Литература:

1. Красовский, О. Г., Берман А. А., Матвеев В. В. Применение программ численного моделирования рабочего процесса дизелей // Труды ЦНИДИ «ЭВМ в исследовании и проектировании двигателей внутреннего сгорания». — Л.: ЦНИДИ, 1986. — С. 100–111.
2. Гончар, Б. М. Применение ЭВМ в дизелестроении // Труды ЦНИДИ «Дизелестроение». — Л.: Машиностроение, 1974. — С. 187–192.
3. Кулешов, А. С., Грехов Л. В. Математическое моделирование и компьютерная оптимизация топливоподачи и рабочих процессов двигателей внутреннего сгорания. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2000. — 64 с.
4. Хрящёв, Ю. В., Блаженнов Е. И. Электронное управление работой автомобильных двигателей: Учеб. пособие. — Ярославль: ЯПИ, 1990. — 92 с.
5. Дизели. Справочник / Байков Б. П., Ваншейдт В. А., Воронов И. П. и др. Под ред. Ваншейдта В. А., Иванченко Н. Н., Коллерова Л. К. — Л.: Машиностроение, 1977. — 480 с.
6. Звонов, В. А. Образование загрязнений в процессах сгорания. — Луганск: Изд-во Восточноукраинского государственного университета, 1998. — 126 с.

Методологии проектирования мультиагентных систем

Зубарева Мария Георгиевна, магистрант;
Цветков Александр Андреевич, магистрант;
Хамуш Анис Ленин, магистрант;
Шорох Данила Кириллович, магистрант;
Шуклин Алексей Владимирович, магистрант;
Юрсков Сергей Валерьевич, магистрант
Московский государственный технический университет имени Н. Э. Баумана

В данной статье рассмотрены методологии проектирования мультиагентных систем, в том числе, применительно к созданию искусственного интеллекта. Приведен анализ популярных подходов к разработке информационных систем на всех этапах создания.

Ключевые слова: *мультиагентная система, объектно-ориентированный подход, искусственный интеллект*

В основе мультиагентного подхода лежит понятие программного агента, который реализован и функционирует как самостоятельная элемент программного обеспечения или элемент искусственного интеллекта.

Суть мультиагентных технологий заключается в принципиально новом методе решения задач. В отличие от классического способа, когда проводится поиск некоторого четко определенного (детерминированного) алгоритма, позволяющего найти наилучшее решение проблемы, в мультиагентных технологиях решение получается автоматически в результате взаимодействия множества самостоятельных целенаправленных программных модулей — агентов. В данной статье были рассмотрены некоторые популярные способы построения мультиагентных систем.

Методология создания МАС на основе онтологии

Создание онтологии для конкретной бизнес задачи имеет преимущества для разработки, совместимости, возможности многократного использования и интеграции разрабатываемой системы. МАС в совместимости с онтологией могут использовать эти преимущества, предоставляя возможность создавать семантически-совместимые приложения. Данный подход фокусируется на развитии онтологии в качестве движущей силы развития системы и предполагает разработку приложения, в большей части, при помощи экспертов в предметной области, требующую минимального участия разработчиков программного обеспечения. Существует два подхода:

1. Поэтапная итеративная методология;
2. Конвейер избранных инструментов, с помощью которых осуществляется методология.

Описание методологии

Онтология представляет собой спецификацию объектов, понятий, сущности и отношения между объектами в определенной области. В качестве альтернативы, онтология определяется как формальная, явная спецификация общей концептуализации.

В контексте мультиагентных систем использование онтологии способствует анализу и проектированию этапов разработки системы, развитию интеллекта отдельных агентов, связи и взаимодействию между агентами во время работы, а также возможности взаимодействия агентов с другими системными компонентами. Полностью такой подход звучит как методология развития МАС с онтологией (eng. MOMA).

Конструктивными целями MOMA являются:

1. Разработка предметно-ориентированных онтологий для повторного использования и обмен информацией между приложениями в определенной области;
2. Перемещение бизнес логики и знаний предметной области из базового кода агента на более высокий уровень развития онтологии;
3. Содействие использованию инструментов для ускорения разработки приложения с использованием онтологии;
4. Отделение эксперта в предметной области от разработчика агента;

Существуют еще четыре методологии разработки MAS, также включающие в себя понятия онтологии, хотя и не в полной мере, как это используется в MOMA. Этими методологиями являются:

- MAS-CommonKADS;
- MESSAGE;
- Multi-agent Systems Engineering (MaSE);
- Процесс спецификации и реализации объединений агентов (PASSI).

