Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds

BPMN PATTERN FOR AGENT-BASED SIMULATION MODEL REPRESENTATION

Bhakti S. S. Onggo

Department of Management Science
Lancaster University Management School
Lancaster, LA1 4YX, UNITED KINGDOM

ABSTRACT

The explicit representation of a conceptual model allows it to be communicated and analyzed by the stakeholders in a simulation project. When communication involves different types of stakeholders, a good representation that can be understood by all stakeholders is essential. Many existing methods for the conceptual model representation of agent-based simulation models are less friendly to business users. This paper advocates use of the Business Process Model and Notation diagrams for agent-based simulation conceptual model representation in the context of business applications. This paper also proposes a BPMN pattern that provides visual representation of an agent and its behavior represented as a set of internal and external functions.

1 INTRODUCTION

An agent-based simulation (ABS) model is a simulation model formed by a set of autonomous agents that interact with their environment and other agents through a set of internal rules to achieve their objectives (Onggo 2010). There is no universal agreement in the literature on ABS on the definition of an agent (North and Macal 2007). Instead, we have observed, from the literature, a spectrum of complexity in the definition of an agent. At one extreme, an ABS model is composed of a set of homogeneous agents with a set of simple attributes and simple behaviors. North and Macal (2007) refer to this type of agent as a pseudo-agent. At the other extreme, we have seen ABS models formed by a set of heterogeneous agents with various complex attributes (such as memory) and complex behaviors (such as communication, perception, planning and learning).

The lack of consensus on what constitutes an agent results in various methods of representing an ABS model. There are a number of representation methods that we have seen in the simulation literature. The behavior of agents in many ABS models is often expressed as logical rules. Hence, it is very common to see *pseudo-code* (or even computer code itself) used to represent agent behaviors in the literature. The pseudo-code representing an ABS model tends to be closer to actual computer code. People without any knowledge of computer programming may have difficulty in understanding such model representations. On the flip side, pseudo-code, if written properly, can be a very effective communication method for people who are familiar with computer programming to discuss ABS model behavior.

DEVS (*Discrete event system specification*) has also been used in the literature (e.g. Zaft and Zeigler 2002, Saple and Yilmaz 2006, Onggo 2010). DEVS is a formal specification language that has been applied to represent a wide range of systems (Zeigler 1976). It provides a mathematical representation of an ABS model that can be translated into a simulator automatically. The correctness of the representation can be validated mathematically.

Another formal language that has been used in the literature is *Petri Nets*, especially in multi-agent systems (MAS). Although there are some differences between ABS and MAS (North and Macal 2007), the representation of agents and agent behaviors is a topic common to the two fields. Similar to DEVS, Petri Nets can represent the static structure and dynamic of behavior of an ABS model. Holvoet (1995) was among the earliest to propose the idea of representing agents in Petri Nets. Since the initial work by Holvoet (1995), there have been significant advances in the research that address their limitations, such as the lack of specification for inter-agent communications, the static nature of the nets and the lack of a specification for intelligence. One notable piece of work was produced by Moldt and Wienberg (1997), who extended Colored Petri Nets to include the intelligent behavior and specifications of agent communications. A more recent example is the work of van der Zee (2009), who used Timed Colored Hierarchical Petri Nets in the representation of ABS models in the context of manufacturing systems.

Allan (2010) conducted a literature review on popular ABS frameworks and found out that the majority of the tools were based on Object Oriented principles, i.e. they used classes and methods to represent agents and agent behaviours. Hence, it is not surprising to find that many articles use UML diagrams (or their variants), which are commonly used in Object Oriented design, to represent agents and agent behaviours. UML provides a multi-faceted representation of an ABS model using various UML diagrams and annotations. Hence, it has the advantage of providing a more complete representation of an ABS model. UML has another advantage in that it is a standard supported by various organizations, especially in the software industry. It can be extended and tailored to suit specific domains, e.g. Bauer (1999) and Odell et al. (2000) extended UML to make it more suitable for modelling agents and agent-based systems in general.

Given its origin in Artificial Intelligence (North and Macal 2007), many existing methods used in ABS conceptual model representation are less friendly to business users who may not be familiar with software engineering or computer programming concepts. The paper advocates the use of *Business Process Model and Notation* (BPMN), a standard designed for business users, for ABS conceptual model representation. This paper extends the earlier work presented at the last Winter Simulation Conference (Onggo and Karpat 2011). Some feedback from the audience was on how to represent interactions and more complex agent behaviour in BPMN. To address this, a BPMN pattern is proposed and will be discussed in Section 3. The outline of this paper is as follows. Section 2 briefly summarizes the earlier work reported in Onggo and Karpat (2011). It is followed by an explanation of the relevance of BPMN for ABS model representation. Section 3 elaborates on the proposed BPMN pattern and demonstrates how agent behaviours and the interactions between agents can be represented using this pattern. Section 4 offers concluding remarks.

