УДК 004.89
DOI: 10.15827/2311-6749.17.3.4

АРХИТЕКТУРА СРЕДЫ МОДЕЛИРОВАНИЯ ДЛЯ ПРОВЕДЕНИЯ ЭКСПЕРИМЕНТОВ С ИНТЕЛЛЕКТУАЛЬНЫМИ АГЕНТАМИ

С.А. Беляев, к.т.н., доцент, bserge@bk.ru;
Ю.С. Черепкова, студентка, yuliyacherepkova@gmail.com
(Санкт-Петербургский государственный электротехнический университет «ЛЭТИ», ул. Профессора Попова, 5, г. Санкт-Петербург, 197376, Россия)

Статья посвящена проектированию и реализации архитектуры среды моделирования для проведения экспериментов с интеллектуальными агентами. Кратко представлен теоретический материал для введения в предметную область. Рассмотрена актуальность проблемы. Показано, что создание среды моделирования позволяет решить задачи разного уровня сложности во многих предметных областях. Представлены существующие решения и их ограничения. Предложен вариант архитектуры с описанием ее функционирования посредством диаграммы деятельности, принципов и особенностей реализации. Продемонстрированы преимущества использования предложенной архитектуры. Описана математическая модель эксперимента, проведенного на базе предлагаемого решения. Проиллюстрирован вариант реализации на примере игры с использованием таких технологий, как Java, JavaScript, WebSocket, JavaReflection, UDP, FlatBuffers.

Ключевые слова: *среда моделирования, многоагентная система, интеллектуальный агент, искусственный интеллект, архитектура.*

Агентное моделирование – подход к моделированию систем, содержащих автономных и взаимодействующих агентов, позволяющий исследовать поведение агентов и то, каким образом оно определяет поведение всей системы в целом [1]. Идея агентного моделирования берет начало с 1940-х годов и связана с именами Джон фон Нейман, Джон Конвей и Станислав Улам.

Первые агентные модели были тривиальными из-за невозможности осуществления трудоемких вычислений, однако с появлением компьютеров они получили широкое распространение. Стремительное развитие данной области информационных технологий объясняется все возрастающей сложностью и пространственной распределенностью информационных систем [2]. Сегодня *интеллектуальные агенты* (ИА) и основанные на них *многоагентные системы* (МАС) применяются практически во всех областях науки и техники, причем как в физических мирах (автопилотируемые транспортные средства, автономные роботы и другое), так и в виртуальных средах (моделирование потребительского рынка, бизнес-процессов, социальных сетей и другое) [3].

Количество задач, которые можно решать с использованием агентного моделирования, достаточно велико, именно поэтому возникает потребность в создании среды моделирования, позволяющей проводить эксперименты с ИА. Основным требованием к такой среде является гибкость, заключающаяся в возможности моделирования процессов из самых разных предметных областей, например, социология (возникновение и развитие конфликтных ситуаций, образование групп, конкуренция, приспособление), распространение информации (новости) в Интернете, эпидемиология (распространение вирусов), технологические процессы на производстве (взаимодействие датчиков в аварийных ситуациях), игры и т.д. Такая среда должна отличаться простотой и обеспечивать основные функции (поддержка жизненного цикла агентов и их взаимодействия, отображение промежуточных и конечных результатов эксперимента). Ее эффективность может быть определена следующими критериями:

– количество поддерживаемых агентов;
– возможность импорта агентов;
– наличие готовых модулей для моделирования;
– горизонтальная расширяемость.

В настоящее время множество сред моделирования основано на парадигме агентного моделирования. С точки зрения проведения экспериментов с ИА наибольшее количество предметных областей охватывает ПО ABLE [4], разработанное компанией IBM по стандартам FIPA [5]. ABLE представляет собой агентную платформу, написанную на языке Java, для разработки и развертывания приложений на базе гибридных интеллектуальных агентов [6]. ABLE предлагает пользователям большие функциональные возможности, но это делает ее сложной в освоении.

