УДК 004.94:528.952
DOI 10.53403/9785951505071_2022_426

ЯЗЫК ПРОГРАММИРОВАНИЯ АГЕНТОВ OPAL++ ВИЗУАЛИЗАЦИОННО-ИНТЕГРАЦИОННОЙ ПЛАТФОРМЫ «ОПТИМУС»

Д. С. Собанин, И. А. Крючков, А. В. Огородников, О. В. Коваленко

Российский федеральный ядерный центр –
Всероссийский НИИ экспериментальной физики, Саров

Разработка программных агентов и эффективное задание их логики поведения являются ключевыми и наиболее сложными элементами при разработке имитационных мультиагентных моделей.

Наращивание сложности и подробности моделей зачастую требует разработки большого количества программного кода в сжатые сроки проекта. Реализация высокоуровневых визуальных и скриптовых средств программирования логики поведения агентов в мультиагентных системах может обеспечить решение этой задачи.

В работе представлены базовые подходы и средства предметно-ориентированного языка программирования агентов Opal++ визуализационно-интеграционной платформы «ОптИМУС». Показаны перспективы развития языка программирования Opal++ с целью упрощения восприятия и сопровождения разрабатываемых моделей агентов, а также интеграции технологий машинного обучения и искусственного интеллекта в имитацию интеллектуального поведения агента.

Ключевые слова: мультиагентное имитационное моделирование, ВИП «ОПТИМУС», агент, решатель, макроязык, Opal++.

Для обеспечения эффективной и качественной разработки специализированных программных комплексов поддержки принятия решений, имитационного моделирования и оптимизации во ФГУП «РФЯЦ-ВНИИЭФ» разработана визуализационно-интеграционная платформа «ОптИМУС» [1, 2]. В ВИП «ОптИМУС», в рамках мультиагентного подхода к моделированию, анализируемая система представляется в виде совокупности взаимодействующих по формализованным правилам программных агентов.

При разработке имитационных мультиагентных моделей наиболее трудоемким процессом на практике является разработка агентов и задание их логики функционирования.

Наращивание сложности и подробности моделей зачастую требует разработки большого количества программного кода в сжатые сроки проекта. При этом для выполнения работы необходимо привлекать к процессу программирования специалистов в исследуемой предметной области, которые могут не обладать требуемым уровнем подготовки в прикладном программировании и мультиагентном моделировании.

Для решения данной проблемы, экспертов предметной области необходимо обеспечить специальными средствами, позволяющими разрабатывать сложные и подробные модели без глубокого погружения в низкоуровневое программирование.

Бурное развитие технологий искусственного интеллекта во всем мире [3] и необходимость их внедрения в различные моделируемые системы также накладывает дополнительные требования к программным средствам [4]. С одной стороны, мультиагентное моделирование является одной из технологий искусственного интеллекта, с другой, в работе мультиагентных систем, при повышении детальности и сложности модели, требуется внедрение других технологий ИИ. В частности,

нейросетевых технологий, нечеткой логики, динамических навигационных сетей, метаэвристической оптимизации, онтологического представления знаний и других.

Таким образом, при создании языка программирования агентов, ставились следующие цели:

− средства программирования агентов должны обеспечивать возможность разработки без погружения в низкоуровневое программирование (это необходимо для специалистов предметной области);

− органичная интеграция технологий искусственного интеллекта в разрабатываемые мультиагентные системы.

На первой стадии реализации ВИП «ОптИМУС» в ее составе были реализованы два крупных программных пакета: мультиагентного моделирования Оптимус.Решатель и препостпроцессинга мультиагентных задач Оптимус.Препост с графическим пользовательским интерфейсом. На этой стадии, разработка агентов выполнялась с помощью языка программирования общего назначения C++. Разработчику агента требовалось, с использованием API решателя и API визуализационных средств платформы реализовать все компоненты агента. Как следствие, средства программирования агентов очень сильно связаны с обоими программными пакетами.

