OSTIS-2012
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

Онтологический подход к интеграции компонентов имитационной модели в Triad.Net

Волегов И.С., Замятина Е.Б.

[*] *Пермский государственный национальный исследовательский университет*
г. Пермь, Российская Федерация

e_zamyatina@mail.ru

volegovis@gmail.com

В работе рассматриваются вопросы, связанные с автоматизацией и оптимизацией этапов имитационного эксперимента. Известно, что имитационное моделирование применяется как метод исследования в различных областях науки и производства. По этой причине сокращение времени на разработку имитационной модели и имитационного эксперимента является актуальным. В работе приводится опыт разработки системы Triad.Net и применение онтологического подхода для автоматического доопределения имитационной модели, добавления нового компонента, а также использования компонентов, подготовленных в других системах имитационного моделирования.

Ключевые слова: имитационное моделирование, онтологический подход, интеграция компонентов, переиспользование кода, оптимизация, автоматизация разработки модели.

Введение

Имитационное моделирование находит все большее применение как метод исследования сложных динамических систем. Этот метод актуален в том случае, если другие методы исследования, например, аналитические или численные, не могут быть применены, поскольку объект исследования сложно формализовать. Примерами сложных динамических систем могут служить вычислительные системы, системы телекоммуникации, бизнес-процессы, цепочки поставок и т.д.

Существует достаточно большое количество систем моделирования, которые позволяют выполнять исследования методом имитационного моделирования. Некоторые системы имитационного моделирования (СИМ) являются универсальными (GPSS WORLD[Sargent R.G.,2004], AnyLogic[Borshev A.,2007], Extend[Krahl D.,2002], , другие же предназначены для исследований в конкретной предметной области, т.е. являются специализированными (здравоохранение, логистика и т.д.).

Так или иначе, при выборе СИМ пользователи руководствуются определенными критериями, в числе которых можно назвать:

- простоту создания модели;
- валидность созданной модели и легкость проверки валидности;
- скорость выполнения имитационного прогона;
- наглядность представления результатов моделирования;
- возможность получения нерегламентированных запросов при проведении имитационного прогона и т.д.

Одним словом, все этапы имитационного эксперимента должны быть в той или иной мере автоматизированы, удобны пользователю и нацелены на оптимизацию времени проведения эксперимента, включая этап создания модели.

В настоящей работе рассматриваются вопросы применения онтологического подхода для автоматизации этапа создания модели, а более точно, интеграции компонентов в имитационную модель (ИМ), в том числе и тех компонентов, которые были созданы в других системах моделирования.

Онтологии находят широкое применение в имитационном моделировании.

Известно, что онтология - это описание типов сущностей, существующих в предметной области, их свойств и отношений между ними [Benjamin P. Et al, 1995]. Каждая предметная область – обычно, в этом контексте, это некая часть реального мира, например, производственная система, университет,

предприятие, - имеет собственную онтологию, которую называют онтологией предметной области. В онтологии предметной области определяются различные виды объектов (например, инструменты и работники), свойства (например, «быть сделанным из металла» или «иметь троих детей»), и отношения между видами и их экземплярами (например, «часть» или «женат на»).

Онтологии создаются во множестве областей знаний. Например, в последнее время было создано несколько онтологий для биологического домена. Создание онтологий для имитационного моделирования осложняется рядом причин[Miller J.A. et al, 2005] , о которых речь пойдет далее.

Моделирование не ограничено0 конкретным доменом, поскольку модели могут представлять биологические, химические, физические, транспортные, военные и т.п. системы;

Моделирование и его методы основаны на математике, вероятностных и статистических расчетах, и, таким образом должно их придерживаться, поэтому онтологии для этих областей должны служить основой для всех остальных (т.н. онтологии среднего уровня)

В работе [Benjamin P. et al, 2006] рассматривается возможность использования онтологий на различных этапах имитационного моделирования начиная с этапа сбора информации о моделируемой системе и заканчивая этапом валидации модели. В работе определяются также цели и задачи использования онтологий на каждом этапе.

Примерами использования онтологий практически на всех этапах моделирования могут служить управляемые онтологиями среды моделирования (например, [Benjamin P. et al, 2005] , а так же подходы к объединению различных федератов, разрабатываемые для HLA [Rathnam T. et al, 2004].

Подход, разрабатываемый для HLA, использует онтологии для описания требований, которым должны удовлетворять интерфейсы федератов для успешного взаимодействия в федерации, а так же для разработки этих требований, с учётом знаний о моделируемой предметной области.

