ОСОБЕННОСТИ РАЗРАБОТКИ АГЕНТНЫХ ИМИТАЦИОННЫХ МОДЕЛЕЙ НА ОСНОВЕ МОДЕЛЬНО-УПРАВЛЯЕМОГО ПОДХОДА[1]

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

Имитационное моделирование на основе агентной парадигмы (ИМАП) – это подход для анализа и моделирования сложных систем в терминах агентов – автономных, целеполагающих и взаимодействующих сущностей, которые могут быть организованы в сообщества. Несмотря на стремительный рост популярности агентного имитационного моделирования в различных областях (логистика, финансы, социальная сфера), его распространение ограничено, так как большинство программных средств обеспечивает возможность создания агентной имитационной модели (АИМ) за счет ее непосредственной реализации, определяя тем самым значительные требования к навыкам моделирования и программирования со стороны пользователей, что особенно затруднительно для специалистов-предметников. В связи с этим создание ориентированных на специалистов-предметников методологий и программных инструментов создания АИМ является актуальной научной и прикладной задачей.

В основе работы лежит два принципа. Первый состоит в том, что с точки зрения парадигмы имитационного моделирования методологии должны обеспечивать сопровождение специалиста-предметника на протяжении всех этапов жизненного цикла создания и использования АИМ, который согласно [1] включает следующие пять высокоуровневых этапов: формулировка проблемы, концептуальное моделирование, спецификация, реализация, проведение экспериментов. Таким образом, в рамках разрабатываемого подхода будет поддерживаться непрерывный интегрированный итеративный процесс создания АИМ.

Второй принцип, что АИМ может рассматриваться как компьютерная программа, поэтому к процессу ИМАП могут быть применены подходы программной инженерии. Одним из наиболее эффективных методов, предназначенных для решения проблемы сложности автоматизации процесса ИМАП, является использование –модельно-управляемого подхода (Model Driven Development, MDD) [2]. Управляемая моделями разработка приложений – это подход, предполагающий систематическое использование моделей как основного инструмента процесса разработки, в рамках которого используются модель, метамодель, метаметамодель, а также трансформации данных моделей, с целью получения хорошо структурированной программной системы и увеличения уровня абстракции.

Полученные к настоящему времени результаты применения MDD-подхода для процесса ИМАП [3, 4, 5] позволяют определить направления дальнейших исследований в данной области. В частности, в работах [6, 7] приведенв подробные описания метамоделей агента, однако этапу реализации исследователями уделено недостаточно внимания. Например, недостаточно подробно рассмотрено или вообще отсутствует описание метамодели, описывающей средства реализации (движок моделирования, машина логического вывода, системы визуализации и обработки данных). По мнению авторов, создание таких метамоделей и их явное использование в процессе ИМАП позволит значительно повысить многосторонность и гибкость этого процесса за счет повторного использования различных артефактов, применяемых на этапе реализации.

С учетом вышеизложенных соображений авторы ведут разработки по созданию методологии и программного комплекса ADSkit (Agent Development Support kit) [8, 9], который должен обеспечить поддержку процесса ИМАП, управляемую моделями и ориентированную на экспертов предметников. В данной работе рассматривается теоретическое обеспечение применения MDD-парадигмы в рамках предлагаемого подхода ADSkit, в частности, представлено описание моделей, метамоделей и их трансформаций.

[1] Работа частично поддержана грантами РФФИ № 16-37-00041 и №15-07-05641.

Архитектура метамоделей ADSkit. В рамках подхода ADSkit предлагается использовать следующую четырехуровневую архитектуру метамоделей:

- M0 – имитационная модель во время выполнения;
- M1 – спецификация АИМ;
- M2 – спецификация элементов M1 согласно выбранной методологии разработки АИМ;
- M3 – определяет наиболее абстрактные элементы для спецификации M2.

Отметим, что в подходе ADSkit модели АИМ условно можно разделить на блоки по следующим критериям: зависимость и независимость от предметной области, описание структуры и поведения.

