УДК 681.3

Особенности моделирования распределенных информационных систем

Е. Б. Замятина, А. И. Миков, Р. А. Михеев
Пермский государственный национальный исследовательский университет
Россия, 614990, Пермь, ул. Букирева, 15
e_zamyatina@mail.ru; (342) 2 396 594

Статья посвящена имитационному моделированию распределенных информационных систем. В настоящее время существует большое количество специализированных программных систем, которые предназначены для построения имитационной модели распределенных систем и последующего проведения имитационного эксперимента. В статье рассматриваются программные и языковые средства системы автоматизированного проектирования Triad.Net, особенности построения Triad-модели и вопросы автоматизации изменения Triad-модели в ходе имитационного эксперимента. Среди свойств распределенных систем можно выделить свойства масштабируемости и отказоустойчивости, в результате которых количество вычислительных узлов системе изменяется. Особое внимание в статье уделяется автоматизации операций добавления и удаления объектов модели, ассоциируемых с вычислительными узлами.

Ключевые слова: *распределенные информационные системы; имитационное моделирование; масштабируемость; отказоустойчивость; автоматизация, онтологии.*

Введение[1]

При проектировании распределенных информационных систем, а также, при разработке алгоритмов и программного обеспечения, поддерживающих их функционирование очень часто применяется имитационное моделирование. В настоящее время существует большое количество специализированных программных систем, которые предназначены для построения имитационной модели распределенных систем, для последующего проведения имитационного эксперимента, сбора статистических данных и представления результатов имитационного моделирования [1, 2, 3]. К имитационным моделям предъявляется требование качества (квалиметрия имитационной модели), которое, в частности, предполагает, что изменение структуры и алгоритмов поведения модели не повлечет за собой ухудшения характеристик модели, определяющих ее способности удовлетворять установленным или предполагаемым потребностям пользователя [4].

Распределенные информационные системы обладают рядом характерных особенностей, к которым можно отнести свойство масштабируемости (наращивание вычислительных мощностей) и отказоустойчивости (неизменность выполнения распределенных вычислений программ при выходе из строя вычислительных узлов). Это означает, что имитационная модель распределенной информационной системы при выполнении операций по добавлению или удалению ее компонентов, ассоциирующихся с наращиваемыми или вышедшими из строя вычислительными мощностями, должна сохранять свои характеристики и предоставлять пользователям достоверные результаты.

Существует целый класс распределенных алгоритмов (волновые, выбора лидера, определения лидера), описание которых можно найти, например, в книгах Э.Таненбаума

[1]

Работа выполнена при финансовой поддержке гранта РФФИ №12-07-00302_а, гранта Министерства образования и науки (проект № 8.5782.2011), гранта РФФИ 13-07-96506 № р-юг-а.

[5] и Ж.Теля [6]. В ряде алгоритмов определены структуры данных (массивы, записи), размеры которых зависят от количества взаимодействующих процессов, функционирующих на различных вычислительных узлах. Примером такого алгоритма служит протокол принятия согласованного решения (задача византийских генералов). Напомним, что это алгоритм, поддерживающий отказоустойчивость распределенной вычислительной системы (маскирует отказы). Более подробно этот алгоритм будет описан ниже. При изменении топологии распределенной вычислительной системы меняются и структуры данных алгоритмов, которые выполняются в этой распределенной вычислительной среде. Следовательно, возникает необходимость и в изменении сценария поведения имитационной модели, а точнее, в изменении алгоритмов и соответствующих структур данных, описывающих поведение моделируемых объектов.

В дальнейшем материал статьи будет изложен в следующей последовательности: программные средства имитационного моделирования распределенных систем и особенности их реализации, алгоритм принятия коллективного решения, языковые и программные средства Triad.Net, реализация алгоритма принятия коллективного решения, слой сообщений, использование онтологий.

Программные средства имитационного моделирования распределенных систем

Распределенные информационные системы в настоящее время находят все большее применение. Это связано с настоятельной необходимостью обрабатывать большие объемы данных (Big Data). Известно, что для обработки больших объемов данных необходимо привлекать дополнительные вычислительные мощности, используя аппаратные средства суперкомпьютеров, распределенных систем (вычислительных узлов, связанных линиями связи), кластерные, Grid и Cloud-технологии. Однако применение распределенных информационных систем связано с разработкой новых алгоритмов (например, алгоритмы маршрутизации), новых протоколов. Для их разработки, анализа и оценки эффективности применяются специализированные программные средства, обзоры которых можно найти в работах зарубежных авторов [7, 8, 9, 10]. Примерами таких систем служат системы Parasol [11], SimGrid [12], NS-2 [1], OPNET [2], OMNet ++[3] и т.д. Подобные специализированные программные системы разрабатываются и в РФ [13, 14].

