УДК 004.9 + 004.358

А.В. Прохоров, Е.М. Пахнина

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков

ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ОНТОЛОГИЙ ПРИ ВЗАИМОДЕЙСТВИИ АГЕНТОВ В СИСТЕМЕ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ПРОИЗВОДСТВЕННЫХ ПРОЦЕССОВ

Проанализированы особенности, связанные с использованием онтологий при взаимодействии агентов в системе имитационного моделирования производственных процессов; проведен анализ существующих подходов в области создания онтологий для дискретно-событийного имитационного моделирования систем; предложена и описана онтология для системы агентного имитационного моделирования; рассмотрены вопросы реализации онтологического подхода в виде прототипа знаниеориентированной системы имитационного моделирования производственных процессов.

Ключевые слова: *имитационное моделирование, интеллектуальный агент, мультиагентная система, база знаний, агентный подход, онтология, взаимодействие агентов.*

Введение

Наиболее эффективный анализ и прогнозирование динамических характеристик производственных систем обеспечивают средства, основанные на методах имитационного моделирования [5]. Для обеспечения гибких механизмов динамического поведения, автономности и адаптации отдельных компонентов имитационной модели актуально использовать интеллектуальные информационные технологий, реализуемые в виде агентно-ориентированных систем. Агентное моделирование предполагает, что модель включает множество взаимодействующих между собой и с внешней средой агентов – информационных (программных) элементов, которые имеют свои цели и задачи, внутреннее состояние и правила поведения [8]. Основным элементом программного агента системы, дающим ему возможность принимать решения, планировать действия, взаимодействовать с другими агентами, является база знаний, содержащая модели концептуальных понятий, отношений и правила для анализа и ситуативной ориентации [5]. В качестве средства структурирования и представления информации в таких системах используются онтологии [6].

Анализ последних исследований и публикаций. Анализ исследований в данной области показал актуальность вопросов использования онтологий при создании и эксплуатации агентных систем моделирования, что реализуется средствами мультиагентной платформы [7].

Важность этих задач особенно проявляется при создании web-ориентированных сред имитационного моделирования. В области создания онтологий для дискретно-событийного имитационного моделирования систем необходимо упомянуть о двух основных существующих на сегодняшний день разработках.

Fishwick и Miller [1] предложили онтологию (*Discrete-event Modeling Ontology*, DeMO), которая нацелена на описание наиболее общих понятий различных подходов дискретно-событийного моделирования: цепи Маркова, конечные автоматы, сети Петри, событийные графы. При создании DeMO был сделан упор лишь на основополагающие понятия, что предполагает ее дальнейшее расширение другими онтологиями.

Альтернативный подход предпринят Lee W. Lacy [2]: онтология формируется конкретно для взаимодействия процессов в рамках дискретно-событийного имитационного моделирования (*Process Interaction Modeling Ontology for Discrete Event Simulations*, PIMODES). Онтология PIMODES описывает основные понятия в рамках подходов к моделированию, которые применяются в существующих программных пакетах моделирования Arena, ProcessModel, AnyLogic. Направленность на процессный подход обусловила тот факт, что онтология в основном построена на развитии понятия активности (Activity) с точки зрения потокового представления (Flowchart). Таким образом, онтология основывается на развитии трех основных классов моделирования: активность (Activity), потоковая диаграмма (Flowchart) и сущность (Entity). В то же время, вопросы, связанные с особенностями использования и взаимодействия в процессах различных типов ресурсов, рассмотрены недостаточно. Основной целью создания онтологии PIMODES было решение задачи концептуального моделирования предметных областей для указанных программных пакетов. Для этого было создано специальное программное обеспечение PIMODES Translation Software, которое обеспечивает экспорт концептуальных PIMODES-описаний в форматы файлов моделей систем Arena, ProcessModel, AnyLogic и обратную процедуру импорта моделей, созданных в этих системах для формирования концептуальных представлений предметных областей.

Цель исследований. Проведенный анализ существующих публикаций показал актуальность рассматриваемых вопросов. В рамках данной работы

ставится иная цель, заключающаяся в создании и использовании онтологии как средства описания знаний непосредственно в самой системе имитационного моделирования дискретно-событийного типа, построенной на основе агентного подхода. Это дает возможность использовать формируемые знаниеориентированные описания при организации процессов принятия решений и эффективного взаимодействия агентов программной имитационной модели [7].

1. Особенности использования онтологического подхода

Онтология представляет собой формальное описание (концептуализацию) предметной области и правил принятия решений [4], которое служит для упрощения программирования поведения агентов и используется ими при взаимодействии.