В качестве основы предлагаемых моделей в аспекте методологии MDD, ADSkit использует метаметамодель M^{Ont} (модель уровня M3) в виде хорошо известной онтологической формы: концепт – атрибут – отношение:

M^{Ont}=<*Понятие, Атрибут, Отношение, Экземпляр*>,

<**Понятие** >=<*Имя*> <*Описание*> {< *Атрибут*>},

<**Атрибут** >=<*Имя*> <*Тип атрибута*> <*Значение по умолчанию*>,

<**Тип атрибута**>= *Литерал* | *Понятие,*

<**Отношение**>=<*Понятие*> <*Имя*> <*Тип отношения*> <*Понятие*>,

<**Тип отношения**>= *Литерал.*

Предметно-зависимые модели. Рассмотрим модели, относящиеся к блоку, где описывается структура и поведение, обусловленные предметной областью (модели уровня M2). Так, на основе метамотамодели M^{Ont} создается концептуальная модель M^{Ont_D}, описывающая структурную составляющую предметной области. Для представления поведенческой части используется формализм продукционных правил как естественный и простой способ представления информации для понимания человеком. Аналогично представлению концептуальной модели, ADSkit использует элементы метаметамодели M^{Ont} для описания метамодели базы знаний $M^{KB:}$

M^{KB}=<*Шаблон, Правило, Начальные условия*>;

<**Шаблон**>=<*Понятие*>|<*Отношение*>;

<**Правило**>=**IF <***Условие***> THEN <***Действие***>;**

<**Начальные условия**>=<*Экземпляр*>,

<**Условие**>=<*Наличие факта*> | <*Ограничение на слот факта (текст или число)*> |

<*Сравнение слота с фактом* > | <*Сравнение слотов фактов*>

<**Действие**>=<*Создать факт*> | <*Изменить факт*> | <*Удалить факт*> | <*Вызов внешнего метода*>.

ADSkit использует стандартные операции с экземплярами шаблонов (фактами) в условиях и действиях правил. Условная часть правила обрабатывает такие операции как сравнение атрибута факта с литералами, фактами или атрибутами другого факта. В части действия правила можно создавать, модифицировать и удалять факты. Для того, чтобы увеличить выразительную силу правил, ADSkit предоставляет возможность осуществлять вызовы внешних процедур, так как для описания некоторых вариантов поведений императивный стиль описания является более приемлемым. На основе моделей M^{KB}, M^{Ont_D} создается конкретная модель поведения M^{KB_D}.

Предметно-независимые методологические модели

Следуя MDD подходу, структурная часть агентно-ориентированной составляющей спецификации АИМ представляется в виде метамодели M^{St_A}, в которой описываются элементы (агенты, объекты, среда, события и т.п.) и отношения между ними в терминах M^{Ont}. Таким образом, согласно подходу ADSkit, метамодель структуры АИМ не является заранее определенной и может быть создана соответствующими специалистами согласно некоторой методологии ИМАП независимо от предметно-ориентированной информации. С учетом вышесказанного конкретные модели M^{St_A} и M^{Ont_D} могут быть разработаны независимо друг от друга и затем агрегированы в модель $M^{St\text{-}A\text{-}D}$, содержащую как методологическую, так и предметную информацию о структуре АИМ.

Методологическая предметно-независимая часть спецификации АИМ может содержать алгоритмы командных циклов агентов и среды, описание различных типов действий/эффекторов (породить событие, послать сообщение, изменить данные), а также методы реализации предметно-ориентированного поведения ролей, сенсоров, объектов и т.д. Таким образом, агентно-ориентированная составляющая АИМ обладает типовой функциональностью, которая реализуется на основе конечного множества операций. Эти операции могут быть сгруппированы в интерфейсы, которые должны быть реализованы соответствующими программными средствами (компонентами). Согласно MDD подходу, ADSkit вводит метамодель M^{Com}, в которой представлено описание операций и интерфейсов на высоком уровне абстракции (модель поведения):

$$\mathbf{M^{Com}}=<Op^{B}, Op^{C}>,$$

$$<\mathbf{Op^{B}}>=<\textit{Интерфейс}> <\textit{Имя}> \{<\textit{Параметр}>\},$$

