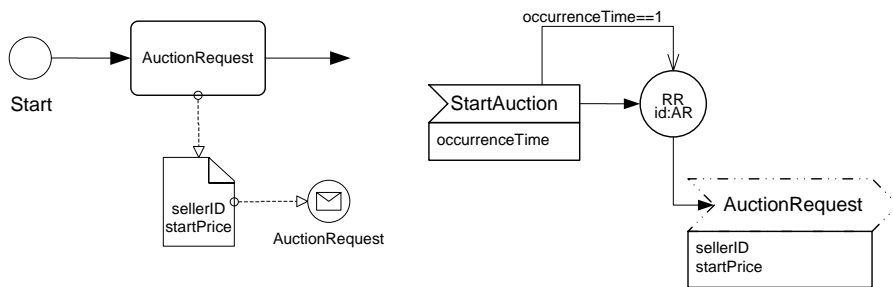


Figure 5: Mapping Activities to Reaction Rules

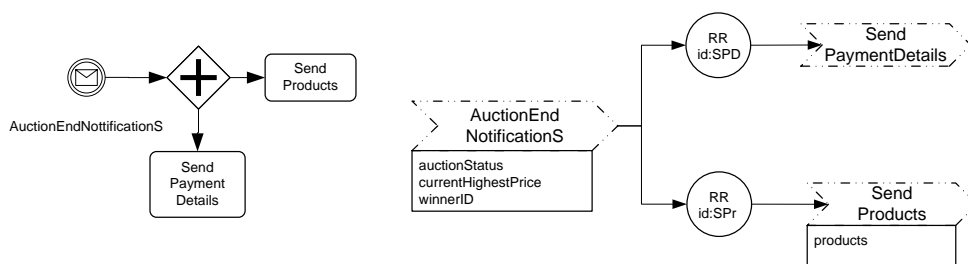


Figure 6: Mapping Gateways to Reaction Rules

Another significant mapping example involves BPMN gateways. The Figure 6 illustrate this mapping. We can see the need of a much better specification of AuctionEndNotifiationsS (such as specifying its properties) to be encoded in a proper AOR event.

The Bidder is very similar with the Seller and follows the same mapping principles. Recall, in the auction can be many bidders but all of them maps to the same agent type.

The Bidder has a loop construction that comprises the Bid activity and the incoming message BidStatus. The mapping is depicted in Figure 7. It must be underlined the

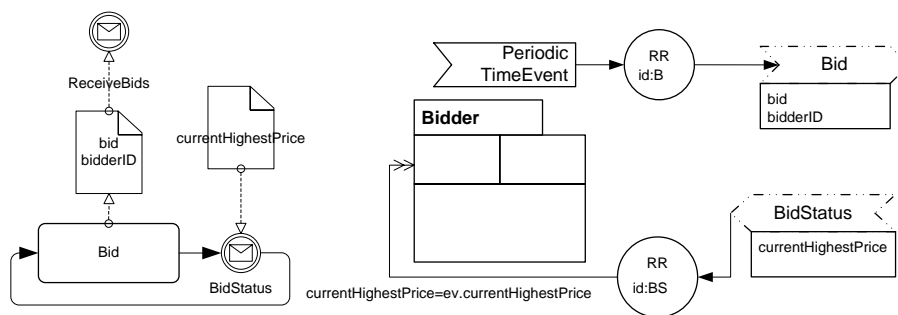


Figure 7: Mapping loops

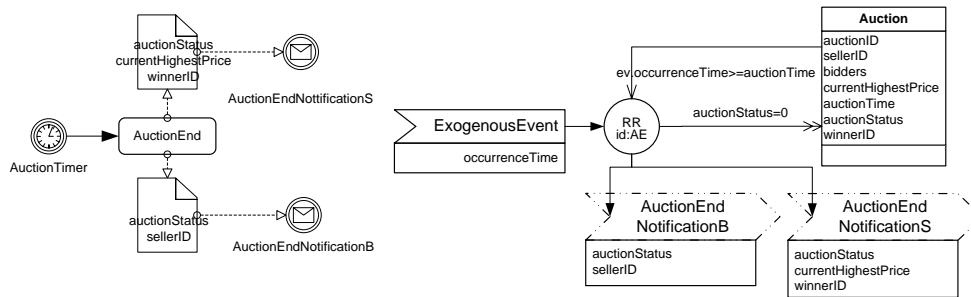


Figure 8: Mapping timers

fact that in AOR simulations the `Bidder` loop as well as the `AuctionHost` loop can not exist without each other. The `Bidder` posts bids periodically and receives `BidStatus` messages periodically because these are sent periodically by the `AuctionHost`. Now the `AuctionHost` sends periodically because itself receives bids periodically. So the two loops are maintaining each other.