Специализация каждого агента отражается подмножеством узлов онтологии. Некоторые узлы онтологии могут быть общими для нескольких агентов. Обычно только один из этих агентов обладает детально структурированным описанием онтологии (содержит фрагменты базы знаний). В то же время некоторая часть онтологических баз знаний является общей для всех агентов, и именно эта часть знаний является тем фрагментом, который должен играть роль общего контекста или общих знаний.

Для сложных динамических систем онтология должна быть концептуализацией, открытой для пополнения новыми знаниями [9]. Подобная специфика означает возможность модификации онтологии непосредственно в сеансе ее использования. Одно из решений этой задачи состоит в поиске общих понятий в требуемых предметных областях, поскольку на уровне базовых понятий все знания совпадают, а конкретными данными эти общие понятия наполняются только при определении конкретного домена. Это приводит к тому, что формально можно разделить онтологию на три уровня: общий, уровень поведений и частный. Верхний уровень должен содержать определения основных понятий и действий. Второй уровень будет использоваться для описания поведения системы. Здесь будут храниться данные о всех алгоритмах, обеспечивающих реакцию агента на различные события. На третьем уровне должны содержаться описания конкретных понятий предметной области моделирования. Элементы этого уровня онтологии будут наследоваться, расширять понятия первого уровня.

2. Онтология для системы агентного имитационного моделирования

Использование онтологического подхода как основы для концептуального моделирования при построении имитационных моделей производственных систем приводит к необходимости более детальной проработки понятийной основы средств построения агентных имитационных моделей. На рис. 1 представлен фрагмент онтологической модели, разделяемой всеми агентами системы моделирования. На рис. 2 представлена диаграмма классов, выполненная в нотации UML, которая детализирует данное онтологическое представление.

Онтологии описываются с использованием различных языков: основанные на исчислении предикатов первого порядка (KIF, CycL), другие языки математической логики, компьютерные языки – семейство языков Semantic Web, основанных на XML – RDF, RDFS и наиболее перспективный из них – OWL (Ontology Web Language) [3].

Для формального описания онтологий в работе используется язык RDF/OWL, а для логического вывода – собственный модуль, реализующий модифицированный метод резолюций для исчисления предикатов [6, 7]. Для этого в системе агентного имитационного моделирования предусмотрен парсер онтологии, который производит преобразование онтологии из формата owl во внутренний формат базы знаний агента на языке исчисления предикатов. Подсистема принятия решений при этом осуществляет логический вывод решений на основе метода резолюций для исчисления предикатов первого порядка.

С точки зрения OWL онтология O = <T, A, R, Dom, C, F> представляет собой описание понятий (классов) T *owl:Class* в рассматриваемой предметной области, свойств *owl:ObjectProperty* каждого понятия, описывающих различные атрибуты A, отношения R понятия, с учетом множества допустимых значений, определяемых доменом *Dom* и ограничений C *owl:Restriction*, наложенных на свойства. Произвольная группа элементов онтологии может образовывать фрагмент F. Существует два заранее определенных класса: *owl:Thing* (наиболее общий класс, содержащий все понятия); *owl:Nothing* (пустой класс). Каждый создаваемый класс является подклассом *owl:Thing* и суперклассом *owl:Nothing*. В OWL имеется два вида свойств: объектные свойства (*object properties*) связывают объекты с другими объектами; свойства типы данных (*datatype properties*) связывают объекты со значениями типов данных.

Элемент *owl:Restriction* в общем случае содержит элемент *owl:onProperty* и одно или более объявлений ограничений. Два типа ограничений: ограничения на тип значений свойства: *owl:allValuesFrom* (описывает класс возможных значений, которые может принимать свойство, специфицированное элементом *owl:onProperty*), *owl:hasValue* (свойство, специфицированное элементом *owl:onProperty* должно иметь конкретное значение) и *owl:someValuesFrom* (описывает класс возможных значений, которые может принимать, по крайней мере, одно из свойств); ограничения кардинальности числа значений.

