УДК: 004.9+519.6

АГЕНТНОЕ МОДЕЛИРОВАНИЕ НА БАЗЕ ЗАЯВОЧНО-СТЕКОВОЙ АРХИТЕКТУРЫ ВЗАИМОДЕЙСТВИЯ

© 2013 А.А. Стеряков

Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет)

Поступила в редакцию 02.12.2013

Рассматривается моделирование агентных систем с использованием заявочно-стековой архитектуры взаимодействия. Предложена структура агентных моделей, обеспечивающая удобную их реализацию в виде компьютерных программ на языках объектно-ориентированного программирования. Описан метод взаимодействия элементов системы на базе заявочно-стековой архитектуры, позволяющий в ряде случаев повысить эффективность выполнения программ.
Ключевые слова: агентное моделирование, заявочно-стековая архитектура, мультиагентные системы, объектно-ориентированный подход, сложные системы.

ВВЕДЕНИЕ

Агентное моделирование (АМ) сегодня является одним из наиболее актуальных типов имитационного моделирования [1-3]. Вместе с тем многие авторы отмечают недостаток эффективных и доступных для понимания и практического использования архитектур агентных моделей [4, 5]. Одной из наиболее актуальных задач в сфере агентного моделирования на сегодняшний день является разработка формализованных и гибких методов построения многоагентных моделей, применимых для систем из различных областей приложения агентного моделирования. Моделирование агентного типа предполагает выполнение двух этапов: 1) описание всех элементов системы и 2) задание способа их взаимодействия. Для этого существуют различные способы [1], и, подходя к решению конкретной задачи, необходимо либо использовать существующий программный пакет (платформу) для проведения АМ (см. [1] и ссылки там же), либо приступить к разработке собственного программного продукта. Существующие платформы, как правило, достаточно трудны для овладения ими (Repast, Netlogo [2]). Кроме того, изучив новый синтаксис и методы разработки, зачастую можно столкнуться с ситуацией, когда инструмент не подходит для конкретной модели или не предоставляет достаточной гибкости для проведения исследования. Существуют также сложности, связанные с невозможностью представления моделей с необходимой точностью (см. [3] и ссылки там же). Другие разработки (например, AnyLogic) являются дорогостоящими коммерческими программными продуктами, которые ориентированы на использование на предприятиях и не всегда применимы в исследовательских целях.

Необходимо также отметить, что одна из центральных проблем современной теории моделирования сложных организационно-технических систем в целом состоит в обеспечение требуемой степени адекватности (в широком смысле), точности, достоверности и корректности рассматриваемого класса моделей по отношению к моделируемым объектам-оригиналам. Вопросам создания теоретических основ имитационного и комплексного моделирования, в рамках которых проблема адекватности была бы формально и фундаментально исследована, в последнее время уделяется большое внимание. Однако, к сожалению, общий объем проводимых теоретических и практических работ и полученных конструктивных результатов по данному направлению исследований еще недостаточен [5]. Существующие ограничения инструментов компьютерного моделирования вынуждают комбинировать их с аналитическими математическими моделями, а также с логико-алгебраическими, логико-лингвистическими моделями с использованием технологий комплексного моделирования [4]. Обсуждается также вопрос о способах применения различных баз эмпирических данных реальных сложных систем для эффективного построения соответствующих агентных моделей ([3, 5] и ссылки там же).

В настоящее время актуальна задача универсализации методов и средств разработки агентных моделей [3, 5]. Существует необходимость

Стеряков Александр Александрович, аспирант кафедры физики. E-mail: steryakov@mail.ru

общего методологического подхода в моделировании такого типа, предоставляющего исследователю некоторый общий алгоритм действий, описывающий весь цикл АМ – от формулировки задачи и её математической формализации до реализации комплекса программ для проведения численных экспериментов. В случае выполнения этого значимого условия развитие данной ветви моделирования ускорится и выйдет на уровень традиционных методов. Особенно важно предложить средства моделирования, которые будут инвариантными относительно сфер применения и сложности описываемых систем.

