Информационные роботы в системах управления предприятием

Д.Е. Намиот, В.А. Сухомлин, Е.В. Стариков, С.П. Шаргалин, А.А. Стяпшин

***Аннотация*—Настоящая работа посвящена рассмотрению использования программных агентов (информационных роботов) в корпоративных информационных системах. Мы рассматриваем общие модели и назначение таких информационных посредников, приводим исследование их возможностей и перспектив. В работе исследуются требования, выдвигаемые к такого рода компонентам. Также проводится исследование существующих сред разработки программных агентов и рассматриваются вопросы проектирования корпоративных информационных систем и систем управления предприятием на базе моделей программных агентов. Финальной целью проекта, в рамках которого была выполнена данная работа, является именно новая модель для корпоративной информационной системы.**

***Ключевые слова*—программные агенты, информационные роботы, ERP, микро-сервисы.**

I. Введение

Эта работа является продолжением и расширением исследований о программных агентах в ERP системах, представленного в работе [1].

Согласно классическому определению, программный агент – это программа-посредник. Эти посредники взаимодействуют с пользователями или другими программами и смыслом этого взаимодействия является выполнение каких-либо действий от имени пользователя или другой программы [2].

Условность многих (большинства) определений в программировании и естественное размывание этого термина привели к тому, что в настоящее время “агентом” может быть названо (и называется) произвольное программное обеспечение (программа). (Естественно, что, как и многие другие определения в программировании - это достаточно условная характеристика. Часто определение программного агента нагружается (снабжается) дополнительными обременениями. Наиболее часто здесь упоминается требование самостоятельного запуска. Согласно этому дополнению, агенты не запускаются пользователем (другим приложением) для решения задачи, а активизируются самостоятельно [2]. Возможно, что это уже излишнее требование – способности к взаимодействию или выполнению каких-либо действий никак не пропадут, если приложение будет запущено “вручную” или по расписанию (crontab, например). Основное, что есть для агента – это именно его поведение. Это основное отличие программного агента от приложения. Агент определяется именно описанием его поведения.

Обзор и классификация программных агентов приведены в нашей работе [1]. По своим свойствам, речь ведут, например, об интеллектуальных агентах, которые обладают способностью к обучению. Отметим также, что дебаты о том, называть это агентом или приложением ведутся уже давно [3].

Автономные агенты характеризуются способностью выбирать задачи и приоритеты. Распределенные агенты, согласно названию, работают на физически различных компьютерах. Мультиагентные системы состоят из агентов, которые для выполнения своих действий должны каким-либо образом общаться с подобными им компонентами [4-6]. В указанных работах приводится описание формальных механизмов для их спецификации. Термин мобильные агенты следует, видимо, признать устаревшим. Он возник еще до повсеместного распространения мобильного программного обеспечения и соответствующих средств разработки.

Формальный спор, является ли какое-то программное обеспечение агентом или нет достаточно бессмысленный. Общая идея (как некая согласованная форма в большинстве определений) заключается в том, что агенты более автономны, чем, например, объекты. У агентов, в целом, должны быть автономность в плане выбора задач и приоритетов, гибкое и проактивное (хотя этот термин можно понимать по-разному) поведение. Здесь можно провести параллели еще с одним определением из программной индустрии – акторы. Согласно работе [7], актор – это автономный, интерактивный, исполняющий несколько функций объект, который обладает внутренним состоянием и участвует в информационном обмене. Вместе с тем, необходимо отметить, что если речь идет о готовой информационной системе (ERP модуле), то, как правило, его программная модель (программная архитектура) уже зафиксирована. Соответственно, все архитектурные вопросы относятся к системам,

Статья получена 10 января 2017.

Намиот Д.Е., МГУ имени М.В. Ломоносова, (email: dnamiot@gmail.com).