2 ABS MODEL REPRESENTATION USING BPMN

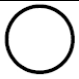
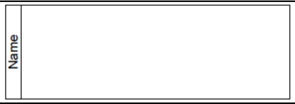
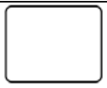
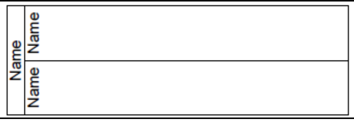
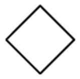
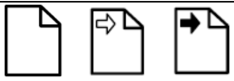
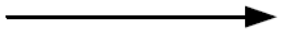
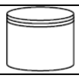
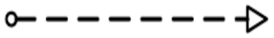
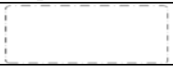
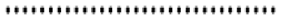
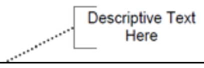
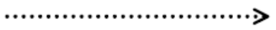
BPMN is a business process modelling language and standard controlled by the *Object Management Group* (OMG). BPMN 2.0 supports a number of diagrams which include: process diagram, collaboration diagram, choreography diagram and conversation diagram. These diagrams share a common collection of core graphical elements that can be grouped into five categories: flow objects, connecting objects, swimlanes, data and artefacts. The shapes of the core graphical components are shown in Table 1.

Flow objects (events, activities and gateways) and connecting objects are the most essential components because they are used to define the structure and behaviour of a process. Connecting objects are used to connect flow objects to each other or to other elements. OMG (2010) provides a detailed specification of all BPMN core elements.

2.1 Relevant BPMN Diagrams: Collaboration and Conversation

Two BPMN diagrams are particularly relevant to the proposed representation method for ABS models. They are the collaboration and conversation diagrams. The *collaboration diagram* is used to represent the interactions between two or more participants. Figure 1 shows an example of a collaboration diagram that describes what happens when a customer places an order with an online trader. The online trader will check the inventory status with the distributor and send the order status to the customer. In this example, three participants are involved and they interact through messages.

Table 1: BPMN core graphical components – adapted from OMG (2010)

Element	**Notation**	**Element**	**Notation**
Event		Pool	
Activity		Lane	
Gateway		Data object, Data input, Data output	
Sequence flow		Data store	
Message flow		Group	
Association		Text annotation	
Data association			