Ниже приведен фрагмент онтологической модели с помощью языка разметки онтологий OWL, описывающий класс *EventSet* как множество событий, которые могут произойти в модели:

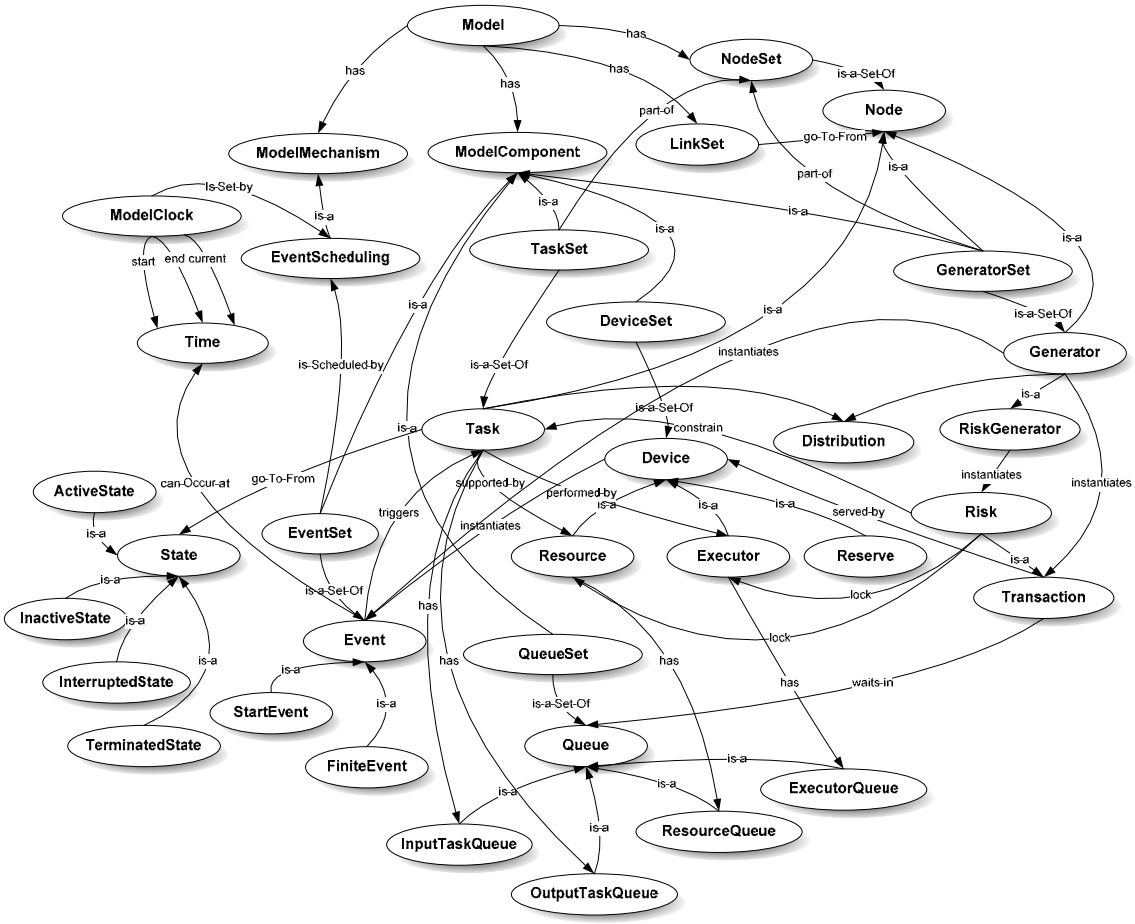

Рис. 1. Фрагмент онтологии системы моделирования

```
<rdf:RDF
 <owl:Ontology rdf:about=""/>
 <owl:Class rdf:ID="EventSet">
  <rdfs:comment rdf:datatype=
  "http://www.w3.org/2001/XMLSchema#string"
  >A set of events, E, that specify the types of
  primitive changes that can occur in a model.
  </rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ModelComponent"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Event"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="is-a-Set-
        of"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
 </owl:Class>
</rdf:RDF>
```

Другие особенности использования ограничений на свойства классов приведены во фрагментах ниже. Данный фрагмент описывает ситуацию, когда задача *task1* выполняется исполнителем *executor1*:

```
<owl:Class rdf:about="#task1">
   <rdfs:subClassOf >
      <owl:Restriction >
         <owl:onProperty rdf:resource=
         "#isExecuteBy"/ >
         <owl:hasValue rdf:resource=
         "#executor1"/ >
      </owl:Restriction >
   < /rdfs:subClassOf >
</owl:Class >
```