В.А. Сухомлин, МГУ имени М.В. Ломоносова, (email: sukhomlin@mail.ru)

Е.В. Стариков, МГУ имени М.В. Ломоносова, (email: evs1@list.ru)

С.П. Шаргалин, Сургутнефтегаз, (email: Shargalin_SP@surgutneftegas.ru)

А.А. Стяпшин, Сургутнефтегаз, (email: Styapshin_AA@surgutneftegas.ru).

проектируемым с “нуля”. В современной терминологии, речь, возможно, должна идти о микросервисах [8]. На сегодняшний день это один наиболее “популярных” архитектурных подходов.

Другой важный момент, который также относится к базовому проектированию – это так называемые контекстно-зависимые вычисления. Под словом контекст понимаются любые измеряемые данные, которые отличаются от позиционирования. Позиционирование (местоположение) – есть объективная всегда существующая характеристика (измерение). Все остальное – показания произвольных датчиков (влажность, освещенность, шум и т.д.), доступные в данный момент беспроводные сети, пользователи из социального круга, находящиеся в данный момент в онлайне и т.д. – все это примеры контекста. Соответственно, контекстно-зависимые вычисления (context-aware computing) [9] – это сервисы, которые зависят от контекста. В глобальном плане, все мобильные сервисы, например, должны быть контекстно-зависимые. Контекст (например, шум, освещение), очевидно, существенен для функционирования сервиса.

Примыкающая к этому область - ambient mobile intelligence [10]. Здесь пользовательские интерфейсы меняются в зависимости от контекста.

И как последнее в ряду этих рассмотрений, отметим так называемые кибер-физические системы. Это направление, выросшее из встроенных систем, в настоящее время рассматривается как система верхнего уровня для Интернета вещей, M2M (межмашинного взаимодействия) и других систем, соединяющих виртуальный и физический мир. Применительно к программных агентам (сервисам, автономным приложениям – как их не называй), это означает возможность включения человека в качестве элемента принятия решения [12].

Для существующих информационных систем для целей развития информационной системы учета и управления крупным предприятием можно обозначить следующие приоритеты [13]:

1) полная автоматизация операционной деятельности, выполняемой сегодня с помощью информационной системы,

2) внедрение решений, позволяющих без программирования строить и перестраивать информационные системы, реализующие определенные бизнес-модели.

Иными словами, речь идет об автоматизации работы с существующей информацией. Отсюда и название – информационные роботы.

Остальная часть статьи структурирована следующим образом. В разделе II мы останавливаемся на моделировании информационных систем на основе агентов. Раздел III посвящен средствам разработки для программных агентов.

II. ПРОГРАММНЫЕ АГЕНТЫ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

Основным моментом является представление информационной системы в виде отдельных взаимодействующих компонент [14].

Это очень важный момент – из компонент можно собрать новую систему, но не всегда существующую можно разобрать. Это зависит от производителя. Вовсе не все информационные системы следуют модульному дизайну. На рисунке 1 представлена такая модель.

По сути – это упоминавшийся выше подход на основе микросервисов. Функциональное разбиение ERP системы, реализация функций, как автономных компонент (агентов), которые взаимодействуют между собой с помощью отправки сообщений.

Здесь выделены три архитектурных уровня:

1) интерфейс агентов. Прием и передача сообщений, представление информации;

2) проблемно-ориентированные агенты: Финансы, Бухгалтерия и т.д.

3) сервисные агенты (стандартные функции в каждой из проблемных областей)

При этом в качестве “агентов” часто выступают просто автономные приложения, которые выполняют какие-либо сервисные функции, не реализованные в базовой информационной системе или ERP. Например, в работе [15] в качестве такого “агента” представлена компонента (модуль), реализующая обмен данными с платежным шлюзом.

Все рассмотренные нами модели информационных систем уровня предприятия (ERP) декларировавшие поддержку агентов представляли собой функциональную декомпозицию информационной системы и обмен данными между такими независимыми компонентами. На рисунке 2 представлена еще одна модель мультиагентной ERP системы.

