OSTIS-2015
(Open Semantic Technologies for Intelligent Systems)

УДК 004.8 + 004.942

ОПЫТ РАЗРАБОТКИ СИСТЕМЫ ИМИТАЦИИ, ОСНОВАННОЙ НА АГЕНТАХ

Замятина Е.Б.[*], Каримов Д.Ф.[**], Митраков А.А.[**]

[*]*Национальный исследовательский университет «Высшая школа экономики» (Пермский филиал)*

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

e_zamyatina@mail.ru,
googlicus@gmail.com,
mitrakov-artem@yandex.ru

В работе рассматривается экспериментальная система имитационного моделирования, основанная на агентах и особенности ее реализации. Обсуждаются вопросы целесообразности применения агентных систем имитации для решения ряда задач, выдвигаются требования, которые необходимо применять при их разработке, особенности организации систем имитации.
Ключевые слова: имитационное моделирование, онтология, адаптируемость, распределенное моделирование, агент, интеллектуальный агент

Введение

С развитием информационных технологий, ростом информационных ресурсов, совершенствованием сетевой инфраструктуры и увеличением производительности вычислительных машин появляются всё новые и новые задачи. Эти задачи являются более сложными и соответственно более затратными с точки зрения использования вычислительных ресурсов, кроме того, еще и менее формализуемыми. Новые задачи требуют поиска новых подходов и методов решения. Порою задачи бывают такими, что использование традиционных подходов и алгоритмов неприменимо в силу специфики области применения: недетерминированности, постоянной изменчивости, невозможности формализовать эти задачи и т.д.

Таким образом, возникает необходимость в более мощных и гибких интеллектуальных программных системах, способных непрерывно приобретать новые знания и изменять свою структуру и функции, развиваясь и адаптируясь к решаемым задачам и условиям внешней среды.

Одним из путей решения подобных задач можно назвать применение мультиагентных систем (МАС) – особой формы области искусственного интеллекта, которая базируется на знаниях и эвристических алгоритмах кооперативного поиска решения задачи.

Ключевым элементом этих систем становится программный агент, способный воспринимать ситуацию, принимать решения и взаимодействовать с другими агентами. В основу успешного коллективного решения задачи положено 3 фундаментальных принципа: кооперация, координация и коммуникация агентов.

В упрощённом виде можно представить агента как некоторую сущность, обладающую памятью и собственной базой знаний, умеющей находить решение некоторой узкой специфической задачи, взаимодействовать с другими агентами и менять правила поведения в динамике. Следует особо оговорить, что ни один агент в МАС не способен решить общую задачу самостоятельно, без взаимодействия с другими агентами.

Агента можно создать таким образом, что он будет иметь определенное отношение к принятию рискованных решений в условиях неопределенности. Агенты функционируют в едином виртуальном мире. В ходе переговоров агентов формируется текущее решение проблемы, которое гибко меняется в соответствии с динамичным изменением среды.

Тем не менее, успешному применению агентного моделирования на практике мешают различные проблемы, связанные в настоящее время в первую очередь с инструментарием для разработки таких моделей. Существующие системы либо достаточно

трудны для обучения с их последующим использованием. Зачастую они требуют навыков в программировании и не предназначены для использования пользователями, которые не являются специалистами в программировании. Кроме того, существуют проблемы, связанные с возможностью полного представления моделей. В этом случае требуется вести доработку системы, написания новых модулей и т.д. И необходимо отметить еще одну проблему – затраты вычислительных ресурсов для завершения имитационного эксперимента в приемлемое время. Для оптимизации имитационного эксперимента по времени необходимо распределить агенты имитационной модели (согласно определенной стратегии распределения) таким образом, чтобы они функционировали на различных компьютерах, или процессорах (или нескольких ядрах). При этом необходимо синхронизировать действия агентов (необходимо соблюдать каузальность событий).

Таким образом, при разработке системы имитации, основанной на агентах, необходимо придерживаться определенных правил и ограничений.

Дадим краткий обзор агентных систем имитации, попутно рассмотрим задачи, решение которых достигается с применением этих систем.