При проработке архитектуры модуля языка программирования логики агентов Opal++ выполнена его интеграция в Оптимус.Решатель и Оптимус.Препост (см. рис. 1).

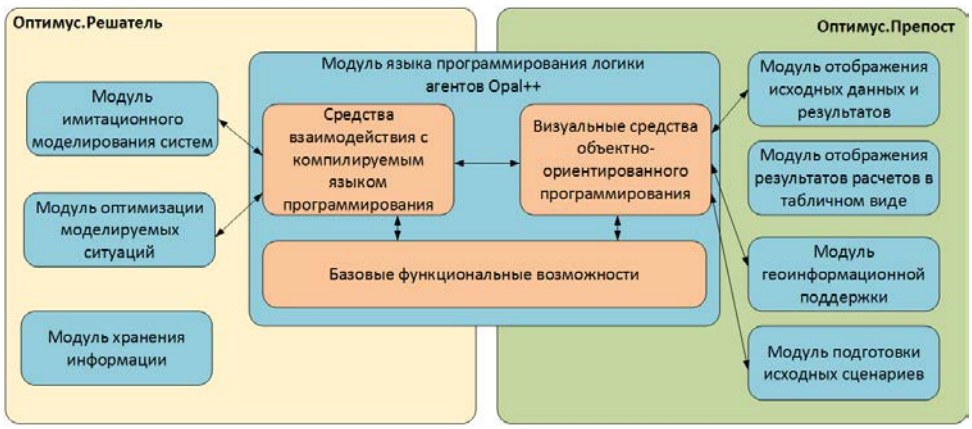


Рис. 1. Интеграция языка Opal++ в архитектуру ВИП «ОптИМУС»

Программный агент представляет собой сложную структуру (см. рис. 2), которая содержит множество параметров, поведения, задачи и т. д. Разработка агента с «чистого листа» на языке C++ – это трудоемкий и сложный процесс, требующий от разработчика множество разноплановых навыков. В частности для этого необходимы, помимо знаний в предметной области, знания особенностей работы API мультиагентного решателя, языка программирования C++, API препостпроцессора интеграционной платформы. Для того чтобы упростить (меньше специализированных знаний) и ускорить (радикальное сокращение рутинных операций) создание новых и развитие существующих агентов, в язык программирования логики агентов Opal++ заложены следующие возможности:

− поддержка функциональных возможностей мультиагентного решателя;

− поддержка функциональных возможностей визуализационно-интеграционной платформы;

− возможность конструирования новых типов агентов;

− средства разработки логики поведения агентов.

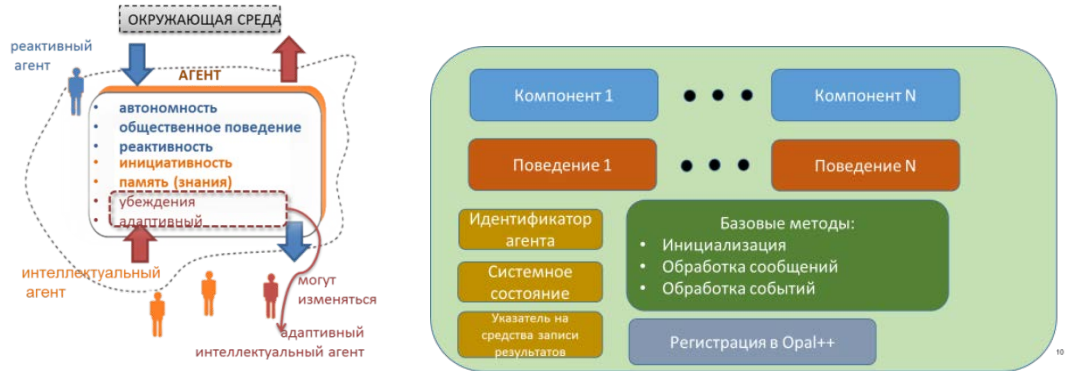
Конструктор агентов должен формировать нового агента с требуемыми предметному специалисту качествами. Новый агент основывается на классе существующего (или базового универсального агента) и содержит:

− базовый класс агента (от которого происходит наследование);

− атрибуты;

− поведение;

− дочерние объекты.