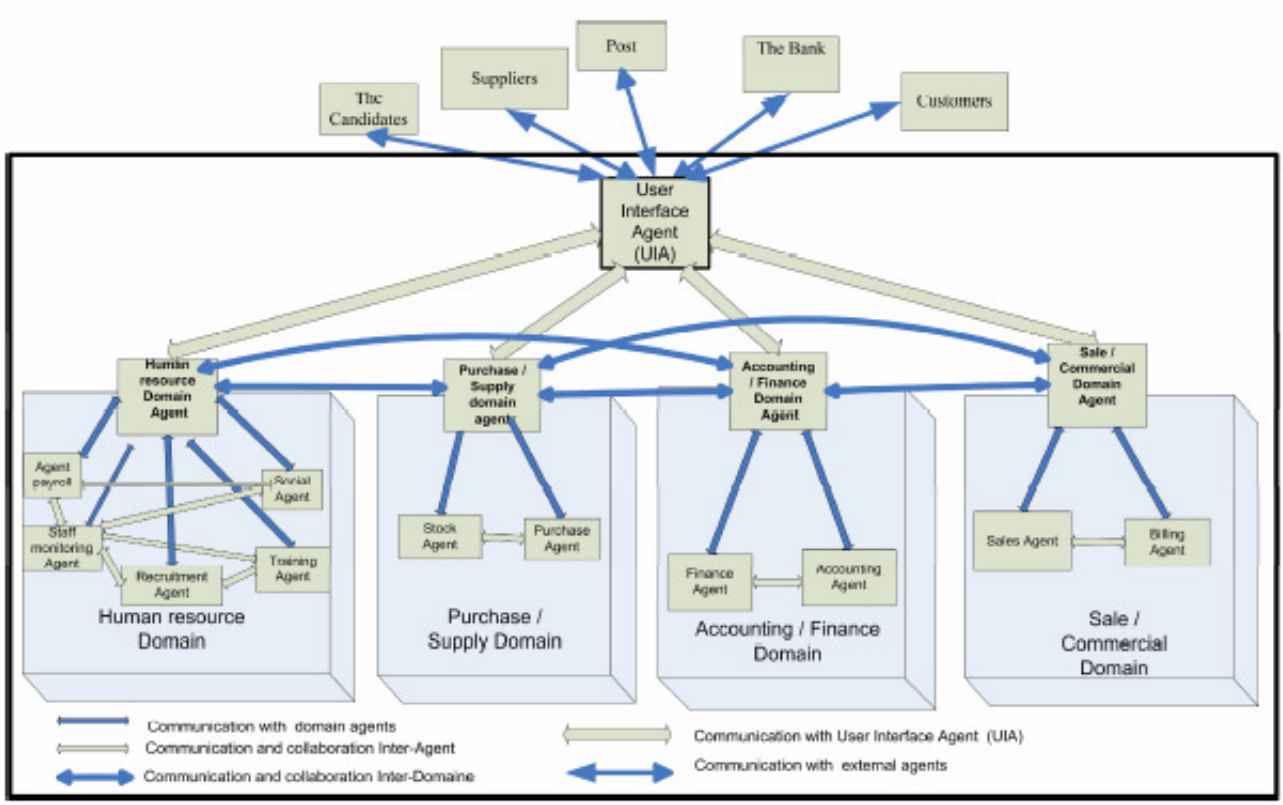

Рис. 1. Мультиагентная информационная система [14].

В этой модели Функционал ERP также разобран на отдельные задачи. Вместо единого интерфейса – агент-координатор и агенты, отвечающие за интерфейсы. Поскольку агенты разрабатываются (обновляются) независимо, то это означает, что различные задачи могут иметь и разные пользовательские интерфейсы. Или вообще какие-то пользовательские интерфейсы могут быть отменены, если решение задач перекладывается на агентов - для межмашинного взаимодействия (M2M) не нужны пользовательские интерфейсы.

