ПРОБЛЕМЫ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ И ГИБКОСТИ СИСТЕМ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

А. И. Миков, Е. Б. Замятина*

Кубанский государственный университет, 350040, Краснодар, Россия
* Пермский государственный университет, 614990, Пермь, Россия

УДК 004.942, 004.896

Исследовано влияние автоматизации и интеллектуализации на эффективность работы систем имитационного моделирования. Рассмотрены вопросы оптимизации времени имитационного эксперимента за счет использования ресурсов нескольких вычислительных узлов и предложены решения, связанные с синхронизацией объектов имитационной модели, а также балансировкой нагрузки на вычислительные узлы. Изучены проблемы сокращения времени создания и отладки имитационных моделей за счет автоматизации этого процесса и повышения адаптируемости инструментальных средств имитационного моделирования. Для решения этих проблем предложено использовать методы искусственного интеллекта.

Ключевые слова: имитационное моделирование, распределенная имитационная модель, консервативный алгоритм синхронизации, оптимистический алгоритм синхронизации, верификация, валидация, мультиагентный подход, онтология.

Simulation is one of the famous and may be only single method for the investigation of complex dynamic systems. The problems being investigated by simulation become more and more complicate and it is necessary to reduce the total time needing for simulation model design, verification, validation and simulation run. The authors suppose that the effectiveness and flexibility of a simulation system (program tools and linguistic construction) depend on degree of automation and intellectualization. This paper discusses the problem of design and implementation of the distributed and parallel simulation systems using the power of several calculating nodes of nodes (or cluster or multiprocessor computer). Authors suggest original approach to the synchronization and load balancing of the objects of distributed simulation model. Furthermore authors consider the problem of the reduction of time needing to simulation model design and debugging, the problems of the simulation tools adaptability and suggest using models and methods of artificial intelligence in these cases.

Key words: simulation, distributed simulation model, conservative algorithm of synchronization, optimistic algorithm of synchronization, verification, validation, multiagent approach, ontology.

Введение. Интерес к имитационному моделированию обусловлен использованием его при исследовании сложных динамических систем и процессов в различных областях науки, бизнеса и производства. Ежегодно проводятся конференции по имитационному моделированию, появляются новые программные системы имитационного моделирования, как универсальные [1–3], так и направленные на решение проблем конкретной предметной области [4–6] (здравоохранение, производство, авиаперевозки, логистика, управление транспортными потоками, управление бизнес-процессами и т. д.).

Однако при использовании в качестве метода исследования имитационного моделирования возникает проблема выбора соответствующей системы имитационного моделирования (СИМ) программного продукта, предоставляющего пользователю те или иные средства для проведения имитационного эксперимента. Как правило, исследователи заинтересованы в

том, чтобы имитационный эксперимент выполнялся за максимально короткий период времени. При этом большое значение имеет надежность проведения эксперимента, а также удобство и эффективность работы с СИМ за счет использования различных средств автоматизации на различных этапах имитационного моделирования.

Одним из способов повышения скорости и надежности имитационного эксперимента является применение вычислительных ресурсов нескольких вычислительных узлов локальной сети, кластера или многопроцессорной ЭВМ. Преимущества проведения распределенного имитационного эксперимента указаны в работах [7–12]. В ходе распределенного имитационного эксперимента возникает необходимость синхронизации распределенных по вычислительным узлам объектов имитационной модели (ИМ) и поддержки каузальности событий во время имитационного прогона. Для решения этой проблемы применяются консервативный и оптимистический алгоритмы [7, 13, 14].

В работах, публикуемых в последнее время, данные алгоритмы постоянно совершенствуются, что позволяет сократить время их выполнения [15]. В настоящей работе для повышения гибкости и ускорения работы алгоритма предлагается использовать знания о модели [16], включающие знания исследователя о предполагаемом поведении модели (представленные в виде продукционных правил) и знания, извлекаемые автоматически алгоритмом синхронизации при функционировании модели.

Еще одним способом уменьшения времени имитационного эксперимента является организация равномерного распределения нагрузки на вычислительные узлы вычислительной системы (ВС) при проведении распределенного (параллельного) эксперимента. Действительно, возникновение дисбаланса нагрузки может свести выигрыш от использования нескольких вычислительных узлов к нулю: часть вычислительных узлов простаивает, а часть работает интенсивно, но вследствие большого количества задач на вычислительном узле скорость их выполнения остается низкой. Для того чтобы обезопасить распределенную имитационную модель от дисбаланса нагрузки, предлагается использовать управляемую динамическую балансировку, основанную на знаниях. В правилах, управляющих балансировкой, используются знания пользователей о модели [17, 18].

В начале итерационного процесса моделирования исследователь может еще не знать некоторых деталей описываемой им имитационной модели (например, не знать подробно поведение ряда устройств в проектируемой компьютерной сети). Тем не менее пользователю необходимо получить хотя бы приближенные результаты моделирования. В этом случае целесообразно использовать не полностью определенную модель, например, можно автоматически доопределять модель. При доопределении модели применяется онтологический подход [19]. Кроме того, исследователь имеет возможность использовать имитационные модели, имеющиеся на сайтах Интернета. Программные средства СИМ Triad.Net преобразуют их в онтологию Triad-модели, а затем в Triad-модель.

Наряду с автоматизированным доопределением и автоматизированным переиспользованием модели в СИМ Triad этап верификации и валидации модели также автоматизирован [20].

В настоящей работе представлена система моделирования Triad.Net — новая версия ранее разработанной системы имитации Triad, в которой учтены современные возможности распределенной обработки информации и используются новые программные технологии. Разработка системы, предназначенной для работы в авиационной промышленности, была начата в Пермском университете в середине 80-х гг. XX в. На идеологию этой системы существенное влияние оказали работы по имитационному моделированию и приложениям теории графов, проводившиеся в то время в ВЦ СО АН СССР [21–23].

