Ontology and automatic code generation on modeling and simulation

Youcef Gheraibia
Computing Department
University Md Messadia
Souk Ahras, 41000, Algeria
youcef.gheraibia@gmail.com

Abdelhabib Bourouis
Computing Department
University of Larbi Ben M'Hidi
Oum El Bouaghi, 4000, Algeria
habib.bourouis@univ-batna.dz

Abstract— **In this paper we present a new approach for using semantic web technologies in modeling and simulation. In recent years ontologies have been used popularly in many fields to represent and structure their concepts. This work is an attempt to create a specific ontology for the process oriented discrete event simulation domain. The ontology instances represent the model instances. This instance described in XML format and then transformed to another form that is used to generate the simulation code via XSLT rules. The code is generated according to the open source library Japrosim. The objective of this work is to enhance interoperability and automation of the transition from the ontology to the code execution.**

Keywords-component; Ontology, Semantic Web, modeling and simulation, code generation, interoperability.

I. INTRODUCTION

The web of today is basically syntactic and the interpretation of the resources content is available only to humans, the machine addresses only document structure. Generally there is no rigid method for classifying semantic content of Web documents. This is one of the reasons of the development of semantic web. The Semantic Web is an extension of the syntactic web, we add semantics layer, its objectives to make the semantic content of Web resources accessible by the software agents through a set of languages, meta-data and formal knowledge representation tools. One of the rich knowledge representation tools is the ontology, which is a set of concepts based on the meaning of an information field [9].

The use of ontologies is now become widespread because many fields have used this technology like medicine, architecture, geography and computing [6]. The simulation is one of the computing fields that can make a successful exploitation of ontologies, especially during the first stages of a simulation project that is the formulation of the problem and develop the conceptual model.

XML (eXtensible Markup Language) is a computer language that allows structuring of information and promotes the exchange of information on the Internet. It ensures high interoperability in the exchange of models [4]. This benefit has motivated to define an XML dialect (noted XPISM) for describing discrete events simulation models according to the process approach.

Today, technology developing and ambitions of researchers are increasing. One of these ambitions is the automatic code generation from a conceptual model, which is not something easy. It allows avoiding several error sources, save the time, and verifying the transition from conceptual model to the executable model with formal methods. With the definition of transformation rules, the passage from an XPISM instance to executable simulation code (especially Java) became possible, these rules are written using XSLT (extensible Stylesheet Language Transformation), which is a declarative language.

By carefully examining the course of our work, we build the domain ontology, the construction of the scheme XPISM and finally defining the XSLT rules to build the conceptual model and executable code generation, it will be obvious to see the interest of this work, which aims to enhance interoperability, define a standard vocabulary to represent concepts of simulation model, and full automate of the process of modeling and simulation.

In the next section, we present the motivation and use of ontologies in modeling and simulation. Section 3 presents the related work on the use of ontologies in process-oriented discrete event simulation. Section 4 is devoted to develop PIDESO ontology. Section 5 deals with devolved XML schema XPISM. In Section 6, we present the whole passage from domain ontology to exactable model and in section 7 a conclusion is given which focus on the path from the domain ontology to simulation executable.

II. ONTOLOGIES IN SIMULATION.

Simulation knowledge representation approaches require the handling of highly structured knowledge, including ontologies. Ontologies are useful in the process of modeling, simulation and analysis cycle, particularly in the problem analysis and in the conceptual model development [12]. One of the motivations for modeling and simulation is the decomposition of the model of the whole system into smaller components and easy manipulated to distribute the development effort of the model to different working groups, and also in communication between deferent groups work [10]. Ontologies play an important role in the development

process of the conceptual model. This occurs mainly in two ways; capture the needs and the formulation of the conceptual model [15].

A. Identifying needs

The simulation model is often designed to achieve a set of modeling objectives or respond to a set of questions. The ambiguity of natural language is always a problem but ontologies can help to facilitate the different tasks as described below [11]:

- Provide a mechanism to interpret and understand the description of the problem.
- Assist the designer to capture the user requirements (the information necessary and sufficient for the model set).

