МЕТОДИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРОЦЕССА ГИБКОЙ РАЗРАБОТКИ АГЕНТНЫХ ИМИТАЦИОННЫХ МОДЕЛЕЙ

О.А. Николайчук, А.И. Павлов, А.Б. Столбов (Иркутск)

Современное состояние информационных и коммуникационных технологий открывают новые возможности и создают ниши для активного применения мультиагентного подхода, многоагентных систем (МАС) и агентных имитационных моделей (АИМ). Наиболее эффективно применять МАС и АИМ для анализа и разработки систем с автономными, проактивными и гетерогенно распределенными элементами, например,в таких областях как финансы, логистика, промышленность, социальная сфера и т.п. В сфере информационных технологий перспективными направлениями являются интернет вещей, виртуальные цифровые помощники, социальные сети, образование.

К настоящему времени тематика АИМ активно развивается. Согласно одному из последних обзоров [1], существует более 80 программных средств, которые могут быть использованы для разработки АИМ. Такое разнообразие инструментов и связанных с ними методологий разработок, может быть одним из сдерживающих факторов при освоении и применении технологий создания АИМ. В связи с этим на данный момент актуальны исследования, позволяющие решить проблему высокого порога входа для разработчиков прикладных АИМ, что особенно актуально для специалистов-предметников, которые, как правило, не обладают высоким навыками в программировании и моделировании.

Для решения указанной проблемы существующие платформы создания МАС разделяют структуру прикладного агента как минимум на две части: «generic agent» – инвариантную часть, в которой представлены функции, общие для всех агентов прикладной системы; и предметно-зависимую часть, которая специализирует «generic agent» в зависимости от его содержательного смысла в МАС. Целью последних работ авторов [2-6] является развитие этой идеи и её применение к процессу разработки прикладных АИМ. При этом в предлагаемом подходе осуществляется дальнейшая детализации составляющих. Из инвариантной части, связанной прежде всего с функциональностью программных средств реализации АИМ, явно выделяется часть, относящаяся непосредственно к методологии разработки АИМ. Также предполагается, что инвариантная часть не привязана к конкретному программному средству, а обеспечивает достаточный уровень общности, чтобы использовать методы различных существующих сторонних библиотек и инструментальных платформ создания АИМ, включая возможность их комбинаций.

Таким образом, на абстрактном уровне в результате процесса гибкой разработки прикладная АИМ будет состоять из следующих частей: 1) инвариантная; 2) методологическая; 3) предметная; 4) конфигурационная (настройка программного обеспечения для реализации АИМ).

Данная статья посвящена преимущественно второй (методологической) части и рассматривает метод описания методологии проектирования АИМ и метод проектирования АИМ с использованием предложенной ранее авторской системы моделей [2], специализирующий модельно-управляемый подход [7] к разработке АИМ.

Применение модельно-управляемого подхода для разработки АИМ

Согласно предлагаемому подходу конкретная реализация агентной имитационной модели может рассматриваться как некоторая программа, обладающая специфичным поведением и функционирующая в гетерогенной среде. Поэтому основная идея подхода – применение к процессу создания АИМ принципов модельно-управляемый разработки MDD (model-driven development) [7]. За последние годы данный подход активно применяется не только для решения проблем, связанных с непосредственной разработкой программного обеспечения, но также используется и в других смежных областях, например, в имитационном моделировании [8]. Подход MDD предполагает обобщение специфичного поведения разрабатываемых программных

систем в виде метамоделей и создание средств их обработки, на основе которых происходит построение конкретных моделей, описывающих проблемную ситуацию.

Так, на основе известных методологий создания многоагентных систем можно формализовать понятия для описания структуры АИМ (агент, среда, ресурс, действие и т.д). На основе описания операций, используемых для проектирования и реализации АИМ, можно формализовать предметно-независимое поведения элементов АИМ [5, 6]. К таим операциям можно отнести, например, функции, связанные с жизненным циклом имитационной модели, коммуникацией между элементами модели; логическим выводом на основе баз знаний; организацией вызова внешних подпрограмм и т.п. Далее необходимо произвести формализацию предметной специфики АИМ, включая создание онтологии предметной области и базы знаний. В результате объединения двух составляющих АИМ будет получена её спецификация, которая может быть реализована либо путем непосредственной интерпретации, либо генерации кода для какой-либо системы имитационного моделирования. Обобщенная модель информационного процесса создания и функционирования системы агентного имитационного моделирования (рис. 1) включает в себя как метод описания методологии проектирования, так и метод проектирования АИМ, которые рассмотрим подробнее в следующих разделах.

