

Мульти-агентная архитектура менеджеров и агентов(исполнителей).

Процик Петр Павлович
Киевский Национальный Университет
им. Тараса Шевченко,
факультет кибернетики,
pprocik@unicyb.kiev.ua

Аннотация

Мульти-агентные системы (МАС) представляют собой новый виток в развитии программных архитектур, в особенности архитектур распределённых программных систем. В работе рассматривается одна из возможных реализаций такой архитектуры. В качестве платформы программирования используется Microsoft .NET.

Введение

Мульти-агентные системы (МАС) представляют собой новый уровень в развитии программных архитектур, в особенности архитектур распределённых программных систем.

Прообразом мульти-агентных архитектур были модели биологических экосистем и экономические (рыночные) модели. Характеристиками таких систем являются децентрализованность источников информации, отсутствие глобального управления, асинхронность и отсутствие полной информации о возможности достижения целей.

Такие структуры позволяют проводить качественно иной способ анализа, проектирования и реализации сложных программных систем.

Область исследований МАС затрагивает системы, в которых много агентов взаимодействует друг с другом. Агентами считают независимые сущности, такие как специальные программы, роботы, сенсорные устройства. Их взаимодействие может основываться как на сотрудничестве, так и на конкуренции. Поскольку агенты могут разделять общие цели (как, например, в колонии муравьёв), либо иметь свои собственные интересы (как в свободной рыночной экономике).

Теоретические вопросы МАС сосредоточены на проблемах уточнения понятия агента, организации взаимодействия при помощи специальных протоколов и языков, координации действий агентов.

В работе описывается распределённая мульти-агентная архитектура и способ её технической реализации на платформе Microsoft .NET.

В предложенной архитектуре рассматриваются агенты двух типов: агенты-менеджеры и агенты-исполнители. Управление назначением заданий агентам-исполнителям полностью возложено на агентов-менеджеров. В процессе работы агенты не могут меняться ролями (в отличие от взаимодействия основанного на CNP, см. ниже).

Описанная архитектура была использована при построении системы для сбора и анализа информации из различных источников разнородной природы и частоты обновления.

Задача агентов-менеджеров состояла в поддержании информационной базы данных системы в соответствии с актуальным состоянием источников данных.

Основные положения

Существуют различные способы определения понятия агент – более или менее полные. Рассмотрим некоторые из них:

(1) *Агент* – это абстрактная или физически автономная сущность, которая выполняет поставленные задачи, используя информацию из своего окружения. Агент должен иметь механизмы адаптации к изменениям в окружении, и адекватно на них реагировать (выдавать ожидаемый результат).

(2) *Агент* – это устойчивая программная сущность, предназначенная для достижения конкретных целей [Smith, Cypher and Spohrer].

(3) *Интеллектуальные агенты* – это программные сущности, которые могут выполнять некоторые операции на стороне пользователя или другой программы с некоторой степенью независимости или автономии, применяя в процессе работы некоторые знания или представления о желаниях или целях пользователя [IBM White paper, 1995].

В оригинале:

Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires.

Если не будет указано другое, далее в работе понятия агент и интеллектуальный агент будут пониматься в смысле определений (2) и (3) соответственно, т.е. рассматриваться будут программные агенты.

Кроме того, агент должен обладать следующими свойствами:

- **Автономность:** агенты должны работать без прямого вмешательства человека (оператора) и иметь механизмы управления своим внутренним состоянием;
- **Коммуникабельность:** агенты могут обмениваться между собой сообщениями;
- **Реактивность:** агенты воспринимают своё окружение и реагируют на возникающие в нем изменения;
- **Активность:** агенты не просто действуют в ответ на изменения в среде, они также могут проявлять собственную целенаправленную активность. Агент способен решать задачи высокого уровня сложности. Способ разбиения задачи на подзадачи и порядок их решения должен выбирать агент;
- **Временная непрерывность:** агенты – непрерывные работающие процессы;
- **Мобильность:** способность агента транспортировать себя с одного вычислительного узла на другой, сохраняя своё текущее состояние.

Вычислительная структура агента должна содержать:

- Статические знания о себе и о других агентах;
- Экспертные знания - представляют действия, которые агент может предпринять, и которые могут быть описаны в различных формах (порождающие правила, фреймы, логические выражения);
- Средства связи - представляют собой коммуникационные протоколы между агентами;
- Стратегии взаимодействия агентов между собой.

Возможны различные классификации агентов по: поведению, задачам, вычислительной модели, логической парадигме и др.

Примем такую классификацию, основываясь на задачах, выполняемых агентом:

- интерфейсные агенты;
- личные информационные ассистенты;
- добытчики данных;
- информационные фильтры;
- планировщики.