В данной работе предлагается достаточно универсальный метод построения моделей для мультиагентных систем. В первой части статьи описана структура агентных моделей, позволяющая унифицировать описание систем из различных предметных областей. Вторая часть посвящена формальному описанию предлагаемого заявочно-стекового взаимодействия агентов. В заключении излагаются основные выводы и результаты работы.

СТРУКТУРА АГЕНТНЫХ МОДЕЛЕЙ

Существование ныне хорошо развитой концепции объектно-ориентированного программирования (ООП) [6-8], во-первых, позволяет использовать языки ООП [8, 9] для компьютерной реализации агентного моделирования, а, во-вторых, наталкивает на мысль использовать некоторые принципы этой концепции для построения самих моделей [6]. Например, Бертран Мейер в своей статье [6] выделяет пять принципов объектно-ориентированной парадигмы. На данный момент в русскоязычной литературе нет соответствующих устоявшихся терминов, поэтому приведены оригинальные термины с частичным наиболее близким переводом:

1. Seamlessness (Согласованность). Одним из принципов является согласованность между этапами цикла разработки программ: анализом, проектированием, реализацией и сопровождением. Суть идеи согласованности состоит в необходимости существования явного структурного соответствия между моделями на различных этапах разработки (анализа, проектирования и реализации). Такой подход к разработке минимизирует сложность перехода между последовательными её этапами.

2. Decentralization (Децентрализация). Данным принципом констатируется, что объектная методология предполагает высокоуровневую структуру программных систем, представляющую собой сеть взаимодействующих агентов-объектов без какого-либо единого центра. Такая модульность системы предотвращает непредвиденные последствия при изменениях в одном модуле, а также способствует возможности повторного использования отдельных модулей.

3. Contracts (Гарантированность результата, контракты). Данный принцип определяет отношения между объектами и предписывает необходимость точного определения того, что одна сторона ожидает, и того, что другая сторона гарантирует, при взаимодействии. Лишь наличие точных «контрактов» гарантирует надёжность архитектуры системы.

4. Selfishness (Самодостаточность). Суть данного принципа схожа с более распространённым понятием инкапсуляции, но более полно и точно раскрывает цель объектной методологии. На самом деле регламентируется обязательным не сокрытие данных объекта, а то, что другой объект при обращении не обязан «знать», что скрывается внутри реализации того или иного метода, а должен лишь получить оговоренный заранее (см. Contracts) результат. Благодаря этому решаются проблемы с повторным использованием модулей, с безопасностью внесения изменений и другие.

5. Classification (Классификация). Важным принципом в объектной методологии является представление моделируемой системы в виде упорядоченной структуры объектов. Одним из способов реализации этого принципа является классификация элементов предметной области, в результате которой возникает иерархия объектов, связанных наследованием.

В целом рассмотренные принципы создают концептуальную основу для решения задач, связанных со сложностью программных систем. Кроме того, важен тот факт, что объектно-ориентированная парадигма – не столько инструментарий в программировании, сколько фундаментальный базис для анализа и проектирования сложных систем произвольной природы [6]. Используя данный подход, в работе предлагается концепция создания агентных моделей на базе объектно-ориентированного подхода. Условимся далее в тексте под объектно-ориентированными моделями (ООМ) понимать модели, относящиеся к классу мультиагентных имитационных моделей и построенные по принципам, которые естественным образом обеспечивают их дальнейшую реализацию в виде комплекса программ на языках объектно-ориентированного программирования.

Реализацию на языках ООП рассматриваемым моделям будет обеспечивать следование основным принципам объектно-ориентированного подхода ещё на этапе формирования моделей, описывающих реальные системы. Такая методика позволит максимально сблизить вид математически формализованной модели и её компью-

терную реализацию. Ниже будут сформулированы правила, по которым построенную таким образом ООМ можно будет реализовать программно на языке ООП, что позволит формировать некоторого рода «техническое задание» специалистам в области языков программирования, не знакомых с конкретными областями приложения. В описанной методике явным образом будет отражаться идея о согласованности этапов проектирования и реализации, описанная в предыдущем разделе.

