DOI 10.37882/2223–2966.2022.10.02

МНОГОАГЕНТНЫЙ ПОДХОД К ПОСТРОЕНИЮ РАСПРЕДЕЛЕННОЙ АРХИТЕКТУРЫ ВЕБ-ПРИЛОЖЕНИЯ

MULTI-AGENT APPROACH TO BUILDING A DISTRIBUTED WEB APPLICATION ARCHITECTURE

Amoa Kouadio-kan Armel Geafroy

Summary. This article is devoted to the analysis of standards what are used to build a distributed architecture in applications. At the same time, taken in the account the multi-factorial and multi-level nature of the tasks that users set for the software, a reasonable assertion arises about the need and expediency of using multi-agent technologies. It is this approach what is now the most in demand in terms of different areas of development and formation of artificial intelligence, as well as different niches of information technology. But at the same time, it is necessary to take into account the problems that this method of building software poses. Since only when certain shortcomings and difficulties are taken into account, unresolved issues can be identified and futher eliminated to futher increase the efficiency of work. This issue requires coverage and detailed analysis, since recently the task conserning the process of designing social-economic systems from the point of view of information support has become more and more urgent. And since here you can immediately identify several niches, more precisely, spheres and subsystems, then the approach to solving each component should be applied individually, taken into account key characteristic. And this can be implemented at a sufficiently high level only by multi-agent technologies for building a distributed architecture for a web application. At the same time, it is worth considering different subspecies of such a system, which are expressed in deliberative, reactive and hybrid ways of building an architecture. And the latter, giving the ever increasing multitasking of software, is becoming the most commonly used when creating web application.

Keywords: system, functioning, program, software, information, environment, intelligence, artificial, agent, multi-agent, technology, application, features, basic autonomy.

Амоа Куадио-кан Армел Жеафруа
Аспирант
Воронежский Государственный Технический Университет (ВГТУ)
г. Воронеж
amoa.armel@gmail.com

Аннотация. Статья посвящена анализу стандартов, которые используются для построения распределенной архитектуры в приложениях. При этом, учитывая многофакторность и многоуровневость тех задач, которые ставят пользователи перед программным обеспечением, возникает обоснованное утверждение о необходимости и целесообразности применения мультиагентных технологий. Именно такой подход сейчас является наиболее востребованным с точки зрения разных направлений развития и становления искусственного интеллекта, а также разных ниш информационных технологий. Но при этом необходимо и учитывать те проблемы, которые ставит такой способ построения ПО. Так как только при учете определенных недостатков и трудностей, можно выделить нерешенные вопросы и в дальнейшем устранить их для еще большего повышения эффективности работы. Этот вопрос требует освещения и детального анализа, так как в последнее время становится все более актуальной задача, касающаяся процесса проектирования социально-экономических систем с точки зрения информационного обеспечения. А так как здесь можно сразу выделить несколько ниш, точнее сфер и подсистем, то и подход к решению каждой составляющей, должен применяться индивидуально с учетом ключевых характеристик. А это под силу реализовать на достаточно высоком уровне только мультиагентным технологиям построения распределенной архитектуры веб-приложения. При этом стоит учитывать и разные подвиды такой системы, которые выражаются в делиберативном, реактивном и гибридном способах построения архитектуры. И последний, учитывая все больше возрастающую многозадачность программного обеспечения, становится наиболее часто используемым при создании веб-приложений.

Ключевые слова: система, функционирование, программа, обеспечение, информационное, среда, интеллект, искусственный, агент, мультиагентый, технологии, применение, особенности, базовые, автономность.

Введение

На сегодняшний день любое веб-приложение представляет собой полноценную программу, при помощи которой пользователь посредством сети интернет осуществляет коммуницирование и работу с разными элементами. Например, при помощи таких программ можно осуществить приобретение товара, услуги, прокомментировать любое действие или новость в социальных сетях. Очевидным преимуществом веб-приложения является тот факт, что такая программа (в преобладающем большинстве случаев) не требует обязательного процесса установки на любой гаджет. А это существенно упрощает ее использование.