1. Обзор агентных систем имитации

В настоящее время существует достаточно большое количество агентных систем имитации. Одна из наиболее известных систем моделирования - *AnyLogic*[1]. Система поддерживает непрерывное моделирование, процессо-ориентированное и агентно-ориентированное. Кроме того, Anylogic располагает современным графическим интерфейсом и предоставляет пользователю набор стандартных библиотек, что упрощает процесс разработки модели. Имитационные модели, которые создаются средствами Anylogic, предназначены для решения достаточно широкого круга задач (на сайте www.xjtek.ru представлены примеры решения таких задач, как моделирование цепочек поставок, моделирование бизнес-процессов, моделирование поведения пассажиров в аэропорту, на вокзале и в торговом центре). Поведение агентов и их взаимодействие описывают с помощью графического языка. Если нет возможности описать модель средствами графического редактора, то пользователь может выполнить расширение модели с помощью языка Java. Интеграция компилятора Java в AnyLogic предоставляет более широкие возможности при создании моделей, а также создание Java-апплетов, которые могут быть открыты любым браузером. Среда моделирования AnyLogic поддерживает проектирование, разработку, документирование модели, выполнение компьютерных экспериментов с моделью, включая различные виды анализа — от анализа чувствительности до оптимизации параметров модели относительно некоторого критерия. Однако в AnyLogic нет средств создания интеллектуальных агентов, кроме того, настоящая версия не поддерживает распределенное моделирование (но, судя по публикациям, работы такие велись и ведутся [Борщев А., 2002]).

Авторами из Екатеринбурга [Аксенов К.А., 2013] разработана агентно-ориентированная система моделирования *BPSim*. Система имитации BPSim изначально была предназначена для моделирования бизнес-процессов. В программной системе BPSim агенты управляют объектами процесса преобразования ресурсов. Агенты анализируют текущую ситуацию, обращаются к базе знаний, вырабатывают решение, контролируют достижение целей, обмениваются сообщениями. Программные средства BPSIM позволяют разработать концептуальную модель, динамическую модель, провести имитационный эксперимент и экспортировать результаты в EXCEL. В системе реализованы (а) реактивные агенты (их описывают с помощью диаграмм деятельности (конечный автомат, (б) реактивно-интеллектуальные агенты (описывают с помощью продукционной базы знаний, для описания моделей ЛПР (лицо принимающее решение), управляющих процессами), (в) интеллектуальные агенты (поведение их описывают планирующей системой, знания хранятся в фреймовой базе знаний, используют для построения сложных советующих ЭС и т.д.), (г) гибридные (построение сложных систем планирования). Однако вряд ли можно говорить о кроссплатформенности этих программных средств, поскольку в BPSIM генерируется код на Delphi, а также о поддержке параллельного (распределенного) моделирования.

REPAST [Repast, 2014] REcursive Porous Agent Simulation Toolkit (Repast)-это открытый и свободно распространяемый источник библиотек для крупномасштабного агентного моделирования. Repast поддерживает разработку гибких моделей из агентов и используется в моделировании социальных процессов, в маркетинге, логистике. Пользователь строит свою модель, включая в свои программы компоненты из библиотеки Repast или используя визуальный Repast для среды Scripting. Существует три версии Repast, названных Repast for Python (Repast Py) (Python), Repast for Java (Repast J) и Repast for the Microsoft.NET framework (Repast .NET)). Так Repast J включает (а) параллельный дискретный планировщик по времени (календарь событий); (б) среду для визуализации модели; (в) средства интеграции с географическими информационными системами с целью моделирования агентов на реальных картах; (г) средства описания поведения агентов (с применением компонентов из библиотек нейронных сетей, генетических алгоритмов, например). Модели Repast могут разрабатываться различными способами, включая использование ReLogo (диалект языка мультиагентного моделирования Logo), блок-схем, языка Groovy (динамического языка виртуальной машины Java) или самого языка

Java. Также все указанные способы редактирования модели можно чередовать без потери качества в ходе разработки модели. Кроме того, среда моделирования располагает средствами для обработки результатов моделирования (например, MatLab, SQL, Excel). Однако для того, чтобы создать модель, описать поведение агентов, требуется специальная подготовка, надо быть специалистом-программистом.

