ЛИНГВИСТИЧЕСКИЕ И ИНТЕЛЛЕКТУАЛЬНЫЕ ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА СИМУЛЯТОРА КОМПЬЮТЕРНЫХ СЕТЕЙ TRIADNS

Елена Замятина, Александр Миков, Роман Михеев

Abstract: *В работе представлен симулятор компьютерных сетей TRIADNS. Широко используемая в настоящее время распределенная обработка информации, развитие Grid-технологий и облачных вычислений, позволяют говорить об актуальности проблемы, связанной с проектированием и моделированием компьютерных сетей. Проблема заключается в разработке программных средств имитационного моделирования, которые позволяют исследовать компьютерные сети, включающие множество узлов (и, следовательно, требующие больших временных затрат на моделирование). Кроме того, возникает необходимость в определении их топологических характеристик (диаметр, ширина бисекции и т.д.), исследовать трафик, изучить поведение устройств и отладить разрабатываемые протоколы компьютерной сети, в частности, исследовать работу алгоритмов маршрутизации, узнать, как будет выполняться тот или иной алгоритм при изменении сети и т.д. Таким образом, к симуляторам предъявляются требования, связанные с их эффективностью и гибкостью, т.е. симуляторы достаточно быстро должны находить решения, связанные с проектированием как аппаратной части компьютерных сетей, так и с проектированием их программного обеспечения, учитывая при этом особенности сетей различного типа (локальных, глобальных, корпоративных, беспроводных и т.д.). В работе представлены подходы, позволяющие решить эти проблемы (иерархическое трехслойное представление модели, использование онтологий и методов Data Mining).*

Keywords: *имитационное моделирование, компьютерные сети, онтологии, алгоритмы маршрутизации, Data Mining, параллельное имитационное моделирование.*

ACM Classification Keywords: *I.6 SIMULATION AND MODELING: I.6.8 Types of Simulation – Distributed : I.2 ARTIFICIAL INTELLIGENCE: I.2.5 Programming Languages and Software – Expert system tools and techniques.*

Введение

Роль компьютерных сетей в настоящее время достаточно велика. Это и распределенная обработка информации (корпоративные информационные системы, Grid-технологии, облачные вычисления), это и социальные сети, без которых многие люди уже не представляют себе жизни.

Широкое распространение компьютерных сетей предъявляет требования к скорости и надежности передачи информации, к эффективной ее обработке. По этой причине возникает необходимость в исследовании трафика, в разработке и изучении новых протоколов, в разработке новой аппаратуры и исследовании алгоритмов работы новых устройств.

Поскольку аналитические методы не всегда возможно применить для исследования компьютерных сетей из-за сложности объекта исследования, а натурные эксперименты не дают возможность исследовать все аспекты проектируемой сети, разработчики прибегают к методам и программным средствам имитационного моделирования, а именно, к использованию симуляторов компьютерных сетей. В

настоящее время существует большое число программных средств такого рода [Salmon S., 2011]. Краткий обзор этих средств приведен ниже.

В связи со сложностью исследуемого объекта к симуляторам можно предъявить ряд требований:

– *Оптимизация имитационного эксперимента по времени*. Такая необходимость возникает при решении задач, требующих значительных вычислительных ресурсов, например, при моделировании крупномасштабных сетей (large-scale networks). Моделирование сетей, насчитывающих сотни, тысячи и десятки тысяч узлов, должно завершаться за приемлемое время [Миков, 2008; Liu, 2009]. А это возможно при использовании многопроцессорной или кластерной аппаратуры, на которой проводится имитационный эксперимент, и соответствующего программного обеспечения, реализующего параллельный алгоритм продвижения времени [Riley, 1999; Fujimoto, 2003; Замятина, 2011]. Здесь же возникает проблема равномерной загрузки узлов при выполнении имитационного эксперимента, надежности и отказоустойчивости симулятора, использующего несколько вычислительных узлов [Zheng, 2005; Миков, 2010]. В настоящее время появился и ещё один класс симуляторов, использующих высокопроизводительную аппаратуру, – это симуляторы, предназначенные для выполнения на графических процессорах [Djinevski, 2012].