Практически все эти работы обладают такими чертами, как [15]:

- модульный подход к построению моделей;
- возможность многоуровневого описания исследуемой системы;
- наличие библиотек готовых модулей, имитирующих работу различного телекомму-никационного оборудования и протоколов;
- развитые средства сбора статистики;
- возможность графического конструирования отдельных фрагментов модели на основе объединения готовых модулей.

К недостаткам подобных программных систем некоторые авторы [16] относят:

- направленность их, прежде всего, на моделирование работы коммуникационных систем, ограниченные возможности описания вычислительного процесса в самих узлах сети;
- скрытость алгоритмов функционирования ряда модельных блоков и отсутствие исходных кодов инструментальных средств;
- отсутствие в некоторых случаях возможности создания собственных блоков и ограничения на выбор уровней детализации;
- высокая стоимость годовой лицензии использования пакетов.

Хотелось бы отметить еще ряд существенных черт, которые, по возможности, должны присутствовать в системах автоматизированного проектирования и моделирования распределенных систем:

- *Наличие программных средств оптимизации имитационного эксперимента по времени*. Такая необходимость возникает при решении задач, требующих значительных вычислительных ресурсов, в частности, при проектировании при проектировании распределенных вычислительных систем (крупномасштабные сети (large-scale networks)). Моделирование сетей, насчитывающих сотни, тыся-

чи и десятки тысяч узлов, должно завершаться за приемлемое время [17, 18]. А это возможно при использовании многопроцессорной или кластерной аппаратуры, на которой проводится имитационный эксперимент, и соответствующего программного обеспечения, реализующего параллельный алгоритм продвижения времени [19; 20, 21, 22]. Здесь же возникает проблема равномерной загрузки узлов при выполнении имитационного эксперимента, надежности и отказоустойчивости симулятора, использующего несколько вычислительных узлов [23, 24, 25, 37]. В настоящее время появился и еще один класс симуляторов, использующих высокопроизводительную аппаратуру, – это симуляторы, предназначенные для выполнения на графических процессорах [26].

- *Наличие программных средств для описания операций над структурой модели.* Во время имитационного эксперимента структура моделируемой распределенной системы может быть изменена: добавление линий связи, удаление линии связи. добавление вычислительного узла, удаление узла. Следовательно, в имитационной модели необходимо предусмотреть операции над моделью.
- *Наличие программных средств совместного исследования аппаратуры и программного обеспечения компьютерных сетей*. При проектировании компью-терных 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 [27]. 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.
- *Адаптируемость программного обеспечения симуляторов распределенных информационных систем к включению в имитационную модель новых устройств и новых алгоритмов, которые управляют их работой.* Наиболее известные программные продукты для проектирования и моделирования распределенных вычислительных систем перечислены выше. 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.
- *Наличием программных средств, позволяющим адаптировать симулятор распределенных информационных систем на конкретную предметную область.* При проектировании распределенных информационных систем целесообразно привлекать специалистов в области проектирования, которые используют

тот или иной математический аппарат, например, теорию графов, теорию очередей, системы массового обслуживания, сети Петри [28, 29]. Зачастую графическое представление объектов имитационной модели представлено в виде диаграмм UML или IDEF [30]. Однако в данном случае речь идет не только лишь о графическом представлении модели, но и о средствах анализа распределенной информационной модели. Так при анализе структуры распределенной информационной системы возникает необходимость в получении такой информации, как множество вершин (структура представлена в виде графа), множество смежных вершин по входу, множество смежных вершин по выходу, степень выбранной вершины, диаметр графа, который представляет собой структуру моделируемой распределенной информационной системы и т.д. При анализе имитационной модели, в основе которой лежит система массового обслуживания проектировщиков интересуют занятость одноканальных и многоканальных устройств, средняя и текущая длина очереди и т.д.