NetLogo [NetLogo, 2014] – среда моделирования, предназначенная для создания моделей, которые используют для описания естественных и социальных явлений. NetLogo позволяет пользователям оперировать моделями “на лету”, динамически меняя поведение системы под действием различных условий. Система достаточно проста, что позволяет пользователям без квалификации программиста открывать и запускать уже готовые модели, или строить их самим. Но также система достаточно “продвинута”, чтобы удовлетворить запросы исследователей из многих областей знаний. Среда моделирования NetLogo является кроссплатформенной и поставляется с библиотекой программных компонентов, которая представляет собой большой набор заранее написанных моделей. Эти модели могут быть использованы вновь или модифицированы.

Следующая агентно-ориентированная система моделирования MASON [MASON, 2014] представлена в виде набора библиотек на Java (кроссплатформенность). MASON включает мощные программные средства визуализации (2D,3D). В то же время, пользователь может не подключать эти средства. Следует отметить, что система MASON устойчива к взломам. Однако MASON не поддерживает распределенное моделирование (но в настоящее время ведутся работы по созданию распределенной версии) и требует серьезных знаний языка Java, т.е. не является удобным средством моделирования для малоопытного пользователя.

Еще одна кроссплатформенная агентная система моделирования *Ascape* [Ascape, 2014], написана на Java и является свободно распространяемой. Довольно удобная система моделирования для широкого круга пользователей.

Swarm [Swarm, 2014](стая, рой) был первой средой разработки АМ приложений, впервые запущен в 1994г. Разработчики Swarm стремились создать распределенную платформу для моделирования АМ. Пользователь создает модели путем включения компонентов из библиотек Swarm в свои программы.

2. Требования к построению агентно-ориентированных систем имитационного моделирования

Итак, обзор позволяет сделать вывод о том, что в основном инструментальные средства агентного моделирования представляют собой библиотеки программных модулей, которые можно включить в программы пользователя. При разработке систем моделирования важно, чтобы модели могли разрабатывать не только пользователи с квалификацией программиста, но и обычные конечные пользователи[Власов С.А.,2013], [Замятина Е.Б., 2012]. Таким образом, агентно-ориентированные системы должны обладать визуальными программными средствами конструирования и редактирования моделей. Кроме того, для агентных систем моделирования характерны: (а) кроссплатформенность (чаще всего используют язык Java или C); (б) модульность, использование объектно-ориентированного подхода (агентов чаще всего представляют в виде объектов, которые обмениваются информацией друг с другом); (в) возможность динамического изменения моделей (изменение настроек, например) во время выполнения имитационного эксперимента. При разработке агентно-ориентированной системы моделирования авторы пытались учесть опыт перечисленных выше разработок и добиться выполнения требований, изложенных ниже:

(a) операции над моделью - при разработке агентной модели целесообразно предусмотреть возможность изменения набора агентов, совместное функционирование которых отображает протекающие в реальном мире процессы, возможность *изменения* связей между агентами [Замятина Е.Б., 2011];

(b) *иерархическое представление модели, возможность ее детализации* - имитационная модель должна быть иерархической и предоставлять возможность детализировать модель, заменяя конкретный агент (и его поведение) группой агентов или, наоборот, заменять группу агентов одним агентом, при этом новый агент должен реализовывать групповое поведение объединенных в группу агентов;

(c) *выделение структуры взаимодействия агентов* - имитационная модель отображает взаимосвязи агентов (структура моделируемой системы), которые обмениваются сообщениями. Целесообразно разработать программное обеспечение, которое в готовой имитационной модели выделяет ее структуру. Выделив структуру агентов, их взаимосвязи, можно провести дополнительный анализ имитационной модели, исследуя ее структурные характеристики (возможно, методами теории графов;

(d) *настройка на конкретную предметную область* - системе имитации приходится решать задачи, связанные с различными предметными областями, кроме того, с имитационной моделью можно работать в разных «терминах», так например, при моделировании компьютерной сети можно работать в «терминах» систем массового обслуживания, «в терминах» сетей Петри или теории графов. В этом случае целесообразно использовать онтологический подход, опыт применения онтологий при работе с

имитационными моделями приводится в [Mikov A., 2009], [Сухов А.О., 2013], [Замятина Е.Б., 2013-а].