Моделирование предполагает описание всех элементов системы в виде объектов, наделённых вектором состояния элемента и функциями, описывающими реакцию элемента на текущее состояние системы, то есть свойствами и поведением в терминах ООП. Предлагается следующий метод построения таких моделей, основные этапы которого, связанные с анализом элементов системы, могут быть сформулированы следующим образом:

1. Описание всех элементов системы в виде уникальных объектов различных типов с присущими им характеристиками. Все элементы системы, следуя принципу инкапсуляции, представляются в виде объектов, наделённых свойствами и поведением (то есть вектором состояния элемента и функциями, описывающими реакцию элемента на текущее состояние системы).

2. Классификация элементов системы с построением их иерархии. Все объекты, составляющие систему, классифицируются таким образом, чтобы они организовали естественную иерархию. Формально данная процедура подразумевает выделение общих частей векторов состояния или функций у объектов различных типов. В случае их наличия создаётся более общий тип объектов, а исходные типы объектов становятся его подтипами. Такой подход, основанный на иерархичности систем, позволит сократить работу по описанию различных подтипов, которое по принципу наследования базируется на описании родительских типов.

3. Выделение функций, отвечающих за автономное изменение состояния элемента.

Поведение элемента системы, не связанное с окружающей средой, и все реакции, результат которых не зависит от других элементов системы, представляется в виде функций, преобразующих вектор состояния объекта.

4. Выделение функций, связанных с взаимодействием элемента с окружающей средой, – поведенческих функций. Все изменения элемента, связанные с другими элементами системы (их реакцией), делегируются специальной системе взаимодействия, которой предоставляется достаточный набор данных о данном элементе в виде запроса – вектора, формируемого поведенческой функцией.

5. Определение необходимой информации, достаточной для описания внешних взаимодействий агентов, то есть формирование минимального интерфейса, позволяющего использовать единообразный способ коммуникации при взаимодействии агентов. Следуя принципу полиморфизма, формируется единый интерфейс у объектов для обеспечения работы системы взаимодействия, которая по запросу к любому объекту должна получать единообразный набор данных, используемых для удовлетворения (исполнения или отклонения) всех возможных действий элементов системы с последующим соответствующим изменением их состояния.

Предлагается определённая архитектура ООМ, обладающая достаточной универсальностью, что иллюстрируют примеры конкретных реализаций [10, 11]. Архитектура включает в себя три основных понятия: агент – заявка – стековое взаимодействие.

Агент. В системе, макросостояние которой характеризуется вектором $\bar{s}(t)$, для каждого отдельного типа элементов строится модель i-го агента, характеризующаяся, как было сказано выше, вектором состояния $\bar{v}^i(t)=\left(v_1^{\,i}(t),v_2^{\,i}(t),...,v_n^{\,i}(t)\right)$ и функциями двух назначений:

1. Первая (в дальнейшем – функция преобразования состояния агента) отвечает за независимое от других элементов системы изменение состояния агента, то есть вектора состояния (или его части), на каждом временном шаге. Данная функция определяет автономность агента и имеет вил $\bar{v}^i(t+\Delta t)=g^i\left(v_1^{\,i}(t),v_2^{\,i}(t),...,v_k^{\,i}(t),\bar{s}(t)\right),\ k\le n$.

2. Вторая (в дальнейшем – поведенческая фукнция агента) формирует вектор параметров $\bar{b}^{\,i}=\left(b_1^{\,i},b_2^{\,i},...,b_m^{\,i}\right)$, который будем называть заявкой и который используется для осуществления взаимодействия агентов в системе. Таким образом, поведенческая функция определяет реакцию элемента системы на внешнюю среду и имеет вид $\bar{b}^{\,i}=\varphi^i\left(\bar{v}^i(t),\bar{s}(t)\right)$.