Основная часть. На сегодняшний день выделяют следующие разновидности веб-приложений, с учетом разных типов классификации:

- MPA вариант построения сайта. Он представляет собой приложение, состоящее из некоторого числа страниц. При коммуницировании с таким приложением пользователь совершает действие, осуществляя фактически запрос на сервер. В этот момент происходит обновление страницы.
- SPA разновидность построения шаблона предполагает под собой одну страницу кода HTML. Здесь обновление происходит динамически, в зависимости от действий, которые совершает пользователь. То есть полная перезагрузка не выполняется.
- И, наконец, вариант построения PWA, которое требует непосредственной установки пользователем. В этом случае программа может полноценно коммуницировать и в режиме оффлайна [8, С. 9].

Но, несмотря на такое распределение с точки зрения основного типа взаимодействия с пользователем, все приложения функционируют по принципу «клиент-сервер» [4, С. 155]. Но это взаимодействие может быть выражено в виде статической или динамической страницы.

И здесь возникает еще одна проблема, которая обусловлена возрастанием сложности действующих веб-приложений. Ожидания пользователя становятся все более комплексными, многоступенчатыми и программное обеспечение должно выполнять все эти запросы.

И в течение последних 10 лет развития телекоммуникационных технологий стали все более интенсивно проявлять себя концепции кросс-плат. Фактически они являются программными системами с форменными, интеллектуальными и распределенными особенностями. Реализовать их можно несколькими методами. Но именно многоагентные системы, дают возможность сконцентрировать все ключевые технологии с максимальной эффективностью, полнотой и выразительностью в итоговых результатах.

Оценивая ключевые особенности теории МАС — многоагентных систем, нельзя не упомянуть принцип и суть самого «агента». Он представлен программной или аппаратной сущностью, которая имеет возможность действовать на получение конкретной цели, которую задает пользователь.

Любой агент в процессе программирования описывается набором характеристик [3, С. 5]. И именно их совокупность будет характеризовать суть агента, которую можно к свести к следующему:

- Принципы реактивности. Здесь заложена максимально быстрая оперативная реакция на любые изменения.
- Принцип автономности. В это понятие вкладывается возможность программы работать самостоятельно на пути достижения цели.
- Принцип адаптивности. В наборе этих характеристик агент должен иметь возможность самостоятельно обучаться, наращиваю комплексы своих умений.
- Принцип коллаборативности. Каждый из агентов нацелен на эффективное взаимодействие с другими аналогичными аппаратными или программными сущностями. В этом виде он может представлять собой либо роль поставщика, либо потребителя информации.
- Принцип активности. Каждый из агентов для того, чтобы считаться эффективным, должен не только генерировать цели, но и обладать набором функций, которые позволят достичь результата.
- Принцип коммуникативности. В этом понимании каждый из агентов полноценно общается между собой для комплексного получения информации и данных.
- И последний принцип — это способность рассуждать. Умение выстраивать логические связи, позволяет работать с разными источниками, выделяя ключевое и приводя собранный набор к одному виду.

Как видно, роль каждого агента является ключевой. И здесь сложно выстроить иерархию, с точки зрения важности для функционирования в приложении, как впрочем и другого программного обеспечения, принцип работы которых построен на многоагентном подходе. Но при этом, исходя из ключевых характеристик, каждый из работающих агентов оценивается по наличествующим ключевым целям. Поэтому среди приоритетов агента можно выделить:

- Цели.
- Желания.
- Взятые обязательства.
- Ключевые намерения.
- Главные убеждения.
- Но, здесь стоит четко осознавать, что такое выделение ключевых свойств отдельных агентов возможно только с точки зрения анализа со стороны других компонентов системы (агентов) [9].

Структуру взаимодействия агентов, как между собой и с главным распределительным агентом, так и с необходимой базой знаний, можно отобразить схемой, представленной на рисунке 1.