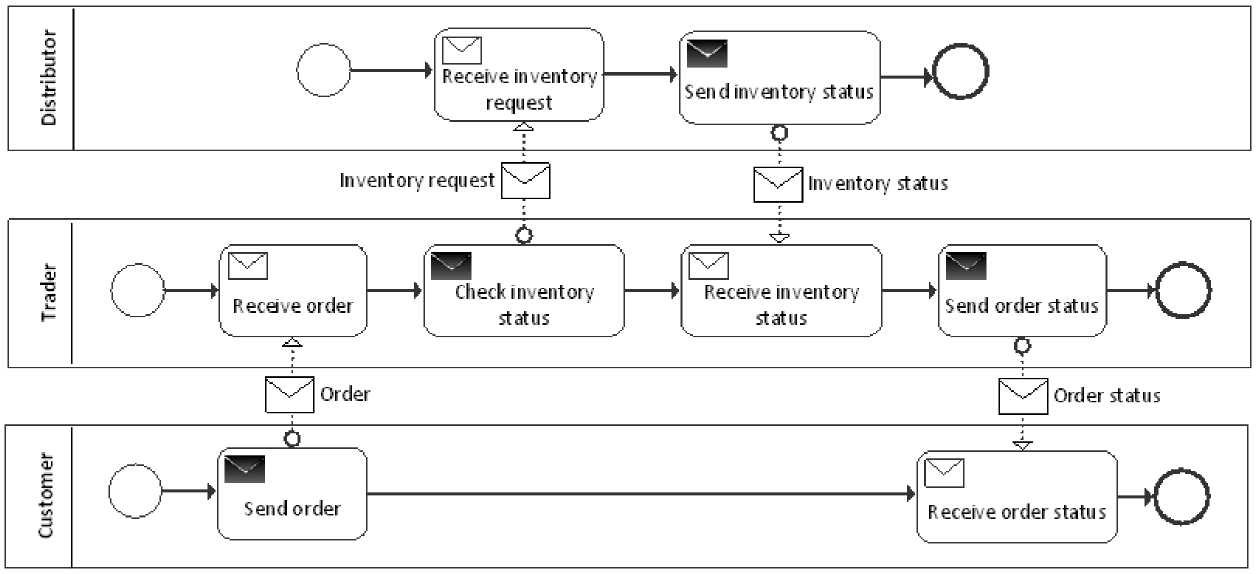


Figure 1: BPMN collaboration diagram

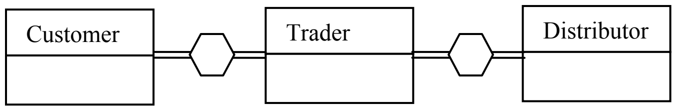


Figure 2: BPMN conversation diagram

The *conversation diagram* provides a high level overview of the interactions between participants in the model. Figure 2 summarizes the interactions between the participants shown in the collaboration diagram in Figure 1. The diagram only shows the participants (as pools), and detailed interactions are encapsulated within the hexagon.

2.2 Representing the ABS Model Using BPMN Diagrams

The main advantage in using BPMN to represent ABS models in the context of business applications is that BPMN is a standard designed for business users and is supported by big vendors such as IBM, Oracle, SAP, Unisys, etc. It is likely that the number of software tools that support BPMN will grow. The main disadvantage of BPMN is that it is designed for a process-oriented modelling tool. This section addresses this issue and explains that BPMN can be used to represent ABS models. Unless otherwise stated, the ABS models in this section onwards refer to ABS models in the context of business applications.

Macal and North (2010) show a wide range of ABS applications and identified business-related applications, such as the artificial society, economics and market analysis. Bonabeau (2002) divided the application of ABS to business into four areas: flows (e.g. customer flow management, traffic management), market (e.g., stock market, ISP market), organization (e.g. organizational behaviour, risk analysis, business operations) and diffusion (e.g. adoption of new products, viral marketing). These applications show that a method for ABS model representation needs to support the representation of agents, agent behaviours, agent interactions and the environment. The examples given in Macal and North (2010) and Bonabeau (2002) show that typical agents in business applications are individuals (such as consumers and investors) and organizations (such as firms and government institutions). These agents plan and execute their plans to achieve their objectives. They are intelligent because they can learn by comparing the results of their actions with their objectives. As a result, they may change their beliefs, plans or actions. They can also learn by analysing the information they receive from other agents and their environment. To an observer, the behaviour of these agents is shown by their actions, including their interactions with other agents and their environment over time. Each agent has its own unique identity and set of attributes (characteristics). These attributes can be static (such as birth date), relatively static (such as an organization's vision, mission and type of individual decision-making) or dynamic (such as memory and wealth). The attributes of an agent can have a significant influence on its behaviour. To demonstrate that BPMN can be used to represent ABS models, BPMN must be able to represent an agent, its behaviour, its environment and the interactions.

The concept of participants in the BPMN collaboration diagram is similar to the concept of agents in ABS. Hence, a BPMN pool that is used to represent a participant can be used to represent an agent. The use of a BPMN pool has at least three advantages. First, the pool creates a clear visual boundary around an agent which encapsulates its attributes and behaviour. A BPMN pool can be divided into a number of lanes. This leads to the second advantage. It allows us to implement an agent which has multiple roles. Since the lanes within a pool can be organized into a hierarchical structure, we can represent the role hierarchy that an agent may have, such as departments within a division and divisions within a firm. Finally, a BPMN pool implies the existence of domain control within the pool, which is in line with the notion of agent autonomy in ABS.

An agent has attributes and behaviour. The attributes of an agent can be represented using BPMN data annotation. This data annotation allows us to show how the attributes are used (e.g. input or output) and which activities use the attributes. The behaviour of an agent can be represented using a combination of flow objects (events, activities and gateways) and connecting objects encapsulated within the pool that represents the agent. A BPMN event can be used to represent (1) an event that triggers an agent to act, (2) an event raised by the action of an agent, or (3) a final event that ends a certain process. In addition, BPMN allows us to specify whether an event will pause, resume or interrupt (and redirect) an action. A BPMN activity can be used to represent an action performed by an agent. It represents a step or action (not a state) in the process. A BPMN gateway can be used to represent various types of decisions, such as branching, selection and merging. The BPMN connecting objects are used to connect the BPMN flow objects. Experienced modellers may appreciate that the combinations of flow objects and connecting objects in BPMN have the potential to represent various complex agent behaviours.