Мульти-агентная система (МАС) – это система, состоящая из коллекции автономных агентов, которые обмениваются между собой информацией для достижения общих целей, в то же время каждый отдельный агент преследует собственные цели.

Координация действий и организация взаимодействия между агентами – ключевые проблемы в теории и технологии МАС.

Существуют различные подходы к решению указанных проблем. Наиболее известный подход к координации взаимодействия основан на Contract-Net Protocol (CNP). В этом случае прототипом служит децентрализованная рыночная структура.

Агенты могут принимать одну из двух ролей – менеджера либо исполнителя (contractor).

Основная предпосылка этой формы координации в том, что если агент не может выполнить, назначенную ему задачу, используя локальные (собственные) ресурсы, то он разбивает задачу на подзадачи и пытается найти свободного агента(ов) способного выполнить образованные подзадачи. Процедура назначения подзадач решается при помощи механизма контрактов. Этот механизм состоит из анонса контракта (подзадачи), который публикует агент-менеджер, сбора предложений от агентов способных выполнить данный контракт и оценки предложений менеджером, которая ведёт к заключению контракта с наиболее приемлемым агентом-исполнителем.

Коммуникация, т.е. обмен информацией, между агентами наиболее часто организуется пересылкой сообщений. Хороший пример языка, который может при этом использоваться – KQML (Knowledge Query and Manipulation Language) – язык запросов и манипуляции знаниями.

Подходя, с практической точки зрения к вопросу реализации МАС исключительно на платформе .NET для организации взаимодействия между агентами используются средства, предоставляемые выбранной платформой.

Координация действий агентов будет происходить в соответствии с фиксированным расписанием и доступными ресурсами. В этом случае предполагается заранее известная частота обновления каждого из источников данных.

За обработку информации из каждого конкретного источника данных в данный момент времени должен отвечать только один действующий в системе агент.

Описание архитектуры

В этом разделе будет описана реализованная мульти-агентная архитектура.

Предложенная архитектура содержит два предопределённых типа агентов – агенты менеджеры и агенты исполнители. Далее в разделе для определения этих типов агентов будут употребляться термины «менеджер» и «агент» соответственно.

Коротко опишем каждый из типов агентов. В следующем разделе будет дана более полная и более формальная спецификация.

В задачи менеджеров входит организация координации действий агентов, обмен информационными сообщениями с другими менеджерами, а также обмен существующими агентами. Совокупность менеджеров представляет собой распределённую сеть, в

которой возможна передача информации между любыми двумя её составляющими. Поскольку предполагается заранее известным частота возникновения задач, которые должны выполнить агенты, то в обязанности менеджеров входит также распределение заданий для агентов и обмен (копирование) между собой агентов необходимых для выполнения задачи. Распределение агентов проводится так, чтобы задействовать как можно меньше менеджеров для выполнения одной задачи. Поэтому в начальной конфигурации только один из менеджеров имеет доступных к запуску агентов. В процессе работы, если возникнет такая необходимость, будет произведена передача (физическое копирование) агента другим менеджерам в сети. Таким образом достигается свойство мобильности агентов-исполнителей. Если один агент доступен нескольким менеджерам, то необходим механизм, разрешающий неоднозначность с его выполнением (так как один агент решает только одну задачу). Таким механизмом будет система приоритетов и заявлений. Менеджер с наивысшим приоритетом для данного агента имеет преимущество. Система начисления приоритетов и публикации заявлений будет рассмотрена в следующем разделе.

При работе системы менеджеры являются статическими агентами. То есть их замена на более новую версию возможна только при перезапуске системы. В то время как агенты-исполнители могут быть динамически добавлены в систему. Менеджер должен выполнять агентов в порядке приоритета версий от самой новой к самой старой при возникновении исключительной ситуации либо состояния ошибки. Если один агент не справился с поставленной задачей, менеджер должен задействовать других доступных агентов способных её решить. Агенты в системе – это независимая программная единица, реализованная в виде библиотеки классов (class library) в соответствии с шаблоном и помещённая в сборку. Каждый агент должен отвечать за решение одной функциональной задачи. После окончания своей работы агент должен возвращать вызвавшему его менеджеру код своего завершения (работа выполнена успешно, нехватка ресурсов, ошибка выполнения и т.д.). Кроме этого агент должен содержать информацию о своей функциональности, требованиям к ресурсам, версии и текущем состоянии для того, чтобы менеджеры могли правильно скорректировать работу системы.

Информация о состоянии агента будет инкапсулирована в нём. Доступ будет осуществляться через специальный предусмотренный интерфейс.