(e) *оптимизация имитационного эксперимента по времени* – необходимость в этом возникает вследствие сложности задач, которые решаются методами имитационного моделирования. Имитационный эксперимент должен завершаться за приемлемое время. Решением в этом случае является использование нескольких вычислительных узлов (сети, кластера и т.д.), распределение агентов по вычислительным узлам и применение специализированных алгоритмов (оптимистического или консервативного, или их модификации) для сохранения каузальности событий и алгоритмов для сохранения равномерной загрузки вычислительных узлов [Zheng G. ,2005], [Миков А.И., 2010]

(f) *оптимизация имитационного эксперимента по надежности* – целесообразно разработать специальное программное обеспечение, применяемое в условиях распределенного имитационного эксперимента для перемещения агентов на «живые» узлы в случае выхода из строя каких-либо вычислительных узлов;

(g) *удаленный доступ* – предполагает возможность использования системы имитации удаленно и реализацию совместной работы пользователей, которые географически находятся на расстоянии друг от друга [Замятина Е.Б.,2012-а].

(i) *интеллектуальная обработка результатов моделирования* - целесообразно использовать для получения структурированной информации с применением методов Data Mining [Kolevatov G.A.,2012] .

(j) *интеллектуальные агенты* – необходимы для того, чтобы наиболее адекватно отображать моделируемые процессы в экономике, маркетинге, логистике очень важно реализовать интеллектуальных агентов. Большинство из рассмотренных выше систем предоставляет пользователю инструментальные средства для описания реактивного поведения, позволяют имитировать достаточно простое поведение. А необходимо сделать так, чтобы интеллектуальные агенты могли изменять свое поведение в зависимости от изменения обстановки (обучаться), преследовать цели, выбирать ту или иную стратегию в зависимости от роли, которую они выполняют.

3. Первый опыт создания агентной системы моделирования

Первая агентная система моделирования была разработана в 2007 году в виде библиотеки с набором основных классов (ядро), необходимых для реализации агентного моделирования [Замятина Е.Б., 2010].

Структура системы отображено на рис. 1 (диаграмме классов UML). Основой всей модели пользователя является класс модели (Model). Именно в нем описана основная семантика поведения всей моделируемой системы в целом. Также методы данного класса (их переопределение) позволяют обеспечить вывод текущих результатов моделирования, а также запускать, останавливать, изменять основные параметры модели. Остановимся кратко на некоторых важных методах.

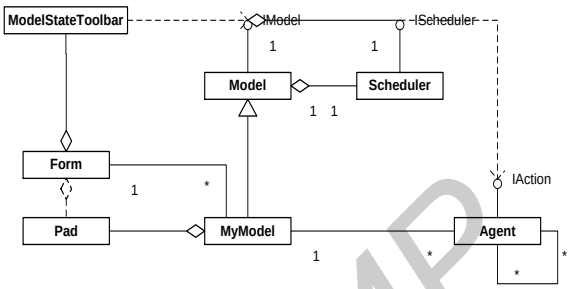

Рисунок 1. - Компоненты системы для построения агентных моделей

Метод инициализации отвечает за инициализацию всех основных внутренних рабочих параметров модели, его вызывают при первом запуске модели, а также после остановки модели. Модель выполняется по шагам (каждый шаг соответствует одной единице времени, то есть за один шаг в модели может произойти несколько событий, если они все были запланированы на одно и то же время). Планировщик (Scheduler) отвечает за продвижение времени в системе (реализуя соответствующий интерфейс). Его главной задачей является определение тех агентов, которые получают право хода, т.е. фактически возможность выполнять активные действия в модели, что собственно составляет суть всего процесса моделирования. Из очевидных преимуществ подхода с центральным планировщиком следует отметить его относительную простоту и эффективность реализации.

Графический интерфейс представлен несколькими реализованными компонентами: это планшет (Pad) для отображения различных графических объектов и привязки к ним агентов и собственно сами графические объекты, инспектор свойств для графических объектов и самой модели и панель инструментов для управления состоянием выполнения модели.

Планшет используется для отображения графических объектов с учетом их порядка по оси OZ. Поддерживаются следующие возможности: масштабирование, перемещение видимой области, перемещение графических объектов, их множественное и одиночное выделение, сжатие рабочей области планшета до реально занимаемой графическими объектами. В случае выхода графических объектов за границы рабочей области планшет автоматически увеличивается в размерах. Работает в многопоточном режиме.