а б

Рис. 2. Программный агент: а – логическое представление; б – представление в формате мультиагентного решателя

Разработка логики агента может вестись разными специалистами. Базовый элементарный уровень может разрабатываться на низкоуровневом языке C++, а для поведенческой логики требуются сложные высокоуровневые конструкции. При создании сложной логики могут быть эффективны средства скриптового и визуального программирования. При реализации интеллектуальных агентов в перспективе потребуется встраивание в состав агента контроллеров на основе различных технологий ИИ (нейросетевой, нечеткологический, программа действий на предметно-ориентированном языке, онтологический и другие).

В состав Opal++ входят средства для программирования на различных уровнях абстракции:

– низкоуровневое задание логики и вычисление пользовательских выражений;

– объектно-ориентированное программирование;

– предметно-ориентированное программирование на основе специализированных языковых средств;

– высокоуровневое программирование с помощью различных графических диаграмм.

Каждый более низкий уровень абстракции позволяет реализовывать средства для ведения разработки на более высоком уровне. В результате каждый уровень агента может быть проработан максимально детально и при этом на уровне логики принятия решений не будет мелких подробностей, которые перегружают восприятие.

Как уже было сказано, средства разработки Opal++ могут использоваться для создания агента с «чистого листа», для разработки высокоуровневой логики поведения и для реализации мелких частных методов. При этом все эти элементы могут быть разработаны на C++. Таким образом, потребовалась возможность осуществлять разнонаправленные вызовы программного кода:

– из визуального кода на Opal++ выполнять вызовы скриптового кода на Opal++;

– из скриптового кода на Opal++ выполнять вызовы визуального кода на Opal++;

– из кода (визуального и скриптового) на Opal++ выполнять вызовы программных компонент, разработанных на C++;

– из кода на C++ осуществлять вызовы программных компонент, разработанных на Opal++ (визуально и/или на скриптах).

Для реализации этих возможностей все видимые для языка сущности (например, классы, методы, свойства, типы сообщений, типы событий и др.) должны быть зарегистрированы в единой интерпретационной системе Opal++. При разработке кода в Opal++ это происходит автоматически, а для сущностей C++ требуется прямая регистрация. Например, регистрация метода, осуществляется C++ кодом:

OPAL_BIND_METHOD (METHOD("step2", "param"), &MyAgent::step2);

А, например, вызов метода *step2* осуществляется:

obj->opal()->call("step2", 10)

при этом неважно на каком языке реализован метод *step2*, на C++ или на Opal++.

В настоящее время выполнение программного на Opal++ реализовано через интерпретацию для возможности запуска в окружениях без поддержки средств разработки. Это накладывает естественные ограничения в производительности разрабатываемого кода. Для решения этой проблемы

в перспективе планируется реализация генерации C++ кода для идентичной замены отработанных ресурсоемких методов.

Для обеспечения возможности функционирования агентов, задаваемых на Opal++, в мультиагентной среде Оптимус.Решатель, реализован доступ к программным интерфейсам (API) ядра мультиагентного решателя через регистрацию в общем интерпретационном механизме Opal++. Это позволяет пользователям в программном коде на текстовом скриптовом языке и в визуальных программах использовать базовые классы модуля мультиагентного моделирования, необходимые для программирования агентов.