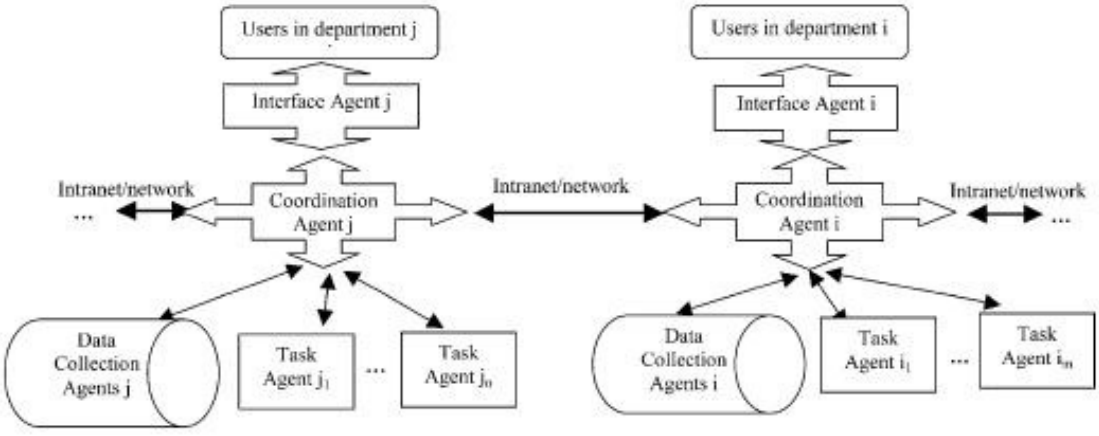

Рис. 2. Мультиагентная ERP [16].

Заметим, что этот подход является достаточно традиционным (попросту говоря – старым) при проектировании информационным систем (в отечественной классификации – автоматизированных систем управления). В этой связи мы хотели бы сослаться, например, на такие модели как Lambda-архитектура. В ней мы имеем дело с некоторой информационной шиной (на базе Kafka [17], например) и процессинг (обработку), реализованную в двух вариантах: пакетном варианте и реальном времени (рисунок 3). Эта архитектурная модель часто используется при обработке больших данных.