Платный инструмент AgentSheets Inc. [7] содержит правила и drug-and-drop-интерфейс, упрощающие внедрение в разработку. С помощью AgentSheets пользователь может создавать свои собственные игры и симуляции различного уровня сложности, что делает данное ПО востребованным как среди школьников, так и среди ученых. Ограничением является возможность использования только готовых модулей.

Программная библиотека для разработки систем мультиагентного моделирования [8] предназначена для разработки приложений, реализующих модели динамических систем в различных предметных областях. Содержит абстракции и примитивы, необходимые для представления сущностей: агентов, моделей их индивидуального и коллективного поведения, виртуальной среды, законов взаимодействия среды и агентов. Программа обеспечивает возможность распределенного запуска и визуализации процесса моделирования. Язык – C#. Среди недостатков – непереносимость решения на некоторые платформы, отсутствие примитивов визуализации и импорта агентов.

Таким образом, ни один из существующих инструментов в полной мере не удовлетворяет приведенным критериям. При разработке новой среды моделирования предлагается выполнить следующие требования:

– наличие готовых типов агентов;

– возможность добавления новых типов агентов с учетом ограничений на интерфейсы;

– обеспечение взаимодействия среды моделирования с агентами, запущенными на другом компьютере;

– возможность участия нескольких пользователей в эксперименте;

– отсутствие ограничений на язык написания кода агента.

Дополнительно к перечисленным требованиям предлагается обеспечить представление результатов моделирования не только в текстовом формате, но и в виде пошагового воспроизведения эксперимента на базе его протокола.

Создание среды моделирования требует разработки модели всех процессов, происходящих в разрабатываемой среде. Такая модель должна:

– описывать начальное состояние среды моделирования, объектов, а также условий завершения эксперимента;

– описывать все этапы жизненного цикла объектов и процессов их взаимодействия;

– не зависеть от способа и среды реализации.

Модель эксперимента будем представлять набором $M = (S, I, R, P, G, N, O, L, Y, K)$, где S – конечное непустое множество состояний эксперимента, содержащее начальное s_0 и конечное s_k состояния; I – начальная конфигурация эксперимента; R – конечное непустое множество операций; $P \subseteq S \times R \times S$ – конечное непустое множество переходов, причем в кортеже $\langle s_i, r, s_j \rangle$, $s_i \neq s_j$; $G \subseteq N \times O \times L$ – вектор состояния объектов среды моделирования (N – множество всех объектов, участвующих в эксперименте, O – множество параметров, характеризующих состояние этих объектов, L – множество значений параметров); $Y : S \rightarrow t \times G$ – функция фиксации состояния, фиксирующая время, когда состояние произошло, и состояние всех объектов среды на текущий момент (t – временная отметка события); K – множество условий продолжения эксперимента; t_r – тактовая частота моделирования.

Рассмотрим функционирование математической модели на примере. Пусть при $t = 5$ среда моделирования находится в состоянии s_5, таком, что существуют объекты «бонус» и «танк». Причем танк движется в сторону бонуса и находится на расстоянии двух тактов от него. Тогда вектор состояния объектов среды моделирования g_5 ($g_5 \in G$) содержит:

– бонус n_0($n_0 \in N$), характеризующийся вектором параметров $o_0[o_x, o_y, o_v, o_{lt}, o_{bt}]$ ($o_0 \in O$), где o_x – координата по оси абсцисс; o_y –координата по оси ординат; o_v – скорость движения; o_{lt} – оставшееся время жизни, выраженное в количестве тактов; o_{bt} – тип бонуса, со значениями из вектора $l_0[2, 0, 0, 10, 1]$ ($l_0 \in L$);

– танк n_1($n_1 \in N$), характеризующийся вектором параметров $o_1[o_x, o_y, o_{vx}, o_{vy}, o_{lt}]$ ($o_1 \in O$), где o_{vx} – проекция скорости на ось абсцисс; o_{vy} – проекция скорости на ось ординат, со значениями из вектора l_1 [0, 0, 1, 0, 85] ($l_1 \in L$).