Агентная система, с точки зрения мультиагентного решателя, состоит из следующих логических компонентов, каждый из которых представляет собой набор логических возможностей:

- агентная платформа (Agent Platform, AP) – обеспечивает инфраструктуру, в которой агенты могут создаваться и функционировать. Агентная платформа функционирует на каждом вычислительном процессе;
- агент – программный объект (существует в единственном экземпляре на каком-либо вычислительном процессе), реализующий автономную функциональность. Агенты обмениваются информацией с помощью систем обмена сообщениями и событиями;
- сервис – программный объект (существует на каждом вычислительном процессе), реализующий функциональность, необходимую для использования агентам. Сервисы получают информацию непосредственно от агентов или от системы обмена событиями;
- справочник вспомогательных функций и сервисов – компонент, предоставляющий для агентов доступ к сервисам;
- система управления агентами – осуществляет управление доступом к агентам, поддерживает актуальное состояние контейнера агентов;
- система обмена сообщениями – средство обмена информацией (передачи сообщений) между агентами (в том числе и на разных вычислительных процессах);
- система обмена событиями – средство публикации и доставки событий (порождаемых агентами) в мультиагентной системе для агентов и сервисов (в том числе и на разных вычислительных процессах);
- планировщик – специальное средство, которое обеспечивает общую синхронизацию вычислительных процессов, продвижение модельного времени, активацию агентов и т. д.

Для доступа к этим средствам, основные компоненты зарегистрированы в интерпретационных средствах Opal++ и доступны, как глобальные переменные в любой программе:

- AMS – система управления агентами (Agent Management System);
- SCHEDULER – планировщик (Scheduler);
- DF – справочник вспомогательных сервисов (Directory Facilitator);
- MTS – сервис обмена сообщениями (Message Transport Service);
- ETS – сервис обмена событиями (Event Transport Service).

Также зарегистрированы и доступны для расширения через наследование необходимые для разработки функционала агентов и средств их взаимодействия классы библиотеки базовых сущностей модуля мультиагентного моделирования:

- BaseAgent – класс базового агента, обеспечивающий базовые средства функционирования агента в мультиагентной среде решателя;
- Component – базовый элемент для расширения функциональности агентов;
- Behaviour – базовый объект, который реализует вызовы поведенческой логики агентов;
- BaseService – класс базового сервиса, обеспечивающего базовые средства для предоставления различных предметных сервисов агентам в мультиагентной среде;
- BaseEvent – класс базового события, с помощью которого агенты могу обмениваться информацией;
- BaseMessage – класс базового сообщения, с помощью которого агенты могу обмениваться информацией.

Для ведения разработки в едином программном пространстве средства Opal++ были интегрированы в программный пакет Оптимус.Препост (см. рис. 3).

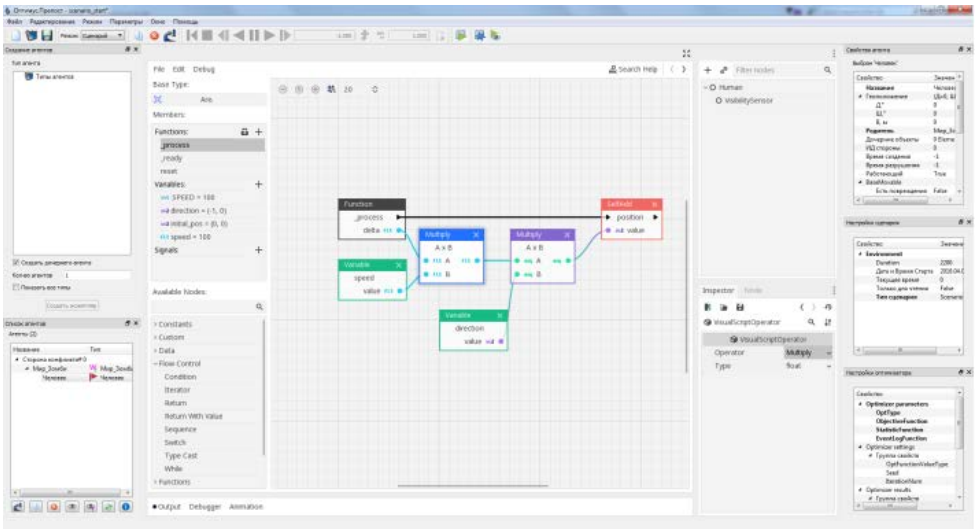


Рис. 3. Интеграция средств OPAL++ в ГПИ ВИП «ОптИМУС»

Основными визуальными интерфейсами Opal++ являются:
- редактор текстовых скриптов;
- визуальный редактор;
- конструктор агентов;
- вспомогательные элементы управления.

