Automatic Construction of Agent-based Simulation Using Business Process Diagrams and Ontology-based Models (Demonstration)

Donghun Kang, Zhen C. Bing, Wen Song, Zehong Hu, Shuo Chen, Jie Zhang, Hui Xi[a]
Rolls-Royce@NTU Corp Lab, Nanyang Technological University, Singapore
[a]Rolls-Royce Singapore Pte Ltd, Singapore
{donghun.kang, zcbing, songwen, zhu, zhangj}@ntu.edu.sg, hui.xi@rolls-royce.com[a]

ABSTRACT

In this paper, we present a tool for the business users to analyze different business scenarios using business process diagrams and ontology-based models. The business scenarios involve different types of entities where the business process diagrams are suitable for describing entities' behaviors. The ontology-based model is proposed to capture entities' attributes and their relations in a hierarchical manner. The tool can automatically construct agent-based simulation models, which can be executed instantly on the agent-based simulation engine without the help of software developers.

Keywords

Process-oriented Agent-based Modeling and Simulation; Business Process Model and Notation; Ontology-based Model

1. INTRODUCTION

Timely and accurate decision making in the presence of uncertain and plentiful information is a key enabler for keeping the companies competitive. Business users heavily rely on software developers in constructing agent-based simulation models to analyze different business scenarios that change over time to cope with their stakeholder's needs. Every change in business scenarios may lead to the implementation of new models, which is time-consuming and may cause a delay in decision support.

To tackle this problem, business process diagrams have been introduced to the agent-based modeling [3, 5, 6]. Business Process Model and Notation (BPMN) [8] is one of such business process diagrams, which has been widely used in the business process modeling. BPMN can depict the flow of activities and their interactions among different types of entities that exist in business scenarios, which are easily understood by business users [5, 6].

The process-oriented agent-based modeling approaches in [3, 5, 6] transform the business process diagrams into the source codes of the agent components, which are not executable agent-based simulation models. This results from the incomplete description of the business scenarios where the business process diagrams including BPMN can only describe the behaviors of the entities. The attributes of each entity and the relations among the entities, so-called *organizational information*, are required to construct a complete and executable agent-based simulation model.

In this paper, we will introduce an ontology-based model to capture the organizational information, named *Operational Entity Model* (OEM) and present a two-phase transformation to construct a complete and executable agent-based simulation model using BPMN and OEM models. We will also provide the problem scenario for the demonstration of our implementation. A demonstration video of this paper can be accessed via `https://youtu.be/AQLrR9W2JGw`.

2. PROCESS-ORIENTED MODELING

Business scenarios involve different types of entities where BPMN is suitable for describing entities' behaviors and their interactions. However, the organizational information cannot be captured solely by BPMN. For example, in the team organizational modeling use case [4], the team formation and team members' attributes cannot be specified in BPMN.

We propose the Operational Entity Model (OEM), which is an ontology-based model to capture the organizational information. OEM specifies the entities that exist in the business scenarios and their relations in a hierarchical manner. In OEM, the *entity node* represents an entity with a set of attributes indicating the specific features of the entity. An intermediate node can be regarded as either an entity or a collection of entities. The OEM Modeler in Figure 1 supports the graphical editing of the OEM models.

The commonly-used entities and their relations for each business domain can be defined by the *OEM template* that consists of entity-type nodes and their hierarchical relations. The concept of OEM template is borrowed from *System Entity Structure* (SES) [13], which is first introduced by Zeigler [11] as a high-level ontology framework for representing the elements of a system and their relationships in a hierarchical manner. SES has been mostly used for defining the meta models in the field of simulation modeling [9, 10, 12]. With the OEM template, the user can easily create an OEM model by providing the admissible structure of entities according to the hierarchy of entity types in OEM template.

