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
Rolls-Royce@NTU Corp Lab, Nanyang Technological University, Singapore
Rolls-Royce Singapore Pte Ltd, Singapore
{donghun.kang, zbing, songwen, zhu, zhang}@ntu.edu.sg, hui.xi@rolls-royce.com

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/AGLzR9W2JGw.

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].
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 execute the flow nodes in the BPMN model have to be mapped to the respective Agent Java classes. The values of Agent Variables that each Agent Java object has are assigned from the values of attributes that correspond to the entities of the OEM model.

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.
REFERENCES