B. Conceptual model formulation

The process of constructing the conceptual model includes the following activities: acquisition and analysis system description, identification and classification of goals in modeling, determining the roles of system objects, boundaries and level of abstraction and the determining the model structure and logic of it, [11], [10].

1) Acquire and analyze the system description :

Ontologies can facilitate the identification of inconsistency and incompleteness in a description of a system. For example, ontologies can be used to interpret the descriptive information on system objects.

2) Identify and classify targets modeling :

An important step in developing the conceptual model is to determine the specific goals of the simulation study based on "the application of decision data" provided by the domain expert. This process of reasoning uses knowledge of the report and stresses in the system description (these are interpreted using domain ontologies).

3) Determine the roles of objects, boundaries and level of abstraction:

The following tasks are performed once the specific aims of the analysis were established.

- Establish the boundaries of the model: the first activity in the development of the conceptual model is to choose the part of the system under study.
- Establish the level of abstraction: A simple rule for determining the appropriate level of abstraction is to "include only those elements of a system that is able to meet the objectives and content and the level of abstraction as up ".
- Identify the roles of objects in the model: this step is to determine the model objects (resource, entity …), and the role of each object, for example queue 'x' is the activity therein '.

4) Determine the model structure and logic

The model structure and logic refer to the characterization of the relationship between activities in the model. An activity represents the dynamic behavior that occurs when objects interact one over the other. Ontologies play a key role in eliminating the ambiguity of interpretation of information contained in the description of the system to correctly understand the logical flow of objects and the decision logic in real-world process.

III. Similar works

A. PIMODES

The Process Interaction Modeling Ontology for Discrete Event Simulation (PIMODES) [7], is a general ontology for the domain of process-oriented discrete event simulation, it is using ontologies to formalize a representation language for process-oriented discrete event simulation models. This formalization is intended to lead to a formal specification of concepts for the automatic interpretation of these concepts.

PIMODES proposes a set of classes for models representation, each model must be identified with a single identifier and a clear description (annotation), the structure of the model represented by a set of processes, a set of activities and a graph of traffic control of entities [7]. PIMODES offers a strong structure and a clear chain of concepts but the management of activities in the same level as the process, causes increased complexity in synchronizing the activities of the control graph.

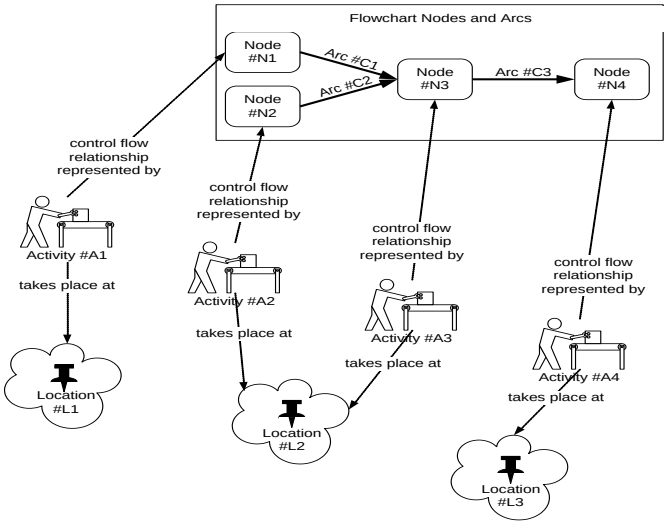

Figure 1. Activities control graph of PIMODES [PER 07]

B. PIMODEL

The Demo (The Discrete Event Modeling Ontology) [13], is an ontology for domain of discrete event simulation. OWL (OWL: Ontology Web Language) has been used to define more than 60 classes and several properties associated with them. This ontology consists of four main parts: Concept Model, DeModel, Model Component Model and Mechanism. DeModel is also divided into four parts representing the simulation approaches, State Oriented Model, Activity Model

Oriented, Event-Oriented Model and Process Oriented Model [13]. PIModel is the DEMO class that focuses on process-oriented simulation models. Models can be represented with OWL instances that can undergo treatment to achieve the automatic programmed model.