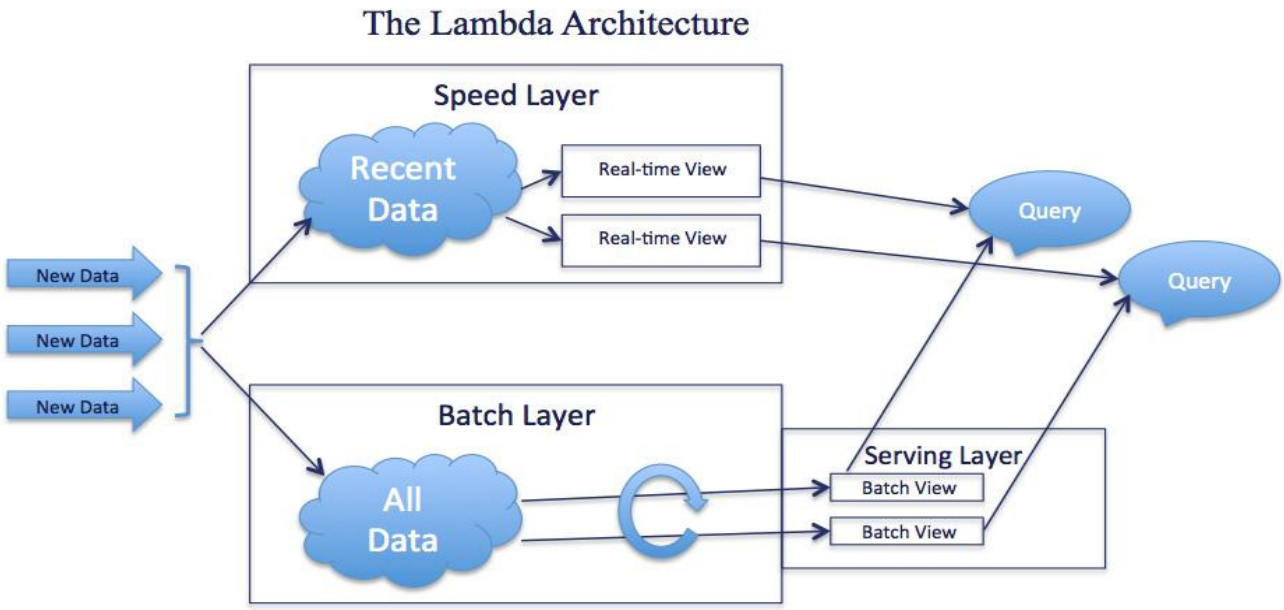

Рис. 3. Lambda-архитектура

Но это относится именно к построению модели. Если модель уже зафиксирована, то автоматизация, выполняемая агентами (тем, что называется агентами) есть автоматизация внутри готовой системы. Естественно, что внутри любой программной реализации, имеющей какой-либо пользовательский интерфейс, все действия, потенциально выполнимые в этом интерфейсе могут быть выполнены программно. Естественно, это может быть недоступно (не всегда доступно) сторонним разработчикам. Это зависит от наличия и доступности программных интерфейсов. Но производитель программного обеспечения может, конечно, это реализовать. Итогом является то, что автоматизация в таком случае сводится к созданию средств повтора (эмуляции) пользовательских действий. То есть, формулируется то, что нужно было бы сделать пользователю (use-case, бизнес-сценарий, регламент и т.п. – пример на рисунке 4), а далее создается инструмент для выполнения таких действий от имени (и с полномочиями) пользователя.

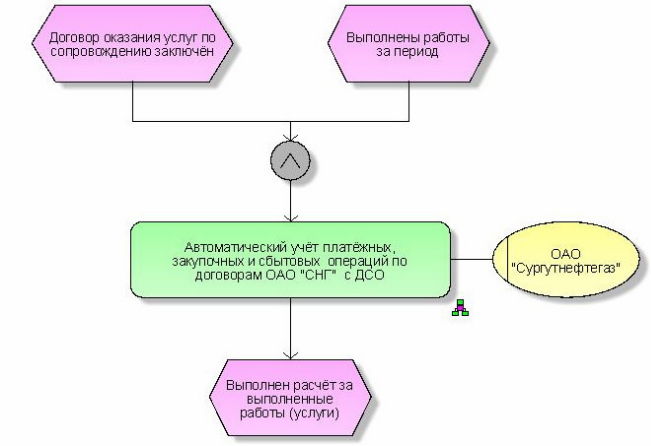

Рис.4. Спецификация применения

Такого рода инструмент может быть назван как скриптом, так и агентом.

III Средства разработки для программных агентов

В этом разделе мы хотели бы остановиться на программных инструментах (например, фреймворках), которые могут быть использованы для построения агентов.

Во-первых, это, конечно, спецификация FIPA (Foundation for Intelligent, Physical Agents) [18]. Спецификация описывает абстрактную архитектуру для системы программных агентов (рисунок 5).

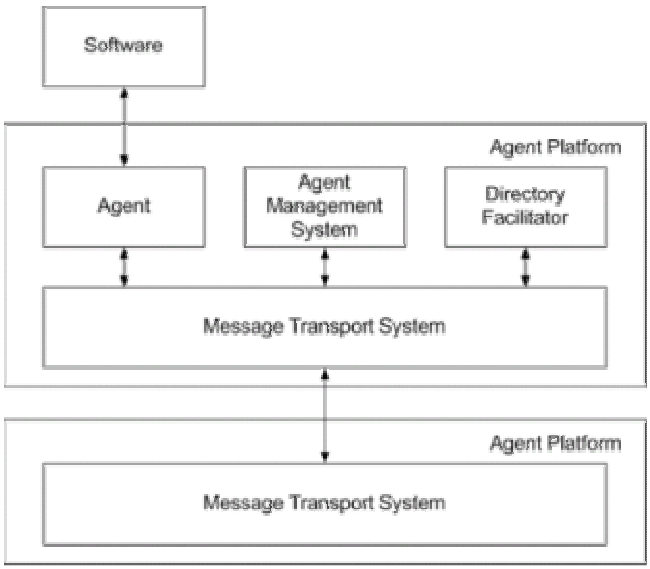

Рис. 5. FIPA

Основная идея платформы – выделение общих уровней, отделение коммуникаций (транспорта) от бизнес-логики. Это уже достаточно старая система, не все из программных реализаций, удовлетворяющих данной спецификации продолжают поддерживаться.