В работе [Liang V.-C. et al,2003] представлена онтология портов, рассматриваемая как средство автоматизации компоновки моделей из компонентов.

Порты описывают интерфейс, определяющий границы компонентов или подсистем в конфигурации системы. Система представляется как конфигурация подсистем или компонентов, соединенных друг с другом через четко определенные интерфейсы. Конфигурация интерфейса компонента состоит из портов, которые определяют предполагаемое взаимодействие объекта с окружающей средой. Взаимодействия могут проходить в виде обмена энергией, материей или сигналов (т.е. информации). При построении модели взаимодействие двух подсистем друг с другом обозначается наличием связи между соответствующими портами подсистем.

Модель представляется в виде графа, в вершинах которого находятся компоненты модели (модели подсистем). Вершины имеют порты – именно через них компонент взаимодействует со своим окружением. Наличие взаимодействия между компонентами представляется на графе в виде наличия связи между соответствующими портами.

То, что взаимодействия описываются при помощи портов, вовсе не означает, что использованы могут быть исключительно компоненты со строго стандартизированными интерфейсами. В том случае, когда описание некоторого типа соединения частей моделируемой системы отсутствует в онтологии портов (например, описание сварочного шва между деталями механизма), взаимодействие всё равно может быть представлено одним из относительно небольшого множества обобщённых взаимодействий (например, жёсткое механическое соединение).

Использование онтологии портов для описания возможных типов соединения частей системы позволяет не только автоматизировать процесс сборки моделей из компонентов путём использования знаний о том, какие виды портов могут быть соединены и каким образом, но и упрощают процесс построения компонентов, разрешая противоречия и многозначность терминологии.

1. Система имитационного моделирования Triad.Net

Система моделирования Triad.Net представляет собой совокупность лингвистических и программных средств имитационного моделирования.

1.1. Архитектура системы имитационного моделирования Triad.Net

СИМ Triad.Net включает следующие компоненты [Миков А.И. и др., 2010]: компилятор, ядро, графический редактор, подсистему отладки, подсистему валидации, подсистему синхронизации распределенных объектов модели, подсистему балансировки (распределенная версия), подсистему организации удаленного доступа, подсистему защиты от внешних и внутренних угроз, подсистему автоматического доопределения модели. Назначение каждого из компонентов представлено ниже: TriadCompile (компилятор языка Triad, переводит описание имитационной модели с языка Triad во внутреннее представление); TriadDebugger (отладчик, использует механизм информационных процедур алгоритма исследования, локализует ошибки и вырабатывает рекомендации для их устранения на основании правил из базы данных, для каждого класса ошибок осуществляется поиск по онтологии соответствующего обработчика ошибок); TriadCore (ядро системы, включает библиотеки классов основных элементов модели), TriadEditor (редактор моделей, предназначен для работы с моделью как в удаленном, так и локальном режимах, локальный режим предполагает работу с системой в том случае, если

нет удаленного доступа), TriadBalance (подсистема балансировки), TriadSecurity (подсистема безопасности, этот компонент используют при удаленном доступе к системе моделрования), TriadBuilder (подсистема автоматического доопределения частично описанной модели), база данных, где хранятся экземпляры элементов модели, TriadMining-набор процедур для исследования результатов модели методами DataMining, TriadRule – алгоритм синхронизации объектов распределенной модели, использующей для вычислительного эксперимента ресурсы нескольких вычислительных узлов.

1.2. Представление имитационной модели в Triad.Net

Описание модели в системе Triad [Mikov, 1995] можно определить как M = (STR, ROUT, MES), где STR – слой структур, ROUT – слой рутин, MES – слой сообщений.

Слой структур представляет собой совокупность объектов, взаимодействующих друг с другом посредством посылки сообщений. Каждый объект имеет полюсы (входные и выходные), которые служат соответственно для приёма и передачи сообщений. Основа представления слоя структур – графы. В качестве вершин графа следует рассматривать отдельные объекты. Дуги графа определяют связи между объектами.