Графические объекты (производные от Graphic) используются для графического представления агентов из модели, а также отображения различного рода связей между ними и обмена

информацией/ресурсами тоже между ними. Инспектор свойств используется для отображения свойств/параметров графических объектов (представляемых ими агентов) или модели. Поддерживается возможность обновления информации о выделенных объектов в динамике с продвижением имитационного времени модели и изменения состояния наблюдаемых агентов (выделенных графических объектах). Извлечение информации из модели для долгосрочного хранения и анализа результатов осуществлены с использованием инфраструктуры .NET.

Итак, первая агентная система моделирования, разработанная группой авторов, была *последовательной* и была основана на совместном функционировании *реактивных* агентов. Следует заметить, что опыт ее разработки был использован при создании системы моделирования «Рудопоток» [Чудинов Г.В., 2013].

Следующим шагом была разработка *распределенной* системы моделирования, основанной на совместном функционировании интеллектуальных агентов, основанных на *продукционных правилах*.

4. Реализация распределенной агентной системы имитации

Для реализации агентной платформы распределенного моделирования был выбран язык программирования Scala.

Язык Scala представляет собой кросс-парадигменный объектно-функциональный язык программирования, совмещающий в себе ООП и функциональное программирование и специализирующийся на создании легко масштабируемого компонентного программного обеспечения (Scala была создана в 2004г. под руководством Мартина Одерски в Университете EPFL (Lausanne, Switzerland)) [Odersky M., 2013].

В настоящее время доступна для платформ Java и .NET Framework. Среди ключевых особенностей языка можно отметить: единую объектную модель, наличие примесей (traits), технику сопоставления с образцом, лямбда-исчисление, виды (type views bounds), линеаризацию типов (type linearization), параметрический и функциональный полиморфизм, вариантность типов, кейс-классы (case classes), вывод типов, поддержка хвостовой рекурсии, наличие многочисленных инструментов по созданию новых языковых конструкций и обработку списков. Обобщённая структурная схема симулятора изображена на рисунок. 2.

Таким образом, архитектура симулятора является не многокомпонентной, а многослойной. Достигается это за счёт того, что к модулю, реализующему некоторую базовую функциональность, «примешиваются» другие компоненты, расширяющие структуру, поведение и семантику.

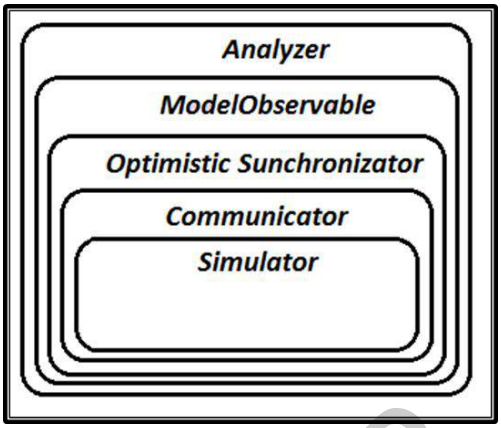

Рисунок. 2. - Многослойная архитектура симулятора

Трейт Simulator является базовым модулем, к которому подмешиваются другие элементы, уточняя его поведение. Фактически, Simulator, – это каркас для логического процесса. Трейт Communicator представляет собой модуль для работы с акторской системой. Он содержит в себе экземпляр актора, который занимается посылкой/приёмом сообщений. Трейт OptimisticSynchronizator является ключевым звеном для организации PDES – он реализует оптимистический алгоритм синхронизации Time Warp. Трейт ModelObservable является реализацией концепции информационных процедур, используемых для организации сбора статистики (информационные процедуры – программные компоненты, накладываемые на модель с целью сбора статистики в ходе имитационного прогона). Трейт Analyzer добавляет «интеллектуальность» алгоритмам синхронизации, используемых для реализации распределенного моделирования.

Особое внимание в разработанной системе имитации уделяется алгоритму синхронизации распределенной модели [Fujimoto R.M. ,2003].

В системе реализован оптимистический алгоритм синхронизации, основанный на знаниях об имитационной модели. Существует ряд работ, в которых используются знания об имитационной модели для сокращения времени выполнения распределенного имитационного эксперимента. Сокращение времени выполнения распределенного алгоритма авторы достигают за счет реализации эффективного алгоритма синхронизации логических процессов, управляемого продукционной экспертной системой, и за счет балансировки нагрузки на вычислительных узлах. Использование знаний об агентной имитационной модели, извлекаемые из онтологий, дали хорошие результаты при реализации алгоритмов синхронизации в агентной системе имитации. Эксперименты показали, что метрики, характеризующие скорость проведения распределенного эксперимента, управляемого модифицированным алгоритмом синхронизации (алгоритм KBASA, основанный на знаниях о модели) существенно сократились по сравнению с метриками классического оптимистического алгоритма Time Warp: количество откатов

сократилось с 71.2 до 3.8, количество отправленных антисообщений сократилось с 108.4 до 12.8.