– *Совместное исследование аппаратуры и программного обеспечения компьютерных сетей*. При проектировании компьютерных систем обычно рассматривают отдельно аппаратную их часть и программную часть. Однако наиболее адекватным решением было бы наличие программных средств как для проектирования и анализа аппаратуры, проектирования и анализа алгоритмов, управляющей этой аппаратурой, так и для совместного проектирования аппаратного и программного обеспечения [Hu, 2011]. При проектировании, к примеру, алгоритмов маршрутизации, возникает необходимость в анализе их работы при изменении топологии, т.е. в данном случае проектировщика интересуют топологические характеристики сети, которые влияют на коммуникационную сложность алгоритма, графовая модель сети, алгоритмы на графах (кратчайшее расстояние, например). В настоящее время наиболее востребованы динамические алгоритмы маршрутизации, которые реагируют на изменение характеристик компьютерной сети и адаптируются к новым условиям. Поэтому важно иметь возможность промоделировать поведение устройств и действия алгоритмов при изменении трафика, пропускной способности линий связи и других характеристик компьютерной сети, промоделировать работу устройств и управляющих ими алгоритмов при выполнении тех или иных событий.

– *Адаптируемость программного обеспечения симуляторов к включению в имитационную модель новых устройств и новых алгоритмов, которые управляют их работой.* Наиболее известные программные продукты для проектирования компьютерных сетей таковы: [OPNET, 2004] (проектирование локальных и глобальных сетей, многопроцессорных и распределенных вычислительных систем, возможность оценивать производительность проектируемой системы и т.д.); OMNeT++ [OMNeT++, 2005] (симулятор дискретных событий, позволяющий исследовать все уровни компьютерных сетей и подключать пользовательские модули), NS-2 [NS-2, 2004] и др. Каждый из продуктов действительно имеет характерные особенности. Одни средства рассчитаны на управление локальными сетями, а другие предназначены для администраторов территориально-распределенных сетей. Одни просто позволяют строить схемы сетей и обладают ограниченными возможностями моделирования, другие же способны производить сложный анализ глобальных сетей. Предлагаемые программные продукты не могут решить все задачи, поскольку одни направлены на решение задач анализа сетей, другие – на решение задач проектирования, одни исследуют только локальные сети, другие – сенсорные. Поскольку технологии работы с сетями развиваются очень быстро, быстро

сменяются типы сетей, то средство проектирования, анализа и моделирования компьютерных сетей должно быть адаптируемыми к этим изменениям.

При проектировании и разработке симулятора компьютерных сетей TriadNS авторы постарались учесть опыт различных разработок подобного рода. Симулятор построен на базе CAD Triad [Миков, 1995]. Идеи, заложенные в Triad, позволяют адаптироваться к быстрой смене технических средств за счет того, что Triad располагает:

– лингвистическими средствами программирования структурных особенностей и поведения проектируемого объекта;
– развитыми средствами подсистемы анализа, которые включают библиотеки стандартных информационных процедур и лингвистические средства для создания новых процедур, а, следовательно, и новых алгоритмов анализа.

Кроме того, эффективность симулятора обеспечивается распараллеливанием имитационного эксперимента и интеллектуальным анализом результатов моделирования (на основе методов Data Mining). Гибкость же достигается за счет использования онтологий, механизма доопределения моделей (поиск по семантическому типу или другим критериям в базе рутин тех процедур, которые определяют сценарий поведения того или иного устройства), интероперабельности (включение в модель в качестве подмодели компонента, разработанного в другой системе моделирования). Прежде всего, следует рассказать о том, как представлена имитационная модель в Triad, архитектуру симулятора и назначение каждой из его подсистем.

Представление имитационной модели в Triad.Net

В Triad принято трехуровневое представление имитационной модели: $M = (STR, ROUT, MES)$, 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 (концентраторы, маршрутизаторы, серверы, рабочие станции, например). Дуги графа определяют связи между объектами. Имитационная модель имеет иерархическое представление. Отдельные объекты, представляющие вершины графа, могут быть расшифрованы подграфом более низкого уровня и т.д.

Объекты действуют по определённому сценарию, 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 сообщения, которые поступают на входные полюса рутины, обрабатываются входным событием. Обработка сообщений, которые генерируются на выходных полюсах рутины, осуществляется обычными 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.