При переходе p_5 из состояния s_5 в состояние s_6 выполняется операция r_5.

Вектор значений параметров l_0 бонуса n_0 принял вид [2, 0, 0, 9, 1]. Уменьшение значения параметра o_{lt} происходит каждый такт, так как данный параметр отвечает за время жизни бонуса. Значения параметров o_x, o_y остались неизменными, так как $o_v = 0$. Значение параметра o_{bt} задается при инициализации среды моделирования и остается постоянным до конца жизни объекта.

Вектор значений параметров l_1 танка n_1 принял вид [1, 0, 1, 0, 84]. Значение параметра o_x увеличилось на единицу, что говорит об изменении координаты по оси абсцисс. Это связано с тем, что $o_{vx} = 1$ и $o_{vy} = 0$, то есть танк движется в направлении бонуса со скоростью, равной единице. Изменение значения параметра o_{lt} объясняется завершением текущего такта моделирования.

Проверка множества K условий продолжения эксперимента содержит

$$k_0 = \begin{cases} \text{true, если } o_{lt} > 0, \\ \text{false иначе.} \end{cases}$$

На текущем такте моделирования k_0 = true, следовательно, эксперимент продолжается. Функция фиксации состояния Y сохранила текущее состояние. Теперь $t = 6$.

При переходе p_6 из состояния s_6 в состояние s_7 выполняется операция r_6.

Вектор значений параметров l_0 бонуса n_0 принял вид [2, 0, 0, 8, 1]. Значение параметра o_{lt} уменьшилось на единицу.

Вектор значений параметров l_1 танка n_1 принял вид [2, 0, 1, 0, 83]. Значение параметра o_x увеличилось на единицу, что говорит об изменении координаты по оси абсцисс: танк продолжает движение в сторону бонуса. Снова уменьшается значение параметра o_{lt}.

Как можно заметить, значения o_x и o_y бонуса n_0 и танка n_1 равны. Значит, два объекта вступают во взаимодействие. В результате этого взаимодействия происходит следующее:

– вектор значений параметров l_1 танка n_1 принимает вид $[2, 0, 1, 0, 113]$; значение параметра o_{lt} увеличилось на 30 – так, как и подразумевалось при $o_{bt} = 1$;

– жизненный цикл бонуса n_0 заканчивается, следовательно, g_6 больше не содержит информацию о нем, его параметрах и их значениях.

Условие продолжения эксперимента k_0 = true. Функция фиксации состояния Y сохранила текущее состояние. Теперь $t = 7$.

И так далее.

Если на каком-либо шаге эксперимента танк выстрелил, в векторе g_i появился бы снаряд $n_j(n_j \in N)$, характеризующийся вектором параметров $o_j[o_x, o_y, o_{vx}, o_{vy}, o_a, o_{lt}, o_d]$ $(o_0 \in O)$ где o_a – ускорение, o_d – урон.

Архитектура среды моделирования, представленная на рисунке 1, была спроектирована с учетом перечисленных требований и описанной математической модели.