- *Наличие средств удаленного доступа к системе проектирования и моделирования распределенных информационных систем*. Эти средства позволяют географически удаленным друг от друга проектировщикам вести совместные работы над одной и той же моделью, обмениваться компонентами имитационной модели [35].

При разработке программных средств проектирования и анализа распределенных информационных систем группой авторов были учтены выдвинутые выше требования (программная система TRIADNS). Программная система TRIADNS построена на основе CAD Triad [31, 32].

Для того чтобы система имитации соответствовала требованиям, предъявленным выше, в системе TRIADNS разработаны:

– языковые конструкции и соответствующее программное обеспечение для описания структуры имитационной модели (граф с вершинами, которые представляют моделируемые объекты, дуги графа представляют собой связи между объектами, которые обмениваются друг с другом сообщениями), сценарии поведения объектов, описания сообщений сложной структуры;
– развитые средства подсистемы анализа, которые включают библиотеки стандартных информационных процедур и лингвистические средства для создания новых процедур, а, следовательно, и новых алгоритмов анализа.
– онтологии, которые позволяют (а) использовать знания о модели при автоматическом доопределении модели (доопределение сценария поведения объектов, которые на начальном этапе проектирования пользователем может быть не определено); (б) для синхронизации объектов имитационной модели в распределенной (параллельной) системе имитации, (в) для верификации и валидации имитационной модели, для организации полимодельных комплексов, (г) для автоматизации операций добавления и удаления объектов, ассоциируемых с вычислительными узлами; (д) для включения в модель компонентов имитационной модели, реализованных в других системах имитации.

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

Компоненты TRIADNS

СИМ TriadNS 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 рассмотри представление имитационной модели.

Представление имитационной модели в TRIADNS

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, или одно-канальное устройство, многоканальное устройство, очередь). 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 моделирования 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.

Triad-модель рассматривается как *переменная*. Она может быть построена с помощью операций над моделью.

Как уже было сказано ранее, модель включает описание структуры, рутин и слоя сообщений.

Синтаксис слоя структур:

structure <имя структуры> ***def*** (<список настроечных параметров>) (<список входных и выходных параметров>) <описание переменных> <операторы>) ***endstr***

В качестве переменных можно использовать переменные типа «вершина», «граф», «полюс» и т.д. Операторы выполняют операции над вершинами, графами, полюсами, структурами.

Ниже приведен пример описания структуры компьютерной сети, 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»*

На рис. 1. представлена структура сети «КлиентСервер», которая строится в виде вершины «Сервер» и присоединенным к ней массивом вершин «Клиент».

Связи между вершинами устанав-ливаются в цикле ***for*** с помощью дуг, при этом указываются входные и выходные полюса (***arc***(Сервер.ВЫДАЧА--Клиент[i].ПРИЕМ)).

Слой структур представляет собой параметризованную процедуру. Изменив значение входного параметра «числоКлиентов», можно получить в результате модель со структурой, в которой определено уже другое количество клиентов.

Переопределить количество клиентов можно перед началом или в ходе имитационного эксперимента. Во втором случае переопределение выполняется в условии моделирования.

Сценарий работы клиента описывают с помощью рутины, синтаксис и текст которой приведены ниже:

Синтаксис:

routine<имя>(<список настроечных параметров>)(<список входных и выходных формальныхх параметров>) ***initial*** <последовательность операторов> ***endi***