3. TWO-PHASE TRANSFORMATION

The transformation of BPMN and OEM models into an executable agent-based simulation model consists of two steps, *automatic code generation* and *model instantiation* as shown in the Transformation module of Figure 1. The agent-based simulation model transformed from the two models will be executed on the agent-based simulation engine, Repast [7].

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

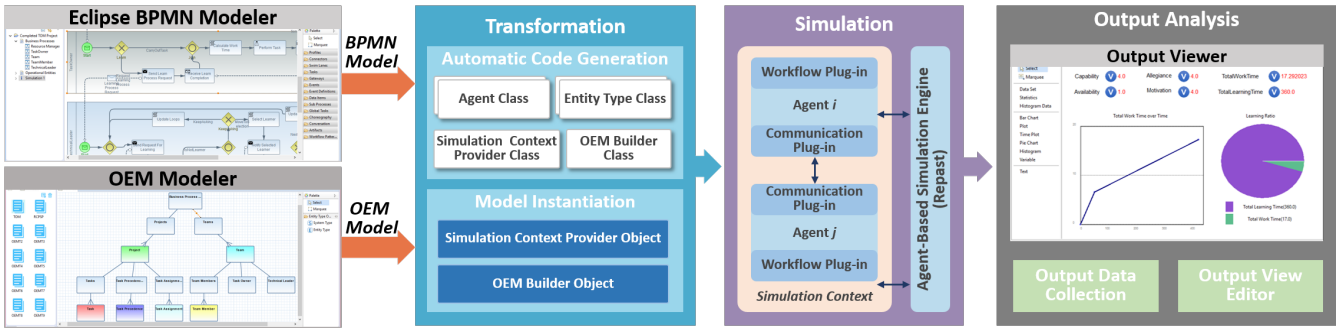

Figure 1: System architecture of proposed process-oriented modeling and simulation tool

3.1 Automatic Code Generation

The automatic code generation focuses on the generation of Java source code required to construct the agent-based simulation model from the BPMN and OEM models. Its outputs include `Agent` Java classes for each participant of the BPMN model, `Entity Type` Java classes for each entity type of the OEM model, `OEM Builder` Java class for the OEM model, and `Simulation Context Provider` Java class.

The `Agent` Java class contains the Agent Variables and the Agent Activity methods. The Agent Variables are mapped from the properties of a participant in the BPMN model. The Agent Activity methods execute the flow nodes in the BPMN model where their bodies vary depending on the types and parameters of the corresponding flow nodes. Detailed mapping mechanisms from the flow nodes to the Agent Activity methods are omitted here for brevity.

The `OEM Builder` Java class allows the agent to access entities during the simulation. The `Entity Type` Java class provides the getter and setter methods for the attributes of each entity type used in the OEM model.

3.2 Model Instantiation

The model instantiation focuses on creating an executable agent-based simulation model that can be executed instantly by the business user on Repast. This step starts with the in-memory compilation of all the Java classes generated from the previous step using Java reflection. Then, it instantiates the `Simulation Context Provider` Java object that contains a method to return the `Context` object. Here, the `Context` object is required by Repast to provide all the agent objects that participate in the simulation. In this method, all the `Agent` Java objects that correspond to the entities of the OEM model are created from the respective `Agent` Java classes. The values of Agent Variables that each `Agent` Java object has are assigned from the values of attributes that the corresponding entity of the OEM model has.

4. SIMULATION

In this paper, lightweight workflow engines (*workflow plug-ins*) are attached to each agent to manage the flow of activities within an agent as shown in the Simulation module of Figure 1. The use of Java threads to control the flow of activities in some approaches [5, 6] can cause heavy loads on threading and exceptional errors. In contrast, the lightweight workflow engine can efficiently control the complex flows of activities with event handling without the heavy use of Java threads. To support the message-based interactions among the agents, the *communication plug-ins* are also attached to each agent and provide subscribe-publish-based communication.