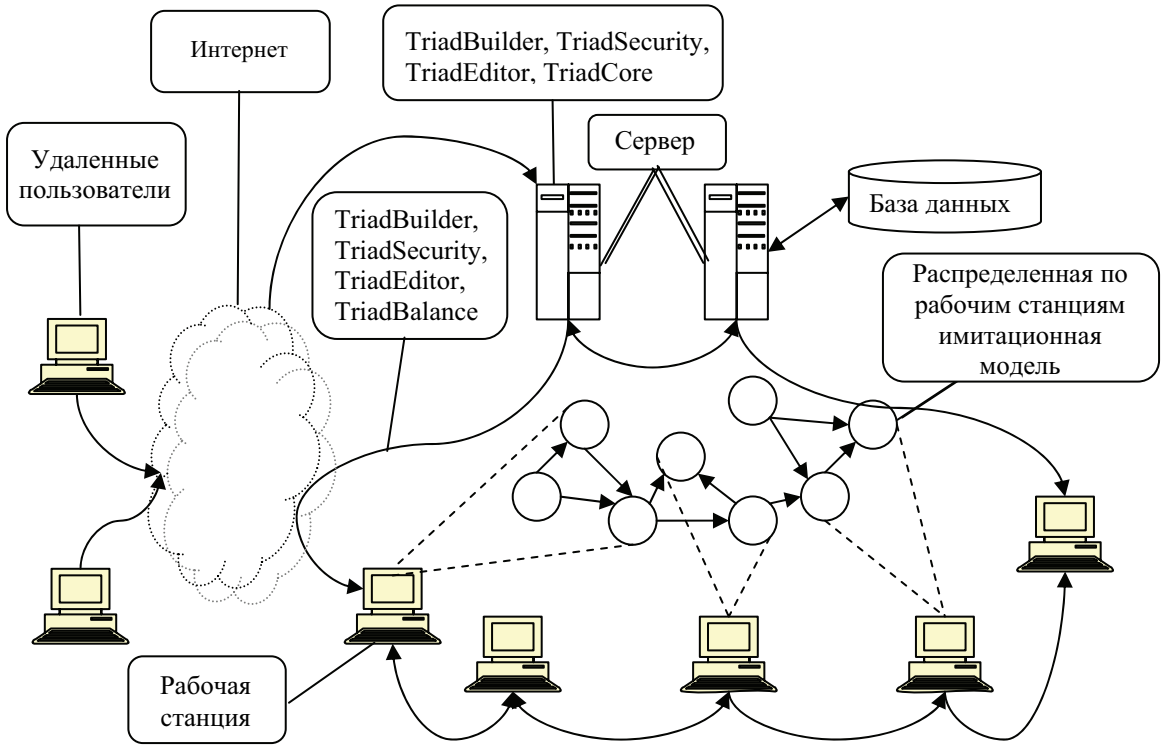

Рис. 1. Архитектура системы имитационного моделирования Triad.Net

Современная система Triad.Net развивает основные теоретические подходы, заложенные в проекте Triad, а в технологическом смысле является новой системой. Рассмотрим более подробно архитектуру новой версии СИМ Triad.Net.

Архитектура системы имитации Triad.Net. Система имитации Triad.Net включает следующие компоненты: компилятор, ядро, графический редактор, подсистему отладки, подсистему синхронизации распределенных объектов модели, подсистему балансировки [24], подсистему организации удаленного доступа [25], подсистему защиты от внешних и внутренних угроз [26], подсистему автоматического доопределения модели (рис. 1).

Ниже указаны функции каждого компонента. 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; TriadCore (ядро системы) включает библиотеки классов основных элементов модели; TriadEditor (редактор моделей) предназначен для работы с моделью как в удаленном, так и в локальном режиме, локальный режим предполагает работу с системой в том случае, если отсутствует удаленный доступ; TriadBalance — подсистема балансировки; TriadSecurity — подсистема безопасности; TriadBuilder — подсистема автоматического доопределения частично описанной модели, база данных, в которой хранятся экземпляры элементов модели; TriadMining — подсистема интеллектуальной обработки результатов моделирования.

Система моделирования Triad.Net может функционировать как в параллельном (распределенном), так и в последовательном режиме имитации. Кроме того, возможен удаленный доступ к системе и предусмотрена разработка программных средств для совместной работы географически удаленных пользователей над одним и тем же проектом.

Представление имитационной модели. Описание имитационной модели в Triad [27, 28] состоит из трех слоев: слоя структур (STR), слоя рутин ($ROUT$) и слоя сообщений (MES). Таким образом, модель в системе Triad можно определить как $M = \{STR, ROUT, MES\}$.

Слой структур представляет собой совокупность объектов, взаимодействующих посредством посылки сообщений. Каждый объект имеет полюсы (входные P_{in} и выходные P_{out}), которые служат соответственно для приема и передачи сообщений. Слой структур можно представить в виде графа. В качестве вершин графа следует рассматривать отдельные объекты. Дуги графа определяют связи между объектами.

Объекты действуют по определенному алгоритму поведения, который описывают с помощью рутины (rout). Рутина представляет собой последовательность событий e_i, планирующих друг друга ($e_i \in E,\ i = 1 \div n;\ E$ — 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 Pr_{in} и выходные Pr_{out} полюсы. Входные полюсы служат для приема сообщений, выходные полюсы — для их передачи. В множестве событий рутины выделено входное событие e_{in}. Все входные полюсы 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. Алгоритмом имитации называются совокупность объектов, функционирующих по определенным сценариям, и синхронизирующий их алгоритм. Последовательный алгоритм является событийно-ориентированным. Каждая рутина имеет свой локальный календарь событий $tloc_i$. При выполнении очередного события осуществляется поиск события с минимальным временем (поиск в календарях событий всех рутин), и управление передается рутине, которая включает это событие.

Сбор и обработка информации о функционировании имитационной модели. Для сбора, обработки и анализа имитационных моделей в системе Triad.Net существуют специальные объекты — информационные процедуры и условия моделирования, реализующие алгоритм исследования.

Информационные процедуры ведут наблюдение только за теми элементами модели (событиями, переменными, входными и выходными полюсами), которые указаны пользователем. Если в какой-нибудь 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.

Запуск модели на выполнение. Имитационная модель, описанная на языке Triad, может быть запущена на моделирование с помощью специального оператора simulate:

simulate <список элементов моделей, за которыми установлено наблюдение> on conditions of simulation <наименование> (список фактических настроечных параметров>)[<список входных и выходных фактических параметров интерфейса>].

Следует отметить, что исследователь может одновременно запустить на моделирование сразу несколько моделей, указав элементы, за которыми будут следить информационные процедуры, перечисленные в условиях моделирования. Кроме того, можно указать сразу

несколько условий моделирования. Например, при моделировании компьютерных сетей целесообразно рассматривать их с разных точек зрения: исследователей может интересовать стоимость проектируемых компьютерных сетей, их производительность, надежность, доступность. Во всех этих случаях набор собираемых данных может быть различным, разными способами обрабатываются наблюдаемые данные, условия завершения моделирования также могут различаться. Для упрощения задачи целесообразно провести несколько имитационных экспериментов с моделью, используя различные критерии (условия моделирования).

Распределенная система имитации. Реализация распределенной системы имитации представляет определенные трудности. Подтверждением тому служат алгоритмы синхронизации объектов имитационной модели, которые распределены по вычислительным узлам вычислительной системы и действуют по определенному сценарию (в Triad сценарий каждого объекта представлен соответствующей рутиной $rout_i$). Распределенная система имитации обычно представляется в виде совокупности логических процессов LP_i, взаимодействующих посредством обмена сообщениями.