event <последовательность операторов> ***ende***

event <наименование события> <последовательность операторов> ***ende*** …

event<наименование события><последовательность операторов> ***ende endrout***

Пример рутины:

```
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.

Графический интерфейс

В TRIADNS реализован графический редактор, с помощью которого можно определить структуру моделируемой распре-деленной системы, разместив соответствующие пиктограммы на экране, соединив их линиями связи и, затем, ввести сценарий поведения отдельных элементов модели, а также, указать информационные процедуры для сбора статистических данных.

Построенная модель может быть сохранена в файле, сведения о модели размещаются в онтологиях (базовой и предметно-ориентированной). Эти сведения в дальнейшем могут быть использованы для автоматизации различных этапов построения имита-ционного модели и проведения имитационного эксперимента.

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.

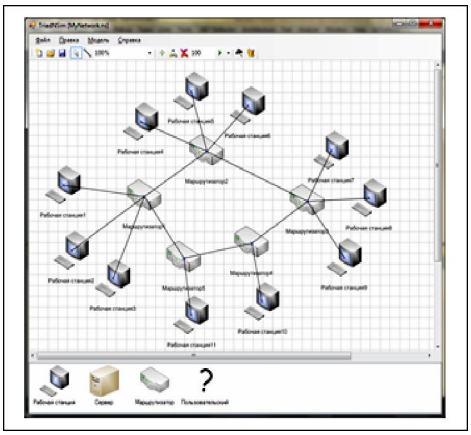

Рис. 3. *Слой структур, определенный средствами графического редактора*

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. Если есть необходимость в исследовании поведения какого-либо объекта, то можно рассмотреть отдельную рутину, подав на вход соответствующие сигналы и т.д. Таким образом, в системе моделирования TRIADNS над моделью в целом определены такие операции, как выделение слоя структур, выделения слоя сообщений, выделение слоя рутин, а также операции наложения рутины, определения типа данных, наложения слоя структур и т.д.

Алгоритм исследования

Алгоритмом имитации называют совокупность объектов, функционирующих по определённым сценариям, и синхронизи-рующий их алгоритм.

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

Информационные процедуры ведут наблюдение за теми элементами модели (событиями, переменными, входными и выходными полюсами), которые указаны 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. Приведем синтаксис условий моделирования:

Conditions of simulation<наименование условий моделирования>(<список настроечных параметров>)(<список входных и выходных параметров>) **initial** <последовательность операторов> ***endi*** <список информационых процедур> <последовательность операторов> ***processing*** <последовательность операторов>…***endcond***

Запуск модели на выполнение осуществляется оператором ***simulate***, приведем пример оператора, который запускает модель M:

simulate M ***on*** Generals(4) (M.MyGeneral [1].result).

Условия моделирования можно использовать для изменения условий проведения эксперимента и выполнения операций над моделью в целом или каких-либо ее элементов или слоев во время проведения имитационного эксперимента.

Так в слое структур можно выполнить следующие операции: добавление, удаление вершины, добавление, удаление дуг и ребер и др. Операции *изменяют* структуру имита-ционной модели во время имитационного прогона, а это в свою очередь требует в некоторых случаях изменения рутин и структур данных, которые определены в них.

В слое структур определены также процедуры и функции: Nodeset (***in ref Graph*** G) –

множество вершин графа;Nodenames (***in ref Graph*** G) – множество имен вершин графа; Nodenumber(***in ref Graph*** G) – количество вершин графа. Процедуры и функции определены также и для смежных вершин, полюсов, событий, типов сообщений.

Далее рассмотри известный алгоритм протокола принятия согласованных решений, который применяется для организации отказоустойчивых распределенных вычислений и покажем, как фрагменты этого протокола могут быть реализованы на языке Triad, а также, как можно решить проблему изменения структур данных в ходе имитационного эксперимента.

Протокол принятия согласованного решения

Рассмотрим классический пример ***протокола принятия согласованных решений*** – задачу "Византийских генералов". Протокол изложен в книге Э. Таненбаума и лекциях В.А. Крюкова [33].

В этой задаче армия зеленых находится в долине, а ***n*** синих генералов возглавляют свои армии, расположенные в горах. Связь осуществляется по телефону и является надежной, но из ***n*** генералов ***m*** являются предателями. Предатели активно пытаются воспрепятствовать согласию лояльных генералов.

Согласие в данном случае заключается в следующем: каждый генерал знает, сколько воинов находится под его командой. Ставится цель, чтобы все лояльные генералы узнали численности всех лояльных армий, т.е. каждый из них получил один и тот же вектор длины ***n***, в котором *i*-й элемент либо содержит численность *i*-й армии (если ее командир лоялен) либо не определен (если командир предатель).

Рассмотрим алгоритм для случая ***n***=4 и ***m***=1. Структуру модели опишем с помощью графического редактора.

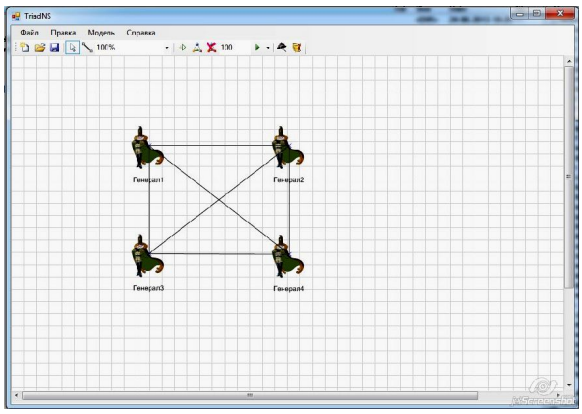

Рис. 4. *Структура модели, реализующей протокол принятия согласованного решения*

В текстовом редакторе для определения структуры необходимо описать ее следующим образом:

```
Structure Army (integer n) def
  Army:=compl (n)(Con(n-1))
End;
```

Здесь структура имитационной модели описана графовой константой ***compl*** – связный граф, n – входной формальный параметр, необходим для указания общего количества генералов.

В этом случае алгоритм осуществляется в 4 шага.

1 шаг. Каждый генерал посылает всем остальным сообщение, в котором указывает численность своей армии. Лояльные генералы указывают истинное количество, а предатели могут указывать различные числа в разных сообщениях. Генерал-1 указал 1 (одна тысяча воинов), генерал-2 указал 2, генерал-3 указал трем остальным генералам соответственно x, y, z, а генерал-4 указал 4.

В имитационной модели этот шаг может быть описан событием Greeting, приведенным ниже.

```
event Greeting;
//шаг первый, каждый генерал посылает сообщение о численности своей армии (номер рабочей станции)
out  NumberWorkStation through Con[0];
//Укажем,что первый шаг выполнен
Mode:=1;
ende
```

Рис. 5. *Сообщение о численности армии лояльного генерала*

```
event GreetingSpy;
 //шаг первый, каждый генерал посылает
сообщение о численности своей армии (но-
мер рабочей станции)
out  NumberX through Con[0];
out  NumberY through Con[1];
out  NumberZ through Con[2];
//Укажем,что первый шаг выполнен
 Mode:=1;
ende
```

Рис. 6. *Сообщение о численности армии генерала-предателя*

Будем считать, что генералы ассоциируются с вычислительными узлами (рабочие станции) и все рабочие станции пронумерованы. Поэтому три лояльных генерала среди событий своей рутины имеют событие Greeting, в котором одно и то же сообщение об истинной численности армии посылается через все три выходных полюса вершины графа, в то время, как генерал–предатель посылает три разных сообщения по трем выходным полюсам.

Выходные полюса являются и входными (тип ***InOut)***. Выходные и входные полюса целесообразно описать в виде массива с количеством элементов, равным (n-1), где n – количество генералов.

Напомним, что описание слоя структур, рутин, а также, слоя сообщений являются параметризованными процедурами. Если в качестве входного параметра будет использовано значение количества рабочих станций, то при создании модели пользователь получит экземпляры структур, рутин и сообщений с конкретным значением количества вершин. В нашем случае будет создана модель, в слое структур которой определены 4 вершины (см. рис. 4.).

2-ой шаг. Каждый генерал формирует свой вектор из имеющейся информации. В результате получается четыре вектора: vector1 (1,2,x,4); vector2 (1,2,y,4); vector3 (1,2,3,4); vector(1,2,z,4).

Второй шаг реализуется в Triad-модели во входном событии. Здесь используется переменная типа ***Vector*** (n), где n – количество генералов (или рабочих станций в сети).

При изменении количества генералов (рабочих станций в сети) соответственно должно измениться и количество элементов этого вектора.

Поэтому целесообразно описать этот тип данных в слое сообщений. О слое сообщений подробнее расскажем несколько позже.

Итак, каждый генерал посылает сформированный на предыдущем шаге вектор всем остальным (генерал-3 посылает опять произвольные значения).

На 3-м и на 4-м шагах каждый генерал будет оперировать с матрицей, описываемой типом ***Matrix***. Действительно, каждый из генералов получает по n-1 (3, в конкретном примере) длины n: первый – ((1,2,y,4), (a,b,c,d), (1,2,z.4)); второй – ((1,2,x,4) (e,f,g,h) (1,2,z.4)); третий – ((1,2,x,4),(1,2,y,4),(1,2,z.4)), четвертый – ((1,2,x,4), (1,2,y,4), (i,j,k.l)).

На 4-м шаге каждый генерал проверяет элементы во всех полученных векторах и, если какое-то значение элемента совпадает, по меньшей мере, в двух векторах, то оно помещается в результирующий вектор, иначе соответствующий элемент результирующего вектора помечается «неизвестен». В результате все лояльные генералы получают один вектор (1, 2, «неизвестен», 4) – согласие достигнуто. Здесь лучше всего воспользоваться процедурой, которая преобразует матрицу типа ***martix*** в результирующий вектор Result типа ***vector***.

Использование слоя сообщений и условий моделирования для изменения модели

Итак, слой сообщений предназначен для определения типов данных сложной структуры, например, ***Vector***(n) ***is*** (***Element***); ***Matrix***[n-1,n] ***is*** (***Element***).

Синтаксические правила для определения типа:

<определение типа>::=<имя типа>(<параметры>) ***is*** (составляющая типа){,<составляющая типа>})

<составляющая типа>::=<объявление селектора>{***or*** <объявление селектора>}

<объявление селектора> ::=<имя типа>(<параметры>) <идентификатор>(<параметры>)[<список граничных пар>]

Пусть составляющая типа Element имеет базовый тип ***integer***;

Задание переменных типа **Vector** и **Matrix** выглядит следующим образом: **Vector** (n) A, где n – входной параметр рутины, опреде-

ляет количество генералов в описанном выше примере.

Для того чтобы во время имитационного эксперимента можно было бы не изменять текст программного кода рутины и заново выполнить трансляцию этой рутины, целесообразно ввести операции (языковые конструкции), позволяющие изменить структуры данных типа ***Vector*** и ***Matrix***.

Соответствующее условие моделирование можно описать следующим образом:

conditions of simulation Generals [***integer*** n] (***input integer (***k); ***output boolean*** answer) (… m = Nodenumber ***if*** m<> n ***then vector*** (m) …)

Здесь, если количество вершин изменилось, то следует заново определить типы ***vector*** и ***matrix,*** т.е.запустить параметризованную процедуру. В качестве фактического параметра используют новое значение количества вершин графа, представляющего структуру имитационной модели. Проверка количества вершин выполняется всякий раз по окончании очередного шага моделирования.

Есть еще способ изменения типов данных во время проведения имитационного эксперимента: выполнить наложение новой рутины на вершины модели (запуск параметризованной процедуры), в качестве фактического параметра опять же использовать количество вершин.