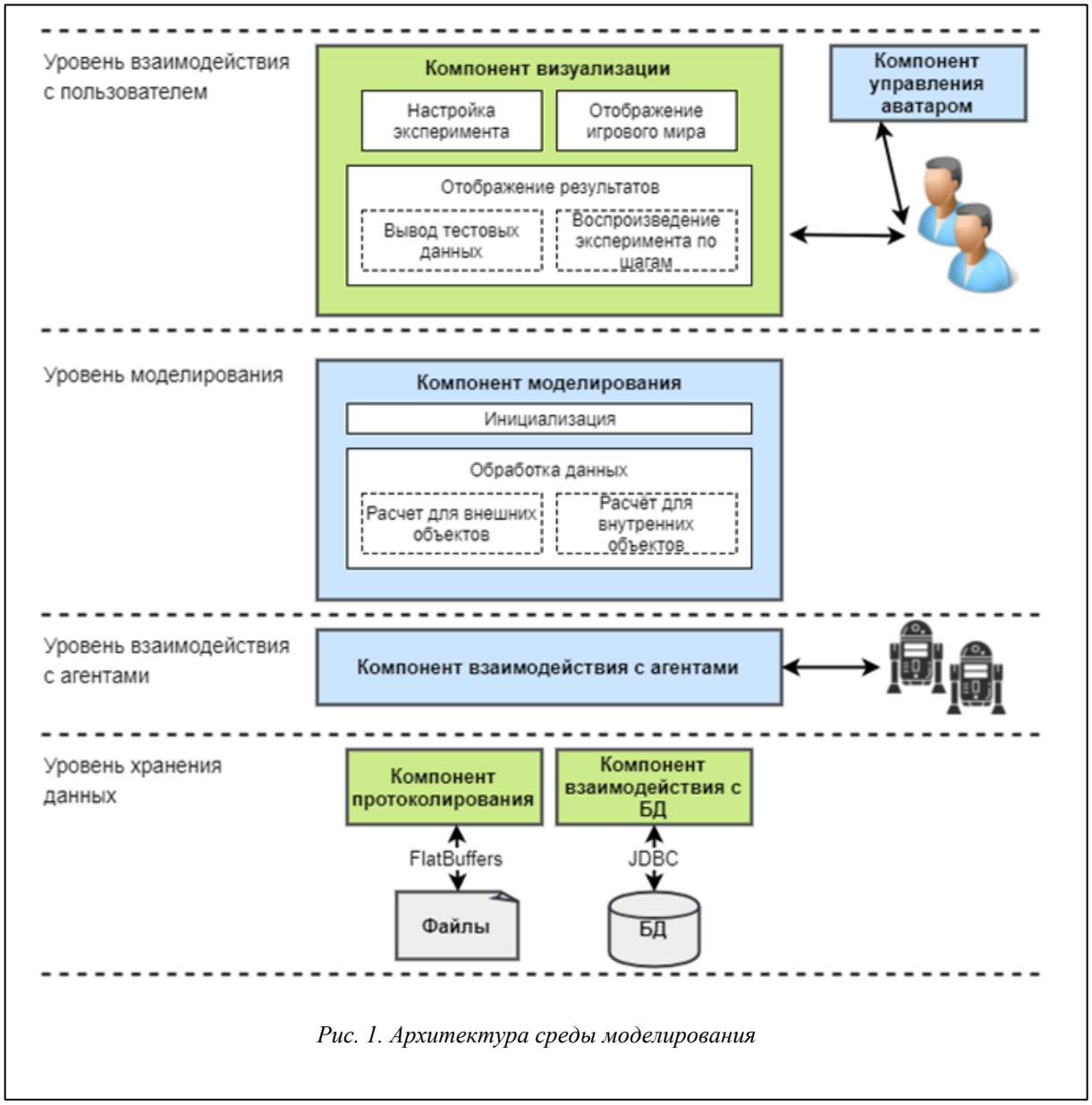

Рис. 1. Архитектура среды моделирования

Функционирование архитектуры представлено на рисунке 2 [9].