Необходимо отметить, что предлагаемая архитектура не накладывает никаких ограничений на вид поведенческой функции и функции преобразования. В данном случае функция понимается в широком смысле, то есть может быть задана аналитически или представлена любым алгоритмом преобразования входных данных в выходные, например, нейросетью. Это особенно важно при реализации когнитивных агентов, способных к адаптации. В общем случае функции преобразования состояния у каждого типа

агента могут быть различными, но поведенческая функция всех агентов должна выдавать вектор одинаковой размерности, выполняя тем самым условие идентичности вида заявки всех агентов системы.

Одинаковые части вектора состояния у различных типов агентов могут быть вынесены в модель более общего типа объекта. Аналогично следует поступать с функциями преобразования состояния и с поведенческими функциями, разделяя преобразования, присущие отдельным подтипам и более общие, свойственные объединяющему типу. Подобное разделение необходимо проводить ещё на этапе формулировки модели, чтобы гарантировать возможность оного при компьютерной реализации, что упростит описание всей иерархии типов моделируемой системы.

Заявка, то есть вектор параметров $\overline{b}^i$, определяемый поведенческой функцией, задаёт протокол информационного обмена между агентом и системой взаимодействия. Таким образом, заявка служит средством внутренней коммуникации в системе, тем самым обеспечивая функцию передачи сообщений.

Заявка должна содержать всю необходимую информацию для того, чтобы могло быть осуществлено взаимодействие.

Стековое взаимодействие. Взаимодействие агентов происходит итеративно и реализуется с помощью стека, под которым подразумевается упорядоченный набор заявок, то есть множество, на котором задано отношение частичного порядка: $L = \{\overline{b}^1, \overline{b}^2, ..., \overline{b}^N\}$, где N – число агентов в системе. В общем случае взаимодействие осуществляется в три этапа: 1) сбор заявок от каждого агента в стек; 2) обработка стека (сортировка, удаление и изменение заявок); 3) осуществление взаимодействия – изменение векторов микро- и макро-состояний системы.

Описанная архитектура представлена в виде структурной схемы на рис. 1.

Предложенная модель полностью формализует два этапа создания агентной модели: 1) описание всех элементов системы и 2) задание способа их взаимодействия. Опишем, каким образом реализуются основные свойства агента [2, 5] в данной модели:

• *Автономность*. Внутренняя динамика агента любой сложности может быть определена посредством функции преобразования состояния агента, на которую в модели не накладывается никаких ограничений.

• *Способность к коммуникации*. Исходя из определения, что коммуникация – это определённый тип действия агента, которое влияет на внутреннюю динамику других агентов, замена прямых сообщений «агент-агент» на взаимодействие посредством заявок не ограничивает сложность коммуникации. Предлагаемый путь осуществления коммуникации через «посредника», то есть отдельную систему взаимодействия, в отдельных случаях может замедлить процедуру влияния одного конкретного агента на другого, но при этом повысить эффективность коммуникации в масштабах всей системы.

• *Реакционная способность* агента в модели частично или полностью делегируется системе взаимодействия, так как лишь посредством последней агент может реагировать на изменение внешней среды или действия других агентов.