Объекты действуют по определённому алгоритму поведения, который описывают с помощью рутины. Рутина представляет собой последовательность событий e_i, планирующих друг друга (E – множество событий; множество событий рутины является частично упорядоченным в модельном времени). Выполнение события сопровождается изменением состояния объекта. Состояние объекта определяется значениями переменных рутины. Таким образом, система имитации является событийно-ориентированной. Рутина так же, как и объект, имеет входные и выходные полюса. Входные полюса служат соответственно для приёма сообщений, выходные полюса – для их передачи. В множестве событий рутины выделено входное событие e_{in}. Все входные полюса рутины обрабатываются входным событием. Обработка выходных полюсов осуществляется остальными событиями рутины. Для передачи сообщения служит специальный оператор *out* (***out*** *<сообщение>* ***through*** *<имя полюса>)*. Совокупность рутин определяет слой рутин ROUT.

Слой сообщений (MES) предназначен для описания сообщений сложной структуры.

Система Triad реализована таким образом, что пользователю необязательно описывать все слои. Так, если возникает необходимость в исследовании структурных особенностей модели, то можно описать только слой структур. В слое структур определены стандартные процедуры, с помощью которых можно определить множество вершин графа, множество ребер, дуг и т.д., найти кратчайшее расстояние между двумя вершинами, компоненты связности GetStronglyConnectedComponents(G), выделение слоя из структур модели GetGraphWithoutRoutines(M).

Ниже приведено описание слоя структур модели, которая представляет собой фрагмент компьютерной сети, состоящей из рабочих станций, передающих сообщения друг другу, и маршрутизаторов, отвечающих за нахождение пути передачи данных.

```
Type Router,Host; integer i;
M:=dStar(Rout[5]<Pol[4]>);
M:=M+node Hst[8]<Pol>;
M.Rout[0]=>Router;
for i:=1 by 1 to 4 do
    M.Rout[i]=>Router;
    M:=M+edge(Rout[i].Pol[1] — Hst[2*i-2]);
    M:=M+edge(Rout[i].Pol[2] — Hst[2*i-1]);
endf;
for i:=0 by 1 to 7 do
     M.Hst[i]=>Host;
endf;
```

Следует обратить внимание, что в Triad модель рассматривается как переменная. Она может быть построена с помощью операций над моделью. При построении модели используют графовые константы, которые соответствуют основным топологиям компьютерных сетей. В приведенном выше описании модели использовали графовую константу «звезда» (dstar). Кроме того, в приведенном выше примере были использованы «семантические» типы (Type Router,Host). В данном случае, это семантические типы «маршрутизатор», и «хост». Семантические типы используют для того, чтобы можно было доопределить модель с помощью рутины, извлеченной из библиотеки экземпляров рутин. Для поиска соответствующего экземпляра рутины.

Система Triad реализована таким образом, что пользователю необязательно описывать все слои. Так, если возникает необходимость в исследовании структурных особенностей модели, то можно описать в модели только слой структур.

Алгоритмом имитации будем называть совокупность объектов, функционирующих по определённым сценариям, и синхронизирующий их алгоритм.

1.3. Алгоритм исследования

Для сбора, обработки и анализа имитационных моделей в системе Triad.Net существуют специальные объекты – информационные процедуры и условия моделирования. Информационные процедуры и условия моделирования реализуют алгоритм исследования модели.

Информационные процедуры ведут наблюдение за элементами модели (событиями, переменными, входными и выходными полюсами), указанными пользователем. Если в какой-либо момент времени имитационного эксперимента пользователь решит, что следует установить наблюдение за другими элементами или выполнять иную обработку собираемой информации, он может сделать соответствующие указания, подключив к модели другой набор информационных процедур. Условия моделирования анализируют результат работы информационных процедур и определяют, выполнены ли условия

завершения моделирования. Кроме того, именно условия моделирования позволяют выполнить операции над моделью.

В системе Triad.Net для анализа функционирования компьютерной сети можно использовать стандартные и пользовательские информационные процедуры. Пользовательские информационные процедуры описывают на языке Triad [Миков,1995]. Для каждого элемента сети можно указать список необходимых информационных процедур, которые будут вести наблюдение во время моделирования за переменными, событиями и полюсами элемента. Система также предоставляет лингвистические средства для создания собственных условий моделирования, в которых можно описывать оригинальные алгоритмы сбора статистики и алгоритмы преобразования модели в динамике.

2. Онтологический подход к интеграции компонентов имитационной модели

При описании проектов по имитационному моделированию, в которых использовались онтологии, приведен пример проекта PortOntology, в котором знания о портах компонентов были использованы для автоматической сборки модели из компонентов.