Редактор текстовых скриптов обеспечивает написание программ на специализированном скриптовом языке, динамически типизированном с опциональной статической типизацией (см. рис. 4). Синтаксис приближен к Python (используются отступы для выделения структурных блоков). Текстовый редактор поддерживает подсветку синтаксиса и предметно-ориентированные возможности дополнения кода.

```
# A file is a class!
# Inheritance
extends UniversalOpalAgent

# variables
var a = 5
var s = "Hello Optimus"
var arr = [1, 2, 3]
var dict = {"key": "value", 2:3}

# Constants
const ANSWER = 42
const THE_NAME = "Charly"

# Enums
enum {UNIT_NEUTRAL, UNIT_ENEMY, UNIT_ALLY}
enum Named {THING_1, THING_2, ANOTHER_THING = -1}
```

```
# Built-in vector types
var v2 = Vector2(1, 2)
var v3 = Vector3(1, 2, 3)

# function
func some_function(param1, param2):
    var local_var = 5
    if param1 < local_var:
        print(param1)
    elif param2 > 5:
        print(param2)
    else:
        print("Fail!")
    for i in range(20):
        print(i)

    while param2 != 0:
        param2 -= 1
```

Рис. 4. Пример кода на текстовом скриптовом языке

Визуальный редактор обеспечивает редактирование визуального кода и высокоуровневых поведенческих схем с различным графовым синтаксисом и семантикой. Диаграмма с визуальным программным кодом (см. рис. 5) состоит из функциональных блоков, имеющих входные и выходные порты, а также настраиваемые атрибуты. Порты соединены связями различных видов: одни определяют порядок выполнения функциональных блоков и задают поток управления, другие – передачу данных (порядок выполнения определяется зависимостями по данным). Реализован большой набор готовых функциональных блоков, в том числе для управления потоком выполнения, математических операций, вставки фрагментов текстовых скриптовых программ, создания и уничтожения объектов,

взаимодействия объектов, доступ к методам и свойствам объектов. Реализована возможность определения новых функциональных блоков различными способами.

Одной из больших проблем визуального программирования является борьба со сложностью (программы становятся слишком большими для восприятия в виде схем). Данная проблема решается путем иерархической декомпозиции (выделение части кода в отдельный функциональный блок).

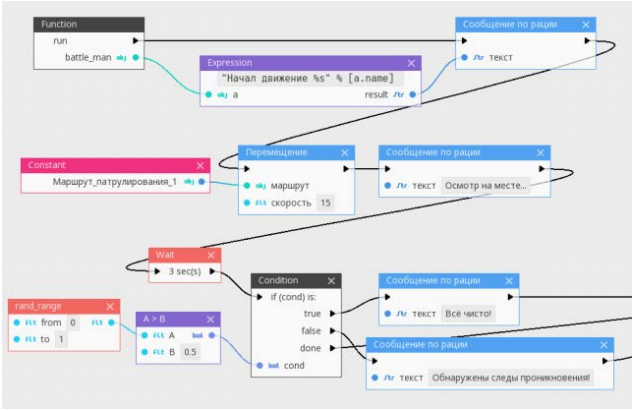


Рис. 5. Пример визуального кода

Процесс редактирования визуального кода включает в себя добавление новых блоков, соединение блоков связями, редактирование свойств блоков. Для быстрого добавления функциональных блоков из предметных библиотек присутствует поиск по нечеткому совпадению названия сущности и за счет перетаскивания визуальных сущностей (агентов, свойств, типов) из других окон ГПИ Оптимус.Препост.