Алгоритмы синхронизации. При реализации алгоритма синхронизации объектов моделирования, распределенных по разным вычислительным узлам, традиционно используются два подхода: консервативный и оптимистический. Основной целью этих алгоритмов является обеспечение выполнения каждым логическим процессом событий в порядке неубывания их временных меток, что позволяет сохранить причинно-следственные связи.

Задача консервативного алгоритма — определить время, когда обработка очередного события из списка необработанных событий является “безопасной”. Событие является безопасным, если можно гарантировать, что в дальнейшем процесс не получит от других процессов сообщение с меньшей временной меткой. Консервативный подход не позволяет обрабатывать событие, до тех пор пока не убедится в его безопасности. В отличие от консервативных алгоритмов, не допускающих нарушения ограничения локальной каузальности, оптимистические методы не накладывают такого ограничения. При нарушении локальной каузальности (логический процесс получает событие, имеющее меньшую временную отметку по сравнению с уже обработанными событиями) происходит восстановление правильного порядка [29, 30]. Наиболее известный алгоритм из класса оптимистических алгоритмов синхронизации — TIME WARP [31]. Для восстановления порядка выполняется откат, последующие события обрабатываются повторно в хронологическом порядке. Откатываясь назад, процесс восстанавливает состояние, имевшее место до обработки событий (все состояния системы сохраняются), и отказывается от сообщений, отправленных в результате обработки событий, которые необходимо отменить в результате отката. Для выполнения отката от этих сообщений разработан механизм антисообщений.

Для некоторых специфических моделей производительность работы консервативных алгоритмов в несколько раз превышает производительность работы оптимистических, для других моделей выигрыш во времени дает оптимистический алгоритм. Для увеличения производительности консервативного алгоритма необходимо расширить горизонт безопасных событий, а для повышения производительности работы оптимистического алгоритма — сократить количество откатов, сдержать чрезмерное продвижение модельного времени. Так или иначе, если алгоритм синхронизации располагает информацией о модели (lookahead, lookback, временная метка следующего события и т. п.), то производительность его работы повышается. Таким образом, для оптимизации времени выполнения распределенного имитационного эксперимента целесообразно использовать информацию о модели, в том числе

знания исследователя о конкретной модели, что позволит адаптировать алгоритм синхронизации в конкретном случае. Для хранения информации исследователя о модели предлагается использовать продукционные правила. Кроме того, знания должны извлекаться из самой модели.

Рассмотрим алгоритм, созданный с использованием такой информации на основе оптимистического алгоритма. С точки зрения моделирования главным является соблюдение оптимального порядка выполнения модели во времени.

Итак, обычно пользователь имеет представление о поведении модели. В результате могут быть построены продукционные правила в виде IF e_1 AND e_2 AND e_3 AND ... AND e_n THEN e_k CF <0...100>. Здесь e_k зависит от $e_1, e_2, ..., e_n; CF$ — коэффициент доверия (0 — доверие отсутствует, 100 — доверие максимальное). Правила отражают каузальную зависимость между событиями, а поскольку знания не являются точными, каждому правилу приписывается коэффициент доверия.

Однако детальное описание поведения модели в виде правил существенно затруднено. Необходимо извлекать информацию о поведении модели автоматически. Эту информацию также будем хранить в виде правил (системные правила). Системные правила формируются на стадии трансляции (процедуры обработки событий и сообщений) и стадии выполнения модели (анализ последовательности обработки событий). Количество правил может достаточно быстро увеличиваться, поэтому для его ограничения следует использовать следующие критерии: 1) генерация правил происходит только для объектов системы, находящихся в разных локальных процессах; 2) наиболее значимыми правилами являются правила, способствующие предотвращению временного парадокса (они генерируются из тех событий, которые ранее вызвали временной парадокс). Для сбора, обработки и распространения информации алгоритма синхронизации целесообразно реализовать специальных агентов на вычислительных узлах. Информация (центральная база знаний) и главный управляющий процесс находятся на выделенном сервере.

Агент сбора и обработки информации определяет безопасное событие следующим образом: происходит поиск события в заключениях всех правил в базе знаний, а затем, в случае если событие обнаружено, выполняется логический вывод по правилам. Далее выбирается событие с максимальным коэффициентом доверия.

Испытания, проведенные для разных моделей и на ВС с различными конфигурациями, показали, что алгоритм дает выигрыш во времени. Наиболее эффективно этот алгоритм работал для модели клиент — сервер.

Балансировка нагрузки на вычислительные узлы. Однако выигрыш от параллельного расчета распределенной имитационной модели может быть сведен к нулю, в случае если возникнет дисбаланс загрузки вычислительных узлов. Ниже перечислены основные причины возникновения дисбаланса: 1) гетерогенность вычислительной системы (узлы вычислительной системы имеют разную производительность, а линии связи — разную пропускную способность); 2) гетерогенность имитационной модели, которая создает большую нагрузку на вычислительные узлы, выполняя одно событие за другим, в то время как другие могут быть приостановлены в состоянии ожидания прихода того или иного сообщения; 3) нагрузка на вычислительные узлы, которую создают сторонние приложения.

Таким образом, целесообразно корректировать возникающий дисбаланс. С этой целью обычно разрабатывается специальное программное обеспечение, которое следит за нагрузкой вычислительных узлов и восстанавливает равномерное распределение приложений по вычислительным узлам, перенося часть компонентов приложений на другие, менее загру-

женные узлы. При этом программное обеспечение, выполняющее балансировку, следит за передачей сообщений по линиям связи. Нагрузка линий связи также должна быть сбалансирована.

Восстановление баланса нагрузки является известной задачей, которая имеет множество решений. Разработано большое количество алгоритмов. Однако очень часто эти алгоритмы применимы только для решения конкретной задачи. Существуют также решения, применимые для оптимизации распределенного имитационного эксперимента [32–34]. Однако, несмотря на то что разработчики подсистемы балансировки SPEEDES и Charm++ пытались создать алгоритмы балансировки, способные адаптироваться к изменяющейся обстановке, к особенностям той или иной имитационной модели, результаты экспериментов показали, что применение этих алгоритмов эффективно лишь в частных случаях. Действительно, найти универсальный алгоритм, эффективный (сокращающий время выполнения имитационного эксперимента) для любой имитационной модели, практически невозможно.

Для хотя бы частичного решения этой проблемы можно применить управляемую балансировку. Управляемая балансировка предполагает наличие в программном обеспечении, используемом для восстановления равномерной загрузки, экспертного компонента, который на основании логического вывода предлагает решения для переноса избыточной нагрузки с перегруженного вычислительного узла на менее загруженный.

Кроме того, можно использовать языковые средства, которые позволяют описать оригинальный алгоритм балансировки, учитывающий особенности конкретной имитационной модели и вычислительной системы.