В качестве внешнего окружения (environment) в системе должна функционировать глобальная БД. Глобальность в данном случае означает, что доступ к ней может получить любой агент, менеджер или иная часть системы. БД

будет служить средством хранения текущего состояния системы, средством хранения опубликованных заявлений менеджеров, а также информационной средой, с которой работают агенты для сбора, хранения и обновления информации.

Спецификация архитектуры агентов и менеджеров

В этом разделе приводится спецификация основных компонентов архитектуры агентов и менеджеров – агентов, менеджеров, протоколов взаимодействия, информационной базы данных. Спецификация будет учитывать особенности и принятую терминологию платформы Microsoft .NET и операционной системы Microsoft Windows.

Спецификация агента-исполнителя (далее агента):

- Для обеспечения совместимости все вспомогательные и базовые типы собраны в сборке (BaseAgent.dll);
- Агент должен быть прямым наследником виртуального(абстрактного) класса BaseAgent.Agent, вся работа выполняемая агентом должна быть инкапсулированной в методе DoWork;
- Метод DoWork должен быть потокобезопасным (thread-safe). Если нужно использовать класс, не обеспечивающий безопасность потоков в многопоточковой среде, следует поместить его в класс-оболочку, обеспечивающий необходимые конструкции синхронизации;
- Версия, тип агента, необходимые для работы ресурсы и другая информация об агенте должна содержаться в атрибутах соответствующего типа;
- Одна сборка может содержать только одну версию конкретного агента. Вообще, в сборке может быть несколько разных агентов;
- Агент должен иметь возможность взаимодействия с глобальной БД;
- Во время работы агент не должен выводить никакой информации на экран, консоль и напрямую взаимодействовать с пользователем. Изменять информацию агент должен только в БД. Допускается использование внешних дисковых файлов, например для ведения протокола работы агента. Для взаимодействия с клиентом должны использоваться другие средства (программы, визуальные агенты), которые модифицируют данные в БД;

- Агент должен указывать своё текущее состояние. Доступ к значению текущего состояния производится вызовом метода `BaseAgent.Agent.GetStatus()`. После выполнения метода `DoWork` в случае успеха, метод `BaseAgent.Agent.GetResult()` должен выдавать результат (код завершения) в противном случае код ошибки;
- Для получения доступа к базе данных каждый агент содержит поле `ConnectionString`. Поле должен заполнить, вызывающий агента, менеджер.

Описание базового класса агента на языке C# может выглядеть следующим образом:

```
public abstract class Agent
{
    public string ConnectionString;
    public int version;

    public abstract void DoWork();
    public abstract string SolvedTask();
    public abstract AgentStatus GetStatus();
    public abstract string GetResult();
    public Agent() { }
}
```

Спецификация агента-менеджера (далее менеджера):

- Менеджер должен быть реализован, как служебное приложение Microsoft Windows (см. глоссарий);
- Один менеджер соответствует одной рабочей станции;
- Агенты принадлежащие менеджеру должны находиться в заранее определённой физической папке (например, `windows\system32\agents`);
- Выполнение агента заключается в создании экземпляра агента (при помощи класса `BaseAgent.AgentLoader`), помещении его в очередь менеджера. Как только у менеджера появляется возможность выделения нового потока из очереди извлекается агент и создаётся отдельный поток для выполнения метода агента `DoWork`. После выполнения метода в потоке должна производиться проверка кода завершения и модифицироваться расписание;
- Расписание составляется на основании данных о частоте обновления источников информации, результатах выполнения агентов и опубликованных заявок. Все эти данные должны храниться в БД и не должны быть доступны агентам-исполнителям;
- Агенты должны представлять собой распределённую сеть. Маршрутизация производится в

- соответствии с таблицей маршрутизации, помещённой в БД;
- Разрешение неоднозначности (т.е. ситуации когда одного агента могут выполнить несколько менеджеров) производится в соответствии с «алгоритмом разрешения» описанным в конце раздела;
- «Алгоритм разрешения» вступает в действие каждый раз когда менеджер намеревается выполнить агента для решения существующей задачи;
- Каждый менеджер должен иметь приоритет.
- Приоритет менеджера – число большее 0. Чем число больше тем приоритет меньше;
- Обращение менеджера к информационной среде – это обращение к глобальной БД. Обращение к БД зависит от реализации и не специфицируется;
- Перед началом функционирования менеджер должен зарегистрироваться в информационной среде. Менеджер, зарегистрировавшийся первым получает наивысший приоритет - 1, вторым – 2 и т.д. После завершения работы менеджер должен выполнить операцию обратную регистрации;
- Регистрация менеджера в среде означает помещение в неё адреса по которому можно связаться с менеджером, получение приоритета, публикации доступных менеджеру агентов, публикация доступных менеджеру ресурсов;
- На основании опубликованных в глобальной среде данных менеджеры могут обмениваться агентами;
- Среда должна содержать всех имеющихся у менеджеров агентов;
- Менеджер должен запрашивать агентов у среды, если у него достаточно ресурсов для решения задачи, а агентов способных её выполнить нет;
- Менеджер должен иметь возможность получать сообщения и обрабатывать их (Передача сообщений менеджеру основана на технологии .NET Remoting), (В реализованной системе сообщения указывают менеджеру агента, которого необходимо выполнить вне очереди).