Ниже приведен пример описания структуры компьютерной сети (рис. 1), состоящей из сервера и нескольких клиентов (в слое структур).

```
Structure КлиентСервер[ integer числоКлиентов ] def
        КлиентСервер := node Сервер<ПРИЕМ,ВЫДАЧА> + node Клиент[ 0 : числоКлиентов - 1 ]
<ПРИЕМ,ВЫДАЧА>;
    integer i;
for i := 0 by 1 to числоКлиентов - 1 do
        КлиентСервер := КлиентСервер + arc ( Клиент[ i ].ВЫДАЧА -- Сервер.ПРИЕМ ) +
                                    arc ( Сервер.ВЫДАЧА -- Клиент[ i ].ПРИЕМ );
endf;
endstr
```

Рис.1. Слой структур с описанием структуры компьютерной сети «Клиент-сервер»

Следует обратить внимание, что слой структур представляет собой параметризованную процедуру. В Triad модель рассматривается как *переменная*. Она может быть построена с помощью операций над моделью.

На рис.1. представлена структура сети «КлиентСервер», которая строится в виде вершины «Сервер» и присоединенным к ней массивом вершин «Клиент». Связи между вершинами устанавливаются в цикле *for* с помощью дуг, при этом указываются входные и выходные полюса (arc (Сервер.ВЫДАЧА -- Клиент[I].ПРИЕМ)).

Сценарий работы клиента описывают с помощью рутины, текст которой приведён ниже (рис. 2).

```
routine Клиент( input ПРИЕМ; output ВЫДАЧА )[ real deltaT ]
                initial     boolean запросПослан;
                            запросПослан := false; schedule ЗАПРОС in 0;       Print "Инциализация клиента";
endi
                event ЗАПРОС;
                            out "Запрос на обслуживание" through ВЫДАЧА;
                            Print "Клиент послал запрос серверу";
                            schedule ЗАПРОС in deltaT;
                ende  endrout
```

Рис.2. Рутина «Клиент»

Рутина также представляет собой параметризованную процедуру, которая наряду с параметрами интерфейса (входные и выходные полюса рутины «Прием» и «Выдача»), включает формальные параметры *deltaT* – временной интервал между запросами Клиента к Серверу.

Экземпляры рутин формируются оператором *let* Клиент(*clientDeltaT*) be клиент , а наложение рутины на соответствующую вершину графа выполняется оператором *put* клиент *on Модель.Клиент*[*i*] <ПРИЕМ=ПРИЕМ, ВЫДАЧА=ВЫДАЧА>. При этом входные и выходные полюса рутины сопоставляются с входными и выходными полюсами вершины.

Ниже приведено описание слоя структур модели (рис. 3, 4), которая представляет собой фрагмент компьютерной сети, состоящей из рабочих станций, передающих сообщения друг другу, и маршрутизаторов, отвечающих за нахождение пути передачи данных.

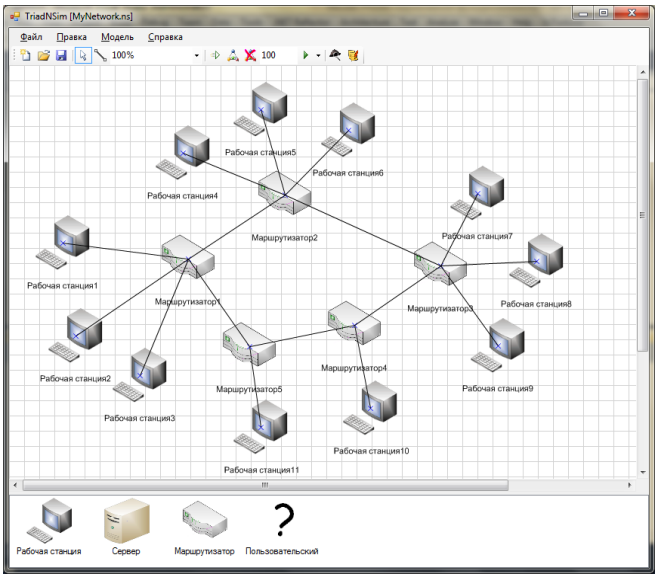

Рис.3. Графическое представление компьютерной сети «Маршрутизаторы», состоящей из маршрутизаторов и рабочих станций

```
Type Router,Host; integer i;
M:=dcycle (Rout[5]<Pol>[5]);
M:=M+node (Hst[11]<Pol>);
for i:=1 by 1 to 5 do
    M.Rout[i]=>Router;
    M:=M+edge(Rout[i].Pol[1] — Hst[i]);
endf
for i:=1 by 1 to 3 do
    M:=M+edge(Rout[i].Pol[2] — Hst[2*i-1]);
endf;
for i:=0 by 1 to11 do
    M.Hst[i]=>Host;
endf;
```