Задача балансировки. Задача балансировки — это задача отображения неизоморфных связных графов $B : TM \to NG$ (TM — множество графов моделей; NG — множество графов-конфигураций компьютерной сети). Граф $G = \{C, Ed\}$ из NG определяется множеством вычислительных узлов C и множеством ребер Ed, обозначающих линии связи. Граф M из TM задает имитационную модель. Имитационную модель M можно представить и как совокупность логических процессов $MP = \{LP_j\}, j = 1, ..., n$, взаимодействующих между собой путем передачи сообщений.

Реализация подсистемы балансировки в Triad.Net. Система балансировки в Triad.Net включает следующие компоненты: 1) подсистему анализа и принятия решения; 2) подсистему миграции; 3) подсистему мониторинга имитационной модели; 4) подсистему мониторинга вычислительной системы; 5) базу знаний с правилами перераспределения нагрузки; 6) редактор правил; 7) механизм вывода (рис. 2).

В базе знаний содержатся правила миграции, которые используют данные о текущем состоянии модели (частота обменов сообщениями между логическими процессами, частота выполнения тех или иных событий и т. д.) и текущем состоянии ВС (загрузка процессоров, загрузка линий связи). Наряду с правилами миграции в базе знаний содержатся данные о поведении конкретной имитационной модели. Например, пользователь знает, что через определенный промежуток времени (100 единиц модельного времени) интенсивность обмена между двумя логическими процессами значительно увеличится. В этом случае целесообразно принять решение о переносе интенсивно взаимодействующих процессов на соседние узлы с соответствующей пропускной способностью линий связи (или расположить их на одном узле).

Подсистема мониторинга ВС собирает информацию о текущем состоянии вычислительной системы, на которой выполняется имитационный эксперимент. Подсистема мониторинга имитационной модели собирает информацию о текущем состоянии имитационной модели (использует механизм информационных процедур). Подсистема анализа получает информа-

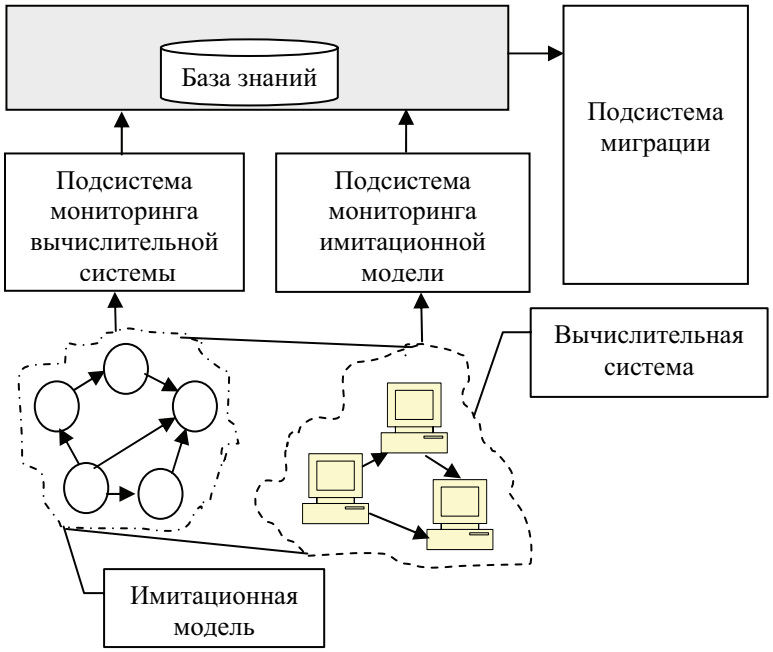

Рис. 2. Архитектура подсистемы балансировки с централизованным алгоритмом

цию от подсистем мониторинга и принимает решение о перераспределении нагрузки, после чего обращается к экспертной системе и получает от нее рекомендации о том, какие объекты имитационной модели следует перенести и на какие вычислительные узлы. Далее управление передается подсистеме миграции, которая и осуществляет перенос объектов.

Однако управление является централизованным, все правила хранятся в единой базе знаний, подсистема мониторинга вычислительной системы также располагается на одном из выделенных вычислительных узлов, взаимодействуя с остальными.

Для того чтобы ресурсы распределенной вычислительной среды были использованы наиболее полно, а также с целью оптимизации времени выполнения алгоритма балансировки предлагается использовать мультиагентный подход.

Мультиагентная система балансировки нагрузки. Динамическая система балансировки TriadBalance является мультиагентной, т. е. состоит из агентов различного типа: 1) агента-датчика вычислительного узла; 2) агента-датчика имитационной модели; 3) агента анализа; 4) агента миграции; 5) агента распределения (рис. 3).

Агенты каждого типа действуют по своему сценарию для достижения цели, а все вместе реализуют балансировку распределенной имитационной модели.

Агент-датчик вычислительного узла постоянно собирает информацию о нагрузке вычислительного узла и состоянии линий связи.

Агент-датчик имитационной модели ведет постоянное наблюдение за объектами имитационной модели во время имитационного прогона, регистрируя интенсивность обмена между объектами, частоту выполнения тех или иных событий, частоту изменения переменных и т. д. Агент-датчик имитационной модели использует механизм информационных процедур.

Агент анализа, взаимодействуя с агентами-датчиками (эти агенты являются реактивными), принимает решение о необходимости перераспределения нагрузки; он является когнитивным объектом и, принимая решения, использует правила из экспертной системы.

Агент распределения получает информацию от агента анализа. Цель агента распределения — выявить порцию нагрузки (выбрать объекты имитационной модели) на вычис-

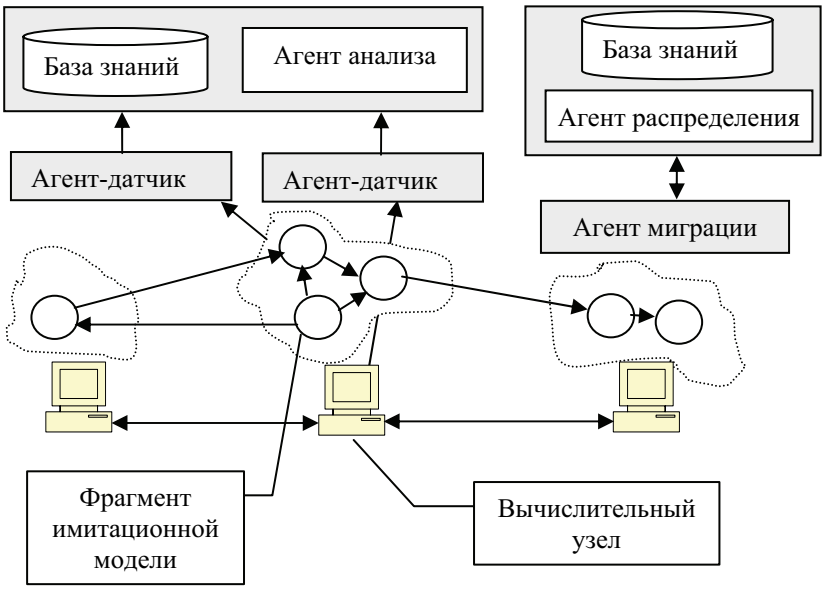

Рис. 3. Мультиагентная система балансировки

лительном узле, которую следует передать другим узлам, чтобы избежать дисбаланса, и определить целевой узел, на который следует перенести нагрузку.