Реализация функциональной и поведенческой логики агентов не может обойтись без взаимодействия агентов и составляющих их объектов между собой. Opal++ поддерживает весь спектр способов взаимодействия, доступный в рамках мультиагентного решателя:

− синхронные (прямой вызов метода, сигнал-слот) и асинхронные (сообщения, события, в том числе с временной задержкой через *SCHEDULER*);

− между агентами и внутри агента между подобъектами, совместно обеспечивающими его функционирование;

− адресно (адресат явно указывается) и широковещательно (адресат сам подписывается).

Для высокоуровневого описания поведения агента в Opal++ задаются способности агента (см. рис. 6) в виде доступных для вызова действий. Действия могут быть мгновенными, протяженными во времени с событием на завершение или бесконечной длительности. Некоторые заданные действия могут комбинироваться и запускаться на выполнение одновременно. Действие может требовать входные параметры, иметь результат успешности выполнения и выходные периметры, иметь предусловия доступности и ограничения на комбинирование с другими действиями, быть связанным с механизмами представления знаний интеллектуальных агентов.

Реализация способностей может осуществляться на любом из заявленных уровней абстракции. Например, элементарные способности могут быть реализованы в низкоуровневом C++ коде, более сложные в виде текстовых скриптов, а высокоуровневые – средствами визуального программирования.

При внедрении в ВИП «ОптИМУС» новых технологий ИИ, возможно их задействование при реализации различных способностей агентов.

Комбинирование этих подходов и технологий обеспечивает иерархическую декомпозицию и комбинирование различных способов описания логики простых и, в перспективе, интеллектуальных агентов, в рамках единого подхода на базе общего набора переиспользуемых сущностей.

Одной из задач создания собственного языка программирования агентов Opal++ в составе ВИП «ОптИМУС» является возможность создавать специальные предметно-ориентированные языки (ПОЯ) (в англ. литературе – Domain-specific language – DSL) (рис. 7).

Такие языки, с одной стороны, специализированы под конкретные задачи и области знаний (домены), а с другой – сохраняют полноту языка, не ограничивая программиста заданными рамками.