Как конкретный пример “живой” реализации этой

спецификации можно назвать JADE (Java Agent Development) – Open Source пакет для создания программных агентов [19]. Реализован на языке Java. Технически - это программный сервер, на котором и исполняются агенты. Помимо среды для запуска агентов, в JADE входят библиотеки (в Java – пакеты) для разработки агентов и графические инструменты для администрирования и мониторинга.

Основными элементами JADE являются:

- контейнер. Это запущенная копия JADE, которая может содержать несколько агентов;
- платформа. Это совокупность активных контейнеров. Среди активных контейнеров выбирается один главный, и он и содержит информацию обо всех остальных контейнерах.

Каждый агент в системе характеризуется своим уникальным именем. Главный контейнер (main container) в JADE запускает два специальных агента. Можно назвать их служебными агентами:

- AMS (Agent Management System). Этот агент обеспечивает сервис управления другими агентами. Например, с его помощью можно создать (запустить) или остановить (удалить) агента;
- DF (Directory Facilitator). Этот агент поддерживает каталог агентов (“Желтые страницы”). Здесь агенты смогут искать других агентов, которые им понадобятся для достижения целей.

Общая структура представлена на рисунке 6.

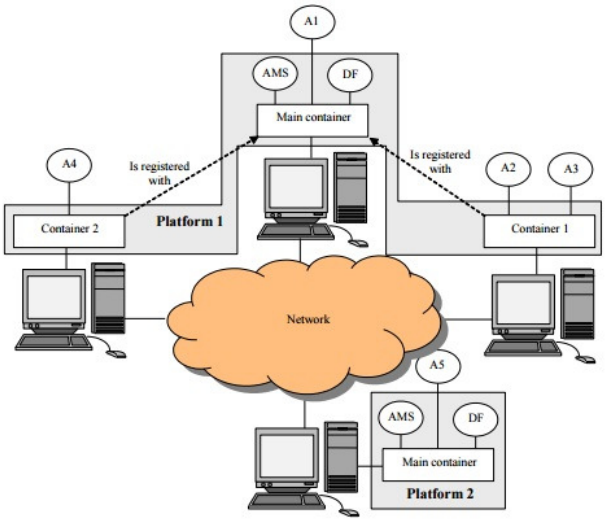

Рис. 6. Контейнеры JADE

Наличие каталога агентов и единственного главного контейнера платформы позволяет агентам общаться (обмениваться информацией) между собой. При этом такое взаимодействие возможно внутри контейнера, между контейнерами одной платформы (между агентами разных контейнеров на одной платформе) и между платформами (между агентами в контейнерах разных платформ). Обмен сообщениями осуществляется асинхронно, с использованием очередей (рисунок 7).

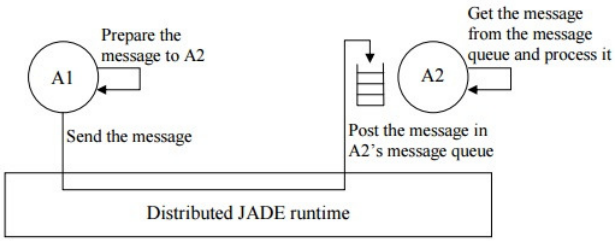

Рис. 7. Обмен сообщениями в JADE

Поведение конкретного агента в системе – это отдельная задача. Но в JADE эти процессы отличаются от базовой модели процессов (Thread) в Java. Программист должен определять, когда заканчивается одно исполнение и начинается следующее. Это позволяет сохранять жесткое соответствие: один агент – один Java процесс. Естественно, что это работает быстрее, чем переключение между процессами, а также позволяет избежать проблем с синхронизацией (например, если бы мы имели несколько процессов для одного агента, которые бы работали, естественно, с одним и тем же ресурсом).