3.2 Mapping the AuctionHost

Any agent-based simulation system has an environment where the agents perform. The main question is how to distinguish between pools that have to be mapped to agents and pools that have to be mapped to environment. In our use case `AuctionHost` is suitable to be modeled as the environment. However since the mapping should be done automatically, we have to underline another question for further research: Is the graphical syntactical representation in the form of a BPMN model enough so that a computer program be able to make the right choice? If not what would be the necessary annotations that have to be added to the BPMN models in order to facilitate this distinction?

The two loops from the `AuctionHost` and `Bidder` are modeled in AOR model by means of periodic time events, as already explained in Section 3.1.

Mapping `AuctionHost` introduces two time-related problems: BPMN timers, and timing for events. Time is very important to multi agent systems. Usually agent based systems are discrete with respect to time. However is not completely clear how BPMN deals with time events. In AOR, which is an agent based discrete event simulation, system timers can be modeled either as periodic time events (time events internal to agents), or as exogenous events (when such timers refer to the environment). In our use case the timer event from the `AuctionHost` is mapped into an `ExogenousEvent` (Figure 8).

3.3 A Discussion on Patterns

Recall that the Single Item English Auction use the following patterns: *Send pattern*, *Receive pattern*, *Send and Receive pattern*, *One-to-many send pattern*, and *One-from-many receive pattern*.

We will discuss only the *Send and Receive pattern* since is the correlated combination of *Send pattern*, *Receive pattern*. This pattern involves in the both pools that are part of the conversation an activity and a message. However in different order. For the auction creation situation, the Seller performs activity `AuctionRequest`. This activity is followed by the `AuctionCreationRequestResponse` incoming message from the `AuctionHost`. Opposed the `AuctionHost` first receives the `AuctionRequest` incoming message and then performs activity `createAuction`. Notice that in both cases activities are started by an event. In the AOR model this pattern comprises 3 rules. Two rules (AR, ACRR) belonging to the seller, and one rule in the environment (CA). The environment rule CA is started by the incoming message `AuctionRequest`. The rule ends up by sending the message `AuctionCreationRequestResponse` (see Figure 4). While in the BPMN model the activity `AuctionRequest` is preceded by a starting event in the AOR model we have the rule AR belonging to the Seller triggered by the `StartAuction` event. This rule actually performs the action of posting the message `AuctionRequest`. The second rule belonging to the Seller that completes the pattern is ACRR. This one is triggered by the incoming `AuctionCreationRequestResponse` message from the environment. Notice that the order of the rules is `Seller:AR -> environment:CA -> Seller:ACRR`. While this order in the BPMN model is imposed by the flow arrows, in the agent based simulation this is imposed by the order of the messages.

The *One-to-many send pattern* and *One-from-many receive pattern* mapping process is similar. The difference occurs by the fact that actually several messages are sent or received from/to different agents.

4 Conclusion and Future Work

The paper here underlined facts and open issues about execution and validation of BPMN process models as agent based simulations. It explained why such an approach is suitable and which are the gains and identified some open issues. The open questions identified such as (a) "How to distinguish between different categories of pools i.e. to be mapped to agents or to environment?", (b) "What are properties/metadata that help the distinction?", or (c) "How complete is BPMN such that a computer program be able to make the right choice?", as well as mapping other BPMN identified patterns to agent based constructs and, in addition, the timing problems are starting points for future research.