5. Реализация интеллектуальных агентов

Известно, что создание интеллектуальных агентов является весьма сложной задачей, требующей теоретического фундамента для концептуального представления агентов. Таким фундаментом служат модели интеллектуальных агентов, по-разному описывающие знания, способы рассуждений, планирование поведения и непосредственные действия агентов.

Модели принято рассматривать в двух точек зрения: с точки зрения анализа свойств и поведения агентов в процессе функционирования системы в целом; с точки зрения изучения и конструирования свойств агента, определяющих его внутренние процессы (получение знаний, выработка целей, принятие решений и т.д.).

Выделено три вила архитектур: (а) делиберативные архитектуры и модели; (б) реактивные архитектуры и модели; (в) гибридные архитектуры и модели.

Реактивный подход позволяет использовать множество достаточно простых сценариев поведения агентов. Эти сценарии являются реакцией на появление того или иного события внешней среды. Недостатком является то, что достаточной полный ситуативный анализ всех возможных активностей агентов.

Делиберативные модели и архитектуры позволяют применять строгие формальные методы и хорошо отработанные технологии традиционного искусственного интеллекта, позволяющие относительно легко представлять знания в символьной форме и переносить их в агентно-ориентированные системы.

Гибридный тип архитектуры совмещает преимущества упомянутых выше архитектур. Таким образом, интеллектуальный агент обладает высокоуровневым выводом и низкоуровневыми реактивными способностями.

Итак, разработка агентной системы моделирования предполагала разработку инструментальных средств набора базовых классов для представления интеллектуальных агентов. Было принято решение представить архитектуру агентов в виде гибридной и делиберативной схем. В качестве средств логического вывода авторы используют продукционные системы и нейронные сети. Известно, что нейронные сети обладают свойством самообучаться, что актуально для реализации интеллектуальных агентов, которые должны приспосабливаться к изменению внешней среды и менять свое поведение, принимая то или иное решение. На первом этапе исследование было использовать три типа нейронных сетей: многослойный персептрон, сеть Хопфилда и сеть Хемминга. Для обучения нейронных сетей предложен генетический алгоритм.

Разрабатываемые инструментальные средства были протестированы. В качестве тестовых задач была выбрана задача поиска достопримечательностей в парке и «Искусственная жизнь».

Задача о поиске достопримечательностей имеет следующую формулировку: некая персона ищет достопримечательность в парке и пытается добыть информацию об ее месторасположении с помощью карты местности и информаторов. Получив информацию от информатора-человека, персона направится к остановке (трамвая), вместо того, чтобы исключительно пешком добираться до интересующей достопримечательности. Если в качестве источника информации используется карта, то персона добирается до цели пешком и т.д. Задача «Искусственная жизнь» достаточно хорошо известна.

Результаты тестовой задачи «Поиск достопримечательностей» приведены ниже в таблице.

Было проведено моделирование в среде размерами 400 на 600 точек, при условии наличия одной цели-достопримечательности, и всего было совершено 100 тестовых прогонов модели. В результате моделирования были получены результаты, показывающие возможность использования такого типа агентов для решения данной задачи (таблица 1).

Таблица 1. Результаты моделирования “поиск в парке” продукционными агентами

Кол-во агентов-информаторов	Среднее время поиска цели (в секундах)
1	29.6
3	27.9
5	19.5

Модель агента, основанная на нейронных сетях, была реализована в виде 2-слойного персептрона, имеющего 6 нейронов входного уровня и 2 нейрона выходного слоя.

В качестве входных данных на входной слой сети от рецепторов передавалась следующая информация: (а) факт наличия информатора в поле видимости – info; (б) расстояние до информатора (если в поле видимости их несколько, то учитывается расстояние до ближайшего) - dinf.;(в) расстояние до цели - df.; и т.д.