Метод описания методологии проектирования

С точки зрения системного аналитика (специалиста в агентном имитационном моделировании) процесс описания методологии проектирования АИМ может быть задан следующей последовательностью:

1. *Онтологическое моделирование структуры агентной модели* $M^{St\text{-}A}$. Метамоделью для $M^{St\text{-}A}$ выступает модель M^{Ont}=<Понятие, Атрибут, Отношение, Экземпляр>, более подробное описание представлено в предыдущей работе авторов [2]. На данном этапе в рамках модели $M^{St\text{-}A}$ необходимо описать множество взаимосвязанных понятий, которые будут использованы в качестве элементов (строительных блоков) для представления архитектуры агента, среды и других объектов АИМ в соответствии с выбранной аналитиком методологией проектирования АИМ.

В качестве примера при создании $M^{St\text{-}A}$ адаптируем вариант эталонной модели GRAMS (General Reference Model for Agent-based Model ingand Simulation) [9] (для краткости здесь приведены пять её базовых понятий):

$M^{St\text{-}A}$=<Модель, Среда, Объект, Событие, Агент>.

Перечисленные компоненты являются понятиями в терминах M^{Ont}, они могут иметь потомков, а также связны через атрибуты и отношения с другими понятиями $M^{St\text{-}A}$. Рассмотрим некоторые из них более подробно.

<Среда>=<Понятие>, {<Свойство среды>}, {<Местоположение>},

{<Механизм обновления среды>},

где<Понятие> $\in M^{Ont}$ – базовое понятие, создаваемое аналитиком для идентификации среды.{Механизм обновления среды} содержит описание функций обновления свойств и местоположений среды.

<Объект> задается через понятия M^{Ont} и будет уточняться на последующих этапах.