Целевой узел отыскивается из числа соседних вычислительных узлов. Если же таковых среди вычислительных узлов-соседей нет, то агент распределения, взаимодействуя с агентами распределения соседних вычислительных узлов, узнает адрес вычислительного узла, способного разместить у себя дополнительную нагрузку.

Если нагрузка на вычислительном узле незначительна, то функцией агента распределения является извещение вычислительных узлов-соседей о том, что узел готов к выполнению дополнительных вычислений. Агент распределения также является когнитивным и действует по правилам из базы знаний. Эти правила определяются или модифицируются пользователем на начальном этапе балансировки (инициализация). База знаний включает информацию (представление агента) об узлах-соседях. При взаимодействии с агентами распределения узлов-соседей представление агента пополняется новыми знаниями.

Агент миграции осуществляет перенос нагрузки на целевой узел. На него возлагается обязанность выполнить это действие наиболее оптимальным способом.

Были разработаны метаправила с целью повышения адаптируемости когнитивных агентов при изменении условий проведения имитационного эксперимента. Проведено тестирование разработанного программного обеспечения. Исследования показали, что применение подсистемы балансировки позволяет сократить время проведения имитационного эксперимента. Итак, установлено, что применение подхода, основанного на знаниях, позволяет повысить эффективность системы моделирования и оптимизировать время выполнения имитационного эксперимента [35]. Ниже показано, каким образом использование онтологий позволяет сократить время, необходимое для создания модели, ее отладки и тестирования.

Доопределение модели. Известно, что онтология — это описание типов сущностей, имеющихся в предметной области, их свойств и отношений. Каждая предметная область (некая часть реального мира) может быть описана с помощью онтологий. Онтологии создаются и используются во множестве областей знаний, в частности, известны примеры

их успешного применения в имитационном моделировании. Однако создание онтологий для моделирования является достаточно сложной задачей, поскольку этот метод используется для исследования разнообразных систем, относящихся к различным предметным областям (химическим, физическим, транспортным и т. д.). Кроме того, методы имитационного моделирования основаны на математических, вероятностных и статистических расчетах, и, таким образом, онтологии для этих областей должны служить основой для всех остальных. Онтологии используются на различных этапах имитационного моделирования начиная с этапа сбора информации о моделируемой системе и заканчивая этапом валидации модели [36].

Примерами использования онтологий моделирования могут служить управляемые онтологиями среды моделирования, а также подходы к объединению различных федератов, разрабатываемые для HLA. Подход, создаваемый для HLA, использует онтологии для описания требований, которым должны удовлетворять интерфейсы федератов для успешного взаимодействия в федерации, а также для разработки этих требований с учетом знаний о моделируемой предметной области.

В работе [37] представлена онтология портов, рассматриваемая как средство автоматизации построения моделей из компонентов. Порты описывают интерфейс, определяющий границы компонентов или подсистем в конфигурации системы. Система представлена как конфигурация подсистем или компонентов, соединенных друг с другом через четко определенные интерфейсы. Онтологии успешно применяются и в других работах по имитационному моделированию [38].

Как сказано выше, на начальных этапах проектирования исследователь может описать модель частично, опустив описание поведения какого-либо элемента модели ($\mu r^* = \{STR, ROUT^*, MES\}$), не указав информационные потоки, воздействующие на модель ($\mu s^* = \{STR^*, ROUT^*, MES\}$), не определив правила преобразования сигналов в слое сообщений ($\mu m^* = \{STR, ROUT^*, MES\}$). Однако для запуска модели и последующего ее анализа все эти элементы должны быть так или иначе описаны (хотя бы приближенно). В Triad.Net доопределение выполняется подсистемой доопределения модели. На рис. 4 представлен процесс обработки имитационной модели.

Следует различать автоматическое и полуавтоматическое доопределение модели.

Полуавтоматическое доопределение моделей предполагает использование условий моделирования и погружение в среду моделирования. При полуавтоматическом доопределении исследователь вводит в часть `initial` условий моделирования: 1) операторы наложения рутин на вершину; 2) операторы наложения слоя сообщений; 3) операторы расшифровки вершины подструктурой; 4) операторы, реализующие операции над структурой модели (добавление и удаление вершин, дуг, входов и выходов и т. д.). При выполнении процесса имитации по оператору `simulate` симулятор сначала выполняет доопределение модели (при обработке операторов, записанных в части `initial` условий моделирования). Полуавтоматическое доопределение позволяет только на один акт имитации изменить воздействие информационных потоков (расшифровка вершины подмоделью, наложение рутины на вершину) или условия преобразования сигналов (наложение слоя сообщений). Для того чтобы провести исследования с другими информационными потоками или с другими правилами преобразования сигналов, необходимо запустить процесс моделирования с другими условиями моделирования: `simulate M on condition of simulation New_Condition (M.N1.a,M.N2.b)`, где фактические параметры M.N1.a, M.N2.b — переменные модели, за которыми ведется наблюдение в системе моделирования Triad.Net.

