White Paper

Agent-Oriented Modeling using BPMN with an example in Manufacturing

WP0127 | January 2014

Bhakti Stephan Onggo

Bhakti Stephan Onggo is a lecturer at the Department of Management Science at the Lancaster University Management School, United Kingdom. His research interests are in the areas of simulation methodology (modeling paradigms and conceptual modeling), simulation technology (parallel and distributed simulation, cloud-based simulation) and business process modeling and simulation applications. He has carried out a number of consultation projects in healthcare, manufacturing and public sector services including the European Commission and a UK Police department.

In an earlier white paper, I explained what an agent-oriented model is and how BPMN can be used to represent an agent-oriented model (Onggo 2013). In agent-oriented BPMN, we model organizations and their processes as a collection of interacting agents. Hence, we emphasize those agents who are involved in the activities or processes within an organization (or organizations in the case of inter-organizational interactions). Agents may include people and social/organizational constructs, such as departments. This will make us more aware of the objectives and concerns of people who require or provide services in the system being modeled, such as customers and key human assets. Agents may also be non-human, such as machines or software. In this case, agent-oriented modeling will help in the performance analysis of individual agents and the impact of their interactions on overall system performance.

In this white paper, I will elaborate the concept of agent-oriented modeling and the steps involved in developing an agent-oriented model using BPMN. To help the reader gain a better understanding of the various steps, I will use a case study based on one of my past projects for a manufacturing company.

Agent-Oriented Modeling

An agent-oriented modeling approach to represent the activities (or processes) within an organization (or across organizations) is a modeling approach in which the resulting model is formed by a set of autonomous agents that interact with other agents and their environment through a set of internal rules to achieve their objectives. This is different from a process-oriented modeling approach, in which the resulting model is formed by a set of interacting processes. These two approaches view the world from different perspectives. Hence, each of them may be useful for modeling different objectives. In some cases, they can be used complementarily to provide a more complete view of the system being modeled. The following sections explain the three main steps in developing an agent-oriented model.

Identifying key agent types

The first step is the identification of key agent types. Some agents may exhibit complex abilities (such as learning and making plans). These agents can usually make independent decisions in order to achieve certain objectives. Some agents can be very simple (e.g. they do not plan and do not learn). These agents may perform certain activities either in response to an external event (reactive agents) or perform autonomous activities (autonomous agents). Academics have different opinions about what an agent should be. However, this should not deter us from applying agent-oriented modeling in practice, given its many benefits. I have found that the understanding that agent-oriented modeling is a modeling approach that views the world as interacting agents is more useful in practice than worrying about whether our agents have to be complex or not before we can call them agents. In fact, we should keep our agent-oriented models as simple as possible.

Regardless of the complexity of agents, each agent has a set of characteristics (or attributes) and is able to perform certain activities (or behaviors). An agent type is a modeling construct that represents all the agents that can be identified by using the same set of characteristics and are able to perform the same set of activities. In a manufacturing company, the types of agent can be a production manager or even a machine (such as a laser-cutting or welding machine). A laser-cutting machine is an agent type because all laser-cutting machines can be identified using the same set of characteristics (such as make, model and laser power) and are able to perform the same activities (e.g. cutting and engraving). The identification of key agent types includes specifying the characteristics and behaviors of each agent type.

Identifying a relevant environment

Agents live (or are located) in an environment. When the environment significantly affects the behavior of agents, it may be necessary to include the environment in the model. If this is the case, we need to represent that environment. *Figure 1* shows a number of possible topologies for the environment. The environment can be spatial when physical locations are important. A physical location can be implemented as a location on a map (linked to a geographical information system) or a Euclidean space (when a map is not necessary or when the environment does not cover a large geographical area, e.g. a building). In some cases, a physical location can be simplified as a cell in a two-dimensional grid (cellular automata). The environment topology can also be in the form of a network when the connections between agents are important. The network can be logical (social network) or physical (road network).