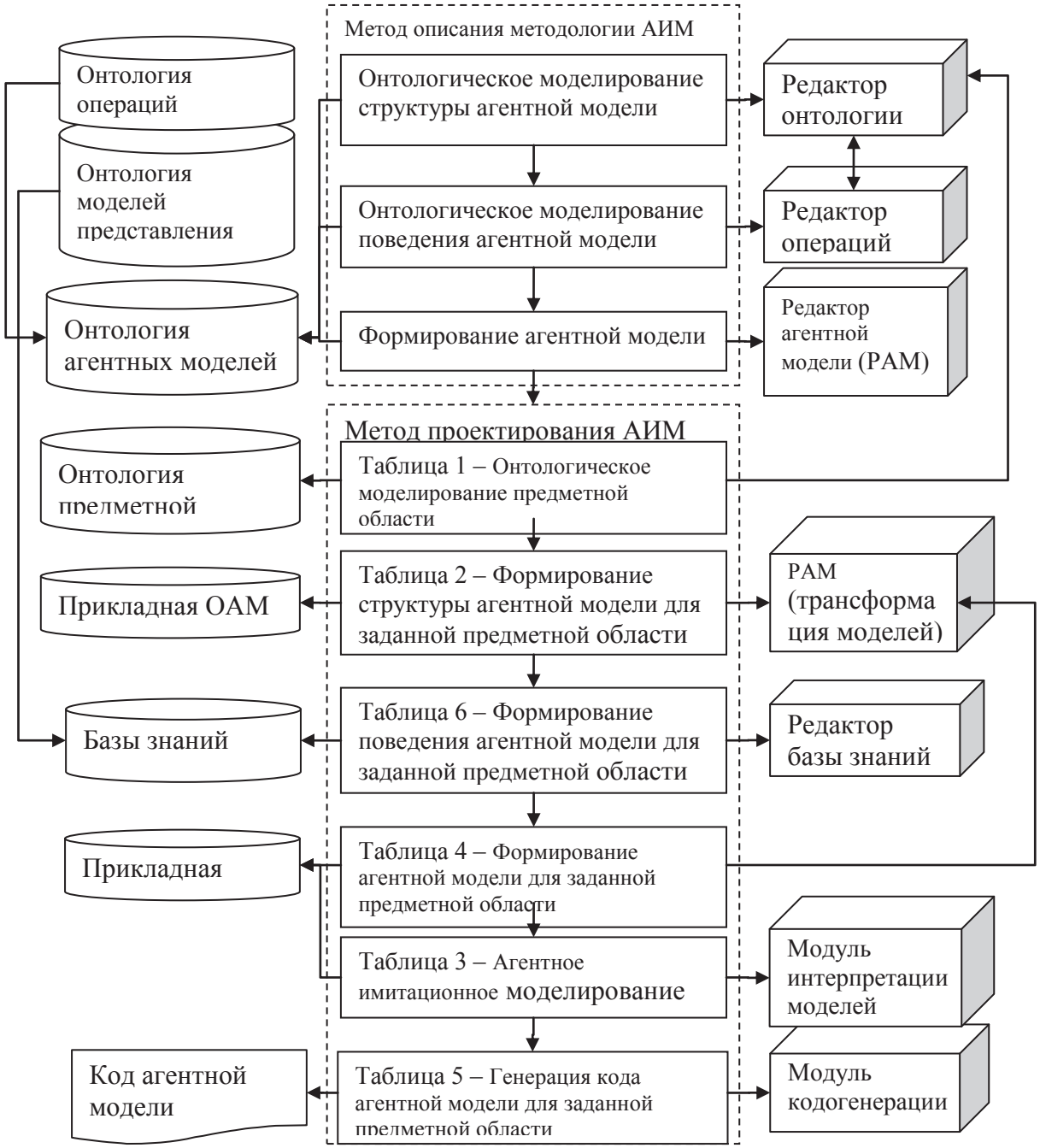

Рис. 1. Модель информационного процесса создания и функционирования системы агентного имитационного моделирования

<Событие> – с использованием этого понятия (или его потомков) описываются явления (мгновенные), возникающие в определенный момент времени $t \in T$ и способные изменить состояние модели. События могут быть эндогенными (возникают внутри агента или среды и не затрагивают другие элементы АИМ) и экзогенными.

<Агент>=<Понятие>,{<Сенсор>},{<Эффектор>},{<Механизм рассуждений>},

где<Понятие> $\in M^{Ont}$ – базовое понятие, создаваемое аналитиком для идентификации агента и определения его свойств; <Сенсор> – с использованием этого понятия (или его потомков) задается описание интерфейса, через который агент воспринимает информацию из окружающей среды и, если это определено в используемой методологии, от других агентов. <Эффектор> – с использованием этого понятия задается описание интерфейса, позволяющий агенту взаимодействовать со средой и (агентами) и активно добиваться свих целей. <Механизм

рассуждений> – является внутренним компонентом агента, обеспечивающим обработку данных, воспринимаемых сенсорами; принятие решений и управление эффекторами. Можно выделить несколько вариантов классификации архитектуры агентов [10]: реактивные, делиберативные и рефлексивные; одноуровневые, горизонтальные многоуровневые, вертикальные многоуровневые. В зависимости от конкретного типа архитектуры агента механизм рассуждений может включать, например, базу знаний, поток работ/задач (workflow), планировщик и т.п.

2. *Онтологическое моделирование поведения агентной модели* $M^{Op\text{-}A}$. На данном этапе необходимо описать поведение активных элементов АИМ, используя в качестве элементов (строительных блоков) методы функциональных компонентов предлагаемой системы агентного имитационного моделирования [2-6]: компонент работы с данными (DataController), коммуникационный компонент (CommunicationEngine), компонент рассуждений на основе правил (Inference Engine), компонент движка моделирования (SimulationEngine), компонент вызова внешних процедур и исполнения операций (Imperative Engine) и т.п. При этом каждый метод функционального компонента является базовой операцией Op^B, которые можно объединять в поток работ с использованием последовательных, разветвляющихся и циклических процессов для реализации более сложного поведения – композитной операции Op^C. Подробнее модель операций M^{Op}=<Op^B, Op^C> и особенности её программной реализации описаны в работах [2-6]. В качестве параметров операций могут выступать элементы M^{Ont}, и $M^{St\text{-}A}$ как частный случай.