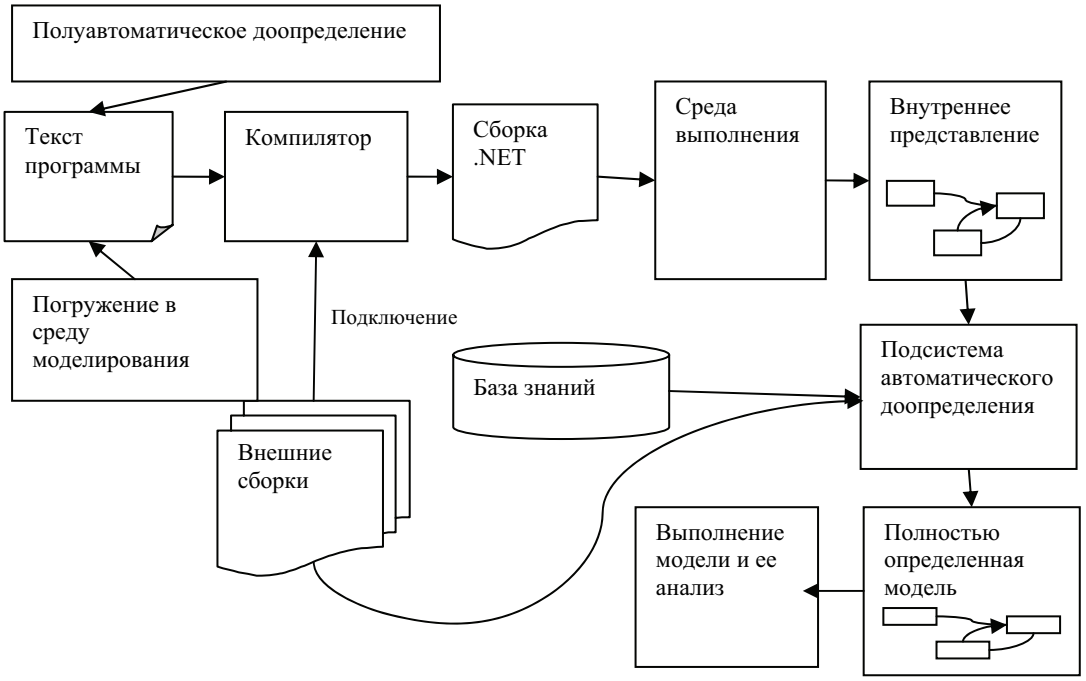

Рис. 4. Структура системы доопределения моделей в Triad.Net

Погружение в среду моделирования позволяет оперативно изменить информационные потоки, поступающие на внешние полюсы имитационной модели, и выполняется с помощью оператора расшифровки вершины v графом, который представляет собой структуру моделируемого объекта, представленного вершиной v.

Автоматическое доопределение модели предполагает, что пользователь работает с частично описанной моделью μr^*, в которой не определены алгоритмы поведения некоторых элементов. Во время компиляции компоненты подсистемы автоматического доопределения моделей выявляют вершины v_i, для которых исследователем не определены рутины $r_i = f(v_i), i = 1, ..., n$. Задача подсистемы автоматического доопределения — найти по определенным критериям подходящие рутины в базе экземпляров рутин и достроить модель.

Подсистема автоматического доопределения модели. Рассмотрим пример моделирования компьютерной сети, представленной на рис. 5, и описание этого фрагмента на языке Triad. Каждая рабочая станция имеет два соседних узла: рабочую станцию и маршрутизатор. Сообщение должно быть передано от одной рабочей станции к другой (не соседней). При передаче сообщений компьютерная сеть использует маршрутизатор. Точный сценарий поведения маршрутизаторов (Router) исследователю неизвестен. Задача системы автоматического доопределения модели состоит в том, чтобы для каждой вершины Router подобрать подходящую рутину из базы экземпляров рутин и выполнить действия, определенные оператором наложения рутины.

Автоматическое доопределение в Triad выполняется на основании дополнительной семантической информации. Семантическая информация включает такое понятие, как семантический тип. Семантический тип вводится для того, чтобы сгруппировать ряд объектов по смысловому, структурному, поведенческому типам. Так, при моделировании вычислительных систем для обозначения множества процессорных устройств может быть введен семантический тип Процессор, для обозначения элементов памяти — тип Модуль Памяти. При моделировании систем массового обслуживания будут уместны семантические типы Очередь, Генератор Заявок и т. п. Для того чтобы причислить объект к тому или иному семан-

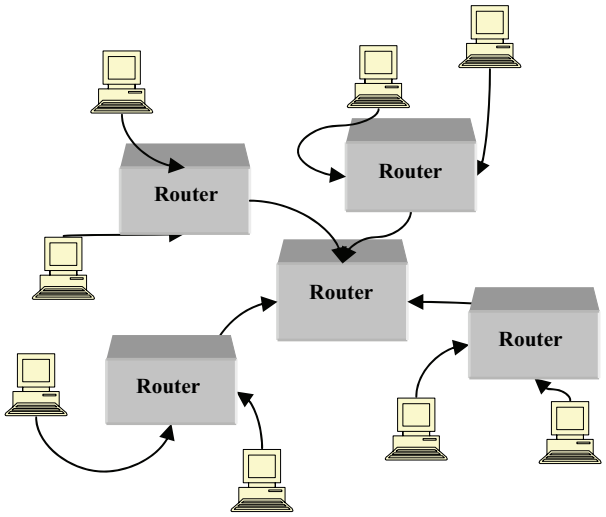

Рис. 5. Фрагмент компьютерной сети

тическому типу, в тексте программы используется специальный оператор: <имя объекта> → <имя типа>. Семантические типы объявляются специальным оператором type <имя типа>. Ниже приведен фрагмент программы, в которой используются семантические типы.

```
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;
```

В результате выполнения данной программы будет построена структура, описывающая небольшую сеть, терминальным вершинам которой будет присвоен семантический тип Host, а промежуточным — Router (рис. 6).

Семантические типы определяют смысловую нагрузку того или иного объекта модели. Для поиска экземпляра рутины используется база знаний, представленная в виде онтологий (рис. 7). В этих онтологиях описываются семантические типы, отношения наследования между ними, а также множества соответствующих этим типам экземпляров рутин и семантической информации, необходимой для проверки условий доопределения. Семантические типы представлены в виде иерархии классов онтологии. Использование такого подхода предполагает, что дочерние семантические типы будут описывать понятия, конкретизирующие понятия, соответствующие родительским типам. Например, при моделировании вычислительных систем на верхние уровни иерархии будут помещены семантические типы, соответствующие базовым понятиям, например Устройство. Дочерние типы будут представлять более конкретные понятия моделируемой предметной области: Устройства разделятся на Процессор, Модуль Памяти и т. д., процессоры могут быть классифицированы в зависимо-