An environment may have certain behaviors (e.g. a location may become less attractive). If an environment demonstrates autonomous behavior (something happens even in the absence of an action performed by any agent), we refer to this as a dynamic environment. In contrast, a static (or reactive) environment does not exhibit any behaviors without actions being initiated by an agent (e.g. an agent may pollute the environment). In a manufacturing system, I have found that the Euclidean space is usually suitable, especially when we need to include the movements of various elements such as a forklift or items on a conveyor belt.

Defining interactions

The final step is to define the relevant interactions between different agent types and between an agent type and the environment. In agent-oriented modeling, an interaction is implemented using message passing or signal broadcasting. An initiating agent will send a message to the target recipient (another agent or the environment). The recipient will then react, based on a predefined behavior, as defined in the previous steps. Similarly, an initiating agent may broadcast a signal and interested agents or the environment will then react to this signal based on a predefined behavior. In this step, we need to specify the communication lines and the types of messages or signals that will be circulated.

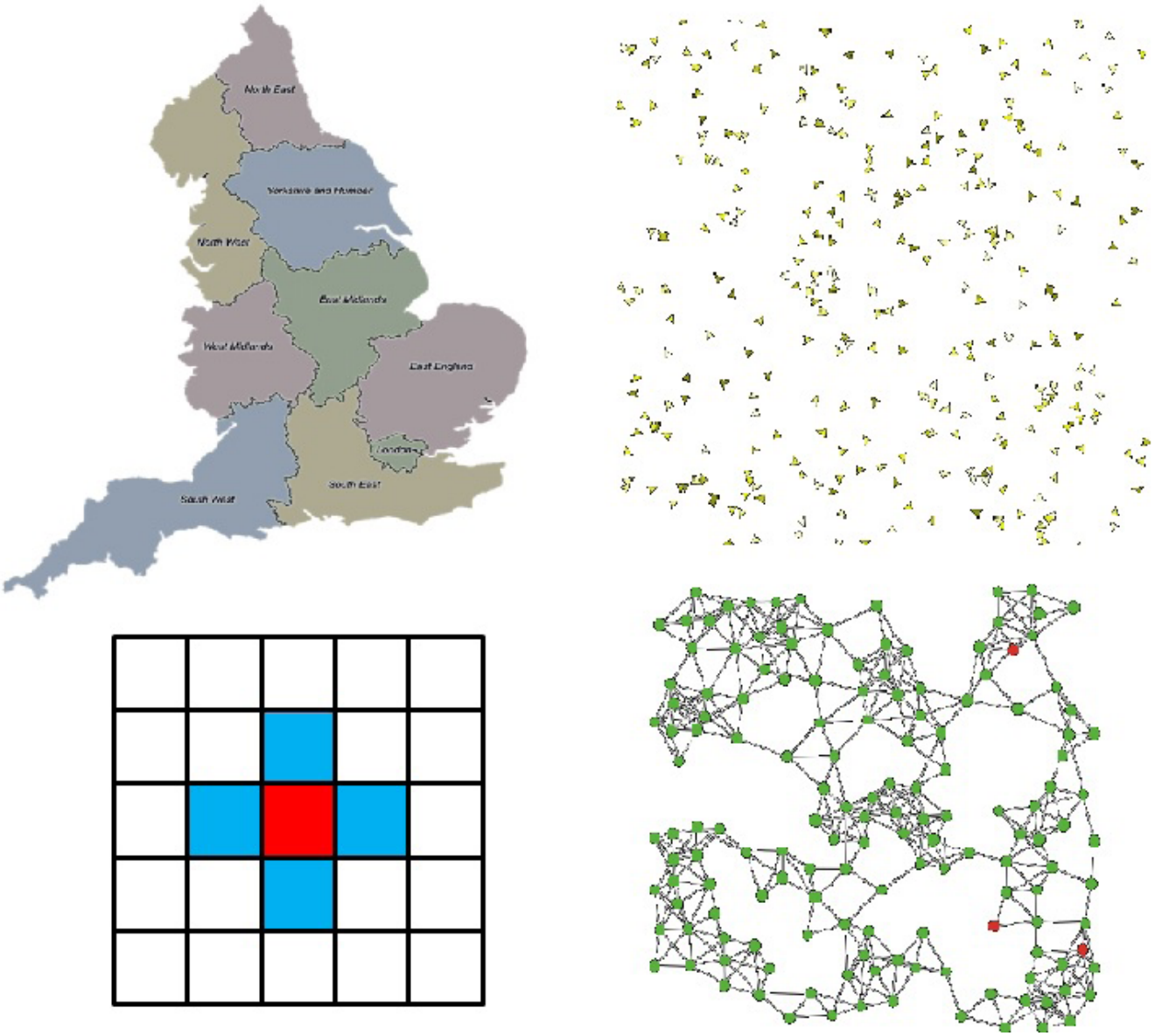

Figure 1: Environment's topologies (from top left, clockwise: GIS, Euclidean, network, cellular automata)

BPMN Pattern for Agent-Oriented Model

I have introduced a generic BPMN pattern that can be used to develop an agent-oriented model (Onggo 2012, Onggo 2013b). The pattern (*Figure 2*) should help practitioners to build an agent-oriented model more easily. Before an agent does its job, it typically starts with an initialization task. A finalization task may be needed before the agent leaves the system.