В СИМ Triad.Net онтологии используются для доопределения частично описанной модели [Mikov A. et al,2007] , а именно, для доопределения поведения некоторого устройства, например, маршрутизатора в компьютерной сети. Это бывает необходимо на начальных этапах моделирования, когда исследователю или проектировщику необходимо получить оценки моделируемого объекта или системы, даже если они являются достаточно грубыми. При этом поиск компонента модели (экземпляра рутины) в базе данных осуществляется с помощью онтологий автоматически.

Для оперативного построения имитационной модели в конкретной предметной области используют графический интерфейс[Замятина Е.Б. и др., 2011], добавление нового компонента модели (экземпляра рутины) и настройка на конкретную предметную область выполняется с помощью онтологий.

Кроме того, в СИМ Triad.Net разрабатываются программные механизмы, позволяющие интегрировать в Triad-модель компоненты, разработанные в других СИМ. Это позволит не только воспользоваться уже готовыми моделями и провести эксперименты с ними в среде Triad, но и организовать распределенный имитационный эксперимент.

2.1. Автоматическое доопределение модели

В процессе проектирования модели, модель может быть определена частично. В настоящее время реализована работа с частично определёнными моделями вида:

$$M = \{STR; ROUT^*; MES\}$$

При этом:

$$ROUT^* \subseteq ROUT$$

т.е. не установлено поведение ряда объектов модели. Иными словами, в модели присутствуют вершины, не расшифрованные структурой (терминальные), не имеющие наложенной на них рутины.

Таким образом, задача системы автоматического доопределения модели состоит в том, чтобы для каждой такой вершины подобрать подходящую рутину. Для этого, очевидно, необходима информация о существующих рутинах, а так же некоторые знания о том, что собственно моделируется. Для этих целей используется понятие семантического типа.

Семантический тип – специальное понятие. Оно вводится для того, чтобы сгруппировать ряд объектов по некоторому смысловому, структурному, поведенческому типам [20]. Так, для обозначения множества процессорных устройств при моделировании вычислительных систем, может быть введён семантический тип Процессор, для обозначения элементов памяти – тип **МодульПамяти**. При моделировании систем массового обслуживания уместны будут семантические типы **Очередь**, **ГенераторЗаявок** и т.п.

Для того, чтобы причислить объект к тому или иному семантическому типу, в тексте программы употребляется специальный оператор:

<имя объекта> => <имя типа>

Сами семантические типы объявляются специальным оператором **type** <имя типа>

Фрагмент программы, использующей семантические типы, приведен выше.

В системе Triad.Net выделены условия доопределения терминальной вершины экземпляром рутины:

- Условия специализации;
- Условия конфигурации;
- Условия декомпозиции.

Условие специализации предполагает, что накладываемый на вершину экземпляр рутины, найденный в базе данных, имеет тот же семантический тип, что и терминальная вершина.

Условие конфигурации налагает ограничение на количество полюсов экземпляра рутины. Количество полюсов должно соответствовать количеству полюсов вершины.

Условие декомпозиции предназначено для проверки «графа окружения». Это означает, что необходимо проверить соответствие топологии графа, который образован вершинами, смежными с терминальной, и «графом окружения» из базы данных.

Если условия выполняются, то соответствующий экземпляр рутины накладывается на терминальную

вершину. В результате исследователь получает полностью определенную модель, готовую к выполнению.

3. Добавление нового компонента модели

3.1. Представление знаний

Для настройки на конкретную предметную область в Triad используют онтологии. В Triad разработана базовая онтология. Основу ее составляют следующие классы:

- TriadEntity – любая логическая сущность языка Triad, имеющая имя. Подклассами TriadEntity являются все классы базовой онтологии, кроме вспомогательных (например, тип полюса), а именно:
- Model – имитационная модель.
- ModelElement – составная часть имитационной модели, а также все, чем может быть специализирована вершина структуры имитационной модели.

Подклассами ModelElement являются классы:

- Structure – структура имитационной модели.
- Routine – рутина.
- Message – сообщение и т.д.

Основными свойствами в базовой онтологии являются следующие свойства:

- Свойства владения чем-либо: модель имеет структуру, структура имеет вершину, вершина имеет полюс и т.д.
- Свойства принадлежности к че0мулибо – inverse properties по отношению к соответствующим свойствам владения – структура принадлежит модели, вершина принадлежит структуре, полюс принадлежит вершине и т.д.
- Свойства, связывающие полюс с присоединенной к нему дугой: connectsWithArc (Pole, Arc), connectsWithPole (Arc, Pole).
- putsOn (Routine, Node) – свойство, связывающее вершину с наложенной на нее рутиной.
- Свойства, связывающие вершину с расшифровывающей ее структурой: explicatesNode (Structure, Node), explicatedByStructure (Node, Structure).
- modelingToCondition (Model, ModelingCondition) – свойство, связывающее модель с условием моделирования.