Фрагмент ниже показывает что, каждая задача имеет, по крайней мере, одного исполнителя:

```
<owl:Class rdf:about="#task1">
   <rdfs:subClassOf>
      <owl:Restriction>
      <owl:onProperty rdf:resource=
      "# isExecuteBy"/>
         <owl:minCardinality rdf:datatype=
         "&xsd;nonNegativeInteger">1
         </owl:minCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Онтология для данной системы разрабатывалась в системе Protégé [5]. На рис. 3 представлен фрагмент визуализации разработанной онтологии в системе Protégé.

3. Особенности использования онтологий при взаимодействии агентов

Для построения прототипа системы была выбрана платформа JADE (*Java Agent Development Framework*), которая упрощает создание мультиагетных систем с помощью реализованных FIPA (*The Foundation for Intelligent Physical Agents*)-спецификаций, а также поддержки фаз отладки и внедрения [5, 7].

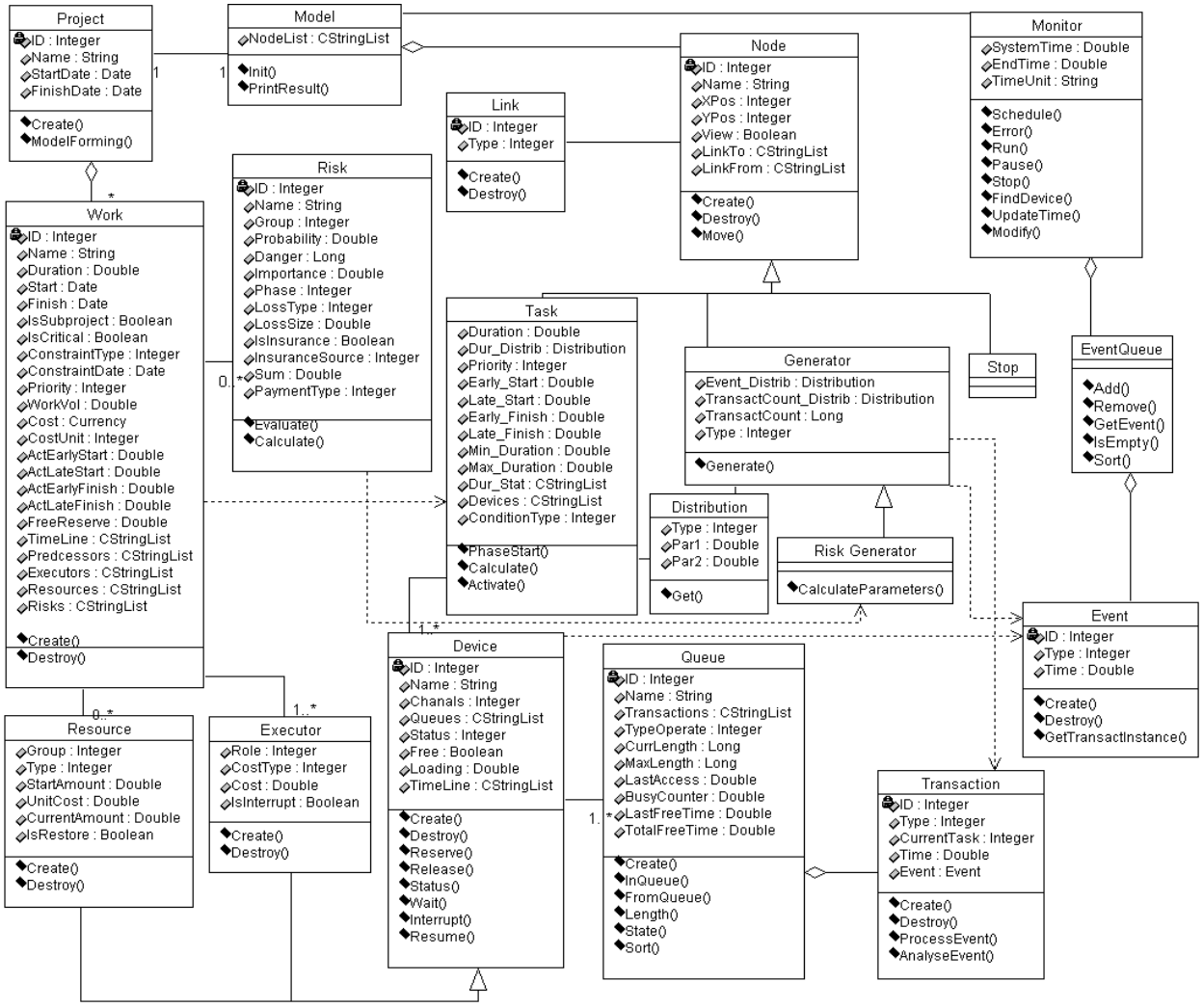

Рис. 2. Фрагмент диаграммы классов для онтологии системы

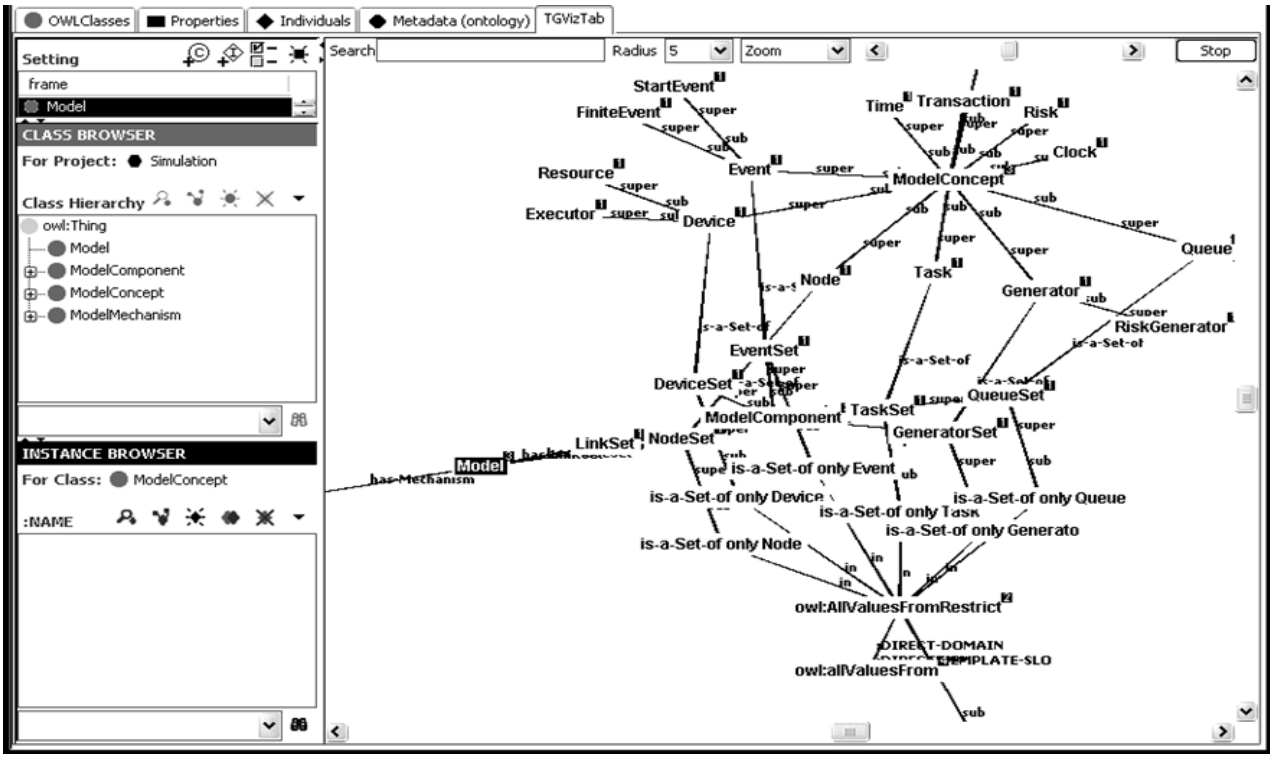

Рис. 3. Разработка онтологии в системе Protégé

На платформе JADE взаимодействие агентов реализовано с помощью механизма сообщений, которые кодируются в соответствии со спецификацией FIPA ACL Message Structure Specification. Формат сообщений представлен в табл. 1. Каждое сообщение представляется классом *jade.lang.acl.ACLMessage*, содержащим множество полей. При взаимодействии программных агентов онтологические модели используются для интерпретации передаваемых данных. По умолчанию для кодирования содержания используется язык FIPA Semantic Language (SL), который позволяет строить запросы агенту в виде логических выражений, но также можно использовать и любой другой язык, для которого разработана система кодирования (декодирования) объектов, например, OWL, RDF, XML и др.