• *Активность и мобильность*. Активность, как способность агента изменять своё состояние только вследствие внутренней динамики, и мобиль-

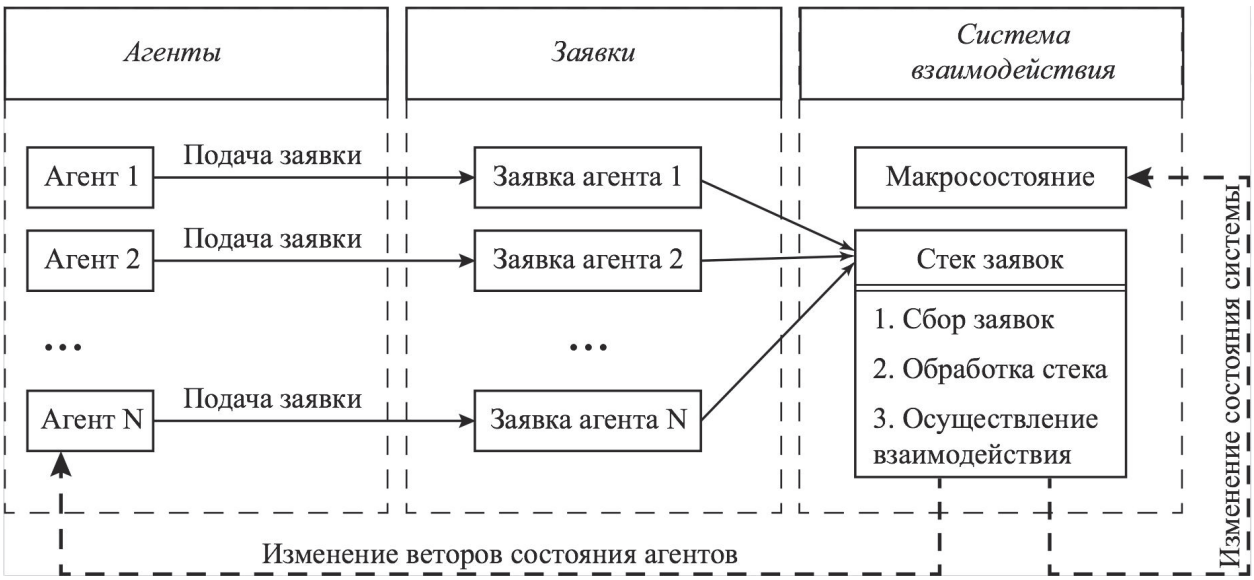

Рис. 1. Архитектура объектно-ориентированных моделей

ность, как один из возможных типов активности, определяется в модели как посредством функции преобразования, то есть вне зависимости от внешней среды, так и с помощью поведенческой функции и в дальнейшем системы взаимодействия, так как, например, возможность передвижения в заданном пространстве может определяться внешними факторами, влияние которых контролируется в модели именно системой взаимодействия.

ЧИСЛЕННЫЙ МЕТОД ЗАЯВОЧНО-СТЕКОВОГО ВЗАИМОДЕЙСТВИЯ АГЕНТОВ

Часто взаимодействие агентов реализуется непосредственно путём передачи сообщений [1, 3, 5], что можно считать наиболее реалистичным способом с точки зрения схожести с процессами, происходящими в жизни. Однако не всегда такой способ будет эффективным, если рассматривать вопросы компьютерной реализации. Предлагаемая архитектура заявочно-стекового взаимодействия увеличивает степень косвенности коммуникации между агентами. Система взаимодействия выполняет роль посредника между «общающимися» агентами, тем самым уменьшая связность модели, то есть зависимость между отдельными её компонентами. Последнее, как известно из теории ООП [12], улучшает архитектуру компьютерной реализации, существенно облегчая задачу модификации всей программы в случае необходимости.

Кроме того, используя опосредованность во взаимодействии агентов, часто удаётся существенно повысить эффективность алгоритмов осуществления последнего. Стоит отметить следующие возможные преимущества введения посредника при реализации взаимодействия между агентами:

1. Предварительный сбор заявок и их сортировка в стеке или в системе стеков уменьшает количество операций взаимодействия вследствие полного или частичного отбора необходимых заявок ещё на этапе сбора.

2. По сравнению с последовательным перебором пар взаимодействующих агентов при подаче заявок от агентов с последующей их обработкой можно обеспечить более точное описание реальных систем при существовании в них сложных правил взаимодействия.