Depending on its behaviour, the environment in an ABS model can be passive or active. The environment is passive if it reacts only as a response to an action performed by an agent. In the absence of actions from agents, the environment will remain the same. An active environment can change its state, even in the absence of any actions by agents. This shows that the environment behaves very much like an agent. Therefore, we can apply the same BPMN constructs we use for the agent to the environment.

Finally, BPMN imposes that the communications between participants must be conducted via messages and signals. Messages are sent to specific recipients and signals are broadcast (and it is up to the participants to decide whether they are interested in those signals). This is consistent with the interactions between agents in ABS. Hence, this section has shown that BPMN has the ability to represent the core components of an ABS model, i.e. agents and their attributes, behaviour, environment and interactions.

3 ABS MODEL REPRESENTATION USING A BPMN-PATTERN

To help modellers use BPMN to represent an ABS model, this paper proposes a BPMN pattern that is built based on DEVS's internal and external functions and which provides visual representation of them. DEVS is one of the most elegant formal specification languages that has been used to represent ABS models (Dávila and Uzcátegui 2000, Zaft and Zeigler 2002). Therefore, we can tap into the significant body of knowledge that proves the general computing power of DEVS. We use an *internal function* and an *external function* to represent an action performed by an agent based on a condition internal to the agent and an action performed in response to an action performed by another agent, respectively. Figure 3 shows the BPMN pattern that provides a visual representation of the internal and external functions of an agent.