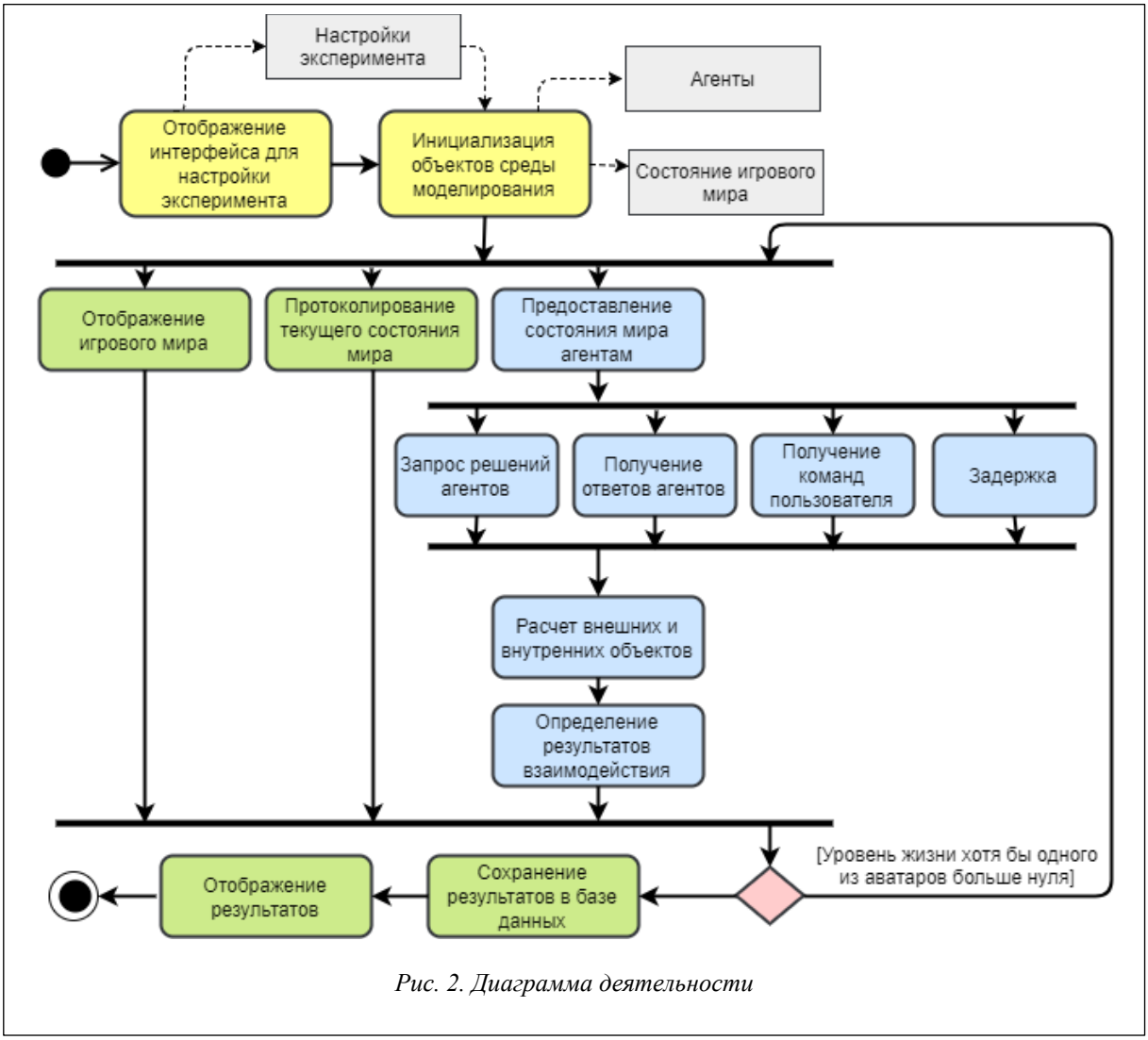

Рис. 2. Диаграмма деятельности

Компонент визуализации предоставляет пользователю интерфейс для настройки эксперимента. Заданные значения параметров передаются компоненту моделирования для инициализации. В результате появляется начальное *состояние игрового мира* (СИМ).

Компонент протоколирования записывает СИМ в протокол. Компонент взаимодействия с агентами передает СИМ агентам. Компонент визуализации, получив СИМ, начинает отображение игрового мира. Во время эксперимента пользователь с помощью мыши может задавать направление движения аватара и создавать новые объекты. Информацию о действиях пользователя на каждом такте компонент управления аватаром отправляет компоненту моделирования. Аналогично компонент взаимодействия с агентами передает компоненту моделирования команды агентов. Компонент моделирования пересчитывает координаты, в которых внешний объект, инициировавший расчет, должен оказаться к концу следующего такта моделирования, инициализирует новые объекты. По окончании расчета для внешних объектов компонент моделирования приступает к пересчету координат для внутренних объектов.