Рис.4. Описание структуры компьютерной сети «Маршрутизаторы» на языке Triad

При построении модели на рис.3. используют графовые константы, которые соответствуют основным топологиям компьютерных сетей. В приведенном выше описании модели использовали графовую константу «ориентированный цикл» (dcycle). 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.

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*).

Кроме того, пользователь располагает лингвистическими и программными средствами, которые дают возможность написать недостающие для исследования процедуры самостоятельно.

Исследование модели на уровне структур является статическим. Итак, после того, как произошло наложение рутин на вершины структуры модели, можно проводить моделирование. Алгоритмом имитации будем называть совокупность объектов, функционирующих по определённым сценариям, и синхронизирующий их алгоритм. Запуск модели осуществляется оператором simulate. Он имеет вид:

Simulate <*список моделей*> *on*

condition of simulation <*имя условия моделирования*>(<*настроечные параметры*>)

(<*параметры интерфейса*>)

(<*список информационных процедур*>; <*последовательность операторов*>)

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 рамках одного сеанса моделирования.

В системе TriadNS для анализа функционирования компьютерной сети можно использовать стандартные и пользовательские информационные процедуры. Пользовательские информационные процедуры описывают на языке Triad. Для каждого элемента сети можно указать список необходимых информационных процедур, которые будут вести наблюдение во время моделирования за переменными, событиями и полюсами элемента.

Система TriadNS предоставляет также лингвистические средства для создания условий моделирования, в которых можно описывать оригинальные алгоритмы сбора статистики и алгоритмы преобразования модели в динамике. Для удобного и оперативного представления имитационной модели компьютерной сети, отладки моделей, анализа результатов моделирования, проведения распределенных экспериментов разработаны специальные программные компоненты, описанные ниже.

Компоненты Triad.Net и их назначение

СИМ TriadNNS включает следующие компоненты: 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),
- 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),
- TriadBalance (подсистема балансировки),
- TriadRule – алгоритм синхронизации объектов распределенной модели, использующей для вычислительного эксперимента ресурсы нескольких вычислительных узлов.

Далее более подробно опишем компоненты, которые придают разработанным программным средствам определенную гибкость, позволяя добавлять новые объекты, доопределять модели, выполнять интеллектуальный анализ результатов моделирования.

Представление знаний

Для настройки на конкретную предметную область в Triad используют онтологии.

Онтологии используются на различных этапах моделирования [Benjamin, 2005, Benjamin, 2006]. В основном, на этапе сборки общей модели из отдельно разрабатываемых, переиспользуемых компонентов, в качестве источника информации о внутреннем устройстве этих компонентов и о возможных взаимосвязях между ними. При использовании более ранних подходов к объединению разнородных компонентов, приходилось создавать некоторый общий шаблон представления компонентами информации о себе, отдельно для каждого случая, а после этого ещё и дорабатывать компоненты, чтобы обеспечить их соответствие этому шаблону. При использовании онтологий достаточно один раз построить онтологию, описывающую компонент, после чего её можно переиспользовать в любом количестве моделей. Основное отличие представления знаний в виде онтологий заключается в том, что онтологии не нуждаются к приведению к общему словарю терминов и понятий. Достаточно определить несколько отношений синонимии терминов, чтобы новая онтология могла использоваться наравне с уже существующими, без её переработки, даже если она создавалась совершенно другими людьми с использованием другой терминологии. При этом работу по объединению и приведению форматов данных при обмене информацией между компонентами, берёт на себя среда моделирования.

Использование онтологий позволяет так же создавать репозитории компонентов, в которых будет храниться не только информация о предназначении и интерфейсе компонента, но и информация о связях между отдельными компонентами.