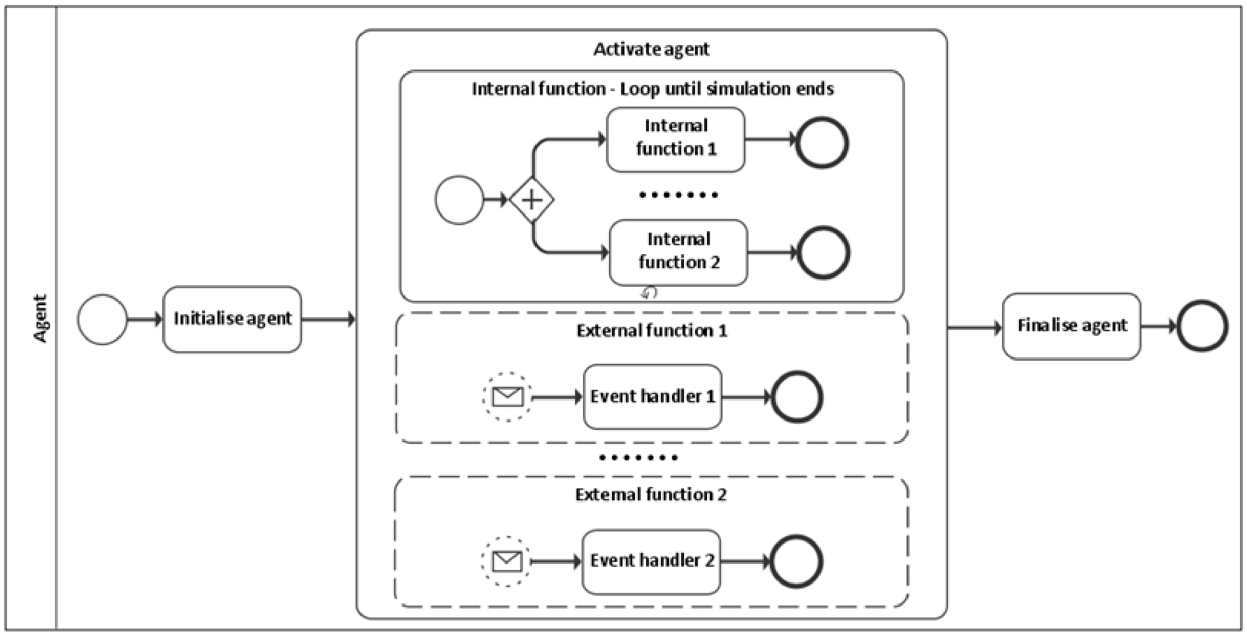


Figure 3: BPMN pattern for a generic agent

The pattern shows that a start event (leftmost circle) activates an agent. A start event is represented as a circle with a single thin line boundary. Once activated, the agent will perform a self-initialization action. This is followed by the main actions that will be performed by the agent throughout its lifetime. This is represented by a sub-process called `Activate agent` in the diagram. The sub-process is used to encapsulate the two types of actions (or functions) that an agent can perform: internal and external. The internal function is repeated until the simulation ends. This is indicated by the loop icon at the bottom of the internal function sub-process. If necessary, modellers can add logical rules to specify the conditions for the execution of certain internal functions. The external functions are represented differently. Each external function is represented as a non-interrupting BPMN event sub-process, i.e. a sub-process that is activated as a response to an event external to the agent, and its execution will not terminate the execution of the currently active internal and other external functions. A non-interrupting event sub-process and the event that triggers it are represented by a dashed-line boundary in the diagram. If the agent needs to interrupt the execution of its internal function, the modeller needs to specify this explicitly in the BPMN diagram. When the simulation ends, the agent will wait until all actively running internal and external functions end before executing the activity `Finalise agent`. When this task is complete an end event is raised and the agent dies. An end event is represented as a circle with a single thick line boundary.