На основании СИМ Triad.Net было разработано программное средство для проектирования компьютерных сетей TRIADNS[Замятина Е.Б. и др.].

Онтология системы TriadNS дополняет базовую онтологию. Введены специализированные для области компьютерных сетей подклассы основных классов базовой онтологии (см. рис.1):

- ComputerNetworkModel – модель компьютерной сети;
- ComputerNetworkStructure – структура модели компьютерной сети;
- ComputerNetworkNode – элемент компьютерной сети, изначально содержит подклассы WorkStation, Server, Router;
- ComputerNetworkRoutine – рутина элемента компьютерной сети, и т.д.

В онтологии есть два специальных свойства для полюса:

- isRequired(ComputerNetworkRoutinePole, Boolean) - обязательно ли полюс должен быть соединен с другим полюсом
- canConnectedWith(ComputerNetworkRoutinePole, ComputerNetworkRoutine) – определяет семантический тип элемента, с которым полюс можно соединить.

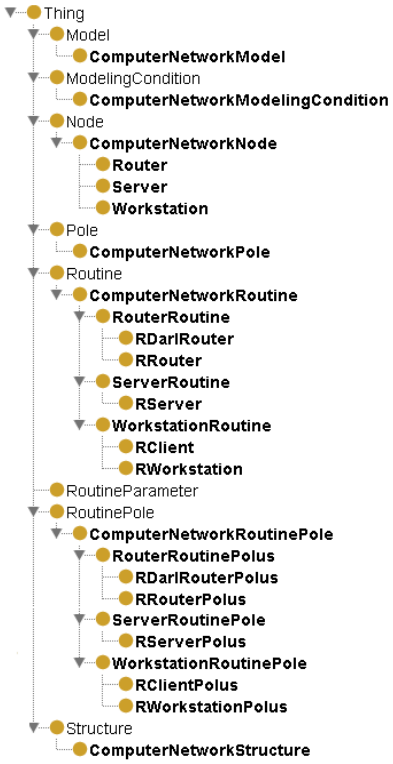

Рисунок 1 Иерархия классов онтологии

Имитационная модель компьютерной сети может быть представлена графически или описана на языке Triad (входной язык CAD TRIADNS). Рассмотрим более подробно особенности работы с графическим интерфейсом.

3.2. Графический интерфейс

Графический редактор позволяет оперативно

проектировать компьютерные сети. Структуру сети формируют путем перемещения пиктограмм из панели элементов, обозначающих конкретные элементы сети, в рабочую область, затем добавляют связи между ними (см.рис.2).

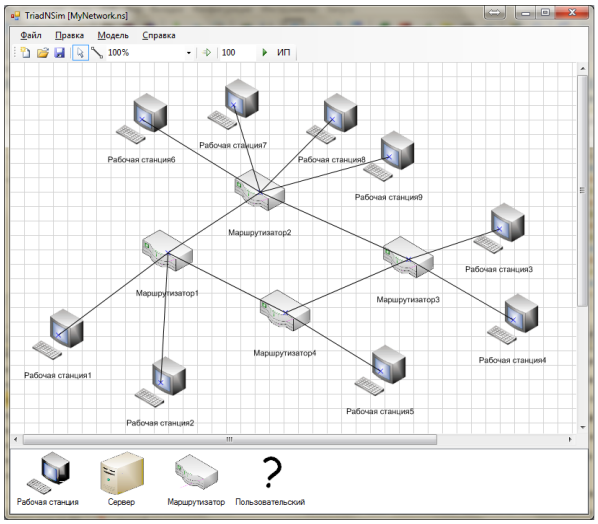

Рисунок 2 – Окно графического редактора

Как уже говорилось ранее, имитационная модель имеет графовое представление. Элементы сети являются вершинами этого графа. Элементы сети загружаются из онтологии предметной области (подклассы класса ComputerNetworkNode, изначально он содержит 3 подкласса: «Рабочая станция», «Сервер», «Маршрутизатор»). Пользователь может также добавлять собственные элементы с помощью диалоговых окон системы, для этого нужно указать имя элемента, описание, суперкласс, изображение. При этом элементы программно добавляются в онтологию с использованием библиотеки OWL API.