C. Automatic generation of simulation code

Automated code generation is a difficult task that falls within the agile development movement. The generation of the code is done automatically from a set of information (model, meta-data ...). In the simulation, the model can be programmed directly from the encoded conceptual model using translation rules (type: IF THEN) with high-level language or languages of the simulation. These rules are written in software which does not facilitate their maintenance in the event of changes in the target language. At this level there is a lack of interoperability is a low reuse [3].

IV. Process Interaction Discrete Event Simulation Ontology (PIDESO)

Process interaction discrete event simulation ontology (PIDESO), it's an ontology specific to represents the concepts of process-oriented discrete event simulation domain. It consists of a set of classes organized in different levels in a hierarchy very clear to help designers to build their models without ambiguity and in a formal framework provided by OWL. A model is a set of processes where each is a set of activities and controlled by a control graph [1]. PIDESO plays an important role in the exchange of simulation models by providing a standard vocabulary for communication and reuse. The construction of ontology PIDESO passes through three stages, Conceptualization, Operationalization and ontologization [7], [12].

A. Conceptualization

This step allows reaching an informal model, semantically ambiguous and therefore usually expressed in natural language. This step is done to identify concepts and relationships between these concepts from raw data, these concepts to describe informally cognitive entities of simulation domain. The ontology in this research is divided into two levels. The first one is to identify the major elements of the system: model, processes, activities (general view) and the graphs of control. The second level is to represent the elements of characterization of first-level classes such as the types of each activity, the components of graph control, additional information on the model ... etc.

- First level:
  - Model
  - Process
  - Graph Activity
  - Activities
- Second level:
  - Project Description
  - Attributes Feature
  - Feature Type
  - Resource
  - Variable
  - Arc
  - Node
  - Transition
  - Connecting Activities
  - Creation of entities Activities
  - Change of activities
  - Queue Activity
  - Duration of activity
  - Manage resource

B. Ontologization

This step leads to a semi-formal. This partial formalization facilitates its subsequent representation in a formal language and fully operational. Here is a diagram used to specify each class of the ontology. Figure 1 shows the classes in the ontology PIDESO and semantic links between different concepts [8].

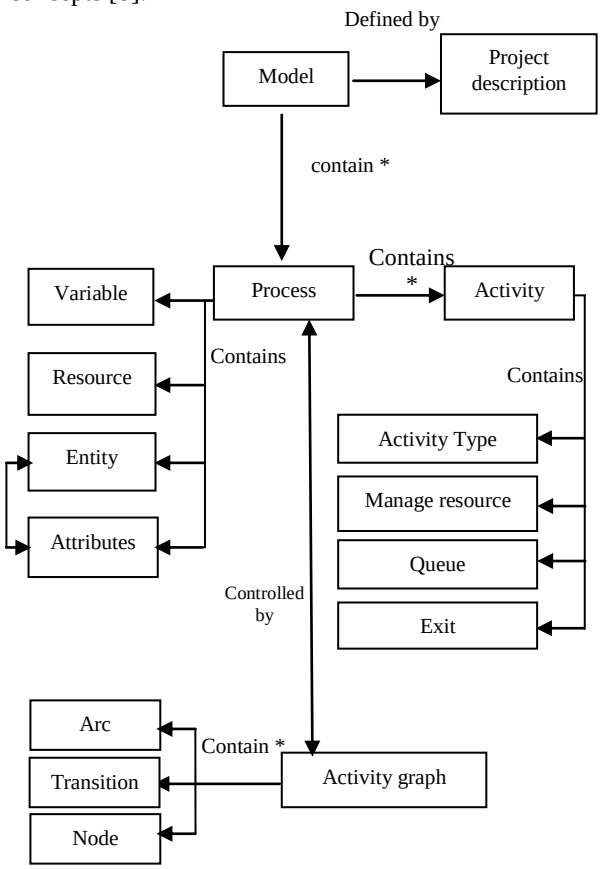

Figure 2. Ontology classes diagram (PIDESO)