Спецификация задач, протоколов взаимодействия менеджеров между собой, агентов и менеджеров с информационным окружением выполняется разработчиком каждой конкретной системы.

Алгоритм разрешения

Алгоритм разрешения будет использовать приоритеты назначенные менеджерам. Алгоритм разрешения выполняется каждый раз, когда менеджер собирается выполнить агента для решения задачи. Информационная среда (ИС) должна содержать информацию об агентах которые запущены на выполнение задачи, статусе задачи и давать менеджерам возможность публиковать свои заявления.

Вход алгоритма – задача и агент для её решения. Выход – разрешение запуска агента данным менеджером.

Алгоритм состоит из таких этапов:

1. Если ресурсы необходимые для выполнения агента есть у менеджера то пункт 2, иначе ответ отрицательный, менеджер ищет другую задачу для выполнения;
2. Менеджер запрашивает ИС о состоянии задачи – «свободна», «выполняется», «идет обмен заявлениями», «обмен заявлениями закрыт»;
3. Если статус задачи - «выполняется» или «обмен заявлениями закрыт», то ответ алгоритма отрицательный и менеджер пытается найти другую задачу;
4. Если статус задачи – «свободна», то её статус изменяется на «обмен заявлениями» и менеджер публикует в среду своё заявление. Заявление получает приоритет равный приоритету менеджера. Менеджер ждёт $2 * T$ секунд. T – системно зависимая переменная;
5. Если статус задачи – «обмен заявлениями» менеджер публикует свое заявление и ждёт $2 * T$ секунд;
6. По прошествии T секунд задача получает статус – «обмен заявлениями закрыт»;
7. По прошествии $2 * T$ секунд разрешение на запуск получает менеджер с наивысшим приоритетом, остальные менеджеры пытаются приступить к решению других задач.

Заключения и дальнейшая работа

В работе было представлено описание реализованной архитектуры, которую можно классифицировать, как разновидность мульти-агентной архитектуры.

Одной из задач реализованной системы был сбор данных о погодных условиях в различных

населённых пунктах. Информация извлекалась из веб ресурсов. Была предусмотрена возможность снятия показаний с устройств слежения за состоянием погоды (например, электронных термометров).

Дальнейшая работа будет заключаться в усовершенствовании протоколов взаимодействия в системе, оптимизации работы менеджеров, разработке оптимальных алгоритмов маршрутизации в сети менеджеров и др.

Сопровождающие исходные тексты описанной архитектуры можно найти на сайте:

<http://unicyb.kiev.ua/~pprociik>

Глоссарий

Содержит краткое определение основных употреблённых в работе технических терминов. За более подробной информацией обращайтесь к библиотеке MSDN, а также другим указанным источникам.

Сборка (orig. Assembly) – представляет собой коллекцию типов и ресурсов, собранных для совместной работы и формирующих функциональную логическую единицу (источник MSDN).

Атрибут сборки – это значение, содержащее сведения о сборке(источник MSDN).

Службы Microsoft Windows (ранее называвшиеся службами Windows NT) - исполняемые приложения, работающие продолжительное время и выполняющиеся в отдельной сессии Windows. Эти службы не содержат элементов пользовательского интерфейса и могут быть автоматически запущены при загрузке компьютера, а также остановлены и запущены повторно(источник MSDN).

Литература

[1]. ASAP Theme: Multi-Agents, Intelligent Agents and Multi-Agents.

(<http://www.asap.cs.nott.ac.uk/themes/ma.shtml>, по данным на 24 января 2006 года)

[2]. Ó Vasant Honavar, Intelligent Agents and Multi Agent Systems, A Tutorial presented at IEEE CEC 99.

[3]. Jeff Ferguson, Brian Patterson, Jason Beres, Pierre Boutquin, and Meeta Gupta, C# Bible, Wiley Publishing, 2002

[4]. Merriam-Webster Online Dictionary, <http://www.m-w.com/>