На этом этапе отдельно рассматривается случай работы с компонентом ImperativeEngine, в котором, определены методы LoadInitialState – установка начального состояния рабочей памяти и ExecuteReasoning – выполнение логического вывода. Первый метод в качестве входных параметров принимает список элементов в формате M^{Ont}. Второй метод возвращает состояния рабочей памяти в формате M^{Ont}, а на входе принимает базу знаний, описанную в терминах метамодели M^{KB}, определение которой представлено в предыдущей работе авторов [2]. При этом конкретное содержание базы знаний определяется на последующем этапе «формирование поведения агентной модели для заданной предметной области», а на текущем этапе разработчиком задается лишь её идентификатор (подробнее см. пример далее).

Согласно рассмотренному выше примеру эталонной модели, на данном этапе описания методологии проектирования можно уточнить, что структурам <Механизм обновления среды> и <Механизм рассуждений> будут соответствовать композитные операции Op^C. При этом системный аналитик может использовать принципы структурной разработки и создавать избыточное множество взаимосвязанных «поведений». Выбор конкретных элементов из этого множества, соответствующих проектируемой методологии, осуществляется на следующем этапе.

3. *Формирование предметно-независимой спецификации агентной модели*. На данном этапе осуществляется объединение описаний элементов структуры $M^{St\text{-}A}$и связанных с ними поведений (либо в форме$M^{Op\text{-}A}$, либо M^{KB}) в единую метамодель – M^A со следующей структурой её элементов:

<Имя,$el^{St\text{-}A}$, Behaviour>,

$el^{St\text{-}A}\in\{\text{Понятие}\}\subset M^{St\text{-}A}$, $Behaviour\in\{M^{Op\text{-}A}|M^{KB}\}$,

где <Имя>– идентификатор пары «элемент АИМ – поведение». Например, разным типам сенсоров соответствуют разные поведения, но они могут носить одно название для удобства их адресации (см. далее пример).

Таким образом, метамодель M^A является формализованным описанием методологии проектирования и в дальнейшем будет использована в процессе разработки прикладных АИМ. В качестве иллюстративного примера поведения элемента АИМ, содержащегося в $M^{A.}$, рассмотрим псевдокод, отражающий логику жизненного цикла агента.

DataController → RetrieveProperty (*Agent,* «*Sensor*»*,* ***Sensors***);

FOREACH (Sensors as **Sensor**) DO

ImperativeEngine →Execute(Sensor, «ProcessAndGetData»,

DataOfAllSensors);

DataController → RetrieveBeahviour(Agent, «Analyze Inputs», **KnowledgeBase1**);
InferenceEngine → LoadInitialState(*Agent,DataOfAllSensors*);
InferenceEngine → ExecuteReasoning**(**KnowledgeBase1, **ResultOfReasoning1);**
DataController → RetrieveBeahviour(Agent, «Make Effectors», **KnowledgeBase2**);
InferenceEngine → LoadInitialState*(ResultOfReasoning*1*);*
InferenceEngine → ExecuteReasoning**(**KnowledgeBase2, **EffectorsList);**
DataController → RetrieveProperty(*ABM,* «*Environment*»*,* ***Environment***);
DataController → RetrieveBeahviour(Environment, «Constraints For Effectors»,
EnvKnowledgeBase);
DataController → LoadInitialStat(*EffectorsList*);
DataController → ExecuteReasoning**(**EnvKnowledgeBase,
CheckedEffectorsToExecute);
SimualtionEngine → Execute(ABM, «Execute action», Agent,
CheckedEffectorsToExecute);

Здесь курсивом выделены входные параметры соответствующих методов функциональных компонентов, а жирным шрифтом– выходные. Для удобства разработки компонент DataControllerпредоставляет следующие методы: RetrieveProperty – возвращает значение свойства понятия или экземпляра по его имени, Retrieve Beahviour – возвращает поведение M^A по его имени для понятия из $M^{St\text{-}A.}$ или соответствующего ему экземпляра. Элементы *Agent,Sensor,ABM,Environment* – это экземпляры из $M^{St\text{-}A}$, соответствующие конкретным элементам АИМ на этапе выполнения модели. Конечное состояние рабочей памяти компонента рассуждений в форме M^{Ont} возвращается через переменные *ResultOfReasoning1, EffectorsList, CheckedEffectorsToExecute.*