Для функционирования сети необходимо определить поведение каждой из вершин графа. Поведение вершин графа представлено рутинами. Все возможные рутины элементов также описывают в онтологии. Стандартные рутины имеют некоторое количество параметров, которые пользователь может изменять, после наложения рутины на элемент сети. Для каждого элемента сети можно определить произвольное количество рутин, описав их на языке Triad и добавив в онтологию. При добавлении рутины следует указать необходимость соединения каждого полюса рутины и семантический тип возможной смежной вершины, значения по умолчанию для указанных в рутине параметров.

4. Средства поддержки интероперабельности имитационных моделей в Triad.Net

За последние годы было создано множество систем имитационного моделирования, среди которых четко прослеживается деление на классы: дискретно-событийные системы, процессно-ориентированные, специализированные системы по моделированию физических процессов и другие. Системы, находящиеся в рамках одного класса очень похожи по своей структуре, при этом каждая обладает рядом преимуществ или недостатков перед конкурентами, но, зачастую, не имеет никаких средств взаимодействия с другими имитационными моделями.

При работе с такими системами пользователь жестко ограничен ее функциональными возможностями, не имея средств переноса своей модели в другую систему имитационного моделирования, расширить язык введением или модификацией существующей конструкции или настройки визуальной среды моделирования на конкретную предметную область. Эти недостатки можно устранить, введя дополнительный уровень в систему имитации: уровень метамоделей, который сможет описать:

- Целевой язык имитационного моделирования, при помощи которого будет производиться имитационный эксперимент, в данном случае это язык Triad – входной язык СИМ Triad.Net.
- Язык-источник, представляющий уже существующий язык имитационного моделирования, модели которого нам хотелось бы перенести на платформу целевого языка для дальнейшей модификации и исследования.
- Конкретные предметные области, т.к. зачастую пользователю бывает сложно работать с абстрактными концептами систем имитационного моделирования.
- Взаимосвязи между вышеперечисленными пунктами

Реализация системы Triad.Net предполагает создание метамоделей, которые будут описывать все важнейшие этапы функционирования: настройку транслятора и конвертера на определенную систему имитационного моделирования и настройку среды разработки на конкретную обобщенную модель, при этом при преобразовании моделей будут использоваться отображения одной метамодели на другую.

Структура информационной системы при таком подходе будет иметь вид, представленный на рис.3. Такой подход позволить создать открытую систему имитационного моделирования, предоставляющую пользователю гибкий набор механизмов взаимодействия с ней и расширения ее функционала:

- Описание дополнительных метамоделей для систем имитационного моделирования, позволяющее расширить круг доступных для работы моделей.
- Расширяемости метамоделей.

Расширяемость метамоделей предполагает:

- Изначально пользователь может создавать не полноценные метамодели систем имитационного моделирования, а неко-

торые их приближения, отражающие лишь необходимые на данном этапе элементы и связи.

- При переходе сторонней системы на более новую версию или добавление в нее функционала метамодель может быть доработана (а не создана с нуля).
- При недостаточных описательных возможностях обобщенной метамодели, используемой в визуальном редакторе, ее так же можно расширить.

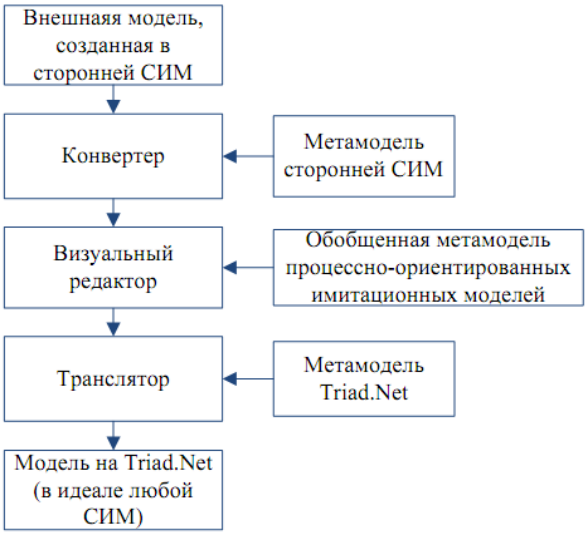

Рисунок 3 Архитектура системы

Важным этапом в создании любой информационной системы является этап определения формализма представления модели. С точки зрения открытых систем решение должно быть масштабируемым и обладать свойством интероперабельности, т.е. в данном случае, с точки зрения моделей, формализм должен предоставлять возможности по переводу модели из одного представления в другое, будь то графовые трансформации или наборы генераторов элементов модели.