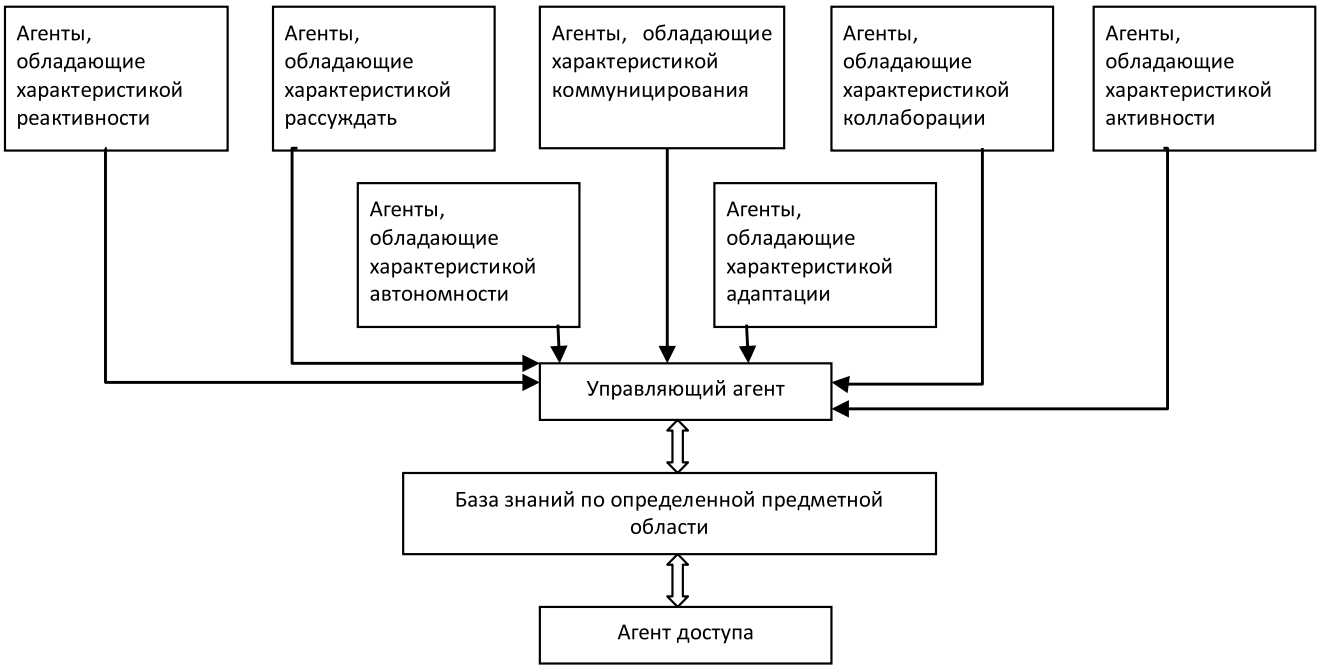

Рис. 1. Принцип построения архитектуры многоагентной системы

После того, как становится понятным, в чем заключается роль каждого агента перед целой системой, стоит вернуться к термину «многоагентные системы». Это определение применяют по отношению к тем системам, в которые входят определенное количество автономных агентов, способных исключительно полноценно коммуницировать между собой для осуществления поставленной задачи. Система коммуницирования проводится через платформу, ключевая суть которой является асинхронизация. Этот принцип является ключевым, так как в каждой конкретной поставленной задаче есть ключевой запрос, а также масса побочных эффектов, которые являются второстепенными по отношению к главной цели.

Что же касается непосредственно класса выстраиваемой архитектуры программного обеспечения, то можно выделить следующие разновидности:

· Делиберативную. Здесь агенты, как в принципе и вся сама архитектура программного обеспечения, применяют точно выстроенные представления о событии или действии, которые чаще всего выражают в виде символов. И это является традиционным подходом, который присущ классическому принципу построения искусственного интеллекта. Все принимаемые действия и получение результатов здесь осуществляются на основе формальных суждений, логических рассуждений, сравнений с существующим образцом.

· Реактивную. Здесь применяемые принципы можно противопоставить делибераттивному принципу построения архитектуры. Здесь отсутствует принцип символьного отображения модели мира. А суть взаимодействия и работа агентов сходится к тому, что решается проблема «ситуация — действие». При этом агенты, которые выполняют определенную задачу подбирают самые оптимальные действия, помогающие ее решению.