В протоколе FIPA ACL имеется около 20 типов сообщений (*performatives*). *Inform* (информирование получателя сообщения) и *Request* (выполнение действия получателем сообщения) являются двумя основными. Все остальные построены на их базе и являются макродействиями. В некоторых случаях общение агентов можно представить в виде стандартных шаблонов, предполагающих определенные последовательности сообщений, которые называются протоколами взаимодействия. Протоколы взаимодействия FIPA – заранее определенные протоколы обмена ACL-сообщениями (существуют спецификации для 11 протоколов). Протоколы взаимодействия FIPA являются частью *FIPA Interaction Protocol Library* (FIPA IPL).

Таблица 1

Структура ACL-сообщения

Элемент	Описание
Performative	Тип сообщения
Sender	Отправитель сообщения
Receiver	Получатель сообщения
Reply-to	Адреса агентов, которым необходимо отправить ответ
Content	Содержание сообщения
Language	Язык кодирования сообщения
Encoding	Кодировка содержания сообщения
Ontology	Онтология, используемая для интерпретации сообщения
Protocol	Протокол взаимодействия агентов
Conversation-id	Идентификатор сообщений
Reply-with	Строка, идентифицирующая сообщение
In-reply-to	Строка, идентифицирующая сообщение, соответствует параметру при ответе на сообщение
Reply-by	Время, к которому необходимо получить ответ

Использование онтологий при взаимодействии агентов позволяет передавать высокоструктурированную информацию, значительно упрощает работу с агентами и их программирование. Единственным ограничением является то, что при этом каждый агент должен располагать информацией обо всех классах предметной области. Однако возможна организация дополнительного взаимодействия между агентами для обучения