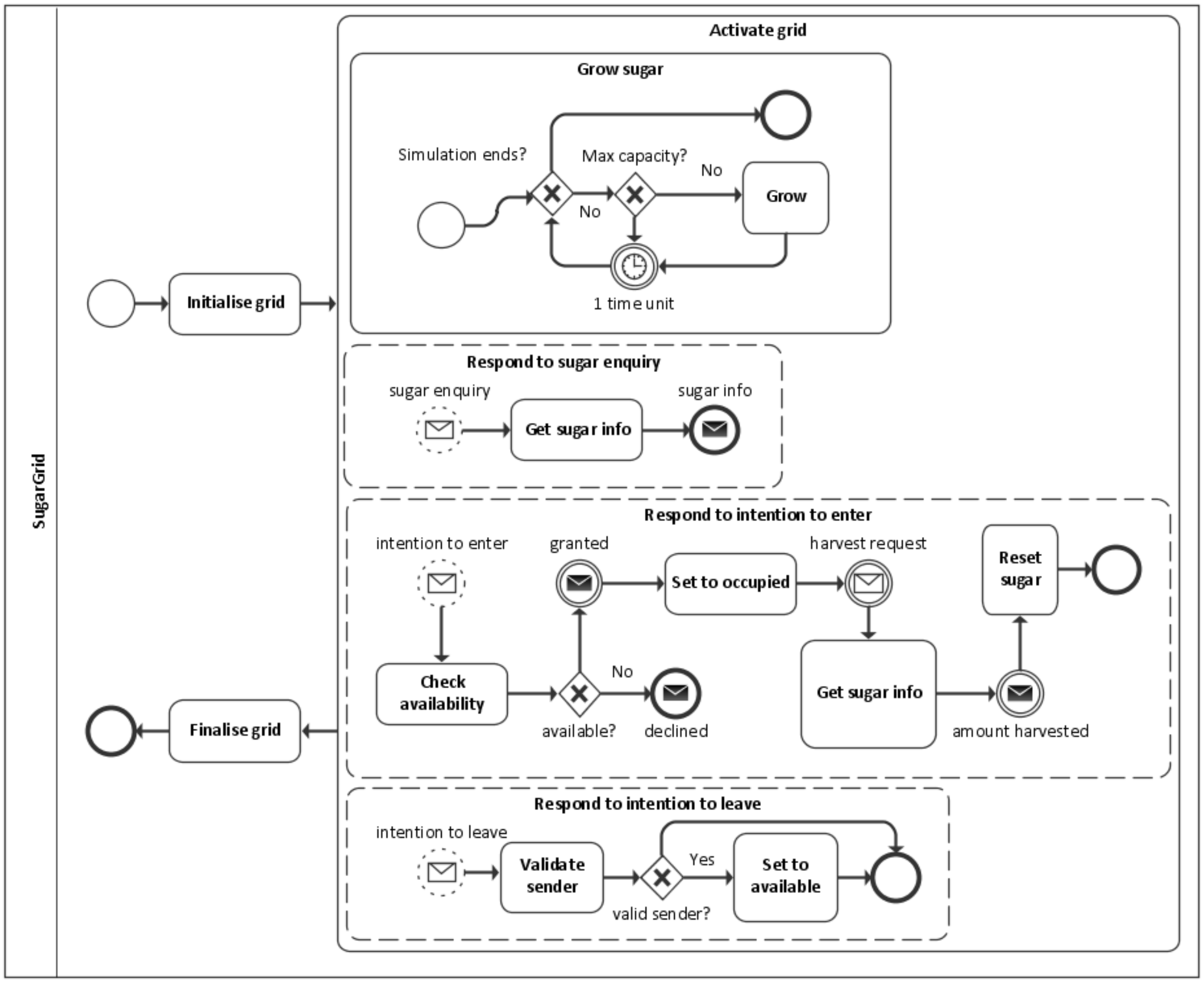


Figure 4: A grid in a SugarScape model represented using a BPMN pattern

3.1 Example: SugarScape

A variant of the `SugarScape` (Epstein and Axtell 1996) model is used to demonstrate how a BPMN pattern is used to represent an ABS model. The `SugarScape` model is chosen because it is one of the most widely known ABS models. Unlike the rumour model used in Onggo and Karpat (2011), this model shows more complex behaviour and interactions between agents. The `SugarScape` model has one type of agent, i.e. `SugarPerson,` and each of them lives in a grid, i.e. `SugarGrid`. The agents can move to different grids in a two-dimensional world that is divided into $n \times n$ grids during their lifetime.

Each grid contains some sugar. The amount of sugar varies from grid to grid and grows over time until it reaches a pre-determined maximum amount. Since each grid is active, i.e. it behaves even in the absence of any action from any `SugarPerson`, each grid is represented as an agent. The representation of each `SugarGrid` using the BPMN pattern is shown in Figure 4. The growth of sugar in a grid is represented as an internal function (see the top sub-process in `Activate grid`), because its execution is not triggered by any external event. The model specifies that each iteration takes one unit of simulation time. We can choose to implement the loop explicitly, as shown in the figure, or implicitly by using a BPMN loop symbol (see Figure 3). The model specifies three external functions. The first external function responds to any sugar enquiry message sent by a `SugarPerson`. This message is sent when a `Sugar-Person` is looking for a grid with the highest amount of sugar. A non-shaded icon in BPMN denotes adjective "catching" (or receiving), and a shaded icon denotes adjective "throwing" (or sending). In this example, when the external function receives the "sugar enquiry" message, it will send a "sugar info" message to the sender. The second external function is executed when a `SugarPerson` sends a message to indicate its intention to enter the grid. If the grid has been occupied by another `SugarPerson`, the request will be declined. Otherwise, the grid will send a message to grant permission and set its status to "occupied". Subsequently, the grid will wait for a harvest request from the `SugarPerson`. When the harvest request is received, the grid will send the amount of harvested sugar requested and reset its sugar level to zero. The last external function is executed when the current occupier leaves the grid. The grid will check whether the sender is the current occupier. If the sender is not the current occupier, the request will be ignored. Otherwise, the status of the grid will be set to "available".