MOMA состоит из двух основных этапов развития: разработки онтологии и разработки агента (рисунок1).

Во время первого этапа знания предметной области моделируются в виде онтологии таким образом, чтобы они не были определены в коде низкого уровня в процессе разработки агента. Результирующая онтология используется в качестве основы для второго этапа, который включает в себя реализацию агентов и среды приложения для конкретного приложения.

Преимущества и недостатки методологии

Плюсы:

При использовании определенных инструментов возможно ускорить процесс разработки приложения с использованием онтологии.

За счет перемещения бизнес логики и знаний предметной области из базового кода агента на более высокий уровень развития онтологии возможно отделение эксперта предметной области от разработчика агентов.

Минусы:

Определение понятий и отношений при помощи стандартных высокоуровневых онтологий для моделирования своей онтологии является трудным процессом и отнимает много времени (для решения данной проблемы используется инструмент GT). Также на данном этапе можно упустить некоторые понятия (необходимо иметь несколько итераций в разработки онтологии)

Во время настройки онтологии для тестирования невозможно использовать слишком сложную логику (для использования более сложной логики необходимо определить больше понятий в редакторе Protege).

Методология разработки мультиагентной системы ASEME

ASEME (Agent Systems Engineering Methodology) — это методология проектирования агентных систем, которая заключается в объединении нескольких подохов или их элементов к проектированию подобных систем, доказавших свою применимость и эффективность на практике. Данная методология основывается на модельно-ориентированном подходе. что на практике означает, что модели предыдущего этапа разработки применяются в последующем этапе, но с определенными видоизменениями.

Основное преимущество данной методологии заключается в том, что она основана на уже существующих языках (например, UML и диаграмма состояний), которые хо-

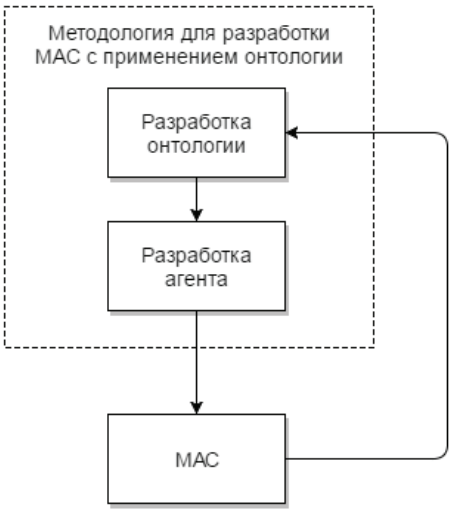

Рис. 1. Этапы развития MOMA

рошо знакомы инженерам в рамках задач проектирования моделей и в процессе проведения системного анализа. Данная методология предусматривает три различных уровня абстракции, что позволяет использовать различные технологии для проектирования высоконагруженных систем. В данной методологии не используются конкретные архитектуры агентов и их параметры, что делает эту методологию агентно-независимой и позволяет гибко выбирать архитектуру каждого конкретного агента, основываясь на требованиях в той или иной предметной области. Методология состоит из нескольких этапов. На каждом этапе разработки происходит адаптация агента согласно определенным новым правилам. Это происходит путем более детализированного процесса сбора и формирования требований к результату предыдущего этапа. Такой итерационный подход позволяет добавлять новые детали на очередном этапе и приводит в результате к точному описанию системы (состояний, жизненного цикла данных и т.д.). Важно отметить, что каждый очередной этап разработки может быть выполнен на новой платформе, что делает данную методологию платформо-независимой.

Этапы создания МАС

Общий подход к разработке МАС представлен на рисунке 2. В данной методологии присутствует 6 различных этапов: первые 4 этапа являются этапами создания системных моделей (фаза разработки), последние два этапа (оптимизация и верификация) оценивают и оптимизируют полученные модели. В общем случае данный подход к разработке является итеративным и, существует возможность возвращаться на любую предыдущую фазу согласно модельно-ориентированному подходу (МОП), применяемому в данной методологии.