неизвестным понятиям (классам) онтологии принятого сообщения. Набор классов для работы с онтологиями в JADE находятся в пакете *jade.onto*. Представление онтологий в JADE основано на содержательно-ссылочной модели (*content-reference model)*. Каждое понятие предметной области классифицируется в предикат или терм, а затем, при необходимости, более детально. Термы представляют собой базовые понятия предметной области, которые могут использоваться агентами при взаимодействии. Предикаты представляют собой подобие двузначных функций, которые принимают значения *true* или *false* в зависимости от подставляемых в них термов. Примером предиката может служить предикат *TaskExecutor*, где подставляемыми в него термами будут название задачи и имя исполнителя. Предикат возвращает *true*, если предметная область предполагает, что исполнитель закреплен за данной задачей. Понятия, или *concept,* расширяют описания в виде термов. Понятия используются для обозначения сложных сущностей, для которых термов недостаточно. Примером понятия может служить сложный используемый алгоритм. Кроме того, элементом модели являются действия агентов. Это особые сочетания понятий, которые определяют действия, выполняемые агентами. Действия, так же как и предикаты, могут иметь аргументы.

Рассмотрим пример использования содержательно-ссылочной модели. В модель входят следующие элементы: предикат *ResCheck*, отвечающий на вопрос, есть ли доступный и в требуемом количестве ресурс для выполнения задачи; терм *Device* (устройство); понятие *Resource*, унаследованное от терма *Device*; действие *GiveRes* (предоставить ресурс задаче). В системе Protégé каждый элемент онтологии представляется полем класса и использует в качестве суперкласса классы терм или действие агента.