```
conditions of simulation Generals [integer n] (input integer (k); output boolean answer) (….m = Nodenumber; (* определим количество вершин*)
 if m<> n
(* если количество вершин изменилось, то надо наложить на вершины новый экземпляр рутины *)
then
let goodgeneral(m) be general; //создание экземпляра рутины
for i:=1 by 1 to m do
    //Наложение рутин на вершины
      put goodgeneral on Army.General[i]
  endf;
```

Знания о модели сохраняются в базовой онтологии.[36, 38, 39] При проведении имитационного эксперимента знания извлекаются из онтологии и используются ядром для изменения всех переменных модели с типами ***vector*** и ***matrix.***

Заключение

Итак, в языке Triad используются языковые конструкции и соответствующее программное обеспечение, которые позволяют автоматизировать операции добавления и удаления вершин в графе, представляющем собой структуру распределенной информационной системы. Добавление и удаление вычислительных узлов в распределенной вычислительной среде возможно, поскольку свойство масштабируемости предполагает наращивание вычислительной мощности системы при увеличении количества вычислительных узлов распределенной системы, а свойство отказоустойчивости – сохранение свойств и работоспособности распределенной системы. В статье представлены языковые конструкции Triad-системы, позволяющие автоматизировать операции добавления и удаления вершин графа, который представляет структуру модели. Сложность заключается не только в том, чтобы изменить структуру имитационной модели, но и сценарий поведения вычислительных узлов, реализующих тот или иной алгоритм или протокол. Изменяются также и структуры данных (сообщений), которыми обмениваются вычислительные узлы. Проблема добавления и удаления узлов, которая сводится к выполнению операций добавления и удаления вершин графа имитационной модели в ходе имитационного эксперимента может быть решена при использовании параметризованной процедуры описания структуры модели. Соответственно изменение сценария поведения узлов выполняется за счет использования параметризованных процедур обработки данных в языке Triad, процедуры описания структуры сообщений и условий моделирования, которые и указывают, какие изменения необходимо выполнить в ходе имитационного эксперимента. Не возникает необходимости во внесении изменений в программный код. Это повышает надежность модели (а, следовательно, соответствует требованиям ее качества), и достоверность эксперимента.

Информация о структуре, рутинах, сообщениях, которыми обмениваются вершины графа, представляющего структуру модели, хранится в онтологиях системы Triad. В системе TRIADNS существуют базовая и предметно-ориентированные онтологии. Базовую онтологию используют для автоматического

доопределения модели, автоматизации операций добавления и удаления, а предметно-ориентированные – для настройки на предметную область.

Языковые конструкции языка Triad, соответствующее программное обеспечение и онтологии позволяют сократить время на разработку модели, упростить проведение имитационного эксперимента.

Список литературы

1. *The Network* Simulator – NS-2. Доступно на сайте: http://www.isi.edu/nsnam/ns (Проверено: 21 ноября 2013).
2. OPNET Modeler. Доступно на сайте: <http://www.opnet.com> (Проверено: 21 ноября 2013).
3. OMNET++ Community Site. Доступно на сайте: http://www.omnetpp.org. (Проверено: 21 марта 2013).
4. *Соколов Б.В., Юсупов Р.М.* Концептуальные основы квалиметрии моделей и полимодельных комплексов // Имитационное моделирование. Теория и практика: сб. докл. 2-й всерос. науч.-практ. конф. ИММОД–2005. Т. 1. СПб: ЦНИИТС. 2005. С. 65–70.