В Triad разработана базовая онтология. Основу ее составляют следующие классы: *TriadEntity* (любая логическая сущность языка Triad, имеющая имя), *Model* (имитационная модель), *ModelElement* (составная часть имитационной модели, а также все, чем может быть специализирована вершина структуры 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 чему-либо – 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*).
– Свойство, связывающее вершину с наложенной на неё рутиной *putsOn* (*Routine*, *Node*).
– Свойства, связывающие вершину с расшифровывающей её структурой: *explicatesNode* (*Structure*, *Node*), *explicatedByStructure* (*Node*, *Structure*).
– Свойство, связывающее модель с условием моделирования *modelingToCondition* (*Model*, *ModelingCondition*).

Онтология системы TriadNS дополняет базовую онтологию. Введены специализированные для области компьютерных сетей подклассы основных классов базовой онтологии (см. рис.5):

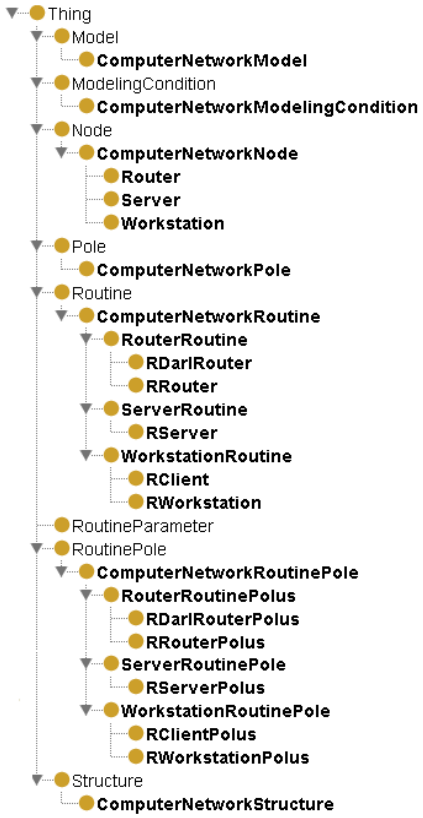

Рис. 5. Иерархия классов онтологии

– 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, которые используются при проверке возможности наложения рутины на элемент сети:

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

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.

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.

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.

Помимо элементов, загруженных из онтологии, на панели элементов есть специальный «Пользовательский элемент». Данный элемент обозначается знаком вопроса, но после добавления на рабочую область можно поменять его изображение. Он позволяет описывать поведение элемента на языке Triad, не добавляя его в библиотеку стандартных элементов. Его удобно использовать в том случае, когда это поведение не придется повторно использовать.

После перемещения элементов на рабочую область они имею только определенный семантический тип («Маршрутизатор», «Рабочая станция» и т.д.), поведение у этих элементов не определено. Поведение элементу можно задать через контекстное меню, все возможные сценарии поведения определяются из онтологии, где располагаются все экземпляры рутин данного семантического типа. Каждая из рутин имеет некоторое количество параметров, которые пользователь может изменять после наложения рутины на элемент. Если у некоторых элементов поведение не задано, то перед началом моделирования система предложит доопределить их поведение автоматически (о доопределении модели чуть ниже).

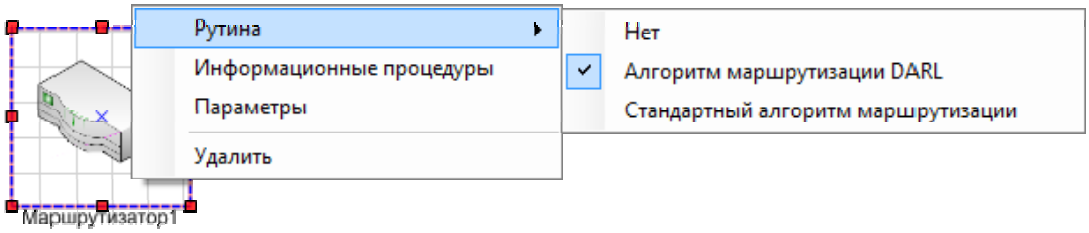

Рис. 6. Выбор поведения для элемента

Каждому элементу можно добавлять произвольное количество сценариев поведения, описав его на языке Triad. Нужно будет указать необходимость соединения каждого полюса и семантический тип возможного соседа, значения по умолчанию для указанных в рутине параметров, примечание - по желанию. Так, например, у стандартной рутины маршрутизатора полюса могут соединяться с рабочей станцией или другим маршрутизатором. Вся перечисленная выше информация и местоположение созданной сборки добавляется в онтологию, в онтологии размещается подкласс класса соответствующего семантического типа и индивид, описывающий созданную рутину.