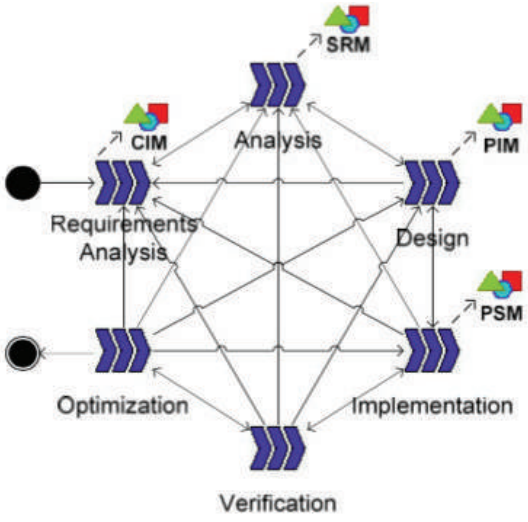

Рис. 2. Общий подход к разработке МАС

МОП заключается в систематическом использовании моделей как главных артефактов во время всего цикла разработки. МОП совместима с модельно-ориентированной архитектурной парадигмой разработки, созданной организацией OMG (Object Management Group). Данная парадигма описывает 3 основные модели:

– Вычислительно-независимая модель (Computation Independent Model, CIM) описывает общие требования к системе, словарь используемых понятий и условия функционирования (окружение). Поэтому CIM рекомендуется выполнять с использованием унифицированного языка моделирования UML.

– Платформенно-независимая модель (Platform Independent Model, PIM) описывает состав, структуру, функционал системы. Модель PIM создается на основе CIM.

– Платформенно-зависимая модель (Platform Specific Model, PSM) описывает состав, структуру, функционал системы применительно к вопросам ее реализации на конкретной платформе.

Фаза анализа требований

Три базовые действия являются примером трех уровней абстракции, о которых было сказано выше: определение действующих лиц, определение целей, определение требований для достижения целей. Входными данными на данном этапе является документ с требованиями, а выходными — список требований для достижения конкретной цели и диаграма действующих лиц.

Фаза анализа

В качестве входных данных используется диаграмма действующих лиц, которая преобразуется в диаграмму вариантов использования, а благодаря документу с требованиями, полученному на предыдущем этапе, на выходе мы получаем также ролевую модель и описание протокола взаимодействия моделей.

Фаза проектирования

Используя результаты двух предыдущим фаз происходит формальное проектирование структуры агентов и результатами фазы являются: диаграмма межагентного

взаимодействия, внутриагентного взаимодействия, процедурные или декларативные знания, список доступных типов сообщений, онтология и таблица функциональности.

Фаза реализации

Фаза реализации заключается в имплементировани агентов на основе выходных документов каждой из предыдущих фаз.

Фазы верификации и оптимизации

Фаза верификации заключается в формальной проверке на соответствие полученной системы требованиям, предъявляемых при начале разработки. Фаза оптимизации заключается в улучшении работоспособности (по временным параметрам, а также с учетом используемых системных ресурсов (нагрузка на сеть, память и т.д.).

Достоинства и недостатки методологии

Данная методология уже доказала свою эффективность в задачах проектиорования сложных систем. Основными достоинствами системы являются:

- модельно-ориентированных подход
- агенто-независимость
- платформо-независимость
- полное соответствие полученного результата функциональным требованиям, предъявляемых к системе.

Из недостатков стоит выделить дополнительную сложность проектирования МАС с использованием данного подхода.

Методология MASITS

MASITS (Мультиагентная система для интеллектуальных систем передачи) является методологией полного жизненного цикла для агентно-ориентированной разработки интеллектуальных систем передачи. Методология MASITS включает в себя наиболее важные результаты исследований в области развития ITS и AOSE методологий. Наиболее подходящие методы из существующих методик AOSE используются на стадиях, когда существующие методики позволяют интегрировать конкретные знания для развития ИТС. Кроме того, новые методы введены в шагах, где известные методы не позволяют интегрировать свои знания.