```
package simulationontology.ontology;

import jade.content.onto.*;
import jade.content.schema.*;
import jade.content.lang.Codec;

public class SimulationOntologyOntology extends jade.content.onto.Ontology {
```

```
// имя, идентифицирующее онтологию
  public static final String ONTOLOGY_NAME = "SimulationOntology";
// экземпляр онтологии
  private static Ontology theInstance = new SimulationOntologyOntology();
// метод для доступа к объектам онтологии
  public static Ontology getInstance() {
    return theInstance;
  }
// описание онтологии
    public static final String RESCHECK_RES="Res";
    public static final String RESCHECK="ResCheck";
    public static final String GIVERES_FRES="FRes";
    public static final String GIVERES_TASKRES="TaskRes";
    public static final String GIVERES_RESAMOUNT="ResAmount";
    public static final String GIVERES="GiveRes";
    public static final String RESOURCE_GROUP="Group";
    public static final String RESOURCE_CURRENTAMOUNT="CurrentAmount";
    public static final String RESOURCE_TYPE="Type";
    public static final String RESOURCE_STARTAMOUNT="StartAmount";
    public static final String RESOURCE="Resource";
  // конструктор
  private SimulationOntologyOntology(){
    super(ONTOLOGY_NAME, BasicOntology.getInstance());
    try {
    // добавление понятий
    ConceptSchema resourceSchema = new ConceptSchema(RESOURCE);
    add(resourceSchema, simulationontology.ontology.Resource.class);
    ConceptSchema deviceSchema = new ConceptSchema(DEVICE);
    add(deviceSchema, simulationontology.ontology.Device.class);
    // добавление действий
    AgentActionSchema giveResSchema = new AgentActionSchema(GIVERES);
    add(giveResSchema, simulationontology.ontology.GiveRes.class);
    // добавление предикатов
    PredicateSchema resCheckSchema = new PredicateSchema(RESCHECK);
    add(resCheckSchema, simulationontology.ontology.ResCheck.class);
    // добавление полей классов
    resourceSchema.add(RESOURCE_STARTAMOUNT, (Term-Schema)getSchema(BasicOnto
            logy.FLOAT), ObjectSchema.OPTIONAL);
    resourceSchema.add(RESOURCE_TYPE, (Term-Schema)getSchema (BasicOntolo
            gy.STRING), ObjectSchema.OPTIONAL);
    resourceSchema.add(RESOURCE_CURRENTAMOUNT, (Term-Schema)getSchema(Basic
            Ontology.FLOAT), ObjectSchema.OPTIONAL);
    resourceSchema.add(RESOURCE_GROUP, (Term-Schema)getSchema(BasicOntology.
             .STRING), ObjectSchema.OPTIONAL);
    giveResSchema.add(GIVERES_RESAMOUNT, (Term-Schema)getSchema(BasicOntolo
            gy.FLOAT), ObjectSchema.OPTIONAL);
    giveResSchema.add(GIVERES_TASKRES, taskSchema, ObjectSchema.OPTIONAL);
    giveResSchema.add(GIVERES_FRES, resourceSchema, ObjectSchema.OPTIONAL);
    resCheckSchema.add(RESCHECK_RES, resourceSchema, ObjectSchema.MANDATORY);
    // добавление наследования
    taskSchema.addSuperSchema(modelConceptSchema);
    executorSchema.addSuperSchema(deviceSchema);
    resourceSchema.addSuperSchema(deviceSchema);
    taskSetSchema.addSuperSchema(modelComponentSchema);
    deviceSchema.addSuperSchema(modelConceptSchema);
   }catch (java.lang.Exception e) {e.printStackTrace();}
  }
  }
```

Рассмотрим процесс пересылки сообщений. Пусть мы хотим спросить агента менеджера ресурсов, есть ли доступный и в требуемом количестве ресурс для выполнения задачи. Для этого в рамках модели есть предикат *ResCheck*. Сообщение формируется следующим образом:

```
ACLMessage msg = new ACLMessage(ACLMes
        sage.QUERY_IF);
msg.addReceiver(ResManagerAID);
msg.setLanguage(codec.getName());
msg.setOntology(ontology.getName());
RESOURCE Res = new Resource();
Res.setName("Equipment");
Res.setAmount(2);
RESCHECK Rescheck = new Rescheck();
Rescheck.setRes(Res);
```

Сначала создаются классы с требуемыми полями, затем они добавляются в сообщение как содержимое. JADE транслирует содержимое в строковый формат, что выполняется следующей командой:

```
getContentManager().fillContent(msg, Rescheck);
send(msg);
```

При получении сообщений имеем такой код:

```
MessageTemplate mt = MessageTemplate.and(
MessageTemplate.MatchLangua
        ge(codec.getName()),
MessageTemplate.MatchOntology(ontolo
        gy.getName()) );
ACLMessage msg = blockingReceive(mt);
try {
ContentElement ce = null;
```