C. *Operationalization*

Operationalization aims to have a formal structure of concepts and relations between them, represented as a web language OWL classes using an ontology editor Protégé-2000 [13]. In the knowledge model of Protégé-2000 ontologies consist of a hierarchy of classes that have properties (slots), which may themselves have certain properties (facets). The edition of these three types of objects is with a GUI, without need to express what was specified in an operational target language, it is enough to fill out the forms corresponding that we want to specify.

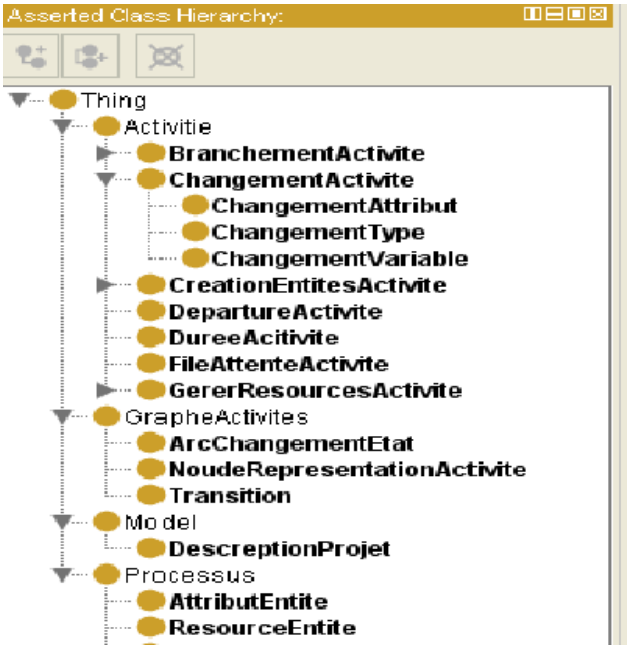

Figure 3. Ontology classes representation with Protégé-2000

V. Extensible Process Interaction Simulation Model (XPISM schema)

An XML document is well formed if it adheres to XML syntax rules that are explicitly designed to make documents easily interpreted by a computer, and an XML document is valid if it adheres to the rules described in a document as an Associate DTD or schema.

