ОНТОЛОГИЧЕСКИЙ ПОДХОД К ИНТЕГРАЦИИ СТОРОННИХ ФУНКЦИОНАЛЬНЫХ ПРИЛОЖЕНИЙ В СИСТЕМЫ ОБРАБОТКИ ЗНАНИЙ

Загорский А. Г.
Кафедра интеллектуальных информационных технологий,
Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: alexandr_zagorskiy@mail.ru

В данной работе описан подход к интеграции сторонних функциональных приложений, которые предоставляют собственный API, в системы, основанные на графодинамической обработке знаний. Также описан процесс интеграции сторонних функциональных веб-приложений.

Введение

Количество интеллектуальных систем (ИС), основанных на обработке знаний, незамедлительно растёт с каждым годом. Однако, человечество успело накопить уже значительный опыт в виде различных программных продуктов, основанных на стандартном подходе обработки информации. Задача интеграции таких продуктов в интеллектуальные системы с целью последующего их использования является ключевой и привлекает всё больше внимания. Возможность интегрировать в систему уже функционирующую подсистему позволяет экономить как значительное количество времени, так и ресурсы.

В данной работе будет рассмотрен подход к интеграции сторонних функциональных приложений в системы, построенные на основе семантических технологий.

I. Подход к интеграции сторонних функциональных приложений

Под функциональным приложением понимается программно реализованный механизм преобразования данных в соответствии с заданной функцией. Функциональность приложения может быть абсолютно любой: прогноз погоды в заданном городе, смена размеров изображений, синтез речи, и т.д. Зачастую такое приложение может предоставить программный интерфейс (API – Application Programming Interface), в котором описаны всевозможные форматы входов и соответствующих им выходов.

Основной проблемой интеграции сторонних функциональных приложений является различие форматов данных, с которыми работают приложения. Даже если два разных приложения предполагают обработку данных из одной предметной области (ПрО), с высокой вероятностью формат данных у них будет различаться.

Для интеграции стороннего функционального приложения в систему обработки знаний выделяются следующие глобальные задачи:

- подготовить базу знаний (БЗ) ИС;
- реализовать агент обработки знаний, который использует стороннее функциональное приложение, для решателя задач (РЗ) ИС.

Подготовка БЗ подразумевает формализацию и наполнение знаниями ПрО, которая содержит необходимые знания для формирования запросов к стороннему функциональному приложению. Также необходимо формализовать ПрО, в которую будут погружаться полученные после обработки данные в виде знаний.

Обобщённый алгоритм агента обработки знаний с использованием сторонних приложений выглядит следующим образом:

1. извлечение из БЗ необходимых структур знаний;
2. преобразование извлечённых знаний в формат, необходимый для подачи на вход стороннего приложения;
3. отправление запроса и ожидание ответа стороннего приложения;
4. формирование конструкции знаний по полученным данным;
5. погружение новых знаний в БЗ ИС.

Проектирование агента предлагается производить на основе шаблона программного проектирования «декоратор» [1].

Таким образом, задача интеграции сводится к формализации структур входа и выхода конкретных функциональных приложений, а также реализации механизма отправки исходных данных и получения уже обработанных данных.

II. Обработка знаний с использованием сторонних веб-приложений

Распространённым видом функциональных приложений являются веб-приложения. Зачастую общение с такими приложениями реализуется на основе HTTP-протокола [2].

Для интеграции веб-приложений предлагается использовать многоагентную структуру. Диаграмма компонентов агентов обработки знаний с использованием сторонних веб-приложений представлена на рисунке 1.