Каждый агент – это Java класс. Каждое действие (акция) агента – это метод класса (в Jade - поведение).

Агенты – каталогизируются. На рисунке 7 показана организация каталога агентов:

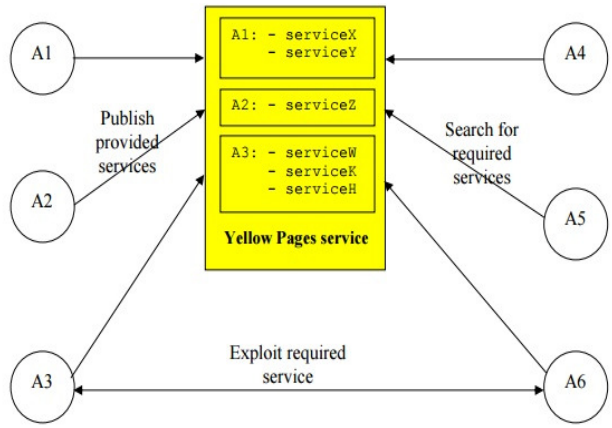

Рис. 7. Каталог в JADE

В целом – это достаточно низко-уровневое программирование. Основная ценность в котором – это даже не программные интерфейсы, а методология создания приложений. Но если приложение сложное, а это типично для ERP – большое количество модулей/сервисов с большим количеством возможных связей, то программирование в такой среде есть, по факту, перепрограммирование всей ERP системы.

Из других поддерживаемых (сопровождаемых) реализаций отметим агенто-ориентированный язык программирования JACK и методику Prometheus. Более точно, JACK - это объектно-ориентированная среда для создания систем на базе агентов [20]. В ее состав включен JAL – Java Agent Language. Это надстройка (препроцессор) для Java. Возможно, что наиболее интересный момент здесь – это программная модель,

которая лежит в основе - BDI: belief–desire–intention (мнение - желание – намерение) [21]. Модель агента (архитектурные компоненты) в такой системе:

Beliefs: информационное состояние агента. Используется именно термин “мнение“, а не знание, чтобы подчеркнуть тот факт, что информация у агента не обязательно может быть истинной.

Beliefset: набор мнений. Например, в заполненной базе данных.

Desires: желание, мотивационное состояние агента. Цели или ситуации, которые агент хочет достичь, задачи, которые необходимо выполнить.

Goals (цель): одно из желаний, которого стремится достичь агент. По сути – накладывание каких-то дополнительных ограничений на желания. Что именно из желаемого агент пытается достичь?

Intentions: намерения описывают то, что агент собирается делать. Некоторый исполняемый план.

Plans: планы представляют собой последовательность действий (акций), которые агент может выполнять для достижения намерений. Планы могут включать в себя другие планы (например, план “поехать на автомобиле” может включать план “найти ключи”). Детали планов могут уточняться в процессе выполнения.

Events: события выступают в роли триггеров для активности агента. События могут обновлять мнения, активировать планы или модифицировать цели. Источники событий могут быть внешними, информация о событиях может приходить от сенсоров или интегрированных систем. Также, события могут генериться изнутри системы.

Это модель расширяется обязательствами для агентов, касательно их функционирования в социальной среде. Эта расширенная модель называется BOID agent architecture [22].

Большой список инструментов приводится в работе [23]. Превалирующая платформа для разработки – Java (JVM – как среда исполнения). На рисунке 8 из этой работы показано распределение агентских сред по областям применения.