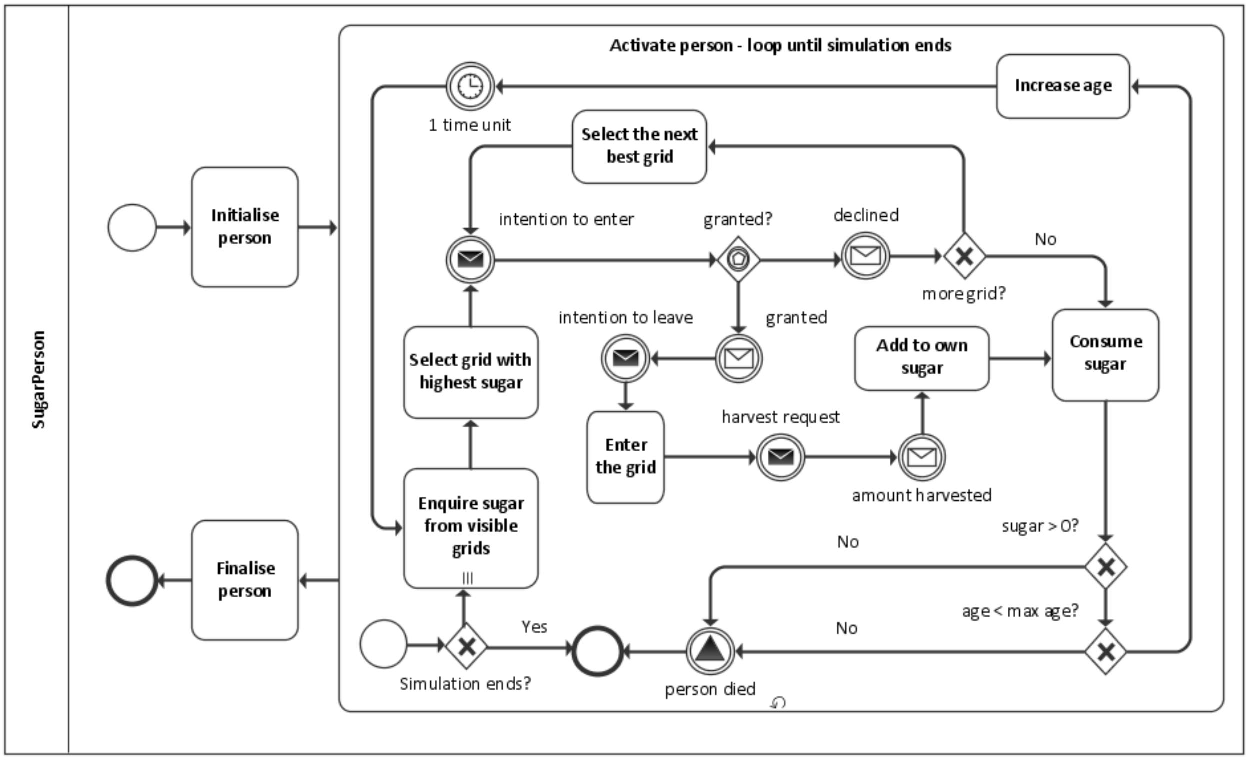


Figure 5: A person in a SugarScape model represented using a BPMN pattern

The agents (i.e., instances of `SugarPerson`) are created with different amounts of sugar in their possession. They have varying levels of vision quality and metabolic rates. Each agent will move to a grid with the highest amount of sugar and harvest all the sugar in the grid. If there is a tie, a grid is chosen at random. An agent can only move to any unoccupied grid within its range of vision. Each time, every agent consumes some of its sugar. The amount of sugar consumption depends on the metabolic rate of the agent. If the amount of sugar in its possession is not enough, the agent will die. Each agent will eventually die when it reaches its maximum age. When an agent dies, a new agent is created somewhere in the world to keep the population level constant but the sugar will not be passed on to the new agent (no sugar inheritance). The BPMN representation for a `SugarPerson` is shown in Figure 5. Since a `SugarPerson` has an internal function without any external function, we can remove one sub-process layer in the diagram. The internal function starts with a `SugarPerson` sending enquiries to grids within its visibility region. The parallel bar at the bottom of the activity indicates that the activity is performed a few times on a list of items, in this case a list of visible grids. The next steps follow the behaviour explained earlier. The whole process will be repeated until the simulation ends, as indicated by the loop icon at the bottom of the sub-process `Activate person`.

Inter-agent communications and the interaction between an agent and its environment can be represented using message flows. The interaction between two agents can be represented simply by connecting the relevant flow object (except for a gateway) in one agent to another flow object (except for a gateway) in the other agent. The same applies for the interaction between an agent and its environment. In the examples given in Figures 4 and 5, we simply connect each throwing event to its corresponding catching event. BPMN also allows us to view the interactions at the higher abstraction level using a conversation diagram, as shown in Figure 6.

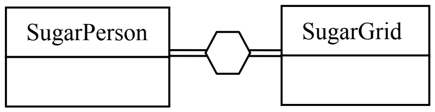


Figure 6: Conversation diagram showing a high-level view of interactions in a SugarScape model

4 CONCLUSION

Although BPMN is designed as a process-oriented modelling language, we have shown that BPMN can be also used as an agent-oriented modelling language. To help modellers use BPMN to represent an ABS model, we have proposed a BPMN pattern. The BPMN pattern provides a visual representation of ABS models where agents and their environment are specified as a set of internal functions, external functions and attributes. This paper does not claim that the proposed BPMN pattern can be used to represent all ABS models. However, the proposed pattern provides a visual representation equivalent to DEVS' internal and external functions which have been used in a number of studies to represent ABS models.

BPMN has some limitations in representing ABS models which need to be addressed in future research to make it applicable to a wider range of ABS models. First, representation of the environment is limited. It is difficult to model an environment that cannot be easily represented as a set of internal and external functions. Secondly, BPMN assumes an implicit queuing for each activity. Hence, it is lacking in the representation of the resources needed for an activity to start. Thirdly, the structure of agents represented in BPMN is static. Hence, it is difficult to represent changes of structure in the model, such as adding new BPMN activities or removing existing activities while the model is running. Finally, historically, BPMN does not provide detailed representation of data. Hence, the representation of complex attributes of an agent (such as memory, perceptions, beliefs and desires) is not adequate.