Если поведение хотя бы одного элемента определено и у него больше одного полюса, то при соединении его с другим элементом необходимо указать свободный полюс для соединения. Способ соединения полюсов можно менять.

При этом элементы программно добавляются в онтологию с использованием библиотеки OWL API, описав их на языке Triad и добавив в онтологию. При добавлении рутины следует указать необходимость соединения каждого полюса рутины и семантический тип возможного соседа, значения по умолчанию для указанных в рутине параметров.

Доопределение имитационной модели

В TriadNS возможно автоматическое доопределение имитационной модели. Автоматическое доопределение модели выполняется компонентом TriadBuilding с использованием онтологий. Доопределение предусматривает сопоставление элементов компьютерной сети с соответствующими рутинами из онтологий. Для поиска нужной рутины используют семантические типы. Семантический тип – это специальное понятие, которое используется в процессе автоматического доопределения модели. Семантический тип используется, чтобы сгруппировать ряд объектов предметной области моделирования по смысловому, структурному, поведенческому признакам.

Для определения рутины, которую требуется наложить на недоопределённую вершину, используется база знаний экземпляров рутин, представленная в виде онтологий. В этой базе знаний описываются семантические типы рутин, отношения наследования между ними, множества соответствующих этим типам экземпляров рутин, и семантическая информация, необходимая для проверки условий доопределения. Существуют следующие три условия доопределения терминальной вершины экземпляром рутины: условие специализации (совпадение семанитческих типов рутины и вершины), условие конфигурации (совпадение количества входных и выходных полюсов рутин и вершины), условие декомпозиции (проверка совместимости рутины и вершины по графу окружения).

Доопределение выполняется автоматически без участия исследователя. Доопределяются элементы компьютерной сети, для которых пользователь не указал точного сценария поведения. Автоматическое доопределение целесообразно использовать на ранних стадиях моделирования при «грубом» первоначальном описании модели. Тем не менее, исследователь может получить представление о функционировании модели, сделать соответствующие выводы, принять решение о ходе дальнейших исследований и о необходимых экспериментах. Более подробно о доопределении имитационной модели было написано в [Mikov, 2007].

Интеллектуальный анализ результатов моделирования

В результате имитационного эксперимента пользователь получает информацию о характеристиках исследуемого объекта. Существуют различные способы сбора и обработки статистики, чаще всего это стандартная информация, которую получают по окончании имитационного эксперимента. Эта

информация чаще всего характеризуются своей избыточностью и недостаточностью. В симуляторе TriadNS для сбора и обработки статистики реализован механизм информационных процедур, который в отличие от многих других систем имитационного моделирования позволяет собирать именно ту статистику, в которой нуждается исследователь (сбор информации по сформированному запросу). Ранее уже упоминалось о наличии библиотеки стандартных информационных процедур, кроме того, исследователь имеет возможность создать оригинальную процедуру сбора статистики, воспользовавшись лингвистическими средствами языка Triad.

Существует еще одна проблема обработки полученной статистики. Дело в том, что анализ полученной в результате имитационного эксперимента информации является трудоемким и зачастую требует высокой квалификации аналитика.

Считается, что эти два фактора существенно снижают популярность моделирования, и делают их фактически недоступными для массового пользователя. В последнее время появились работы [Naumann, 2010], которые предлагают структурировать полученные результаты моделирования. В других работах [Brady, 2005] предлагался новый способ формирования выходных данных имитационного моделирования. Вместо обычного для моделирующих систем отчета, содержащего различные статистические данные, предлагалось применить методы Data Mining для анализа полученных данных. После обработки стандартного отчета получают зависимости между входными данными. Анализ этих зависимостей позволяет сократить общий объем данных, которые приходится анализировать, а информативность отчета повышается. Выявление зависимостей между данными позволяет сократить размерность задачи и тем самым оптимизировать имитационный эксперимент.