Процесс развития ITS состоит из следующих этапов: анализ, проектирование (внешний и внутренний дизайн), внедрение, тестирование, развертывание и обслуживание. Эти фазы представлены последовательно, хотя процесс разработки является итеративным. Итерации используются как внутри фазы, так и в разных фазах.

Этапы создания МАС

Фаза анализа

Первым шагом является моделирование цели, в результате диаграмма целей отображает иерархию целей системы. В первую очередь определяются глобальные цели на уровне системы. Затем каждая цель разбивается на подцели с помощью разложение на «И« и «ИЛИ». Эти цели разбиваются до тех пор, пока не смогут быть достигнуты с помощью простых действий.

На втором этапе создается диаграмма использования системы. В ходе моделирования первоначальные акторы определяются путем указания действующих лиц, которые необходимы для достижения целей.

Внешнее строение агента

Фаза разработки для интеллектуальной транспортной системы может быть разделена на 2 стадии — внутренний и внешний дизайн агентов. На протяжении первой фазы (внешнего дизайна) агенты разрабатываются с точки зрения функциональных возможностей и взаимодействий. Даная фаза состоит из таких этапов как: определение задачи, распределение задач агентам, дизайн взаимодействия и дизайн онтологии.

После определения задач и их назначения должна быть определена модель взаимодействий. Схема взаимодействия состоит из агентов, акторов (пользователей) и связей между ними.

Внутреннее строение агента

В ходе дизайна внутреннего строения агента, должна быть определена внутренняя структура агентов. Этот этап должен дать ответы на вопрос, как агенты ведут себя для достижения поставленных целей.

Выполнение, тестирование, развертывание и обслуживание

Холоническая архитектура мультиагента поддерживается MASITS и состоит из мельчайших агентов которые могут быть использованы повторно с подобным функционалом.

На первом этапе мелкомасштабные агенты используются повторно из предыдущих проектов. Второй шаг заключается в генерации кода, используя инструменты MASITS для агентов, которые не могут быть использованы повторно из созданных ранее систем.

В третьей стадии разработчик должен выполнить действия из поведенческих классов, сгенерированных системой и создать интерфейс пользователя системы.

Для фазы тестирования используются JADE тесты и происходит деление на 3 этапа. Во-первых, отдельные агенты тестируют путем создания группы тестов для каждого агента. Во-вторых, каждый холон проходит испытания в качестве составной сущности, то есть в качестве агента. В-третьих, тестируют функциональность системы в целом с использованием традиционных методов тестирования.

Развертывание выполняется с использованием модифицированной версии схемы развертывания UML, который показывает JADE контейнеры и экземпляры агентов, развернутых в каждом контейнере. Диаграмма развертывания используется MASITS, чтобы сформировать пакетный файл.

MASITS поддерживает внедрение изменений в систему, во время фазы технического обслуживания.

Преимущества и недостатки

Преимущества:

– Изменения в системе на подобие изменения функциональных возможностей, соответствующих открытому холону или изменения внутри одного холона, или изменения на уровень выше холона могут реализованы в MASITS, но не могут быть реализованы в ITS (только при переработке системы полностью).

– В методологии используются разработки ITS, путем интегрирования в отдельные шаги, чтобы облегчить различные этапы развития.

– Поддерживается весь жизненный цикл, включая тестирование, внедрение, развертывание и обслуживание фаз, которые слабо поддерживаются в большинстве методологий.

– Обеспечиваются соответствующие методы для развития системы в течение всего жизненного цикла.

Недостатки

– Основным ограничением методики является задание конкретной цели. Для того, чтобы применить к разработке других систем её придется изменять или использовать какую-либо другую платформу, нежели JADE.

Методология X-machine

В данном подходе используется формальный метод, а именно метод «X-machine», который удовлетворяет потребностям развивающихся агентов и позволяет использовать их на практике. Здесь представлена формализированная методология для разработки систем, основанных на реактивных агентах. Агенты, а также все необходимые расширения модели, представляются формальным образом, которые, в свою очередь, оптимизируют агентную систему. Предложенная методика использует априорные методы (формального моделирования и верификации), чтобы избежать дефектов на ранних стадиях разработки совместно с апостериорными методами (стратегия тестирования черного ящика) чтобы обнаружить любые неоткрытые недостатки в более поздних стадиях.