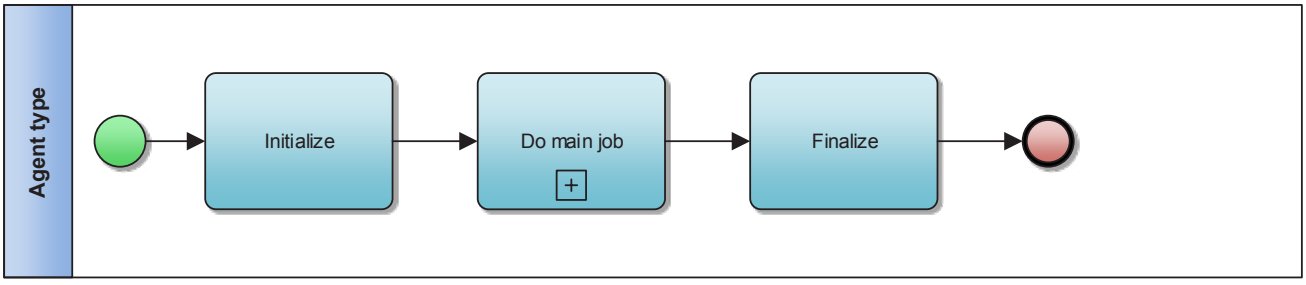

Figure 2: BPMN pattern for an agent

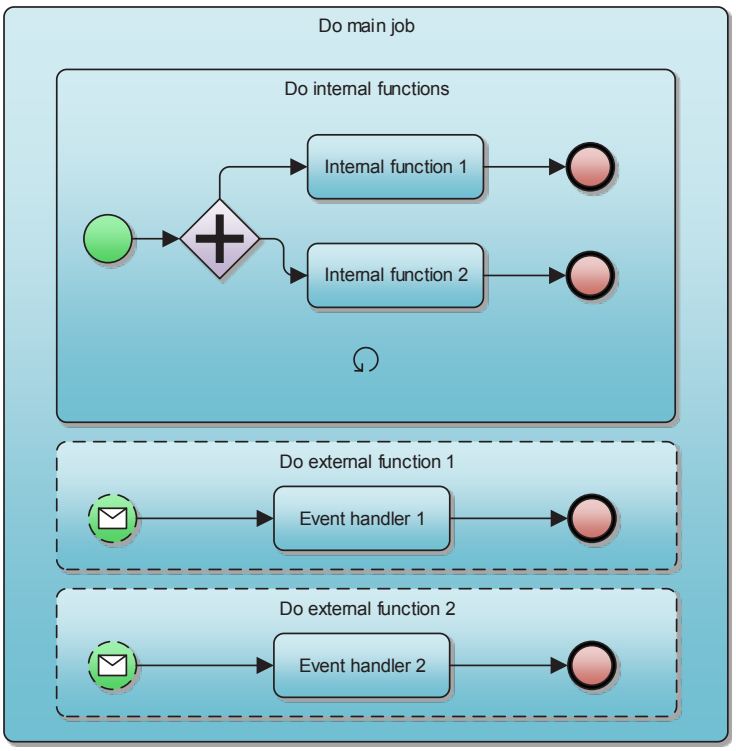

Figure 3: Agent's internal and external functions

The main job (or behavior) of an agent can be split into two types: internal (done based on conditions internal to the agent) and external (done in response to a message or signal). *Figure 3* shows the expanded sub-process "Do main job". The agent will keep checking (see the loop icon) to see if any of the internal functions can be carried out, depending on the conditions internal to the agent. External functions are represented as BPMN event sub-processes. These external functions are usually non-interrupting (represented by dashes around the sub-process). A non-interrupting event sub-process is activated in response to an event external to the agent, and its execution will not terminate the execution of the currently active internal or other external functions. If required, an external function can be modeled as an interrupting sub-process which will terminate the execution of other sub-processes.

Case Study

In this example, I will apply the generic BPMN pattern to map the processes in a manufacturing company. The company manufactures various types of metal containers. The production process for each product is relatively complex. Hence, given the limited space, I need to limit our discussion to parts of the production process and to one product only, which is called a metal box. The product is formed by a base plate, a top plate and a side plate, which will be folded to form four sides. A process-oriented view of the production process is shown in *Figure 4*. The figure shows that once the top plate, base-plate and side plate are ready, they will be welded to make a metal box.