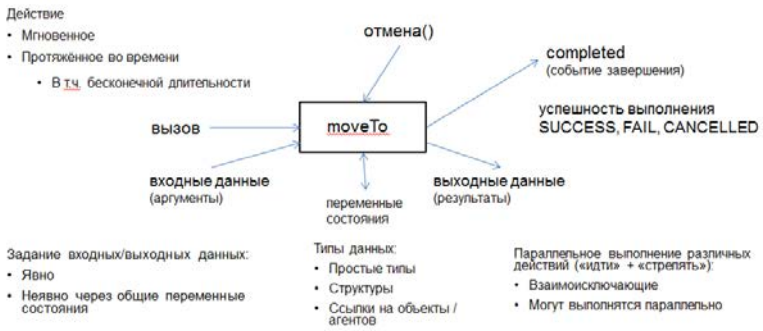


Рис. 6. Пример высокоуровневой способности агента

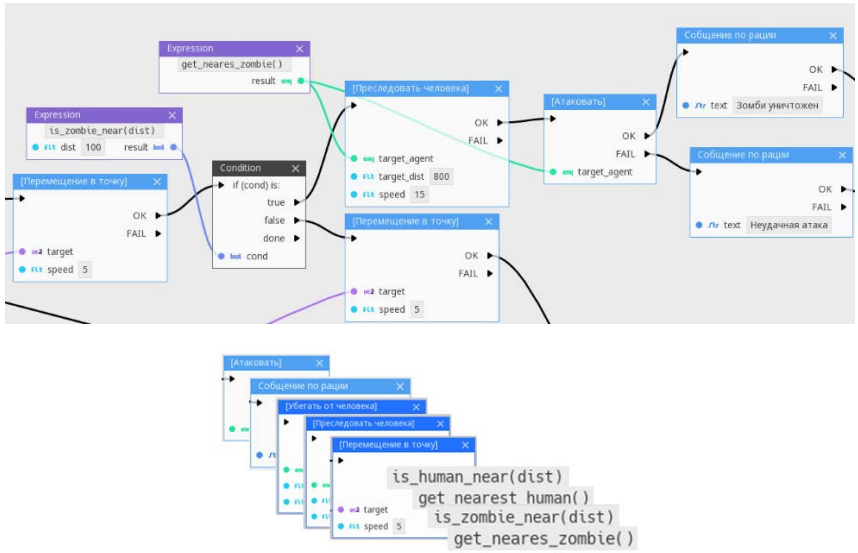


Рис. 7. Пример использования предметно-ориентированного языка (ПОЯ / DSL) для описания логики поведения агента

Использование ПОЯ в сочетании с визуальным программированием позволяет очень эффективно привлекать к процессу программирования агентов специалистов в исследуемой предметной области, а также существенно упрощает разработку, понимание и сопровождение разрабатываемых моделей агентов. Пример использования ПОЯ для описания логики поведения агента приведен на рис. 7.

Специализированный в сторону искусственного интеллекта ПОЯ может позволять внедрять блоки для интеграции технологий машинного обучения, онтологического подхода к представлению знаний, вероятностного анализа для облегчения процесса создания программной модели ИИ. Само создание и использование ПОЯ также относится к технологиям ИИ.

Для ИИ более важным является то, что пространство программ на ПОЯ имеет значительно меньшую размерность по сравнению с аналогичной программой, написанной на языке программирования более низкого уровня, что позволяет в пространстве таких структур проводить эффективный поиск оптимальных программ методами оптимизации. Для этих целей целесообразно использовать имеющиеся в составе ВИП «ОптИМУС» инструменты метаэвристической оптимизации.

Таким образом, в составе ВИП «ОптИМУС» реализован язык программирования агентов Opal++. Выполнена его интеграция с платформой на уровне пользовательского интерфейса и мультиагентного решателя. Изложены реализованные подходы и технологии программирования агентов,

показаны перспективные направления. В частности, приведены пути развития Opal++ с целью интеграции с технологиями искусственного интеллекта. Особенный интерес представляет реализация и использование:

- средств по созданию предметно-ориентированных языков;
- средств модификации в процессе функционирования поведенческой логики;
- механизмов принятия решений на основе онтологического подхода;
- средств постановки оптимизационных задач;
- процессного подхода при разработке моделей;
- средств обучения агентов (в том числе с подкреплением);
- средств нечеткологического вывода.

Литература

1. Свидетельство о государственной регистрации программы для ЭВМ №2019617157 «Визуализационно-интеграционная платформа для оптимизационного имитационного моделирования и управления системами (ВИП «ОптИМУС»)» / Коваленко О. В., Крючков И. А., Ежов Д. В., Огородников А. В., Ерошкина И. В., Собанин Д. С., Хочкин Н. И., Варгина Е. Ф., Тихомиров Ю. В., Рыжих А. В., Васильева Е. А., Кондратьев А. Б.

2. Коваленко О. В., Крючков И. А., Огородников А. В., Ежов Д. В., Собанин Д. С. Возможности визуализационно-интеграционной платформы ОптИМУС для имитационного моделирования вооружений, военной и специальной техники // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2019. Вып. 4

3. Рассел С., Норвиг П. Искусственный интеллект: современный подход, 2-е изд.: Пер. с англ. – М., 2006.