Рассмотрим основные особенности языков имитационного моделирования, на которые стоит обратить внимание при выборе формализма:

- В основе модели лежит граф, описывающий основные элементы модели и связи между ними.
- Большинство языков предоставляет возможности по иерархическому описанию моделей и ее элементов.

С точки зрения метамоделирования формализм должен предоставлять возможности по отображению семантических связей, таких как наследование, агрегация и декомпозиция. Отдельно можно заметить, что предпочтение стоит отдать формализму, обладающему общедоступной спецификацией, чтобы по возможности обеспечить и перенос самих метамоделей.

Всем этим требованиям соответствуют онтологии. Использование онтологий в качестве метамоделей имитационного моделирования позволяет абстрагироваться от конкретной реализации и создать систему, которая будет способна работать с большинством современных СИМ. Пользователи легко смогут переходить к другим системам имитационного моделирования, объединять модели, написанные в разное время на разных языках имитационного моделирования или разрабатывать модели для исследования сразу в нескольких СИМ. Это позволит использовать совокупный потенциал различных СИМ без необходимости многократного описания одной и той же модели.

Главным недостатком сред распределенного имитационного моделирования, помимо их громоздкости и большого ряда ограничений и правил, накладываемых на структуру имитационной модели, является отсутствие семантики в передаваемых сообщениях. При использовании таких систем появляется необходимость в дополнительной обработки сообщений, определении его структуры и содержания, в то время, как в терминах конкретной парадигмы имитационного моделирования каждое сообщение можно соотнести с тем или иным концептом, понятным как источнику сообщения, так и приемнику.

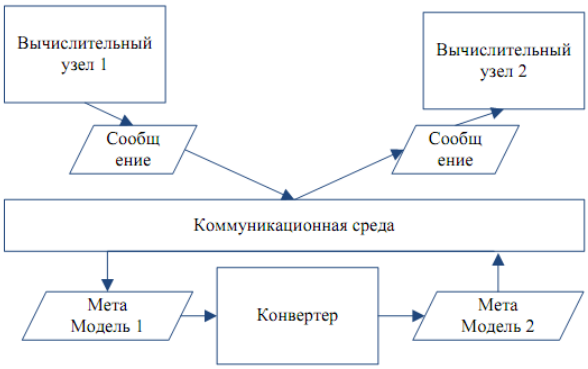

Рисунок 4 Архитектура распределенного имитационного эксперимента

Как мы видим на рисунке 4, этот недостаток можно устранить, добавив уровень метамоделей в коммуникационную прослойку, позволив узлам отсылать непосредственно элементы моделей. В коммуникационной прослойке выполняется конвертация сообщений, основанная на отображении метамоделей СИМ-источника и СИМ-приемника.

В настоящий момент ведутся работы по реализации этого проекта, разработана базовая онтология СИМ Triad.Net, онтология процессо-ориентированных СИМ, выполнены работы по конвертации онтологии процессо-ориентированной модели в Triad-модель.

Заключение

Итак, в работе сделан краткий обзор и представлены примеры применения онтологического подхода для интеграции компонентов имитационной мо-

дели: автоматическое добавление недостающих компонентов в частично определенной модели, добавление в Triad модель компонентов, разработанных в других СИМ. Онтологический подход позволяют сделать систему открытой и адаптируемой, позволяет автоматизировать этап создания модели, а также оптимизировать этот процесс.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

[Sargeant R.G., 2004] Sargent R.G. Some Recent Advances In The Process World View. // Proceedings of the 2005 Winter Simulation Conference / M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.,– pp 293-298

[Borshev A., 2007] Borshev A. AnyLogic Professional. Winter Simulation Conference (WSC'07), Washington, (December 2007), Final Program. Abstracts. p. 1.

[Krahl D.] Krahl D. The Extend Simulation Environment. Proceedings of the 2001 Winter Simulation Conference (WSC'01), Crystal Gateway Marriott, Arlington, VA, (December 2001). pp. 217-225.

[Benjamin P.,1995], Benjamin P., Menzel C, Mayer R. J. Towards a method for acquiring CIM ontologies. // International Journal of Computer Integrated Manufacturing, 8 (3) 1995, pp. 225-234.