```
if (msg.getPerformative() == ACLMes
        sage.QUERY_IF) {
// JADE конвертирует String в Java объекты
ce = getContentManager().extractContent(msg);
if (ce instanceof Rescheck) {
RESCHECK Rescheck = (RESCHECK) ce;
Res nres = Rescheck.getRes();
// Далее проверка наличия свободного ресурса
в требуемом количестве
...
}
}
catch (CodecException ce) {
ce.printStackTrace();
}
catch (OntologyException e) {
e.printStackTrace();
}
}
```

При таком подходе знания предметной области можно формулировать независимо от реализации системы в виде онтологии предметной области. При этом сценарии агентов могут включать динамические правила, загруженные из базы знаний или сконструированные «на ходу» агентами метауровня.

Заключение

Рассмотренный в работе подход позволяет решить задачу совместного применения онтологий программными агентами для возможности накопления и повторного использования знаний, для создания имитационных моделей и программ, оперирующих онтологиями, а не жестко заданными структурами данных, для анализа знаний в предметной области. Логические правила вывода при работе с онтологиями дают возможность манипулировать понятиями и данными гораздо эффективнее, позволяя извлекать новые знания. Практическое реализация онтологического подхода выполнена в виде прототипа знаниеориентированной системы имитационного моделирования производственных процессов, на основе агентного подхода, на базе платформы JADE, в составе которой функционируют интеллектуальные агенты, осуществляющие принятие решений и взаимодействие с помощью онтологической базы знаний и механизма логического вывода [7].

Список литературы

1. *Gregory A. Silver Ontology Based Representations of Simulation Models Following the Process Interaction World View / Gregory A. Silver, Lee W. Lacy, John A. Miller // Proceedings of the 2006 Winter Simulation Conference, Monterey, California. – 2006. – P. 1168-1176.*
2. *Lacy L. W. Interchanging Discrete Event Simulation Models using PIMODES and SRML / L. W. Lacy // Proceedings of the Fall 2006 Simulation Interoperability Workshop. Orlando, Florida, USA. – 2006. – P. 121-127.*
3. *Булгаков С.В. Подход к построению мультиагентной системы содержательного поиска во множестве разнородных структурированных источников данных / С.В. Булгаков // Труды 9-й нац. конф. по искусственному интеллекту КИИ' 2004. – М.: Физматлит, 2004. – Т. 2. – С. 706-714.*
4. *Гаврилова Т.А. Онтологический подход к управлению знаниями при разработке корпоративных информационных систем / Т.А. Гаврилова // Новости искусственного интеллекта. – 2003. – № 2. – С. 24-30.*
5. *Жигулина Е.М. Использование мульти агентных технологий при создании системы имитационного моделирования производственных процессов / Е.М. Жигулина // Труды 2-й межд. науч. конф. «Современные информационные системы. Проблемы и тенденции развития»: сб. материалов конф. – Х.: ХНУРЭ, 2007. – С. 203-204.*
6. *Жигулина Е.М. Агентно-ориентированная система имитационного моделирования производственных процессов / Е.М. Жигулина // Міжн. н.-т. конф. «Інтегровані комп'ютерні технології в машинобудуванні '2007»: тези доповідей. – Х.: ХАІ, 2007. – С. 307.*
7. *Пахнина Е.М. Использование онтологий в имитационном моделировании производственных процессов / Е.М. ахнина, А.В. Прохоров // Х Міжн. молодіжна НПК "Людина і космос": зб. тез. – Дніпропетровськ: Нац. центр аерокосм. молоді ім. О.М. Макарова, 2008. – С. 523.*
8. *Пахнина Е.М. Механизмы информационного взаимодействия агентов в системе имитационного моделирования производственных процессов / Е.М. Пахнина // Міжн. НТК «Інтегровані комп'ютерні технології в машинобудуванні '2008»: тези доповідей. – Х.: ХАІ, 2008. – Т. 2. – С. 77.*
9. *Тузовский А.Ф. Построение модели знаний организации с использованием системы онтологий / А.Ф. Тузовский, С.В. Козлов // Диалог 2006: Труды межд. конф. – М.: РБП-СУЗ, 2006. – С. 508-512.*

Поступила в редколлегию 22.10.2008

Рецензент: д-р техн. наук, проф. Э.Г. Петров, Харьковский национальный университет радиоэлектроники, Харьков.

ОСОБЛИВОСТІ ВИКОРИСТАННс ОНТОЛОГІЙ ПРИ ВЗАЄМОДІЇ АГЕНТІВ У СИСТЕМІ ІМІТАЦІЙНОГО МОДЕЛЮВАННс ВИРОБНИЧИХ ПРОЦЕСІВ

О.В. Прохоров, О.М. Пахніна

Проаналізовано особливості, пов'язані з використанням онтологій при взаємодії агентів у системі імітаційного моделювання виробничих процесів; проведено аналіз існуючих підходів у галузі створення онтологій для дискретно-подієвого імітаційного моделювання систем; запропоновано і надано опис онтології для системи агентного імітаційного моделювання; розглянуті питання реалізації онтологічного підходу у вигляді прототипу знанняорієнтованої системи імітаційного моделювання виробничих процесів.

Ключові слова: *імітаційне моделювання, інтелектуальний агент, мультиагентна система, база знань, агентний підхід, онтологія, взаємодія агентів.*

THE FEATURES OF ONTOLOGY USE IN THE INTERACTION BETWEEN AGENTS IN THE SIMULATION MODELING SYSTEM OF PRODUCTION PROCESSES

O.V. Prohorov, O.M. Pakhnina

This article analyses features connected with ontology use in the interaction between agents in the simulation modeling system of production processes. The analysis of the existing approaches in the ontology development field for discrete-event simulation modeling systems is made. The ontology for agent simulation modeling system is proposed and defined. The problems of ontology approach realization in the form of prototype of knowledge-oriented simulation modeling system of production processes are considered in this paper.

Keywords: *imitation design, intellectual agent, multiagent system, base of knowledges, agent approach, ontology, cooperation of agents.*