Для обработки результирующей информации по окончании имитационного эксперимента в TriadNS используют средства Data Mining (компонент TriadMining). Инструментальные средства TriadMining включают средства регрессионного анализа, временные ряды, баейсовские сети, сиквенциальный анализ и т.д. Известно, что информационные процедуры следят за изменением локальных переменных рутины, за сигналами, посылаемыми и получаемыми объектами модели и за событиями, происходящими в модели. Таким образом, с помощью информационных процедур можно проследить за последовательностью интересующих событий, определив их в качестве фактических параметров информационной процедуры. Известно, что информация о последовательности событий может быть использована для выявления аварий в узлах телекоммуникационных систем [Барсегян, 2009]. Поскольку информационная процедура активизируется при изменении значения локальной переменной рутин, то может быть сформирован временной ряд, содержащий информацию об изменении некоторой переменной во времени. Анализируя схожесть временных рядов, выявляют зависимости между элементами модели, а это позволит сократить объем исследуемых данных [Замятина, 2011].

Заключение

Итак, в работе рассматриваются вопросы применения онтологического подхода для настройки на предметную область. Использование онтологий позволило достаточно просто реализовать инструментальные средства моделирования и проектирования компьютерной сети TriadNS, используя в качестве основы CAD Triad. Симулятор TriadNS оснащен удобным графическим интерфейсом. Симулятор позволяет моделировать как аппаратное обеспечение компьютерных сетей, так и алгоритмы, управляющие устройствами компьютерных сетей. Еще одной отличительной особенностью симулятора является возможность проведения распределенного имитационного эксперимента. Использование методов Data Mining позволяет сделать анализ полученной в результате имитационного эксперимента информации менее трудоемкой, а использование онтологий – автоматизировать процесс построения

модели, ее отладки и добиться интероперабельности программных средств моделирования (переиспользования компонентов, разработанных в других системах моделирования).

Библиографический список

[Salmon, 2011] Salmon S, ElAarag H/ Simulation Based Experiments Using Ednas: The Event-Driven Network Architecture Simulator. In Proceedings of the 2011 Winter Simulation Conference S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.. The 2011 Winter Simulation Conference 11-14 December 2011 Grand Arizona Resort Phoenix, AZ, pp. 3266-3277.

[Liu, 2009], Liu Y., He Y. A Large-Scale Real-Time Network Simulation Study Using Prime. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds. The 2009 Winter Simulation Conference 13-16 December 2009. Hilton Austin Hotel, Austin, TX, pp. 797-806.

[Fujimoto, 2003] Fujimoto R.M. Distributed Simulation Systems. In Proceedings of the 2003 Winter Simulation Conference S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds. The 2003 Winter Simulation Conference 7-10 December 2003. The Fairmont New Orleans, New Orleans, LA, pp. 124-134

[Wilson, 1998] Wilson L. F., Shen W. Experiments in load migration and dynamic load balancing in Speedes // Proc. of the Winter simulation conf. / Ed. by D. J. Medeiros, E. F.Watson, J. S. Carson, M. S. Manivannan. Piscataway (New Jersey): Inst. of Electric. and Electron. Engrs, 1998. P. 487‑490.

[Zheng, 2005] Zheng G. Achieving high performance on extremely large parallel machines: Performance prediction and load balancing: Ph.D. Thesis. Department Comput. Sci., Univ. of Illinois at Urbana-Champaign, 2005. 165 p. [Electron. resource]. http://charm.cs.uiuc.edu/.

[Миков, 2008] Миков А.И., Замятина Е.Б. Технология имитационного моделирования больших систем // Труды Всероссийской научной конференции «Научный сервис в сети Интернет» – М.: Изд-во МГУ, 2008. С.199-204.

[Mikov, 1995] Mikov A.I. Simulation and Design of Hardware and Software with Triad// Proc.2nd Intl.Conf. on Electronic Hardware Description Languages, Las Vegas, USA, 1995. pp. 15-20.

[Замятина, 2011] Замятина Е., Ермаков С. Алгоритм синхронизации объектов распределенной имитационной модели в TRIAD.Net. Applicable Information Models. ITHEA, Sofia, Bulgaria, 2011, ISBN: 978-954-16-0050-4, стр.211-220.