5. *Таненбаум Э.* Современные операционные системы. 3-е изд. Спб.: Питер, 2010. 1120 с. (Сер. "Классика computer science").
6. *Тель Ж.* Введение в распределенные алгоритмы. Изд-во МЦНМО, 2009. 616 с.
7. *Karatza H.D.* Current Trends In Modelling And Simulation Of Parallel And Distributed Systems. [Электронный ресурс]URL: ijssst.info/Vol-03/No-&2/ Karatza.pdf, 2001. P. 1–4 (проверено: 30.11.2013).
8. *Sulistio A., Yeo C.E., Buyya R*. Simulation of Parallel and Distributed Systems: A Taxonomy and Survey of Tools. [Электронный ресурс] URL: w.cs.mu.oz.au/~raj/papers/simtools.pdf. P. 1–19 (проверено: 30.11.2013).
9. *Sulistio A., Yeo C.E., Buyya R.* A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation Softw. Pract. Exper. 2004; 34:653–673 [Электронный ресурс] URL: www.cloudbus.org/papers/simulationtaxonomy.pdf (проверено: 30.11.2013).
10. *Salmon S, Elarag 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 2011Grand Arizona Resort Phoenix, AZ. P. 3266–3277.
11. *Neilson J.I.* PARASOL: A Simulator for Distributed and/or Parallel Systems. [Электронный ресурс] URL: www.sce.carleton.ca/rads/lqns/papers/parasol.pdf, 1991 (проверено: 30.11.2013).
12. *Casanova H.* Simgrid: A toolkit for the simulation of application scheduling. Proceedings 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2001), Brisbane, Australia, May 2001. IEEE Computer Society Press: Los Alamitos, CA, 2001.
13. *Бродский И.Ю.* Модельный синтез и модельно-ориентированное программирование как технология реализации имитационных моделей сложных многокомпонентных систем с ориентацией на параллельные и распределенные вычисления // Матер. конф. «Имитационное моделирование. Теория и практика» ИММОД–2013. Казань: Фэн Академии наук РТ. 2013. Т.1. С. 114–119.
14. *Бродский Ю.И.* Распределенное имитационное моделирование сложных систем. М.: ВЦ РАН, 2010. 156 с.
15. *Бродский Ю.И., Лебедев В.Ю.* Инструментальная система имитации MISS. М.: ВЦ АН СССР. 1991. 180 с.
16. *Бродский Ю.И., Павловский Ю.Н.* Разработка инструментальной системы распределенного имитационного моделирования // Информации-онные технологии и вычислительные системы. 2009. № 4. С. 9–21.
17. *Миков А.И., Замятина Е.Б.* Технология имитационного моделирования больших систем // Тр. Всерос. науч. конф. «Научный сервис в сети Интернет». М.: Изд-во МГУ. 2008. С.199–204.
18. *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 Con-ference 13–16 December 2009. Hilton Austin Hotel, Austin, TX. P. 797–806.
19. *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.
20. *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 Simula-

tion Conference 7–10 December 2003. The Fairmont New Orleans, New Orleans, LA. P. 124–134.

21. *Замятина Е., Ермаков С.* Алгоритм синхронизации объектов распределенной имитационной модели в TRIAD.Net. Appli-cable Information Models. ITHEA, Sofia, Bulgaria, 2011. ISBN: 978-954-16-0050-4. С. 211–220.
22. *Волканов Д.Ю. и др.* Методика использования системы имитационного моделирования РВС РВ Диана, основанной на HLA. Матер. конф. «Имитационное моделирование. Теория и практика» ИММОД–2013. Казань: Фэн Академии наук РТ. 2013. Т.1. С. 322–326.
23. *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.
24. *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/ (проверено: 30.11.2013).
25. *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. P.173–180.
26. *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. P. 111–114.