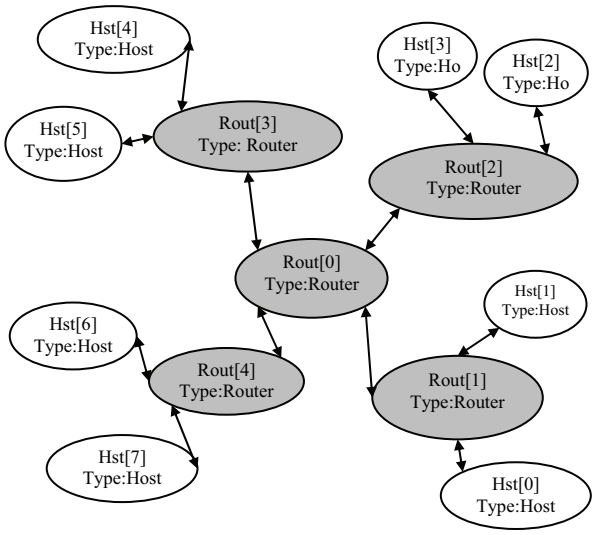

Рис. 6. Фрагмент модели

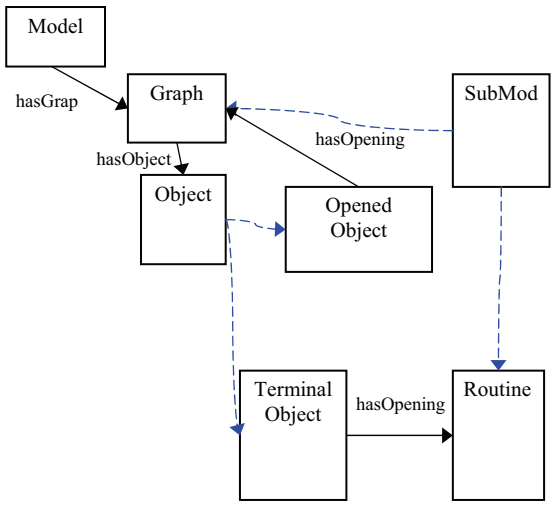

Рис. 7. Фрагмент базовой онтологии

сти от их архитектуры. Таким образом, семантический тип представляется в базе знаний как класс объектов, являющийся подклассом общего типа Object, соответствующим всему множеству вершин. При определении семантических типов возможно указание нескольких семантических типов для одного объекта.

В системе Triad.Net выделены условия доопределения терминальной вершины экземпляром рутины: условия специализации (совпадение семантических типов вершины и рутины из базы знаний), условия конфигурации (совпадение количества входных и выходных полюсов рутины и вершины) и условия декомпозиции (поиск рутины по графу окружения).

Представление знаний и базовая онтология. Для представления семантических знаний, необходимых для доопределения моделей, были выбраны онтологии, для представления онтологий — язык OWL, поскольку существует большое количество инструментальных средств работы с онтологиями OWL, поддерживающих возможность публиковать созданные онтологии в сети Internet и объединять информацию из различных источников, как локальных, так и находящихся в глобальной сети. Для работы с онтологиями используется инструментарий Jena OWL API. Семантические знания, необходимые для проверки условий автоматического доопределения, извлекаются из онтологического описания слоя структур модели. В качестве такого описания используется базовая онтология, импортируемая всеми создаваемыми онтологиями. В этой онтологии определены следующие классы: Model (класс, описывающий множество моделей языка Triad), SubMod (класс, описывающий множество всех экземпляров рутин и структур), Graph (класс, описывающий множество структур моделей и являющийся подклассом SubMod), Routine (класс, описывающий множество экземпляров рутин и являющийся подклассом SubMod), Object (класс, описывающий множество всех вершин структуры модели и являющийся суперклассом для всех семантических типов) и т. д. Фрагмент базовой онтологии представлен на рис. 7. Для работы с самими онтологиями реализован класс OntoManager, поддерживающий загрузку нескольких онтологий, их создание и сохранение. Кроме того, реализован класс TypeManager для работы с семантическими типами и поиска соответствующего экземпляра рутин.

Заключение. На примерах алгоритма синхронизации распределенных по вычислительным узлам ВС и алгоритма, реализующего балансировку нагрузки на вычислительных уз-

лах, показано, каким образом использование методов искусственного интеллекта, в том числе баз знаний, позволяет повысить эффективность разрабатываемой системы имитационного моделирования.

На примере процедуры доопределения частично описанной модели показана возможность применения онтологий. Следует отметить, что онтологии применяются и на других этапах жизненного цикла имитационного эксперимента (например, на этапе верификации и валидации). Использование онтологий позволяет, с одной стороны, ускорять процесс поиска элементов модели для ее доопределения, с другой — управлять процессом верификации и валидации. В настоящее время авторами данной работы проводятся исследования по импортированию с помощью онтологий в среду Triad.Net готовых моделей, созданных в других средах.

Список литературы

1. Krahl D. Extend: An interactive simulation tool // Proc. of the Winter simulation conf., New Orleans, 7–10 Dec. 2003 / Ed. by S. Chick, P. J. Sánchez, D. Ferrin, D. J. Morrice. Piscataway (New Jersey): Inst. of Electric. and Electron. Engrs, 2003. P. 188–196.

2. Nordgren W. B. Flexsim simulation environment // Proc. of the Winter simulation conf., New Orleans, 7–10 Dec. 2003 / Ed. by S. Chick, P. J. Sánchez, D. Ferrin, D. J. Morrice. Piscataway (New Jersey): Inst. of Electric. and Electron. Engrs, 2003. P. 197–200.

3. Bapat V., Sturrock D. T. The arena product family: Enterprise modeling solutions // Proc. of the Winter simulation conf., New Orleans, 7–10 Dec. 2003 / Ed. by S. Chick, P. J. Sánchez, D. Ferrin, D. J. Morrice. Piscataway (New Jersey): Inst. of Electric. and Electron. Engrs, 2003. P. 210–217.

4. Reindl S., Mönch M., Mönch L., Scheider A. Modeling and simulation of cataract surgery processes // Proc. of the Winter simulation conf., Austin, 13–16 Dec. 2009. / Ed. by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, R. G. Ingalls. Piscataway (New Jersey): Inst. of Electric. and Electron. Engrs, 2009. P. 1936–1945.

5. Brown J. E., Sturrock D. Identifying cost reduction and performance improvement opportunities through simulation // Proc. of the Winter simulation conf., Austin, 13–16 Dec. 2009. / Ed. by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, R. G. Ingalls. Piscataway (New Jersey): Inst. of Electric. and Electron. Engrs, 2009. P. 2145–2153.

6. Kontoyiannakis K., Serrano E., Tse K., et al. A simulation framework to evaluate airport gate allocation policies under extreme delay conditions // Proc. of the Winter simulation conf., Austin, 13–16 Dec. 2009 / Ed. by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, R. G. Ingalls. Piscataway (New Jersey): Inst. of Electric. and Electron. Engrs, 2009. P. 2332–2342.

7. Fujimoto R. M. Distributed simulation systems // Proc. of the Winter simulation conf., New Orleans, 7–10 Dec. 2003 / Ed. by S. Chick, P. J. Sánchez, D. Ferrin, D. J. Morrice. Piscataway (New Jersey): Inst. of Electric. and Electron. Engrs, 2003. P. 124–134.

8. Nance R. E. Distributed simulation with federated models: expectations, realizations and limitations // Proc. of the Winter simulation conf., Phoenix, 5–8 Dec. 1999 / Ed. by P. A. Farrington, H. B. Nembhard, D. T. Sturrock, G. W. Evans. Piscataway (New Jersey): Inst. of Electric. and Electron. Engrs, 1999. P. 1026–1031.

9. Meyer R. A., Bagrodia R. Parsec user manual. Release 1.1. (Univ. of California, Los Angeles), 1998. [Electron. resource]. pcl.cs.ucla.edu/projects/parsec.