Был проведен имитационный эксперимент в среде размерами 400 на 600 точек, при условии наличия одной цели-достопримечательности, и всего было совершено 100 тестовых прогонов модели. В результате моделирования были получены результаты, показывающие возможность использования такого типа агентов для решения данной задачи (таблица 2).

Таблица 2. Результаты моделирования “поиск в парке” нейронными агентами

Количество агентов-информаторов	Среднее время поиска цели (в секундах)
1	37.6
3	33.5
5	25.1

Для настройки системы имитации можно воспользоваться либо программными средствами, которые предоставляют языковые инструментарии (или DSM-платформы), предназначенные для создания предметно-ориентированных языков (DSL, Domain Specific Language), либо программными средствами на основе онтологий. Рассмотрим первый способ.

Заключение

Авторы разработали прототип агентной системы имитации. Агентная система соответствует требованиям, изложенным в статье (кроссплатформенность, возможность использования нескольких вычислительных узлов для оптимизации имитационного эксперимента по времени, использование онтологий и т.д., участие в имитационном эксперименте интеллектуальных агентов, в том числе, и основанных на нейронных сетях).

Разработанный алгоритм синхронизации KBASA, основанный на знаниях о модели, позволяет проводить эффективные имитационные эксперименты (метрики, указывающие на скорость проведения имитационного эксперимента, значительно улучшились). Кроме того, получены хорошие результаты при тестировании интеллектуальных агентов, основанных на нейронных сетях.

Благодарности

Работа выполнена при поддержке РФФИ и № 13-07-96506_р-юг-а «Разработка деонтических основ мультиагентных технологий формирования баз знаний для виртуальных лабораторий

Библиографический список

[**Борщев А., 2013**] Борщев А.В. Как строить простые, красивые и полезные модели сложных систем. Сборник докладов шестой всероссийской научно-практической конференции «Имитационное моделирование. Теория и практика» (ИММОД-2013). Том 1. // ISBN 978-5-9690-0221-0 // Издательство «ФЭН» Академии наук РТ, Казань, 2013, с. 21-34.

[Borshchev A.,2002] Borshchev A., Karpov Y., Kharitonov V. Distributed Simulation of Hybrid Systems with AnyLogic and HLA. // Parallel computing technologies. – 2002. № 18(6). – P.829-839.

[Аксенов К.А., 2013] Аксенов К.А., Ван Кай, Аксенова О.П. Разработка и применение метода анализа узких мест на основе мультиагентного имитационного моделирования // Сборник докладов шестой всероссийской научно-практической конференции «Имитационное моделирование. Теория и практика» (ИММОД-2013). Том 2. // ISBN 978-5-9690-0221-0 // Издательство «ФЭН» Академии наук РТ, Казань, 2013, с. 19-23.

[Repast, 2014] Система моделирования “Repast” [Электронный ресурс] [Режим доступа: http://repast.sourceforge.net] [Проверено:25.12.2014]

[NetLogo, 2014] Система моделирования “NetLogo” [Электронный ресурс] [Проверено:25.12.2014]

[MASON, 2014] Система моделирования “MASON” [Электронный ресурс] [Режим доступа: http://cs.gmu.edu/~eclab/projects/mason] [Проверено:25.12.2014]

[Ascape, 2014] Система моделирования “Ascape” [Электронный ресурс] [Режим доступа: http://ascape.sourceforge.net/index.html#Contact] [Проверено:25.04.2014]

[SWARM, 2014] Система моделирования SWARM. [Электронный ресурс][режим доступа:www.swarm.org] [Проверено:25.04.2014]

[Власов С.А., 2013] Власов С.А., Девятков В.В., Назмеев М.М. Имитационная экспертиза: опыт применения и перспективы // Сборник докладов шестой всероссийской научно-практической конференции «Имитационное моделирование. Теория и практика» (ИММОД-2013). Том 1. // ISBN 978-5-9690-0221-0 // Издательство «ФЭН» Академии наук РТ, Казань, 2013, с. 54-63.

[Замятина Е.Б., 2012] Замятина Е.Б., Миков А.И. Программные средства системы имитации Triad.Net для обеспечения ее адаптируемости и открытости. Информатизация и связь. №5, 2012, АНО «Редакция журнала «Информатизация и связь», ISSN 2078-8320, С.130-133.