27. *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. P. 124–134.
28. *Ломазова И.А.* Вложенные сети Петри: моделирование и анализ распределенных систем с объектной структурой. М: Научный мир, 2004. 208 с.
29. *Башкин В.А., Ломазова И.А.* Эквива-лентность ресурсов в сетях Петри. М.: Научный мир, 2007. 208 с.
30. *В.И.Гурьянов.* Моделирование классифи-каций на визуальном языке имитационного моделирования UMLSP. Матер. конф. «Имитационное моделирование. Теория и практика» ИММОД–2013. Казань: Фэн Академии наук РТ, 2013. Т.1. С. 128–133.
31. *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. P. 15-20.
32. *Mikov A.I.* Formal Method for Design of Dynamic Objects and Its Implementation in CAD Systems // Gero J.S. and 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, 1995. P. 105–127.
33. *Крюков В.А.* Операционные системы распределенных вычислительных систем (распределенные ОС). Доступно на сайте www.-parallel.ru (Проверено: 25.11.2013).
34. *Замятина Е.Б., Михеев Р.А.* Использование мультиагентного подхода и онтологий для моделирования компьютерных сетей // Матер. 4 междунар. науч.-техн. конф. «Инфокоммуни-

кационные технологии в науке, производстве и образовании» 28–30 июня 2010. Ставрополь. С. 175–180.
35. *Замятина Е.Б., Миков А.И.* Программные средства системы имитации Triad.Net для обеспечения ее адаптируемости и открытости. Информатизация и связь. 2012. № 5. С. 130–133.
36. *Миков А.И., Замятина Е.Б.* Инструментальные средства имитационного моделирования для анализа бизнес-процессов и управления рисками // Информатизация и связь. 2011. № 3. С. 14–16.
37. *Mikov A.,Zamyatina E.,Kozlov A. Ermakov S.* Some Problems of the Simulation Model Efficiency and Flexibility. Proceedings of «2013 8th EUROSIM Congress on Modelling and Simulation EUROSIM 2013», Cardiff, Wales, United Kingdom, 10–13 of September. P .532–538.
38. *Замятина Е.Б., Миков А.И.* Применение онтологий и принципов организации сервис-ориентированно архитектуры при проек-тировании и реализации системы имитационного моделирования // Матер. 3-й междунар. науч.-техн. конф. «Технологии разработки информационных систем ТРИС-2012». Т.1. Таганрог. Изд-во Технол. ин-та ЮФУ. Ростов-на-Дону, 2012. 9 сентября. С. 61–65.
39. *Mikov A., Zamyatina E., Mikheev.* Linguistic and Program Tools For Debugging and Testing Of Simulation Models Of Computer Networks. International Journal “Information Models and Analyses, 2013. V. 2. № 1. Sofia. 1000. P.O.B. 775. Bulgaria. P. 70–80.

Some features of simulation of the distributed information systems

E. B. Zamyatina, A. I. Mikov, R. A. Mikheev
Perm State University, Russia, 614990, Perm, Bukirev st., 15
e_zamyatina@mail.ru; (342) 2 396 594

The problems of simulation of distributed information systems are discussed. There are a large number of specialized software systems dedicated to design a simulation model of distributed systems and to run a simulation experiment. The article deals with software and language tools of CAD Triad.Net, considers the main features of the simulation model and how to automate changes during simulation experiment. Among the properties of distributed systems one can emphasizes the properties of scalability and fault tolerance, as a result of which the number of computing nodes of the system being modelling may be changed. Particular attention is paid to the automation of operations to add and remove model objects associated with the calculating nodes.

Key words: *ddistributed information systems; simulation; scalability; fault tolerance; automation; ontology.*