ACKNOWLEDGMENT

The author is grateful to Professor Mike Pidd (Lancaster University) for summarizing the feedback from the audience on the paper presented at the last Winter Simulation Conference (i.e., Onggo and Karpat 2011).

REFERENCES

Alan, R.J. 2010. "Survey of Agent-Based Modelling and Simulation Tools." Science and Technology Facilities Council, UK. http://epubs.stfc.ac.uk (Accessed April 3, 2012).

Bauer, B., J. P. Müller, and J. Odell. 2001. "Agent UML: A Formalism for Specifying Multiagent Interaction." In *Agent-Oriented Software Engineering*, Edited by P. Ciancarini, and M. Wooldridge, 91–103. Berlin: Springer-Verlag.

Bonabeau, E. 2002. "Agent-Based Modeling: Methods and Techniques for Simulating Human Systems." In *Proceedings of the National Academy of Sciences of the United States of America*, 99 (Suppl 3), 7280–7287.

Dávila, J., and M. Uzcátegui. 2000. "GALATEA: A Multiagent Simulation Platform." In *Proceedings of the International Conference on Modeling, Simulation and Neural Networks*. http://iies.faces.ula.ve/Amse2000/papers/simulation/MSNN-JDMU00.pdf (Accessed April 3, 2011).

Epstein, J. M., and R. Axtell. 1996. *Growing Artificial Societies: Social Science from the Bottom Up*. Washington, D.C.: Brookings Institution Press.

Holvoet, T. 1995. "Agents and Petri Nets." In *Petri Nets Newsletters*, Edited by O. Herzog, W. Reisig, and R. Valk, 49, 3–8.

Macal, C. M., and M. J. North. 2010. "Tutorial on agent-based modelling and simulation." *Journal of Simulation* 4:151–162.

Moldt, D., and F. Wienberg. 1997. "Multi-Agent Systems Based on Coloured Petri Nets." In *LNCS 1248: Application and Theory of Petri Nets*, Edited by P. Azéma, and G. Balbo, 82–101. Berlin: Springer-Heidelberg.

North, M. J., and C. M. Macal. 2007. *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation*. Oxford, UK: Oxford University Press.

Object Management Group. 2010. "Business Process Model and Notation (BPMN) version 2.0." http://www.bpmn.org (Accessed April 3, 2012).

Odell, J., H. Parunak, and B. Bauer. 2000. "Extending UML for Agents." In *Proceedings of the Agent-Oriented Information Systems Workshop*, 3–17.

Onggo, B. S. S. 2010. "Running Agent-Based Models on a Discrete-Event Simulator." In *Proceedings of the 24th European Simulation and Modelling Conference*, 51–55. Ostend, Belgium: Eurosis-ETI.

Onggo, B. S. S. and O. Karpat. 2011. "Agent-Based Conceptual Model Representation using BPMN." In *Proceedings of the 2010 Winter Simulation Conference*, Edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, 671–682. Los Alamitos, California: IEEE Computer Society Press.

Saple, A., and L. Yilmaz. 2006. "Agent-Based Simulation Study of Behavioral Anticipation: Anticipatory Fault Management in Computer Networks." In *Proceedings of the 44th annual Southeast regional conference*, 383–388. New York, NY: ACM Press.

Van der Zee, D. J. 2009. "Building Insightful Simulation Models using Formal Approaches – A Case Study on Petri Nets." In *Proceedings of the 2009 Winter Simulation Conference*, Edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 886–898. Los Alamitos, California: IEEE Computer Society Press.

Zaft, G., and B. P. Zeigler. 2002. "Discrete Event Simulation of Social Sciences: The XeriScape Artificial Society." In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics*.

Zeigler, B.P. 1976. *Theory of modeling and simulation*. New York, NY: John Wiley.

AUTHOR BIOGRAPHY

BHAKTI STEPHAN ONGGO is a lecturer in Business Process Modeling and Simulation at the Department of Management Science at the Lancaster University Management School, Lancaster, United Kingdom. He completed his PhD in Computer Science from the National University of Singapore and his MSc in Management Science from the Lancaster University. His research interests are in the areas of simulation methodology (modeling paradigms and conceptual modeling), simulation technology (parallel and distributed simulation) and business process modeling and simulation applications. His email address is s.onggo@lancaster.ac.uk.