References

- [BGW06] Costin Badica, Adrian Giurca, and Gerd Wagner. Using Rules and R2ML for Modeling Negotiation Mechanisms in E-Commerce Agent Systems. In Dirk Draheim and Gerald Weber, editors, *Proceedings of the 2nd International Conference on Trends in Enterprise Application Architecture, TEAA2006*, volume 4473 of *Lecture Notes in Computer Science*, pages 84–99. Springer, November 2006.
- [BPJ02] Claudio Bartolini, Chris Preist, and Nicholas R. Jennings. A Generic Framework for Automated Negotiation. Technical report, January 2002.
- [BPJ03] Claudio Bartolini, Chris Preist, and Nicholas R. Jennings. Architecting for Reuse: A Software Framework for Automated Negotiation. In *Proceedings of the 3rd International Workshop Agent-Oriented Software Engineering, Bologna, Italy, AOSE02*, volume 2585 of *Lecture Notes in Computer Science*, pages 88–100. Springer Berlin / Heidelberg, 2003.
- [DASK02] Esther David, Rina Azoulay-Schwartz, and Sarit Kraus. An English Auction Protocol for Multi-Attribute Items. In *Proceedings of the Workshop on Workshop on Agent Mediated Electronic Commerce on Agent-Mediated Electronic Commerce IV, Designing Mechanisms and Systems*, volume 2531 of *Lecture Notes in Computer Science*, pages 52–68. Springer-Verlag London, UK, 2002.
- [DRS⁺05] Esther David, Alex Rogers, Jeremy Schiff, Sarit Kraus, and Nicholas R. Jennings. Optimal Design Of English Auctions With Discrete Bid Levels. In *Proceedings of the 6th ACM conference on Electronic commerce, Vancouver, BC, Canada*, pages 98–107. ACM New York, NY, USA, 2005.
- [DvdAtH05] Marlon Dumas, Wil M. van der Aalst, and Arthur H. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. Wiley, 2005.
- [EHKA07] Holger Endert, Benjamin Hirsch, Tobias Küster, and Sahin Albayrak. Towards a Mapping from BPMN to Agents. In Aalst Wil, Hofstede Arthur, and Weske Mathias, editors, *Proceedings of Service-Oriented Computing: Agents, Semantics, and Engineering 2007*, volume 4504 of *Lecture Notes in Computer Science*, pages 92–106. Springer Berlin / Heidelberg, 2007.
- [EKHA07] Holger Endert, Tobias Küster, Benjamin Hirsch, and Sahin Albayrak. Mapping BPMN to Agents: An Analysis. In *Proceedings of Workshop MALLOW-AWESOME'007 Durham, September 6th7th, 2007*, pages 43–58, 2007. <http://awesome007.disi.unige.it/EndertEtAl-AWESOME007.pdf>.
- [HTvdAW01] Verbeek H, Basten T, and van der Aalst W. Diagnosing workflow processes using Woflan. *The Computer Journal*, 44(4):246–279, 2001. <http://comjnl.oxfordjournals.org/cgi/reprint/44/4/246>.
- [JFL⁺01] Nicholas R. Jennings, Peyman Faratin, A. R. Lomuscio, Simon Parsons, Michael Wooldridge, and Carles Sierra. Automated Negotiation: Prospects, Methods and Challenges. *Group Decision and Negotiation*, 10(2):199–215, March 2001.
- [Kra97] Sarit Kraus. Negotiation and cooperation in multi-agent environments. *Special issue on economic principles of multi-agent systems*, 94(1-2):79–97, 1997.
- [OMG08] OMG. Business Process Modeling Notation, V1.1. <http://www.omg.org/spec/BPMN/1.1/PDF>, January 2008.

- [PGW09] Emilian Pascalau, Adrian Giurca, and Gerd Wagner. *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches*, chapter The Agent Object Relationship Simulation as a Business Process. IGI Publishing, 2009. accepted.
- [RE05] Daniel Rolli and Andreas Eberhart. An Auction Reference Model for Describing and Running Auctions. *Wirtschaftsinformatik 2005*, pages 289–308, 2005.
- [vdAdMW06] Wil M. P. van der Aalst, A. K. Alves de Medeiros, and A. J. M. M. Weijters. Process Equivalence: Comparing Two Process Models Based on Observed Behavior. In *Proceedings of Business Process Management Conference*, volume 4102 of *Lecture Notes in Computer Science*, pages 129–144. Springer Berlin / Heidelberg, 2006. <http://is.tm.tue.nl/staff/aweijters/ComparePM.pdf>, 21 July 2008.
- [vdAtHW03] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske. Business Process Management: A Survey. In Aalst Wil, Hofstede Arthur, and Weske Mathias, editors, *Proceedings of Business Process Management Conference*, volume 2678 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin / Heidelberg, 2003. <http://is.tm.tue.nl/staff/wvdaalst/publications/p183.pdf>, 21 July 2008.
- [Wag03] Gerd Wagner. The Agent-Object-Relationship Meta-Model: Towards a Unified View of State and Behavior. *Information Systems*, 28(5):475–504, 2003.
- [Wag04] Gerd Wagner. *Agent-Oriented Information Systems*, volume 3030 of *LNAI*, chapter AOR Modelling and Simulation – Towards a General Architecture for Agent-Based Discrete Event Simulation, pages 174–188. Springer-Verlag, 2004.
- [Wes07] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag Berlin Heidelberg, 2007.
- [WG08] Peter Y.H. Wong and Jeremy Gibbons. A Process Semantics for BPMN. In *Proceedings of 10th International Conference on Formal Engineering Methods.*, LNCS, October 2008. To appear. Extended version available at <http://web.comlab.ox.ac.uk/oucl/work/peter.wong/pub/bpmnsem.pdf>.
- [Woo02] Michael Wooldridge. *Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.
- [WWW01] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. A Parametrization of the Auction Design Space. *Games and Economic Behavior*, 35:304–338, 2001.
- [WWW02] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. Specifying Rules for Electronic Auctions. *AI Magazine*, 23:15–23, 2002.