Представленный алгоритм означает следующее. В начале для каждого сенсора выполняется, связанная с ним, композитная операция с одинаковым именем «*Process And Get Data*», возвращающая информацию *DataOfAllSensors* в терминах M^{Ont}. Например, для типа сенсора, считывающим сообщения для агента, пришедшие к нему напрямую, может использоваться функция движка моделирования:SimualtionEngine → RetrieveNotifications(*Agent*, ***Notifications***). Далее последовательно с использованием соответствующих баз знаний (KnowledgeBase1, KnowledgeBase2, EnvKnowledgeBase) осуществляется анализ сенсоров и формирование множества возможных реакций-действий агента; выбор соответствующих эффекторов; их анализ со стороны среды; и их активация.

Метод проектирования АИМ

Метод проектирования АИМ обеспечивает поэтапную трансформацию исходной концептуальной модели предметной области (онтологии) в спецификацию АИМ в соответствии с созданным описанием методологии (в форме M^A). С точки зрения непрограммирующего пользователя процесс разработки АИМ может быть задан следующей последовательностью:

Онтологическое моделирование предметной области (создание $M^{Ont\text{-}D}$) в форме M^{Ont}.

Выбор методологии АИМ, описанной в форме метамодели M^A.

Формирование структуры агентной модели для заданной предметной области. На этом этапе происходит объединение описаний элементов структуры АИМ – $M^{St\text{-}A}$ и онтологии предметной области $M^{Ont\text{-}D}$ в единую модель $M^{St\text{-}A\text{-}D}$со следующей структурой ее элементов:

$<\{el^D\}, el^A>$, $el^A \in \{\text{Понятие}|\text{Экземпляр}\} \subset M^{St\text{-}A}$, $el^D \in \{\text{Понятие}\} \subset M^{Ont\text{-}D}$,

где el^* – элемент модели, * – вид модели.

Формирование поведения агентной модели для заданной предметной области. На этом этапе происходит разработка базы знаний $M^{KB\text{-}D}$в терминах метамоделиМ^{KB} для исследуемой предметной области, включая продукционные правила и шаблоны фактов, создаваемые на основе понятий, атрибутов и отношений из модели $M^{Ont\text{-}D}$. При этом $M^{KB\text{-}D}$ может разрабатываться с нуля или основываться на существующей базе знаний, созданной без учета агентной специфики. В

последнем случае происходит дополнение или уточнение $М^{KB\text{-}D}$ правилами с использованием элементов из $М^{St\text{-}A}$ и $М^{Ont\text{-}D}$ с учетом их связи через $М^{St\text{-}A\text{-}D}$.

Формирование агентной модели для заданной предметной области. Здесь создается спецификация агентной модели $М^{A\text{-}D}$, являющаяся объединением всех сформированных на предыдущих этапах моделей. Также на этом этапе через создание экземпляров понятий из $М^{A\text{-}D}$ происходит задание начальных условий, настройка параметров вычислительного эксперимента, включая конфигурацию обращений к вычислительным модулям.

Программное обеспечение процесса разработки АИМ

Программный комплекс, реализующий авторский подход к разработке АИМ (рис.2), на данный момент включает в следующий набор компонентов [4]: компонент управления данными – $К^{дата}$, компонент концептуального моделирования – $К^{км}$, компонент рассуждения на основе правил – $К^{БЗ}$, компонент организации двухстороннего обмена данными – $К^{ком}$, компонент диалогового взаимодействия с пользователем – $К^{диалог}$, компонент управления процессом создания спецификации АИМ – $К^{агент}$, компонент управления вычислительным экспериментом – $К^{эксп}$, компонент реализации имитационного моделирования (интерпретации спецификации АИМ) – $К^{АИМ}$. При этом часть из этих компонентов являются служебными и используются для реализации функциональности других (прикладных) компонентов.