Взаимодействие X-машин рассматривается как метод моделирования, в котором сложная система может быть представлена в виде малых элементов (агентов) и смоделирована используя простые модели X-машин.

Связи всех этих агентов задаются отдельно таким образом, чтобы можно было рассмотреть полную систему в виде сообщающихся X-машин. Такая модель предполагает модульный подход «снизу вверх» и поддерживает итеративное постепенное развитие. Это также облегчает возможность многократного использования существующих моделей данного типа, что делает управление всем проектом более гибким и эффективным, достигая его завершения с меньшей стоимостью и меньшим временем разработки.

Шаги создания MAC

Связи метода X-машина предполагает формализированный подход к модульной разработке, позволяя разработчикам разделять систему на стадии на связанных агентов и, таким образом, моделировать взаимодействующих систем на основе агентов. Алгоритм разработки модели системы можно представить в следующих четко определенных различных действиях, которые представлены далее:

1) Разработка моделей типа X-машина (типов агентов X-машины)

Разработка идет независимо от целевой системы, в том числе, можно использовать существующие стандартные типы модели

2) Кодирование модели X-машина

Кодирование модели типа X-машина на языке разработки облегчает последующие шаги. Можно использовать аниматора, что сопровождающего язык разработки что бы получить обратную связь (неофициальная проверка).

3) Выражение желаемых свойств

Необходимо выразить желаемые свойства в подходящем формате(Временная логика) и использовать формальный метод проверки (Проверка модели) чтобы убедиться, что модели соответствуют требованиям.

4) Тестирование

Использование стратегий тестирования с целью проверки осуществления заданных целей. Модульное тестирование, при котором модулем считается блок агентного типа

5) Создание агентов X-машины

Создание агентов X-машины и определение способов их взаимодействия между собой.

6) Расширение модели для достижения требуемой функциональности.

Достоинства:

- Реактивность;
- Ситуационный подход;
- Гибкость;
- Расширяемость
- Высокое качество

Недостатки:

- Глубокое понимание предметной области разработки
- Жесткая формализация
- Сложность формализации требований

Выводы

В данной статье были рассмотрены методологии проектирования мультиагентных систем, в том числе, применительно к созданию искусственного интеллекта. Приведен анализ популярных подходов к разработке информационных систем на всех этапах создания.

Литература:

1. Nikolaos Spanoudakis, Pavlos Moraitis. Agent Systems Engineering Methodology: The Development Process
2. Egons Lavendelis. MASITS — A MULTI-AGENT BASED INTELLIGENT TUTORING SYSTEM DEVELOPMENT METHODOLOGY
3. Zambonelli, F., et al., 2005. Multi-Agent Systems as Computational Organisations: The Gaia Methodology. Agent-
4. Oriented Methodologies, (Eds. Henderson-Sellers B., Giorgini P.) Idea Group Publishing, London.
5. V. Julian and V. Botti. Developing Real-Time Multi-Agent Systems
6. DeLoach, S., 2001. Analysis and Design Using MaSE and agentTool. Proceedings of the 12th Midwest Artificial
7. Intelligence and Cognitive Science Conference, Oxford OH, March 31 — April 1 2001
8. Petros Kefalas, George Eleftherakis, Evangelos Kehris. A methodology for developing component-based agent systems focusing on component quality

Реализация алгоритма поиска ближайших объектов с помощью K-D tree

Омаров Марат Булатович, студент
Омский государственный университет имени Ф.М. Достоевского

В данной статье разработан алгоритм поиска ближайших объектов с помощью вспомогательной структуры, основанной на K-D tree, а также рассматривается приложение на языке Java, реализующее данный алгоритм.

Ключевые слова: *K-D tree, поиск объектов*

В современных информационных системах часто появляется необходимость найти объекты в определенной области. Ключевое значение при реализации такой возможности имеют используемые структуры данных и алгоритмы их обработки.

Постановка задачи

В системе будут храниться следующие географические данные: широта, долгота, название объекта.

```
public class Geo {
        private final double latitude;
        private final double longitude;
        private final String title;
}
```

Листинг 1. Структура объекта