3. В случае наличия сложных протоколов коммуникации между агентами, при которых используется большое количество сообщений, последовательно передаваемых агентами друг другу, введение посредника также приводит к более эффективной реализации. Для этого необходимо внести все необходимые данные от агентов в заявки и выполнить операцию взаимодействие в одном методе при обработке соответствующей пары заявок.

Применение стекового взаимодействия при реализации конкретных агентных систем [10, 11] позволило эффективно организовать перебор агентов. В случае биологической системы использовалась система стеков, в которые уже на этапе приёма распределялись заявки от агентов в зависимости от локализации в пространстве последних [11]. Такое разделение существенно сократило объёмы перебираемых пар агентов при осуществлении взаимодействия. При реализации агентной модели финансовой модели стек представлял собой два набора заявок на продажу и на покупку акций, упорядоченных по возрастанию и убыванию цены соответственно [10], что позволило осуществить алгоритмизированный обход дерева условий для совершения операций торга. Последнее существенно усложнилось бы при отсутствии заявочно-стекового взаимодействия.

Опишем более формально численный метод для стекового взаимодействия в ООМ. Итак, взаимодействие агентов в системе предполагается итеративным и реализуется с помощью стека (или системы стеков), где стек – это частично упорядоченное множество заявок:

$$L = \left\{\overline{b}^1, \overline{b}^2, ..., \overline{b}^N\right\}.$$

Отношение частичного порядка на данном множестве задаётся в каждой конкретной модели, исходя из типа решаемой задачи. Каждая итерация взаимодействия осуществляется в три этапа:

1) сбор заявок от каждого агента в стек:

$L = \left\{\overline{b}^1, \overline{b}^2, ..., \overline{b}^N\right\}$, где N – число агентов в системе.

Каждая заявка $\overline{b}^i = \phi^i\left(\overline{v}^i(t), \overline{s}(t)\right)$ является результатом поведенческой функции i-го агента.

2) обработка стека: $\hat{L}_{R_\leq} = G(L)$.

На данном этапе происходит преобразование множества заявок. Во-первых, на множестве задаётся отношение частичного порядка $R_\leq$, что соответствует сортировке заявок и/или их распределению по системе стеков при компьютерной реализации. Конкретное отношение $R_\leq$ определяется при моделировании в определённой предметной области. Во-вторых, множество заявок может быть преобразовано путём уменьшения количества заявок или изменением самих заявок.

Сортировка заявок, а также уменьшение их количества облегчает их обработку при осуществлении взаимодействия и повышает эффективность выполнения компьютерной программы, что иллюстрируют примеры конкретных реали-

заций [10, 11]. Изменение заявок может быть использовано для осуществления дополнительной координации без дополнительных операций с векторами состояния агентов.

3) взаимодействие – изменение векторов микро- и макро- состояний системы:

Взаимодействие осуществляется путём удовлетворения заявок. При рассмотрении каждой заявки производится анализ представленной в ней информации, исходя из правил, определённых предметной областью моделируемой системы. На основании результатов этого анализа происходит поиск других необходимых заявок для взаимного удовлетворения. Перебор стека может проходить в несколько итераций до выполнения условий удовлетворения всех возможных заявок. При каждой итерации происходит изменение векторов микро- и макро- состояний системы четырьмя возможными способами:

- $\overline{v}^i\left(t+\Delta t\right)=\overline{f}^i\left(\left\{\overline{v}\left(t\right)\right\},\left\{\overline{b}\right\},\overline{s}\left(t\right)\right).$

Изменяются только вектора состояния агентов.

- $\overline{s}\left(t+\Delta t\right)=\overline{F}\left(\left\{\overline{v}\left(t\right)\right\},\left\{\overline{b}\right\},\overline{s}\left(t\right)\right).$

Изменяется только вектор макросостояния системы.