[Miller J.A., 2005] Miller J.A., Baramidze G. Simulation and the Semantic Web // Proceedings of the 2005 Winter Simulation Conference / M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.. – pp. 2371-2377

[Benjamin P., 2006] Benjamin P., Patki M., Mayer R. J. Using Ontologies For Simulation Modeling // Proceedings of the 2006 Winter Simulation Conference/ L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds. –pp.1161-1167

[Benjamin P., 2005] Benjamin P., Akella K.V., Malek K., Fernandes R. An Ontology-Driven Framework for Process-Oriented Applications // Proceedings of the 2005 Winter Simulation Conference / M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.,– pp 2355-2363

[Rathnam T.,2004], Rathnam T., Paredis C.J.J. Developung Federation Object Models Using Ontologies // Proceedings of the 2004 Winter Simulation Conference / R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.,– pp 1054-1062

[Liang V.-C.,2003], Liang V.-C, Pardis C.J.J. A Port Ontology for Automated Model Composition // Proceedings of the 2003 Winter Simulation Conference / S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds., - pp 613-622

[Mikov, 1995] Mikov A.I. Formal Method for Design of Dynamic Objects and Its Implementation in CAD Systems // Gero J.S. and F.Sudweeks F.(eds), Advances in Formal Design Methods for CAD, Preprints of the IFIP WG 5.2 Workshop on Formal Design Methods for Computer-Aided Design, Mexico, Mexico, 1995. pp.105-127.

[Mikov A.,2007], Mikov A., Zamyatina E., Kubrak E. Implementation of simulation process under incomplete knowledge using domain ontology //Proceedings of the 6-th EUROSIM Congress on Modeling and Simulation. University of Ljubljana, Slovenia, 2007, Vol.2. Book of full papers. p.1-7.

[Замятина Е.Б., 2011], Замятина Е.Б., Миков А.И., Михеев Р.А. Применение онтологий в системе проектировании и моделирования компьютерных сетей TRIADNS. Труды Конгресса по интеллектуальным системам и информационным технологиям. «IS&IT'11», Научное издание в 4-х.томах, М. Физматлит, 2011, Т.1, стр. 253-260

[Миков А.И., 2010] Миков А.И., Замятина Е.Б. Проблемы повышения эффективности и гибкости систем имитационного моделирования. Проблемы информатики,№4(8), Новосибирск, Институт вычислительной математики и математической геофизики СО РАН, 2010, стр.49-64

THE INTEGRATION OF SIMULATION MODEL COMPONENTS INTO TRIAD.NET USING ONTOLOGICAL APPROACH

Volegov I.S. , Zamyatina E.B.

Perm State Natonal Research University, Perm, Russion Federations

Volegovis@gmail.com

e_zamyatina@mail.ru

INTRODUCTION

The paper discusses the problems of the use of ontology in various stages of simulation. More precisely, it focuses on the problem of automatic simulation model путукфешщт using ontological approach. The following sections consider simulation system Triad.Net and some issues of simulation components integration into simulation model Triad: the completion of partly defined simulation model (the behavior of some objects is not defined), the definition of the behavior of new object in graphical interface, the integration of the simulation components made in another simulation systems.

MAIN PART

Triad.Net – is a simulation system for the investigation of the complex computing systems. A simulation model μ={STR, ROUT, MES} consists of three layers, where STR is a layer of structures, ROUT – a layer of routines and MES – a layer of messages.

The layer of structures is dedicated to describe the physical units and their interconnections, but the layer of routines presents its behavior. Each physical unit can send a signal (or message) to another unit. Many objects being simulated have a hierarchical structure. Thus their description has a hierarchical structure too. One level of the system structure is presented by graph P = {U, V, W}.

As it was described above in a completely described model each terminal node $v_i \in V$ has an elementary routine $r_i \in$ROUT. An elementary routine is represented by a procedure. This procedure has to be called if one of poles of node v_i receives a message.

But some of the terminal nodes v_i of partly described model have no routines. The routine has to be found in special library using ontology being defined for Triad model.

Moreover the behavior of new object in graphical interface can be defined using this ontology. And the next example: the simulation component being designed in another simulation system may be integrated in Triad simulation model using this ontology too.

CONCLUSION

Thus the suggested approach allows to automate the process of simulation model generation. This approach supposes use of ontologies. Ontologies are the convenient path to domain describing. So a simulation system using ontologies becomes a powerful tool for scientific research because it complies with principles of open source software.