4. Rafael H. Bordini, Mehdi Dastani, Jurgen Dix, Amal El Fallah Seghrouchni. Multi-Agent Programming. Languages, Tools and Applications – Springer Science+Business Media, LLC 2009

THE *OPAL*++ AGENT PROGRAMMING LANGUAGE OF THE VISUALIZATION AND INTEGRATION PLATFORM “OPTIMUS”

D. S. Sobanin, I. A. Kryuchkov, A. V. Ogorodnikov, O. V. Kovalenko

Russian Federal Nuclear Center –
All-Russian Scientific Research Institute of Experimental Physics, Sarov

The development of agents and effective programming of their behavior logics are the key and most complex stages in the development of multiagent simulation models. The paper presents the basic approach and means of the domain-specific Opal++ agent programming language in the “OptIMUS” visualization and integration platform used to resolve the given problems.

Key words: visualization and integration platform “OptIMUS”, multiagent model, agent behavior logics, Opal++ agent programming language.

УДК 539.3
DOI 10.53403/9785951505071_2022_434

ИСПОЛЬЗОВАНИЕ МЕТОДОВ ГАМИЛЬТОНОВОЙ ДИНАМИКИ В ЧИСЛЕННЫХ РАСЧЕТАХ ЗАДАЧ МЕХАНИКИ СПЛОШНОЙ СРЕДЫ

В. Н. Софронов, М. В. Ветчинников, М. А Демина

Российский федеральный ядерный центр –
Всероссийский НИИ экспериментальной физики, Саров

В докладе приводится построение дискретной модели численного метода решения задач механики сплошной среды на основе дискретной гамильтоновой динамики. Для численного решения гамильтоновых уравнений используются симплектические разностные схемы. Приведены примеры моделирования бездиссипативных процессов.

Ключевые слова: дискретная гамильтонова динамика, фазовый объем, симплектические разностные схемы, динамические задачи упругости, комплекс программ.

Введение

Гамильтоновы методы, основанные на приведении уравнений к гамильтоновой форме, используются в различных разделах теоретической физики. В задачах механики сплошной среды(МСС) почти всегда присутствуют диссипативные процессы, разрушающие многие свойства гамильтоновых систем (например, сохранение фазового объема), и проблема полной интегрируемости становится почти неразрешимой. Действительно, примеры полной интегрируемости динамических систем малочисленны и является редким исключением из общих правил [1]. В данной статье построение бессеточного численного метода для МСС основано на использовании методов дискретной гамильтоновой динамики [2].

В бессеточных методах (подвижные клеточные автоматы (МСА) [3], перидинамика (PD) [4]–[5], кластерная динамика (КД) [6]) сплошная среда заменяется системой взаимодействующих мезо-частиц. Для методов этого класса не всегда удается сконструировать микроскопические взаимодействия так, чтобы получить заданные макроскопические свойства. Уравнения движения мезо-частиц обычно записаны в интегро-дифференциальной форме и не всегда устанавливается их связь с уравнениями гамильтоновой динамики. Последние обладают множеством интересных свойств. Гамильтоновы системы, кроме стандартных интегралов движения, имеют N дополнительных интегральных инвариантов (N – число частиц). Одним из таких инвариантов является фазовый объем. Естественно, что разностные методы должны наследовать все свойства дифференциальных уравнений. Можно надеяться, что в этом случае будет лучше сохранена структура решения в фазовом пространстве.

Построение дискретной модели, представленной в статье, начинается с задания упругого потенциала (упругой энергии деформирования) системы частиц. Для изотропной среды эта энергия зависит от двух модулей упругости. Таким образом, мы получаем гамильтониан системы с разделяющимися переменными (координатами частиц и их импульсами). При построении дискретной модели используется процедура вычисления тензора дисторсии для системы частиц [7]. В исходном виде модель не предполагает наличия диссипативных процессов и может использоваться только для решения ограниченного круга задач (акустики, сейсмики, динамики гипо-упругих сред). Учет диссипации возможен в рамках диссипативной динамики частиц (dissipative particle dynamics, DPD) [8].