5. DEMONSTRATION

The common problem in industrial projects is how to organize multi-functional teams to meet a project's targets [4]. We will demonstrate how our approach can make it easier to model team members' behavior to estimate time and cost outputs without programming expertise. As depicted in Figure 2, a project consists of a set of tasks. Each task is associated with the *complexity* parameter that determines the level of difficulty in executing the task. An individual team member has the social-technical attributes, such as *capability*, *availability*, *allegiance*, and *motivation*.

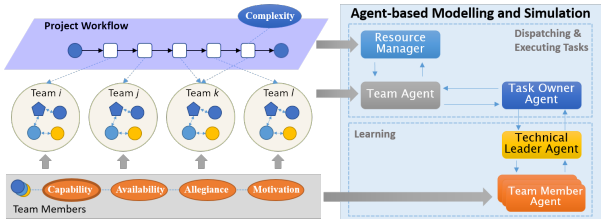

Figure 2: Application scenario of team organization

A task in the project is assigned to a team with a task owner, technical leader, and some team members. The technical leader helps team members to learn a new skill or to improve the skill so as to carry out the assigned task.

In the demonstration, first, we model team members' behavior using the Eclipse BPMN2 Modeler [1], a graphical BPMN modeling tool. Then, we model the organizational information using the OEM Modeler that we developed based on the Eclipse plug-in development environment [2]. The output data collected during the simulation is rendered by the Output Viewer with various types of charts (see the right most sub-figure in Figure 1). Business users can easily analyze the effects of different task assignments by changing the attributes of respective entity nodes in the OEM Modeler.

Acknowledgments

This work was conducted within the Rolls-Royce@NTU Corporate Lab with support from the National Research Foundation (NRF) Singapore under the Corp Lab@University Scheme.

REFERENCES

[1] Eclipse BPMN2 modeler. `https://www.eclipse.org/bpmn2-modeler/`, 2017.

[2] Eclipse plug-in development environment. `http://www.eclipse.org/pde/`, 2017.

[3] G. Caire, D. Gotta, and M. Banzi. Wade: A software platform to develop mission critical applications exploiting agents and workflows. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008) - Industry and Applications Track*, pages 29–36, 2008.

[4] P. Dutta, C. Pepe, and H. Xi. Analyzing organization structures and performance through agent-based socio-technical modeling. In *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems - Volume 2*, pages 39–53, 2015.

[5] T. Küster, A. Heßler, and S. Albayrak. Towards process-oriented modelling and creation of multi-agent systems. In *Proceedings of the 2nd International Workshop on Engineering Multi-Agent Systems (EMAS 2014)*, pages 163–180, 2014.

[6] T. Küster and M. Lützenberger. An overview of a mapping from BPMN to agents. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, pages 1783–1784, 2015.

[7] M. North, N. Collier, and J. Vos. Experiences creating three implementations of the Repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation*, 16(1):1–25, 2006.

[8] Object Management Group. Business process model and notation (BPMN). `http://www.omg.org/spec/BPMN/2.0.2/`, 2014.

[9] J. F. Santucci, L. Capocchi, and B. P. Zeigler. System entity structure extension to integrate abstraction hierarchies and time granularity into DEVS modeling and simulation. *Simulation: Transactions of the Society for Modeling and Simulation International*, 92(8):747–769, 2016.

[10] A. Schmidt, U. Durak, and T. Pawletta. Model-based testing methodology using system entity structures for MATLAB/Simulink models. *Simulation: Transactions of the Society for Modeling and Simulation International*, 92(8):729–746, 2016.

[11] B. P. Zeigler. *Multifaceted Modeling and Discrete Event Simulation*. Academic Press, 1984.

[12] B. P. Zeigler. Discrete event system specification framework for self-improving healthcare service systems. *IEEE Systems Journal*, PP(99):1–12, 2016.

[13] B. P. Zeigler and P. E. Hammonds. *Modeling and Simulation-based Data Engineering*. Elsevier Academic Press, 2007.