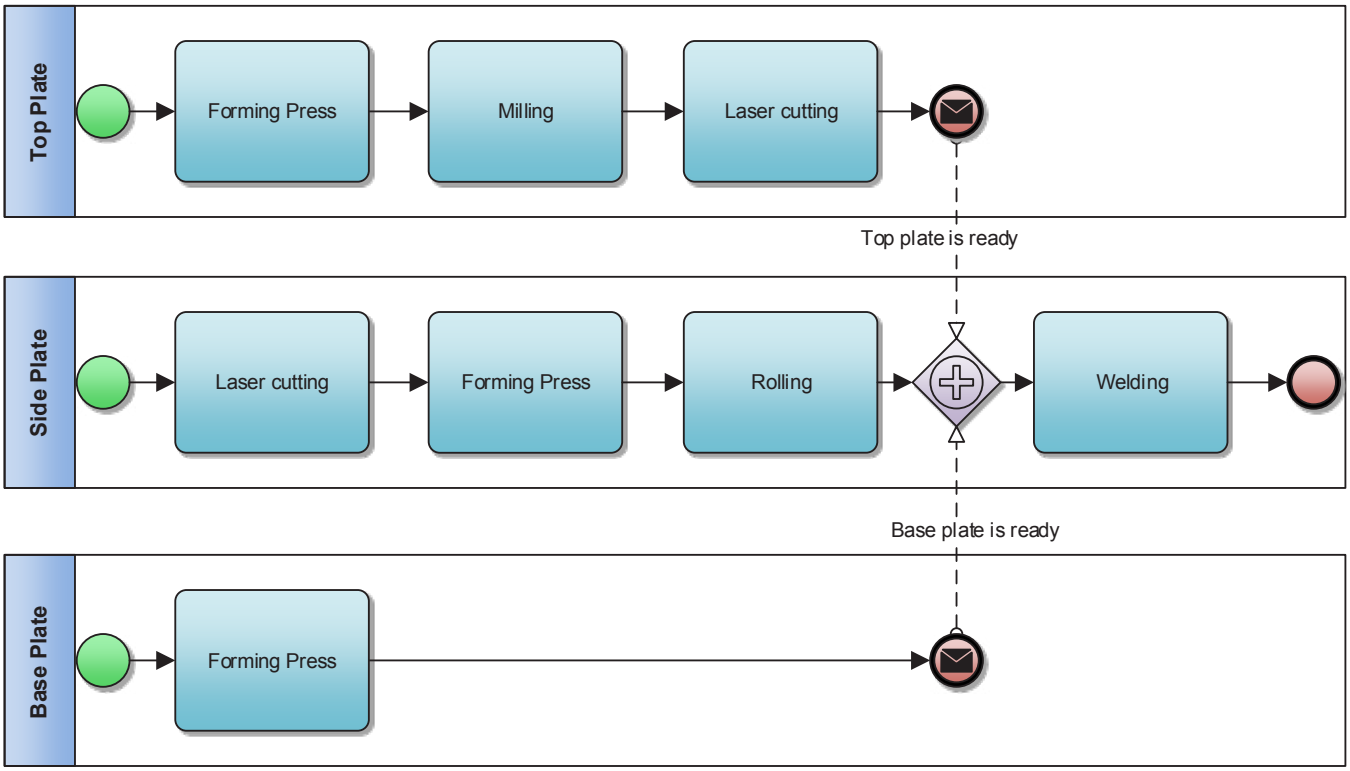

Figure 4: Process-oriented view of a manufacturing process

Agent types

In this example, we can consider the different machine types needed in the process to be agent types. Each machine type will have certain characteristics (e.g. a mean time between failures) and behaviors (e.g. setting up and processing). We can also consider operators as agent types, depending on our assumptions. If we assume that production managers are more concerned with the number of machines needed to increase production throughput, we can exclude operators from the model. In this case, the agent types in our model are: press machine, roller, milling machine, laser-cutting machine and welding. Typical characteristics that need to be represented in the model include set-up time, minimum and maximum numbers of items in a batch, processing time and mean time between failures. For simplicity, in this example, we assume we have enough forklifts to move items around the manufacturing plant.

Environment

Although we assume that we have enough forklifts, we still want to know whether the plant layout can support an increase in traffic. Hence, a Euclidean space environment is needed. In this case, the location of each machine, the paths and the plan layout will be included in the analysis. BPMN does not support a representation of the environment but most simulation software supports the use of a Euclidean space.

Interactions

In this example, interactions between agent types occur when a work-in-progress is ready for the next machine. Each machine type in the model is represented using a pool, as shown in the pattern (*Figure 2*). Given the limited space, in *Figure 5*, we only show the details inside the "Do main job" sub-process of each machine type. The figure shows that the laser-cutting machine has two main job types (represented as sub-processes): cutting the side plate and cutting the top plate. Each of milling machine, roller and welding has only one job type. The laser-cutting machine has three job types. All job types are shown as sub-processes, because they may consist of a number of tasks, such as set-up and processing. Each time a job is completed, a message is sent to trigger the next production stage. Each passing message represents the movement of a work-in-progress in the manufacturing plant. In this scenario, the agents are simple agents (e.g. they do not have the ability to learn). However, it is possible to include more complex behaviors (e.g. learning) into various agent types (e.g. operators, maintenance team, intelligent control systems).