· Гибридная. В связи с возрастающими запросами пользователей разных программных обеспечений, включая веб-приложения, стало невозможным удовлетворять ставящиеся задачи только методами построения архитектуры по делиберативным или реактивным принципам. Поэтому многие разработчики на ранних этапах становления такого подхода к формированию архитектуры, стали объединять два этих ключевых варианта. Это и позволило создать разные виды гибридных архитектур. Фактически, стало возможным описывать ставящуюся задачу и поиск решения не только символьными, но и другими методами.

Фактически, если описывать суть многоагентного подхода к построению архитектуры, то он заключается в корректном распределении функций системы между теми интеллектуальными компонентами, которые реализуют возложенные на них обязательства и задачи.

Также стоит учитывать, что многие из агентов, которые обеспечивают работу разрабатываемого приложения, сталкиваются с обязательным выбором одного или нескольких вариантов из существующих и подходящих заданным условиям. Поэтому, процесс, по которому

осуществляется выбор необходимых ресурсов, можно представить следующим образом:

$$CM=QCM1 \cup QCM2 \cup \ldots \cup QCMi$$

где CM — необходимая база данных, удовлетворяющая условию запроса;

QCMi — количество (множество) уникальных наборов, которые подходят для того, чтобы удовлетворить запрос пользователя под одному из параметров.

i — количество возможных переборов, осуществляемых при поиске.

При этом, по каждому из наборов QCMi, которые в совокупности и формируют итоговый результат, осуществляется перебор, выражаемый следующим отношением:

$$QCMi=N\{NK, NA, RN, NI\},$$

где — NK — количество подходящих концептов;

NA — количество удовлетворяющих условию поиска атрибутов;

NI — количество экземпляров;

RN — множество отношений между составляющими.

Среди преимуществ такого подхода МАС можно выделить:

- Корректное распределение вычислительной нагрузки.
- Более высокое качество поставленных задач. Достигается поиском оптимальных решений в момент коллаборации и взаимодействия между разными типами агентов.
- Возможность системе проявлять гибкость [2, С. 27].
- Также появляются способы масштабирования системы. Достигается это децентрализованностью.

Но, оценивая преимущества, нельзя не упомянуть и о недостатках такого принципа построения архитектуры web-приложений. Заключается они, как правило, только в алгоритмических сложностях описания. В результате это может создать некую неопределенность.

Но после того, как рассмотрены непосредственно вопросы принципов построения многоагентной распределенной архитектуры веб-приложений, ее преимущества и возможные недостатки, стоит перейти к ключевым сферам применения. Это позволит оценить оправданность ресурсозатратности и развитие дальнейших перспектив с точки зрения внедрения в разные сферы деятельности человека.

Первым, что стоит осознать — это практически безальтернативность такого способа построения архитектуры программного обеспечения в той ситуации, когда одну ставящуюся задачу необходимо разбить на 2 и более подзадач. В этом случае каждый компонент ключевой цели распределяется между агентами. Выполняется такая декомпозиция ключевой задачи отдельным агентом. Среди его ключевых особенностей должно выделяться умение разбивать целое на части. При этом от него не требуется подобрать решение к каждой из них. Он выполняет функцию своеобразного «сортера» — и не более того. Это определяется теми ресурсами, которыми наделен каждый из агентов.

После того, как каждая подзадача находит оптимальное решение, все тот же агент, который проводил декомпозицию ставящейся цели, проводит согласование и интеграцию полученных результатов, для выдачи единого итога.

Конечно, то на сколько итоговым результатом будет удовлетворен конечный пользователь, влияет эффективность работы каждого агента. Фактически, оценивается умение подобрать правильное логическое решение каждой подзадачи [5]. А для агента, осуществляющего декомпозицию, оценивается способность быстро и эффективно перемещаться по сети, осуществляя коммуникацию с внешними хостами. Ведь этот агент должен эффективно осуществлять подбор информации с последующей ее композицией после осуществления ряда корректных действий.