$$<\mathbf{Параметр}>=<\textit{Атрибут}><\textit{Тип параметра}>,$$

$$<\mathbf{Тип\ параметра}>= In\ |Out,$$

$$<\mathbf{Op^{C}}>=<\textit{Имя}> \{<\textit{Параметр}>\}\ \{<Op^{B}>|<Op^{G}>|<Op^{C}>\},$$

$$<\mathbf{Op^{G}}>=<\textit{Оператор}\ \{<Op^{B}>|<Op^{G}>|<Op^{C}>\},$$

$$<\mathbf{Оператор}>= O_{if} \mid O_{L},$$

где $<Op^{B}>$ – описание базовой операции, которая может быть выполнена непосредственно ADSkit; $<Op^{C}>$ – описание композитной операции; <Interface address> – описание интерфейса средства/инструмента реализации; $<Op^{G}>$ – множество операций, сгруппированных относительно некоторого оператора; O^{if} – оператор ЕСЛИ; O^{L} – оператор цикла.

Описание агентно-ориентированного поведения в форме M^{Com} может быть повторно использовано при создании различных спецификаций путем интерпретации элементов АИМ в терминах предметной области и назначения предметно-ориентированного поведения этим элементам.

Интерфейсы компонентов реализации

Рассмотрим конкретные примеры интерфейсов и их методы:

– Интерфейс движка моделирования (**Simulation Engine**) содержит стандартный для АИМ набор операций, включая посылку широковещательных и адресных сообщений/уведомлений, в которых в качестве содержимого передается информация в форме Concept Instance; поддержку жизненного цикла элементов АИМ (создать, удалить, запустить и т.п.).

– Интерфейс машины логического вывода (**Inference Engine**) предоставляет операции для интерпретации и вывода на основе базы знаний, описанной в форме M^{KB_D}, а также загрузку начального состояния и получение результатов в форме Concept Instance.

– Интерфейс интерпретации процедур (**Imperative Engine**) описывает стандартный инструмент для выполнения вызовов к внешним процедурам. Предполагается, что входы и выходы этих процедур являются либо литералами, либо Concept Instance.

– Интерфейс контроллера времени выполнения (**Runtime Controller**) разработан с целью поддержки выполнения разнообразных рутинных операций с элементами АИМ в процессе программной имитации модели: получить и изменить данные; получить и выполнить операцию (описанную в форме Op^B или Op^C).

Трансформации моделей в процессе создания АИМ

В рамках подхода ADSkit на основе метамодели M^{Com} и элементов структурной составляющей метамодели $M^{St\text{-}A}$ (таких как, «Роль», «Событие», «Действие») выбранной методологии агентного моделирования формируется модель $M^{Com\text{-}A}$, содержащая операции, которые в дальнейшем используются для реализации поведения элементов $M^{St\text{-}A}$ (таких как «Агент», «Среда»), что в совокупности позволяет определить предметно-независимую агентно-ориентированную метамодель M^{A} (модель уровня M1):

$$\begin{gathered} M^{St-A} \times M^{Com} \to M^{Com-A}, \\ M^{Com-A} \times M^{St-A} \to M^{A}. \end{gathered} \quad (1)$$

На основе метамодели M^{A} осуществляется формирование предметно-зависимой модели M^{A-D} с использованием модели предметной области в форме онтологии M^{Ont-D}, а также описания поведения объектов предметной области в виде продукционных баз знаний M^{Rl-D} ($M^{Rl} \times M^{Ont-D} \to M^{Rl-D}$), результатом данного процесса является спецификация агентной имитационной модели M^{A-D} (модель уровня M1):

$$\begin{gathered} M^{Ont\text{-}D} \times M^{St-A} \to M^{St-A\text{-}D} \\ M^{A} \times M^{St-A-D} \times M^{Rl-D} \to M^{A-D}. \end{gathered} \quad (2)$$

Завершающим этапом подхода ADSkit является осуществление имитационного моделирования на основе интерпретации полученной спецификация M^{A-D} или генерации кода на ее основе (уровень M0):

$$M^{A\text{-}D} \to codI \,|\, codG, \quad (3)$$

где codI – выполнение спецификации программой-интерпретатором, а codG – сгенерированный исходный код имитационной модели.