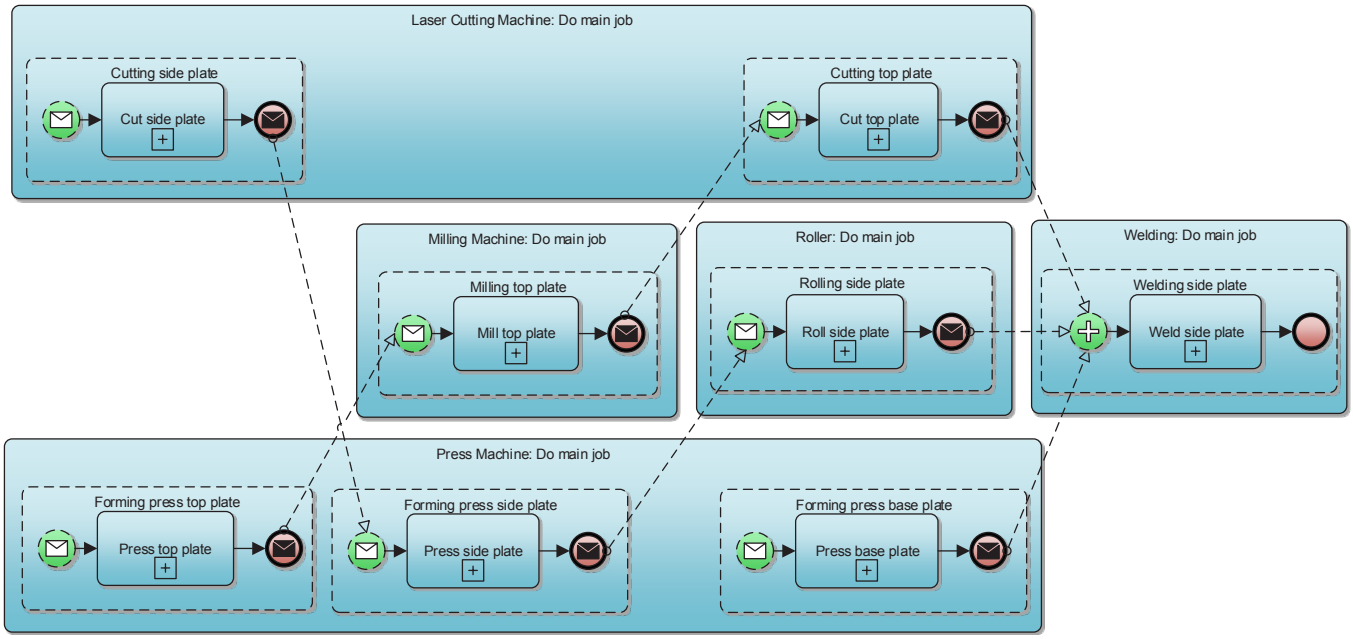

Figure 5: Agent-oriented view of the manufacturing process

Both process-oriented modeling and agent-oriented modelling are suitable for this example. The use of an agent-oriented model, as shown in *Figure 5*, emphasizes the various work done by each machine and the interactions between machines (or work centers). Hence, it has clear benefits such as allowing us to conduct analyses that require information at the machine level (e.g. machine utilization, production scheduling, and bottleneck analysis) and to analyze the impact of interactions on overall performance (e.g. production throughput).

Conclusion

This work paper is the continuation of an earlier white paper (Onggo 2013a). In this white paper, I have elaborated and explained agent-oriented modeling and applied it to a case study in manufacturing. To build a model using an agent-oriented approach requires three main steps: identifying agent types, identifying a relevant environment and specifying interactions.

I have also proposed a generic BPMN template that can be used to make the development of an agent-oriented model easier. People have identified the need to integrate BPMN and simulation, because simulation is one of the best techniques for analyzing and designing a business process model. We have seen that the number of simulation software packages that support BPMN has increased. Although BPMN can only represent agent types and their interactions, most simulation software offers a facility to link a simulation model with the environment, such as Euclidean space or Geographical Information System.

References

Onggo, B.S.S. (2013a) Agent-Oriented BPMN. Orbus white paper series. Available from http://www.orbussoftware.com/downloads/white-papers/agent-oriented-bpmn/

Onggo, B.S.S. (2013b) 'Agent-Based Simulation Model Representation using BPMN', in Fonseca, P. et al. (Eds) Formal Languages for Computer Simulation: Transdisciplinary Models and Applications. IGI Global, pp. 378-399.

Onggo, B.S.S. (2012) 'BPMN pattern for agent-based simulation model representation', Proceedings of the 2012 Winter Simulation Conference, 9-12 December, Berlin, Germany. Los Alamitos, California: IEEE Computer Society Press. Available from http://informs-sim.org/wsc12papers/includes/files/con536.pdf

Orbus Software
3rd Floor
111 Buckingham Palace Road
London
SW1W 0SR
United Kingdom

+44 (0) 870 991 1851
enquiries@orbussoftware.com
www.orbussoftware.com

orbus software