10. Premore B. J., Nicol D. M. Parallel simulation of TCP/IP using TeD // Proc. of the Winter simulation conf., Atlanta, 7–10 Dec. 1997 / Ed. by S. Andradóttir, K. J. Healy, D. H. Withers, B. L. Nelson. Atlanta, 1997. P. 436–447.

11. Ferenci S., Perumalla K., Fujimoto R. An approach to federating parallel simulators // Workshop on parallel and distributed simulation, 2000. [Electron. resource]. http://www.cc.gatech.edu/ computing/pads /papers.html.

12. Yucesan E., Yah Chuyn Luo Y., Chen C-H., Lee I. Distributed Web-based experiments for optimization // Simulat. Practice Theory. 2001. V. 9, iss. 1/2. P. 73–90.

13. Вознесенская Т. В. Математическая модель алгоритмов синхронизации времени для распределенного имитационного моделирования // Программные системы и инструменты: Темат. сб. М.: Изд-во Моск. ун-та, 2002. № 1. С. 56–66.

14. Окольнишников В. В. Представление времени в имитационном моделировании // Вычисл. технологии. 2005. Т. 10, № 5. С. 57–77.

15. Carothers C. D., Perumalla K. S., Fujimoto R. M. Efficient optimistic parallel simulations using reverse computation. [Electron. resource]. www.cc.gatech.edu.

16. Замятина Е. Б., Ермаков С. А. Реализация распределенных алгоритмов для системы имитационного моделирования Triad.Net // Математика программных систем: Межвуз. сб. науч. ст. Пермь: Перм. ун-т, 2008. С. 120–129.

17. Замятина Е. Б., Миков А. И. Мультиагентная система балансировки распределенной имитационной модели // Тр. Междунар. науч.-техн. конф. "Интеллектуальные системы" (AIS'08) и "Интеллектуальные САПР" (CAD-2008). М.: Физматлит, 2008. Т. 2. С. 285–292.

18. Миков А. И., Замятина Е. Б., Осмехин К. А. Динамическое распределение объектов имитационной модели, основанное на знании // Proc. of the 13th Intern. conf. "Knowledge-Dialogue Solution". Varna, June 18–24, 2007. Sofia, 2007. P. 618-624.

19. Mikov A., Zamyatina E., Kubrak E. Implementation of simulation process under incomplete knowledge using domain ontology // Proc. of the 6th EUROSIM congress on modeling and simulation, Ljubljana (Slovenia), 9–14 Sept. 2007. Ljubljana: Tiskarna Pleško, 2007. V. 2.

20. Миков А. И., Замятина Е. Б. Интеллектуальные языковые и программные средства валидации имитационных моделей // Тр. конгресса по интеллектуальным системам и информационным технологиям AIS-IT'09, Дивноморское (Россия), 3–10 сент. 2009 г. М.: Физматлит, 2009. Т. 2. С. 54–61.

21. Миков А. И. Формализация процесса моделирования вычислительных систем // Системное моделирование. Новосибирск, 1985.

22. Миков А. И. Оценка сложности иерархического моделирования вычислительных систем // Системное моделирование. Новосибирск, 1985.

23. Миков А. И. Автоматизация синтеза микропроцессорных управляющих систем / Под ред. М. И. Нечепуренко. Иркутск: Иркут. гос. ун-т, 1987. 288 стр.

24. Миков А. И., Замятина Е. Б., Козлов А. А. Оптимизация параллельных вычислений с применением мультиагентной балансировки // Параллельные вычислительные технологии (ПаВТ'2009): Тр. Междунар. науч. конф., Нижний Новгород, 30 марта – 3 апр. 2009 г. Челябинск: Изд-во Юж.-Урал. ун-та, 2009. С. 599–604.

25. Mikov A., Zamyatina E., Firsov A. Software for remote parallel simulation // Inform. Theories Appl. 2007. V. 14, N 4. P. 389–395.

26. Миков А. И., Замятина Е. Б., Панов М. П. Мультиагентная система защиты распределенной имитационной модели с удаленным доступом // Advanced studies in software and knowledge engineering: Intern. Book Ser.; № 4 (Suppl. Intern. J. Inform. Technol. Knowledge). Sofia: ITHEA, 2009. V. 2. P. 90–97.

27. Mikov A. I. Simulation and design of hardware and software with triad // Proc. of the 2nd Intern. conf. on electronic hardware description languages. Las Vegas, 1995. P. 15–20.

28. Mikov A. I. Formal method for design of dynamic objects and its implementation in CAD Systems // Advances in formal design methods for CAD: Prepr. of the IFIP WG 5.2 Workshop on formal design methods for computer-aided design / Ed. by J. S. Gero, F. Sudweeks. Mexico, 1995. P. 105–127.

29. Bryant R. E. Simulation of packet communications architecture computer systems. MIT-LCSTR-188. 1977. Technical Report TR-188, MIT Laboratory for Computer Science, November, 1977. [Electron. resource]. http://www.cs.cmu.edu/ bryant/pubdir/MIT-LCS-TR-188.pdf.

30. Chandy K. M., Misra J. Distributed simulation: a case study in design and verification of distributed programs // IEEE Trans. Software Engin, 1978. V. SE-5(5). P. 440–452.

31. Jefferson D. R. Virtual time II: storage management in distributed simulation // Proc. of the 9th Annual ACM symp. on principles of distributed computing. Quebec City (Canada), Aug. 22–24, 1990. P. 75–89.

32. 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.

33. Wilson L. F., Nicol D. M. Automated load balancing in SPEEDES // Proc. of the Winter simulation conf. Piscataway (New Jersey): Inst. of Electric. and Electron. Engrs, 1995. P. 590–596.

34. 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/.

35. Миков А. И., Замятина Е. Б., Козлов А. А. Программные средства оптимизации распределенного имитационного эксперимента // Тр. Всерос. суперкомпьютер. конф. "Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность". 2009. С. 275–282.

36. Fishwick P., Miller J. A. Ontologies for modeling and simulation: Issues and approaches // Proc. of the Winter simulation conf. P. 259–264.

37. Liang V-C., Paredis C. J. J. A port ontology for automated model composition // Proc. of the Winter simulation conf. 2003. P. 613–622.

38. Benjamin P., Patki M., Mayer R. J. Using ontologies for simulation modeling // Proc. of the Winter simulation conf. / Ed. by L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, R. M. Fujimoto. P. 1161–1167.

Миков Александр Иванович — проф., зав. кафедрой Кубанского государственного университета; e-mail: alexander_mikov@mail.ru;
Замятина Елена Борисовна — доц. Пермского государственного университета; e-mail: e_zamyatina@mail.ru

Дата поступления — 3.09.2010