Рассмотрим подробнее основные компоненты структуры:

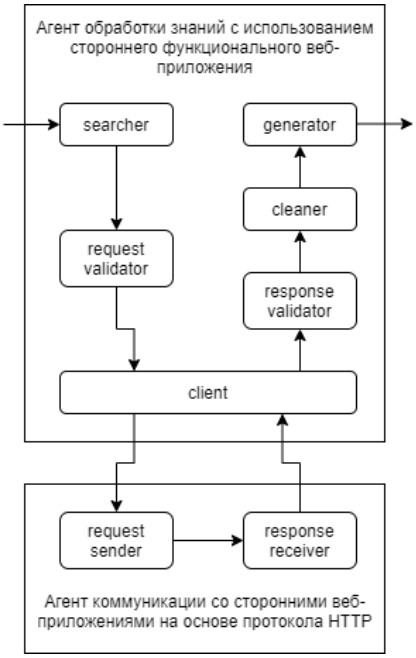

Рис. 1 – Основные компоненты агентов обработки знаний с использованием сторонних веб-приложений

- компонент *searcher* производит поиск необходимых конструкций знаний в БЗ по заданным шаблонам;
- компонент *request validator* проверяет полученные значения;
- компонент *client* формирует запрос для сервера, передаёт его *агенту коммуникации со сторонними веб-приложениями на основе протокола HTTP*, после чего ожидает ответа агента;
- компонент *response validator* обрабатывает полученные данные ответа сервера;
- компонент *cleaner* удаляет из памяти БЗ временные конструкции;
- компонент *generator* формирует структуры знаний по заданным шалблонам с последующей интеграцией в БЗ.

Агент коммуникации со сторонними веб-приложениями на основе протокола HTTP реализует механизм общения с серверами веб-приложений. Фрагмент БЗ в виде SCg-кода, сорежащий примеры входных и выходных конструкций агента, представлены на рисунках 2 и 3 соответственно.

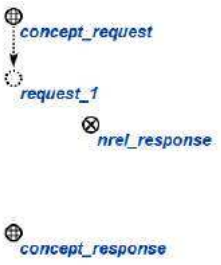

Рис. 2 – Пример входной конструкции агента коммуникации со сторонними веб-приложениями на основе протокола HTTP

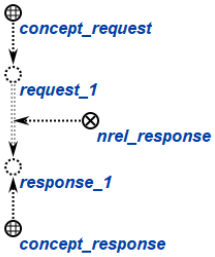

Рис. 3 – Пример выходной конструкции агента коммуникации со сторонними веб-приложениями на основе протокола HTTP

Функционирование агента базируется на множествах *concept_request* и *concept_response*. Вызов агента происходит при создании выходной дуги из множества запросов. Он извлекает полученный на входе запрос, отправляет его и ожидает ответа сервера. При получении ответа, он заносится во множество ответом БЗ, а также соединяем его с соответствующим запросом отношением. Так как информация после отработки агента извлекается моментально, после чего система формирует новые знания, не имеет смысла хранить и накапливать эти конструкции.

Такая структура агентов позволяет следовать принципу единой ответственности [3]: реализация первого агента является интеграцией конкретных функций посредством формализованных входов и выходов веб-приложения, в то время как агент отправки запросов является унифицированным средством коммуникации с любыми серверами сторонних веб-приложений. Также структура является легко расширяемой, что позволит, например, в будущем добавить механизм ограничения времени на запросы.

Аналогичным способом могут быть спроектированы агенты обработки знаний с использованием сторонних локальных функциональных приложений.

Заключение

Сформулированный подход к интеграции сторонних функциональных приложений, а также разработанная методология проектирования агентов обработки знаний с использованием сторонних функциональных веб-приложений позволяют ускорить процессы проектирования и разработки решателей задач интеллектуальных систем, основанных на технологии OSTIS [4].

1. Decorator [Electronic resource] – Mode of access: https://refactoring.guru/design-patterns/decorator. – Date of access: 18.10.2021
2. RFC 7230: HTTP/1.1 Message Syntax and Routing [Electronic resource] – Mode of access: https://datatracker.ietf.org/doc/html/rfc7230. – Date of access: 18.10.2021
3. Robert, M. SRP: The Single-Responsibility Principle. / Robert, M. // Agile Software Development, Principles, Patterns, and Practices – 2003. – pp. 95-98.
4. Семантические технологии проектирования интеллектуальных систем и семантические ассоциативные компьютеры / В. В. Голенков [и др.] // Доклады БГУИР. - 2019. - № 3 (121). - С. 42 - 50.