- $\overline{v}^i\left(t+\Delta t\right)=\overline{f}^i\left(\left\{\overline{v}\left(t\right)\right\},\left\{\overline{b}\right\},\overline{s}\left(t\right)\right),$

$$\overline{s}\left(t+\Delta t\right)=\overline{F}\left(\left\{\overline{v}\left(t\right)\right\},\left\{\overline{v}\left(t+\Delta t\right)\right\},\overline{s}\left(t\right)\right);$$

Изменяются вектора состояния агентов, что приводит к изменению вектора макросостояния системы.

- $\overline{s}\left(t+\Delta t\right)=\overline{F}\left(\left\{\overline{v}\left(t\right)\right\},\left\{\overline{b}\right\},\overline{s}\left(t\right)\right),$

$$\overline{v}^i\left(t+\Delta t\right)=\overline{f}^i\left(\left\{\overline{v}\left(t\right)\right\},\overline{s}\left(t\right),\overline{s}\left(t+\Delta t\right)\right).$$

Изменяется вектор макросостояния системы, что влечёт изменение векторов состояния агентов.

На этом же этапе осуществляются мета-операции: удаление агентов из системы или добавление новых. Формально они означают изменение вектора макросостояния системы, а именно его части, связанной с количеством агентов в системе.

Рассмотрим достаточность предлагаемого метода взаимодействия относительно реализации языка коммуникации агентов, различных протоколов коммуникации и механизмов координации агентов:

• Язык коммуникации в предлагаемой модели определяется на этапе формирования интерфейса у объектов, то есть определением набора данных, которые будут передаваться в заявке системе взаимодействия. Таким образом, заявка является способом отправки сообщения. Получение сообщения в модели заменяет непосредственное изменение состояния агента системой взаимодействия после удовлетворения всех возможных заявок.

• С помощью подобного языка коммуникации можно реализовать практически все стандартные виды протоколов коммуникации между агентами (протокол типа “объявление”, “вопрос” и “передачи задачи”). Технически протоколы будут сильно отличаться в реализации, так как сообщения передаются не от агента к агенту, а обрабатываются системой взаимодействия, однако по сути выполняемых задач останутся прежними. Например, протокол типа “объявление” будет регулировать не рассылку сообщений всем агентам, а лишь определять, что передаваемые данные могут быть использованы системой взаимодействия по отношению ко всем предусматриваемым агентам.

• Механизм координации в модели может быть как централизованным, правила которого будут заложены в систему взаимодействия, так и распределённым. В последнем случае, связанном, как правило, с моделированием сложных по структуре когнитивных агентов, координационные механизмы будут задаваться внутренними функциями агента (функцией преобразования состояния и поведенческой функцией) и взаимодействием между агентами, осуществляемом посредством заявочно-стековой системы.

ЗАКЛЮЧЕНИЕ

В настоящей работе рассмотрено моделирование агентных систем с использованием заявочно-стековой архитектуры взаимодействия. Предложен метод создания объектно-ориентированных агентных моделей, описана структура таких моделей, обеспечивающая удобную их реализацию в виде компьютерных программ на языках объектно-ориентированного программирования. Предложенный в работе метод создания агентных моделей на базе объектно-ориентированного подхода частично решает проблемы АМ, обозначенные во введении. Следование основным принципам объектно-ориентированного подхода ещё на этапе формирования моделей, описывающих реальные системы, может существенно ускорить процесс реализации моделей на языках ООП. Кроме того, предложенная методика позволяет максимально сблизить вид математически формализованной модели и её компьютерную реализацию и тем самым облегчает задачу постановки «технического задания» специалистам в области языков программирования, не знакомых с конкретными областями, для которых разрабатывается модель.