Определив результаты взаимодействия объектов среды, компонент моделирования создает новое СИМ и проверяет истинность условия продолжения эксперимента. В случае, когда условие истинно, начинается новый такт моделирования, иначе процесс моделирования завершается.

По окончании эксперимента компонент моделирования передает результаты эксперимента компоненту взаимодействия с БД для сохранения и компоненту визуализации для отображения.

Агенты – программы, не зависящие от среды моделирования и взаимодействующие с ней по заданному протоколу (например UDP). Агент получает от среды моделирования текущее состояние, тем самым воспринимая окружающий мир, самостоятельно принимает решение о действиях, которые он выполнит, и отправляет соответствующую команду среде моделирования.

Агенты могут обладать различным уровнем интеллекта. Так, простейший агент принимает решение, базируясь на схеме условие–действие. Агент, преследующий цель, понимает, какие ситуации для него желательны, что позволяет ему выбрать среди многих путей тот, что приведет к этой цели. Агент, действующий на основе полезности, умеет ранжировать состояния с помощью функции полезности. Обучающийся агент способен анализировать полученный опыт и приспосабливаться к изменяющимся обстоятельствам.

Рассмотрим частный вариант решения на примере игры-шутера. Пусть танки – это ИА, принимающие решения, в каком направлении двигаться и стрелять. Среда моделирования, внутри которой «живут» такие агенты, представляет собой поле боя с «бонусами». В игре бонусы разделены на типы и в соответствии со своим типом меняют свойства танка тем или иным образом: бонусы, укрепляющие броню; бонусы, увеличивающие скорость танка, и т.д. Помимо бонусов, существуют и другие внутренние объекты среды – снаряды, создаваемые танками. При попадании снаряда в танк уменьшается прочность его брони.

Настройки игры могут включать в себя имя игрока (обязательно), тип танка, количество танков-противников и их типы.

Последние два параметра дополнительные и генерируются случайным образом, если пользователь не указал иного.

Клиентская часть, включающая в себя компонент визуализации и компонент управления аватаром, реализована на JavaScript [10], а серверная часть – на Java. В качестве протокола взаимодействия выбран WebSocket.

В программе существует один экземпляр класса, управляющий процессом моделирования, и он предоставляет глобальную точку доступа к нему (паттерн «Синглтон» [11]).

Танки могут выступать и как объекты, управляемые пользователем, и как самостоятельные программные модулеи, взаимодействие которых со средой осуществляется с использованием Java Reflection или по протоколу UDP с JSON-сообщениями. Предложенная архитектура поддерживает расширение перечня протоколов для подключения агентов. Единственный критерий при выборе протокола – быстродействие. Компонент взаимодействия с агентами реализует паттерн «Фабрика» [11], что гарантирует одинаковую обработку сообщений агентов, полученных разными способами доставки.

Пример сообщения, полученного от агента (в формате JSON): {“x”: “10”, “y”: “48”, “vx”: “5”, “vy”: “7”, “a”: “0”, “missle”: {“create”: “true”, “vx”: “16”, “vy”: “11”}}.

В качестве агента в данном случае выступает танк. Он передает среде моделирования следующие данные: текущие координаты, проекции скорости, ускорение и объект missile (в переводе с англ. – снаряд), содержащий флаг создания и значение урона, наносимого этим снарядом.

Пример сообщения, переданного агенту, аналогично представленному танком, от среды моделирования (в формате JSON):

{“tanks”: [“tank0”: {“x”: “153”, “y”: “55”, “vo”: “0”, “vx”: “5”, “vy”: “2”, “a”: “0”, “r”: “55”, “lifeTime”: “100”}, “tank1”: {…}…],

“missiles”: [“missile0”: {“x”: “153”, “y”: “55”, “vo”: “0”, “vx”: “5”, “vy”: “2”, “a”: “0”, “r”: “55”, “lifeTime”: “10000”, “damage”: “40”}, “missile1”: {…}…],

“bonuses”: [“bonus0”: {“x”: “17”, “y”: “20”, “lifeTime”: “120”, “bonusType”: “1”}]}.