We have defined an XML dialect to describe process-oriented discrete event simulation models, named XPISM (extensible process interaction simulation model). It describes the simulation models in a hierarchy. The model consists of a project description (sets the name of analyst / designer / author, title of project, etc ....) And the whole model process components such as activities, resources associated with each activity, variables, attributes associated with such entities and the graph of activities [14]. A simulation model for discrete event oriented process consists of a set of entities that flow through the system. Entities arrive according to a probability distribution and perform activities that are supported by resources and managed queues [5]. These model elements are represented by an XML schema by limiting the principles of discrete event simulation, process-oriented. Each class of XPISM is associated with a class of ontology (PIDESO). The OWL ontology instances are transformed into another form for code generation according to the scheme XPISM. This intermediate representation simplifies the transformation of the conceptual model into a model program, provides a clear structure of the model concepts.

```
- <xsd:element name="Model">
  - <xsd:ComplexType>
    - <xsd:Sequence>
        <xsd:element ref="DescreptionProjet" maxoccures="1" minoccures="1" />
        <xsd:element ref="Processus" maxoccures="unbounded" minoccures="1" />
      </xsd:Sequence>
      <xsd:attribute name="ID" type="xsd:string" use="required" />
    </xsd:ComplexType>
  </xsd:element>
  <!-- ////////////////////****////////////////////// -->
  <!-- ////////////Descreption de projet/////////////// -->
  <!-- ////////////////////****////////////////////// -->
- <xsd:element name="DescreptionProjet">
  - <xsd:ComplexType>
    - <xsd:all>
        <xsd:element name="Nom_d_analyseur" type="xsd:string" maxoccures="1" minoccures="1" />
        <xsd:element name="titre_De_Projet" type="xsd:string" maxoccures="1" minoccures="1" />
      </xsd:all>
      <xsd:attribute name="ID" type="xsd:string" use="required" />
    </xsd:ComplexType>
  </xsd:element>
  <!-- ////////////////////****////////////////////// -->
  <!-- //////////Les ingrediants de processus//// -->
  <!-- ////////////////////****////////////////////// -->
- <xsd:element name="Processus">
  - <xsd:complexType>
    - <xsd:sequence>
        <xsd:attribute name="ID" type="xsd:string" use="required" />
        <xsd:element name="Nom_De_Processus" type="xsd:string" />
        <xsd:element ref="GrapheActivites" maxoccures="1" minoccures="1" />
        <xsd:element ref="ResourceEntite" maxoccures="unbounded" minoccures="1" />
        <xsd:element ref="TypeEntite" maxoccures="unbounded" minoccures="1" />
        <xsd:element ref="AttributEntite" maxoccures="unbounded" minoccures="1" />
        <xsd:element ref="VariableProcessus" maxoccures="unbounded" minoccures="1" />
        <xsd:element ref="Activite" maxoccures="unbounded" minoccures="1" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

Figure 4. XPISM Schema

VI. Automatic generation of Java code from the ontology instance

A. *From PODESO instance to XPISM instance*

Process-oriented discrete event simulation models can be described as instances of OWL (Ontology instance). The idea is to change the shape of the model (OWL instances) to another more appropriate form (XPISM instance). This new description of the model (XPISM pending) is an intermediate representation for the executable model. An XSLT stylesheet containing the transformation rules allows instances of classes in ontology elements XPISM by an XSLT processor. XSLT rules are based on a locator called XPath to identify nodes in the source document (OWL instance) and build a result document (instance XPISM). This transformation is independent of all simulation languages or programming that ensures and strengthens interoperability and facilitates reuse.

B. *From XPISM instance of java code*

Generating java code contains a main class containing a main () method that initializes the variables of the simulation

model reflects its original state (number of replications, the simulation time ... etc.). It also launches the first arrivals in the simulation, then it call the start method (start ()) that initializes the coordinator and the simulation begins. After initialization of the first arrivals of each process, must be defined for each body that represents its life cycle in the system. Each process contains its own resources, variables, queues… etc