[Замятина Е.Б., 2011] Замятина Е.Б., Миков А.И., Михеев Р.А. Лингвистические и интеллектуальные инструментальные средства симулятора компьютерных сетей TRIADNS. International Journal “Information theories & Applications (IJ ITA). Vol 19, Number 4, 2012, pp.355-368. ITHEA, Sofia, 1000, P.O.B. 775, Bulgaria. ISSN 1310-0513 (printed)

[Mikov A., 2009] Mikov A., Zamyatina E., Kubrak E. An Ontology-based Approach to the Incomplete Simulation Model Analisis and its Automatic Completion. International Journal “Information Technologies & Knowledge”, 2009, Volume 3, Number 2, pp. 169-186.

[Сухов А.О.,2013] Сухов А.О. Трансформация визуальных моделей в системе MetaLanguage / Современные проблемы математики и ее прикладные аспекты – 2013. Сборник тезисов конференции. – Пермь: Пермский государственный национальный исследовательский университет, 2013. – С. 44.

[Замятина Е.Б.,2013-а] Замятина Е.Б, Лядова Л.Н., Сухов А.О.. Мультиязыковое моделирование с использованием DSM платформы MetaLanguage. Информатизация и связь. №5, 2013, АНО «Редакция журнала «Информатизация и связь», ISSN 2078-8320, С.11-15.

[Чудинов Г.В., 2013] Чудинов Г.В. Архитектура и разработка инструментального средства с ориентацией на предметную область транспортировки руды в шахтах – ПВК «Рудопоток». Информатизация и связь. №5, 2013

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

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

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

[Замятина Е.Б.,2012-а] Замятина Е.Б., Миков А.И. Применение онтологий и принципов организации сервис-ориентированно архитектуры при проектировании и реализации системы имитационного моделирования. Материалы 3-ьей Международной научно-технической конференции «Технологии разработки информационных систем ТРИС-2012», Т.1., Таганрог, издательство Технологического института ЮФУ, Ростов –на-Дону, 2012, 9 сентября, стр. 61-65.

[Kolevatov G.A.,2012] Kolevatov G.A., Zamyatina E.B. Simulation Analysis Framework Based on Triad.Net. Proceedings of the 6-th Spring/Summer Young Reseachers' Colloquium on Software Engineering. SYRCoSE 2012, Perm, May 30-31, 2012-Perm, Russia,pp.160-163.

[Odersky M., 2013] Odersky M. The Scala Programming Language [Электронный ресурс]. URL: http://www.scala-lang.org/node/25 (дата обращения: 06.06.2013)

[Замятина Е.Б., 2010] Замятина Е.Б., Чудинов Г.В. Разработка и использование программных средств для построения и исследования агентных имитационных моделей. / Е.Б. Замятина, Г.В. Чудинов // Вестник пермского университета. Математика, механика, информатика, 2010, №2(2), С. 80 – 84.

THE EXPIERANCE OF AGENT-BASED SIMULATION SYSTEM IMPLEMENTATION

Zamyatina E.B.[*], Karimov D.F. [*], Mitrakov A.A.[*]

**Perm State National Researching University, Perm, Russian Federation*

e_zamyatina@mail.ru,

googlicus@gmail.com,

mitrakov-artem@yandex.ru

This paper discusses the problem of agent-based simulation system implementation. Authors consider more precisely the design and impementation of distributed simulation system (the design of special knowledge-based synchronization algorithm) and the design of agents based in neural networks

INTRODUCTION

Introduction considers the actuality of the problem. Indeed the agent-based simulation system are wide spread now, but some features and characteristics of these simulation system do not permit to use them for the solving of some problems. One of the problems is an optimization of simulation experiment in the respect to time. Next problem – the design and implementation of the intellectual agents.

MAIN PART

Main part includes the overview of agent-based simulation systems. The overview shows that it is necessary to make effective and flexible simulation system in order to solve complicate problems. The effectiveness of simulation system may be achieved by the using of special algorithm based on knowledge about simulation model. The flexibility of simulation system may be achieved by using of intellectual agents. Intellectual agents may change their behavior which depends from the characteristics of an environment (it is well known that agent is a program entity which communicate with the environment and other agents). Authors suggest to use agents based on neural network

CONCLUSION

The results of investigations and new directions in investigations are presented. The investigations shows that solutions of authors allow to design more flexible and efficient simulation systems.