[Djinevski, 2012] Djinevski L., Filiposka S, Trajanov D.Network Simulator Tools and GPU Parallel Systems. In Proceedings of Small Systems Simulation Symposium 2012, Niš, Serbia, 12th-14th February 2012, pp.111-114

[Riley,1999] Riley G., Fujimoto R.M., Ammar, M., A Generic Framework for Parallelization of Network Simulations", in Proc. 7th Int.Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 1999, p. 128-135.

[Hu, 2007] Hu W., Sarjoughian H.S. A Co-Design Modeling Approach For Computer Network Systems. . In Proceedings of the 2007 Winter Simulation Conference S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds. The 2007 Winter Simulation Conference 9-12 December 2007 J.W. Marriott Hotel, Washington, D.C., pp. 124-134

[Замятина, 2011] Замятина Е.Б., Колеватов Г.А., Миков А.И. Использование методов Data Mining при проектировании компьютерных сетей в CAD TRIADNS. Знания-Онтологии-Теории. 3-5 октября 2011, (ЗОНТ-2011). Материалы Всероссийской конференции с международным участием. Институт математики им. С.Л.Соболева СО РАН Новосибирск, Т.1., стр.146-154.

[Миков, 2010] Миков А.И., Замятина Е.Б., Козлов А.А. Мультиагентный подход к решению проблемы равномерного распределения вычислительной нагрузки. Natural and Artificial Intelligence, ITHEA, Sofia, Bulgaria, 2010, pp.173-180.

[Neumann G.,2010] Neumann G, Tolujew J, From Tracefile Analysis to Understanding the Message of Simulation Results, proceeding of the 7th EUROSIM Congress on Modeling and Simulation, Prague, Czechia, 2010, 7 pp.

[Brady, 2005] Brady T., Yellig E., Simulation Data Mining: a new form of simulation output, 37th Winter Simulation Conference, Orlando, USA, 2005, pp 285-289.

[Замятина, 2010] Замятина Е.Б., Михеев Р.А. Использование мультиагентного подхода и онтологий для моделирования компьютерных сетей //Материалы Четвертой международной научно-технической конференции «Инфокоммуникационные технологии в науке, производстве и образовании (Инфоком-4)» 28-30 июня 2010, Ставрополь, стр. 175-180.

[NS-2, 2004] The Network Simulator - NS-2. Доступно на сайте: http://www.isi.edu/nsnam/ns [Проверено 21 марта 2012]

[OPNET, 2004] OPNET Modeler. Доступно на сайте: <http://www.opnet.com> [Проверено: 21 марта 2012]

[OMNeT++, 2005] OMNeT++ Community Site. Доступно на сайте: http://www.omnetpp.org. [Проверено: 21 марта 2012]

[Benjamin,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

[Benjamin, 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

[Mikov, 2007] Mikov A., Zamyatina E., Kubrak E.Implementation of simulation process under incomplete knowledge using domain ontology. In proceedings of 6-th EUROSIM Congress on modeling and Simulation. 9-14, September, 2007, Ljubljana, Slovenia, Vol.2. Full papers, 7 pp.

[Барсегян, 2009] Барсегян А.А. Анализ данных и процессов: учеб.пособие /А.А.Барсегян, М.С.Куприянов, И.И.Холод, М.Д.Тесс, С.И.Елизаров.-Спб.:БХВ-Петербург,2009.-512с.

Информация об авторах

Елена Замятина *– Пермский государственный национальный исследовательский университет, РФ, Пермь, 614990, Букирева,15; e-mail: e_zamyatina@mail.ru*

доцент: имитационное моделирование, искусственный интеллект, распределенное моделирование

Александр Миков *– Кубанский государственный университет, г. Краснодар, ул. Ставропольская, 149 e-mail: alexander_mikov@mail.ru*

заведующий кафедрой вычислительных и информационных технологий: информационные системы, имитационное моделирование, искусственный интеллект

Роман Михеев *– Пермский государственный национальный исследовательский университет, РФ, Пермь, 614990, Букирева,15; e-mail: romanmiheev@gmail.com*

магистр: имитационное моделирование, искусственный интеллект, онтологии