Как видно из всего выше представленного, наиболее актуальными, с точки зрения применения мультиагентных технологий при построении архитектуры любого программного обеспечения, включая разнообразные веб-приложения, являются такие отрасли, которые представляют собой:

- Объемные базы данных и знаний.
- Системы любого моделирования, независимо от того, основана она на математических расчетах, имитации процесса или ситуативного стечения обстоятельств.
- Геоинформационные системы, в которых осуществляется ключевые многоуровневые запросы, как по ситуативным описаниям, так и по символьным кодам, отображающим ту или иную информацию [1, С. 100].

Важно четко осознавать, что именно в этих представленных сферах человеческой деятельности и полноценных системах, именно мультиагентные технологии построения архитектуры программного обеспечения позволяют выбрать лучший вариант решения поставленной задачи. Так как в этом случае ключевой запрос разделяется агентом на несколько отдельных информационных компонентов. И далее с каждым из них

работает свой собственный агент, который нацелен на решение конкретной задачи. Чаще всего, в наиболее сложных системах оптимально с поставленными задачами справляются гибридные системы построения мультиагентных архитектур.

При этом не стоит упустить из виду и возможность обучения интеллектуальных агентов. Этот вопрос является также актуальным, в связи с все более разветвляющимся и усложняющимся, по своему ключевому принципу, запросам к программному обеспечению. Наряду с этим интерес представляет дальнейшее расширение работы и перспектив применения мобильных агентов, которые способны осуществлять подбор необходимой информации и коммуницировать с другими программами в процессе перемещения по интернету и интранету.

И это является перспективной нишей дальнейшего исследования, наряду с развитием и усложнением мультиагентных технологий.

ЛИТЕРАТУРА

1. Владимирская Е.Н. Особенности разработки мультиагентных систем на основе платформы Jade в рамках парадигмы Semantic Web / Сб. матер. Междунар. науч.-практ. конф. Минск,— Мн.: Институт математики НАН Беларуси, 2009.— 112 с.
2. Городецкий В.И., Грушинский М.С, Хабалов А.В. Многоагентные системы // Новости искусственного интеллекта, 1998.— № 2.— С. 28–29.
3. Ландсберг С.Е. Некоторые аспекты проектирования мультиагентных систем с использованием языка UML [Текст] / С.Е. Ландсберг, А.А. Хованских // Вестник Воронежского государственного технического университета.— 2012.— Т. 8.— № 9.— С. 4–8.
4. Ландсберг С.Е., Хованских А.А. Основы агентов и многоагентных систем // Оптимизация и моделирование в автоматизированных системах: межвуз. сб. науч. тр. Воронеж: ГОУВПО ВГТУ, С. 151–156.
5. Ландсберг С.Е., Хованских А.А. Особенности построения информационных систем с использованием мультиагентных технологий // Вестник ВГТУ. 2014. № 3–1. URL: https://cyberleninka.ru/article/n/osobennosti-postroeniya-informatsionnyh-sistem-s-ispolzovaniem-multiagentnyh-tehnologiy (дата обращения: 05.06.2022).
6. Леденева Т.М. Системы искусственного интеллекта и принятия решений: учеб. пособие / Т.М. Леденева, С.Л. Подвальный, В.И. Васильев. Уфа, 2005.
7. Основина О.Н. Мультиагентная система оценки и прогнозирования надежности АСУ // Труды II школысеминара молодых ученых «Управление большими системами». Воронеж, 2007.— С. 168–176.
8. Подвальный, С.Л. Интеллектуальные системы моделирования: принципы разработки [Текст] / С.Л. Подвальный, Т.М. Леденева // Системы управления и информационные технологии.— 2013.— № 1(51).— С. 4–10.
9. Anatomy of a Web Service: XML, SOAP and WSDL for Platformindependent Data Exchange [Электронный ресурс].— Режим доступа: http://www.webreference.com/authoring/web_service/index.html (дата обращения 01.06.22 г.).

Журнал «Современная наука: актуальные проблемы теории и практики»