В работе предложен численный метод для осуществления взаимодействия между агентами в системе, основанный на заявочно-стековой архитектуре, которая увеличивает степень косвенности в коммуникации между агентами. Система взаимодействия выполняет роль посредника между агентами, тем самым уменьшая связность модели, то есть зависимость между отдельными её компонентами. Последнее улучшает архитектуру компьютерной реализации, существенно облегчая задачу модификации всей программы. Предложенный метод опосредованного взаимодействия агентов во многих случаях повышает эффективность алгоритмов его реализации. Более конкретно, метод обеспечивает более точное описание реальных систем при существовании в них сложных правил взаимодействия; использование посредника увеличивает эффективность при наличии сложных протоколов коммуникации между агентами; введение заявочно-стекового механизма уменьшает количество перебираемых пар взаимодействующих агентов при предварительном отборе только необходимых заявок.

Предложенные в работе подход и численный метод взаимодействия являются инвариантными относительно сфер применения и сложности описываемых систем и позволяют получать адекватное описание исследуемых сложных систем, формулируя исходную модель с учётом природы тех объектов, которые представляют интерес изучения. Это подтверждается успешным применением предложенной методики для создания двух объектно-ориентированных моделей из конкретных областей: имитационной модели пространственно-временного взаимодействия конкурирующих биологических популяций с автономными неоднородными участниками [10] и имитационной модели финансового рынка одного типа акций с автономными неоднородными участниками [11].

СПИСОК ЛИТЕРАТУРЫ

1. *Nikolai Cynthia, Madey Gregory.* Tools of the trade: A survey of various agent based modeling platforms // Journal of Artificial Societies and Social Simulation. 2009. T. 12, No 2. p. 2.
2. *Macal Charles M., North Michael J.* Tutorial on agent-based modeling and simulation // Proceedings of the 37th conference on Winter simulation. WSC '05. Winter Simulation Conference, 2005. С. 2–15
3. *Швецов А.Н.* Агентно-ориентированные системы: от формальных моделей к промышленным приложениям // Информационно-телекоммуникационные системы: Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению. 2008. 101 с.
4. *Аврамчук Е.Ф., Вавилов А.А., Емельянов С.В.* [и др.] Технология системного моделирования. М.: Машиностроение; Берлин: Техник, 1988. 520 с.
5. *Городецкий В.И.* Самоорганизация и многоагентные системы. I. Модели многоагентной самоорганизации // Известия РАН. Теория и системы управления. 2012. № 2. С. 92–120.
6. *Meyer Bertrand.* The Conceptual Perspective // Computer (IEEE). 1996. T. 29, No 1. P. 86–88.
7. *Буч Гради.* Объектно-ориентированный анализ и проектирование. М.: Бином, 2000.
8. *Floyd Robert W.* The paradigms of programming // Communications of the ACM. 1979. T. 22, No 8. P. 455–460.
9. *Pierce Benjamin C.* Types and programming languages. The MIT Press, 2002.
10. *Стеряков А.А.* Имитационное моделирование фондовых рынков // Вестник Самарского государственного аэрокосмического университета. 2012. № 2. С. 274–281.
11. *Стеряков А.А.* Динамика сложных биологических систем в моделях агентного типа на примере взаимодействия двух конкурирующих популяций // 20-я международная конференция серии «Математика. Компьютер. Образование». Биофизика сложных систем. Анализ и моделирование. Тезисы. г. Пущино. Москва-Ижевск, 2013. С. 87.
12. *Гамма Э. и др.* Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2001. 368 с.

AGENT-BASED MODELING WITH STACK OF BIDS INTERACTION ARCHITECTURE

© 2013 A.A. Steryakov

Samara State Aerospace University named after Academician S.P. Korolyov
(National Research University)

In this paper we consider a "stack of bids" interaction architecture for agent-based models (ABMs). We propose a particular ABM structure, which provides a natural way to implement the ABM model as a computer program in object oriented language. We describe in details the interaction method on the basis of "stack of bids" architecture, which allows to increase efficiency of the program in certain cases.

Key words: agent-based modeling, stack of bids architecture, multy-agent systems, object-oriented method, complex systems.

Alexander Steryakov, Graduate Student at the Physics Department. E-mail: steryakov@mail.ru