```
Public static void main(String[] args)
 {
new Passenger().beginAfter(0.0);
new Employee().beginAfter(0.1);
new Avion().beginAfter(0.0);
SimProcess.sched.nbReplications = 1;
SimProcess.sched.time = 0.0;
}
```

Figure 5. Generated simulation code

C. *XPISM instance to java code*

Models of discrete event simulation can be described as an instance of the schema XPISM well organized according to the process approach (a model contains several processes each process has several activities etc ....). An XSLT stylesheet contains the transformation rules allow instances XPISM (conceptual model) to a java code. These rules are executed via an XSLT processor.

D. *Experiment of the result*

JAPROSIM, (JAva PRocess Oriented SIMulation) [2], Java is a framework for building discrete event simulators oriented process. The code generated by our application is directly executable on the machine using the java library JAPROSIM. Figure 5 shows an example of experimentation using JAPROSIM.

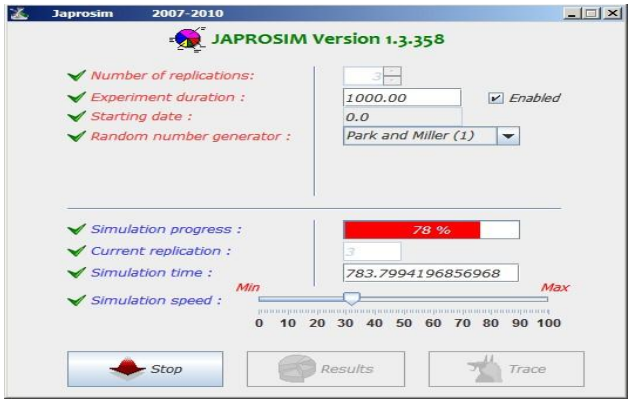

Figure 6. Experimentation of generated code

VII. CONCLUSIONS

PIDESO is a complete and comprehensive ontology for the process interaction discrete event simulation domain. It allows analysts and designers to improve their models with a semantic dimension. Models can be represented with an instance PIDESO and this instance will be transformed into another form more appropriate XPISM noted. Also, it is easy to obtain a conceptual model based on domain ontology. A set of rules defined XSLT transforms this model into an executable Java code based on the framework JAPROSIM.

The interest of our approach is summarized in three main points. First, the introduction of semantics in the simulation models, automatic generation of executable model, and the automation of the first steps of a simulation project. The result is certainly a gain in productivity, security development, enhanced interoperability and ease of maintenance and updating. To this is added clarity of approach and is now more rigorous.

We envision in the future work, to use the results to extend the code generation part to other simulation languages, given that the approach is independent of any simulation languages and simulation tools. And it is possible to provide new ontologies for other approaches to discrete event simulation, especially for events.

REFERENCES

[1] Bachimont B, “Engagement sémantique et engagement ontologique: conception et réalisation d’ontologies en ingénierie des connaissances”. Eyrolles, 2000.

[2] Bourouis A et Belattar B, “JAPROSIM: A Java framework for Process Interaction Discrete Event Simulation: JOURNAL OF OBJECT TECHNOLOGY Vol. 7, No. 1, pp. 103-119, Janry-February 2008 , http//www.jot.fm/issues/issue_2008_01/article3/

[3] Bourouis A et Belattar B, “Using XML IN SIMULATION MODELING automatique code generation for XML based models" CARI 2008 MAROC , pp. 101-109.

[4] Charlet, P. Laublet & C. Reynaud. “Web sémantique”. Octobre 2003.

[5] Fishwick P, “Handbook of Dynamic System Modeling “, Chapman & Hall/CRC, New York 2004.

[6] T.R. Gruber, “Toward principles for the design of ontologies used for knowledge sharing”. International Journal of Human Computer Studies. 1995.

[7] Lacy L, “Interchanging Discrete Event Simulation Models using PIMODES and SRML”. Proceedings of the Fall 2006 Simulation Interoperability Workshop.

[8] M.Leclere, F. Trichet & F. Furst, “Operationalising domain ontologies: towards an ontological level for the SG family”, in Foundations and Applications of Conceptual Structure, contributions to the International Conference on Conceptual Structures. 2002.

[9] T. Berners-Lee, J. Hendler & O. Lassila. “The Semantic Web”. Scientific American. 2001.

[10] Miller A et Fishzick,” ontolgies for modeling and simulation: issues and approaches”, Proceedings of the 2004 Winter Simulation Conference, pp. 259- 364.

[11] Miller A, Silver G, et Lacy L,” ontology based representation of simulation models following the processes interaction world of view “Proceedings of the 2006 Winter Simulation Conference,p 1168-1176.

[12] Natalya F. Noy et Deborah L. McGuinness, ” Ontology Development 101: A Guide to Creating Your First Ontology”. Stanford University, Stanford, CA, 94305, 2005.

[13] N. Noy & D. McGuinness “Ontology Development 101: A Guide to Creating Your First Ontology”, Stanford Medical Informatics Report, SMI-2001-0880. 2001.

[14] R. Studer, V. Benjamins & D. Fensel. “Knowledge Engineering: Principles and Methods”. In Data and Knowledge Engineering. 25, 1998.

[15] M.Uschold & M.Grüninger, “ONTOLOGIES: Principles, Methods and Applications”. Knowledge Engineering Review. 1996.