Таким образом, предлагаемый подход ADSkit обеспечивается совокупностью моделей и их преобразований:

$$ADSkit = \left\langle M^{Ont}, M^{Rl}, M^{Com}, M^{St-A}, M^{Com-A}, M^{Ont-D}, M^{Rl-D}, M^{St-A-D}, M^{A-D}, R, codI \,|\, codG \right\rangle,$$

где R – множество описанных выше преобразований моделей (1)–(3).

Иллюстративный пример

Для демонстрации предложенного подхода рассмотрим пример реализации командного цикла модели в ADSkit, который может состоять из следующих этапов:

1) агенты принимают решения;

2) среда выполняет свое поведение;

3) результаты формируются и отправляются внешним системам.

Для упрощения понимания представим эти этапы в форме «псевдокода»:

RuntimeController→ RetrieveData(АИМ, “Состояние”, State)

ЕСЛИ State==FINISH ТО RuntimeController → Execute(АИМ, “завершение”);

```
RuntimeController → RetrieveData(АИМ, “Агент”, Агенты);
ДЛЯ ВСЕХ (Агенты as Агент) ВЫПОЛНИТЬ
        RuntimeController → Execute(Агент, “Командный цикл”);
RuntimeController → RetrieveData(АИМ, “Среда”, Env);
RuntimeController → Execute(Env, “Командный цикл”);
RuntimeController → RetrieveData(АИМ, “Внешние клиенты данных АИМ”,
                ExternalDataProcessors);
ДЛЯ ВСЕХ (ExternalDataProcessors as ExternalDataProcessor) ВЫПОЛНИТЬ
RuntimeController → RetrieveData(ExternalDataProcessor,
                “Процедура подготовки данных”, PrepareDataFunction);
RuntimeController → RetrieveData(ExternalDataProcessor,
                        “Процедура отправки данных”, SendDataFunction);
Imperative Engine → Execute(PrepareDataFunction, Агенты, DataToSend);
Imperative Engine → Execute(SendDataFunction, DataToSend).
```

Заключение

Ориентированность ADSkit на специалиста-предметника накладывает определенные ограничения на способы создания АИМ. Но вместе с тем эти ограничения позволяют формализовать некоторые методологические этапы и элементы процесса ИМАП и описать средства программной разработки АИМ, обеспечивающих создание и обработку необходимых артефактов, начиная с разработки концептуальной модели предметной области до автоматического создания модели времени выполнения.

Согласно предложенному в статье подходу предполагается, что пользователи ответственны за создание спецификации (моделей) АИМ, в то время как ADSkit должен обеспечить необходимую функциональность для редактирования спецификаций и автоматизации этапа реализации, согласно MDD-подходу.

Литература

1. **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.
2. **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.
3. Duarte, Jaidermes Nebrijo and Juan de Lara ODiM: A Model-Driven Approach To Agent-Based Simulation. ECMS, 2009.
4. **Garro A., Russo W.** EasyABMS: A domain-expert oriented methodology for agent-based modeling and simulation // Simulation Modelling Practice and Theory. 2010. Vol. 18. P. 1453–1467.
5. **Siegfried R.** Modeling and Simulation of Complex Systems: A Framework for Efficient Agent-Based Modeling and Simulation. Springer Fachmedien Wiesbaden, 2014.
6. **Challengera M., Mernikb M., Kardasa G., Kosarb T.** Declarative specifications for the development of multi-agent systems // Computer Standards & Interfaces. Vol. 43, 2016. P. 91–115.
7. **Beydoun G., Low G., Henderson-Sellers B., Mouratidis H., Sanz J.G., Pavon J., Gonzalez-Perez C.** FAML: A generic metamodel for MAS development // IEEE Transactions on Software Engineering. 2009. Vol.35 (6). P. 841–863.
8. **Павлов А.И., Столбов А.Б.** Разработка системы поддержки проектирования имитационных моделей сложных систем на основе декларативного метода описания агентов // Всероссийская научно-практическая конференция «Имитационное моделирование. теория и практика» ИММОД-2013 (21-23 октября 2013 г., С-Петербург). 2013. С. 267–270.
9. **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. С.256–261.