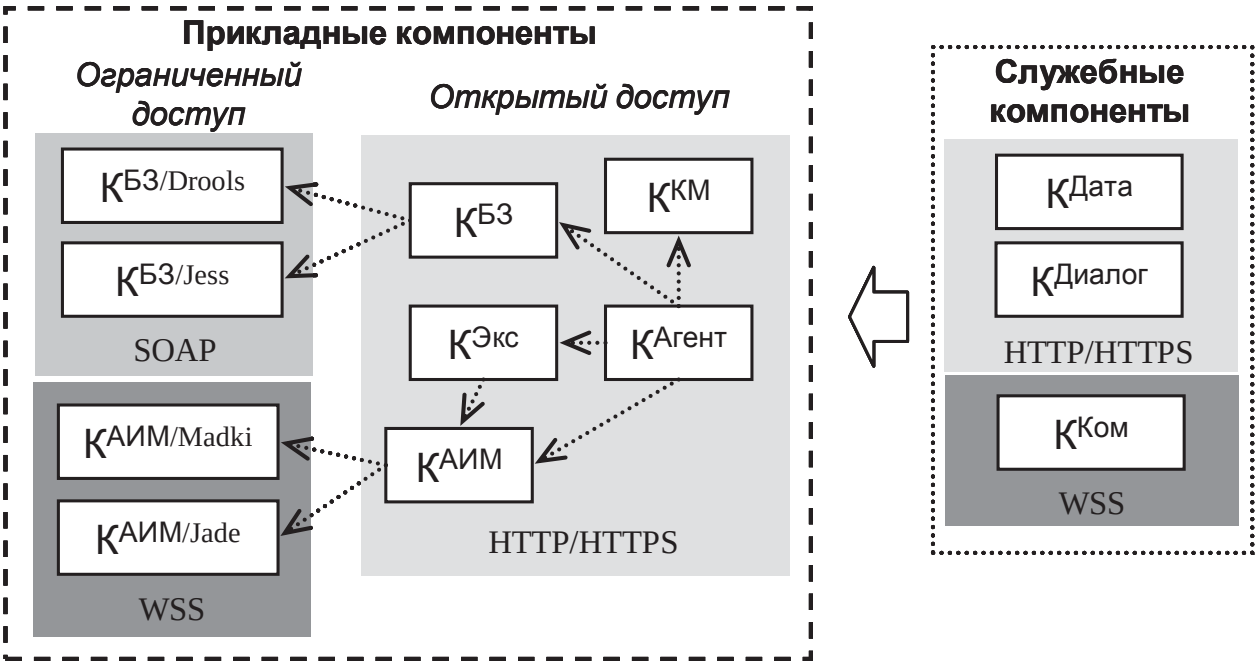

Рис. 2. Архитектура программного комплекса

Базовая функциональность компонентов $К^{дата}$, $К^{км}$, $К^{ком}$, $К^{диалог}$, $К^{БЗ}$ базируется на возможностях, авторской инструментальной платформы создания систем, основанных на знаниях [11].$К^{км}$ обеспечивает создание концептуальной модели для выбранной предметной области (КМ) с заданным уровнем детализации. $К^{БЗ}$ обеспечивает создание продукционных баз знаний на основе КМ предметной области, визуальное конструирования правил, формирования начальных условий, генерацию кода и выполнение рассуждений. Компонент двухстороннего обмена данными ($К^{ком}$)обеспечивает асинхронный обмен данными между пользователями и/или компонентами комплекса, а также обеспечения возможности начала обмена данными по инициативе сервера. Компонент $К^{диалог}$ содержит стандартный набор графических элементов управления.

Компонент $К^{агент}$ является ключевым с точки зрения обеспечения организации процесса разработки спецификации АИМ и предоставляет единый пользовательский терминал, через который осуществляется взаимодействие с другими компонентами. К основным задачами $К^{агент}$ относятся следующие: контроль за последовательностью проектирования спецификации АИМ; хранение всей необходимой информации о состоянии процесса проектирования, поддержка

преобразования моделей согласно используемой методологии [2-6]; адаптация функциональности других компонентов к текущему контексту и вызов соответствующих методов. Компонент управления и поддержки вычислительного эксперимента ($К^{эксп}$) позволяет формировать начальные условия АИМ и сохранять результаты симуляции, а также непосредственно управлять процессом имитации.