Приведенное сообщение содержит информацию об объектах, существующих на текущий момент моделирования в среде, представленную в виде трех массивов: танков, снарядов и бонусов.

Информация об агентах хранится в файле формата xml:

```
<agents>
<agent>
<through>JavaReflection</through>
<class>ru.spb.leti.fkti3304.model.Tank</class>
<name>Friendly</name>
</agent>
<agent>
<through>JavaReflection</through>
<class>ru.spb.leti.fkti3304.model.InvulnerableTank</class>
<name>BornToWin</name>
</agent>
<agent>
<through>UDP</through>
<ip>192.168.1.25</ip>
<port>3082</port>
<name>Agent007</name>
</agent>
</agents>
```

Во время проведения эксперимента все СИМ записываются в протокол. Для реализации эффективного хранения используется инструмент FlatBuffers [12], в основе которого лежит механизм сериализации. Использование этой технологии позволяет осуществлять повторное проигрывание эксперимента на базе десериализованных данных из протокола.

В результате среда моделирования решает задачу проведения экспериментов с интеллектуальными агентами, запрограммированными на любом языке программирования.

Представленная среда моделирования предназначена для проведения экспериментов с интеллектуальными агентами. Она позволяет подключать агентов, проводить эксперимент и пошагово воспроизводить действия агентов в различных ситуациях. Предложенная реализация не предусматривает сохранение внутреннего состояния агента, так как это усложняет протоколы взаимодействия со средой моделирования. В качестве развития данная среда может быть расширена за счет данной функции, которая позволит не только качественно оценивать действия агентов, но и пошагово анализировать их внутреннее состояние.

Литература

1. Бродский Ю.И. Лекции по математическому и имитационному моделированию. М.–Берлин: Директ-Медиа, 2015. 240 с.

2. Зайцев И.Д. Многоагентные системы в моделировании социально-экономических отношений: исследование поведения и верификация свойств с помощью цепей Маркова: дисс… канд. ист. наук. Новосибирск, 2014. 142 с.

3. Среда имитационного моделирования агентных систем реального времени. URL: https://cyberleninka.ru/article/n/sreda-imitatsionnogo-modelirovaniya-agentnyh-sistem-realnogo-vremeni (дата обращения: 20.07.17).

4. ABLE: Agent Building and Learning Environment. IBM Research. URL: http://researcher.watson.ibm.com/researcher/view_group.php?id=979 (дата обращения: 16.07.17).

5. The Foundation of Intelligent Physical Agents (FIPA). URL: http://www.fipa.org/ (дата обращения: 18.07.17).

6. Инструменты для построения мультиагентных систем. URL: https://techweirdo.wordpress.com/2010/12/01/p73/ (дата обращения: 20.07.17).

7. What is AgentSheets? URL: http://www.agentsheets.com/products/index.html (дата обращения: 20.07.17).

8. Князьков К.В., Чуров Т.Н. Программная библиотека для разработки систем мультиагентного моделирования Свид. о регистр. прогр. для ЭВМ № 2013619909; зарегистр. 18.10.13.

9. Особенности протокола в IO-играх. URL: https://habrahabr.ru/post/323466/ (дата обращения: 20.07.17).

10. Беляев С.А. Разработка игр на языке JavaScript. СПб: Лань, 2016. 128 с.

11. Фаулер М., Райс Д., Фоммел М. [и др.]. Архитектура корпоративных программных приложений. СПб: Вильямс, 2007. 544 с.

12. FlatBuffers Documentation. URL: https://google.github.io/flatbuffers/ (дата обращения: 20.07.17).