Заключение

В статье рассмотрено методическое и программное обеспечение процесса гибкой разработки агентных имитационных моделей на основе применения модельно-управляемого подхода. Представлены метод описания методологии проектирования агентных имитационных моделей (АИМ), метод проектирования АИМ, а также рассмотрена архитектура программного комплекса. Результатом работы методов является спецификация АИМ, содержащая информацию о составе и архитектуре элементов АИМ, включая набор действий, выполняемых на шаге моделирования агентом и средой; концептуальную модель предметной области и базу знаний агентов. В качестве иллюстративного примера использована эталонная модель GRAMS и рассмотрен псевдокод, отражающий логику жизненного цикла агента.

Работа выполнена при частичной финансовой поддержке РФФИ (грант № 18-07-01164, 18-08-00560).

Литература

1. AbaraS., Theodoropoulos G.K., Lemarinier P., O'Hared G.M.P. Agent Based Modelling and Simulation tools: A review of the state-of-art software // Computer Science Review. 2017. Vol.24. P. 13–33.
2. Николайчук О.А., Павлов А.И., Столбов А.Б. Особенности разработки агентных имитационных моделей на основе модельно управляемого подхода // Восьмая всероссийской научно-практической конференции по имитационному моделированию и его применению в науке и промышленности «Имитационное моделирование. Теория и практика» (ИММОД-2017): Труды конф., 18-20 октября 2017 г., Санкт-Петербург. – С. 288–293.
3. Павлов А.И., Столбов А.Б. Особенности реализации системы поддержки проектирования имитационных моделей на основе декларативного описания агентов // Седьмая всероссийская научно-практическая конференция «Имитационное моделирование. Теория и практика» (ИММОД-2015): Труды конф., 21-23 окт. 2015 г., Москва: в 2 т. / Ин-т проблем упр. им. В.А. Трапезникова Рос. Акад. наук ; под общ. ред. С.Н. Васильева, Р.М. Юсупова. – Т. 1. – М.: ИПУ РАН, 2015. ISBN 978-5-91450-172-0. С.256–261.
4. Павлов А.И., Столбов А.Б. Архитектура программного комплекса для формирования агентных имитационных моделей на основе MDD подхода // Восьмая Международная конференция «Системный анализ и информационные технологии» (САИТ – 2019): Труды конф., 8-14 июля 2019 г., г. Иркутск – Листвянка / М.: ФИЦ ИУ РАН, 2019. ISBN 978-5-904466-62-6. С. 549–555.
5. Павлов А.И., Столбов А.Б. Модели базовых компонентов системы проектирования агентных имитационных моделей и их реализация в программном комплексе Adskit // Информационные и математические технологии в науке и управлении. 2018. Вып. 4. С. 155–162.
6. Nikolaychuk O.A., Pavlov A.I., Stolbov A.B. The Agent-Based Modeling Method For The Study Of Unique Mechanical Systems // Proceedings of the 7th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2019). Advances in Intelligent Systems Research. 2019. Vol. 166.P. 201–206.
7. France R., Rumpe B. Model-Driven Development of Complex Software: A Research Roadmap // Proc. of the Intern. Conf. «Future of Software Engineering». Minneapolis. 2007. P. 37–54.
8. Cetinkaya D., Verbraeck A., Seck M.D. Model continuity in discrete event simulation: A framework for model-driven development of simulation models // ACM Trans. Model. Comput. Simul.2015.Vol. 25(3), Article 17.
9. Siegfried R. Modeling and Simulation of Complex Systems: A Framework for Efficient Agent-Based Modeling and Simulation. Springer Fachmedien Wiesbaden, 2014.

10. Николайчук О.А., Павлов А.И., Столбов А.Б. Анализ архитектур агентов и их представление в аспекте MDD-методологии // ИТНОУ: информационные технологии в науке, образовании и управлении. 2018. № 5. С. 47–53.
11. Nikolaychuk O.A., Pavlov A.I., Stolbov A.B. The software platform architecture for the component-oriented development of knowledge-based systems // Proceedings of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). 2018. P. 1234–1239.