General purpose distributed simulations	JADE, Jadex, Jason, EMERALD, MaDKit, CybelePro, JIAC, AgentScape, AnyLogic, GAMA
General purpose agent based simulations	JAS, AgentBuilder, Agent Factory (AFSE), Swarm, MASON, INGENIAS Development Kit, Repast, MaDKit, AgentScape, SeSAm, AnyLogic, NetLogo, GAMA, JAMES II
Scientific simulations	Swarm, MASON, Cormas, Repast (Social Sciences), MaDKit, CybelePro, SeSAm, AnyLogic, GAMA, JAMES II
Dynamic and complex environments	JACK, Cougaar, CybelePro, AnyLogic
Real - world - GIS	AGLOBE, Cougaar (integrated with OpenMap), Repast, CybelePro, SeSAm, AnyLogic, GAMA
Large scale simulations	Cougaar, CybelePro, JIAC, AgentScape, GAMA, JAMES II
Scheduling & planning	Jadex, CybelePro, SeSAm, AnyLogic, NetLogo, GAMA
Mobile computing	JADE, Jadex, Agent Factory
Multiple domains	JADE, Jadex, MaDKit, CybelePro, Cougaar, JIAC, AgentScape, Agent Factory, Repast, SeSAm, AnyLogic, GAMA, JAMES II
Artificial life and behavioral observation	JADE, Jadex, Jason, SeSAm, EMERALD, JACK, Cougaar, CybelePro, AnyLogic, NetLogo, GAMA
Biological & social studies	JADE, SeSAm, Jadex, Jason, EMERALD, JACK, Cougaar, CybelePro, MaDKit, Repast, AnyLogic, NetLogom, GAMA, JAMES II
Economics/eCommerce	JADE, EMERALD, JACK, Cougaar, CybelePro, MaDKit, AnyLogic
Natural resources & environment	Cormas, Swarm, SeSAm, AnyLogic. NetLogo, GAMA, JAMES II

Рис. 8. Инструменты для агентов

Вместе с тем необходимо отметить еще раз, что такого рода программные инструменты не содержат каких-либо “волшебных” инструментов для реализации аналитических систем или систем искусственного интеллекта. Все они по-разному решают один и тот же круг проблем (вопросов):

- идентификация и поиск агентов. Работа с каталогом агентов (каталогом сервисов, скриптов и т.д.)
- поддержка жизненного цикла агентов, включая вопросы инсталляции и возможного перемещения. Перемещение актуально для мобильных систем, когда сервис (агент) явно загружается на другое устройство (на мобильный телефон, например)
- планирование исполнения,
- расстановка и поддержка приоритетов и синхронизации
- взаимодействие агентов друг с другом. Это, как правило, обмен сообщениями
- описание, анализ и учет контекста в работе агентов

На верхнем уровне, функциональность любого агента можно разделить на две большие части. Требование автономности, очевидно, означает, что агент должен быть в какой-то степени самодостаточным в смысле работы с данными. Соответственно, всегда есть некоторый внутренний функционал по обработке данных и функционал (уровень), отвечающий за взаимодействие с другими агентами или пользователями.

Одну из моделей привязки действий к внешним событиям предлагает сервис ifttt [24]. Триггеры (каналы с терминах сервиса) привязаны, в основном, к Интернет

сервисам (e.g. получение письма или уведомления), но могут взаимодействовать и с открытыми API.

Для автоматизации действий в готовой (закрытой) информационной системе используются роботы, эмулирующие работу человека в пользовательском интерфейсе. Здесь смысл состоит в том, что даже если декомпозиция информационной системы невозможна, детального API нет, но действия, которые необходимо выполнять есть, в любом случае, некоторая последовательность операция, которые пользователь выполняет в UI (пользовательском интерфейсе). Ничего другого просто нет. При определенных условиях (триггер, событие) нужно выполнять определенную последовательность действий - создавать документы, менять атрибуты (реквизиты) существующих документов и так далее. Вот выполнение таких последовательностей действий и автоматизируется.

То, что выполнял человек – выполняет записанных скрипт, который, по факту, эмулирует действия человека. Остальное – это уже вопрос терминологии, называть это скриптом, макросом или агентом.

Как примеры такого рода продуктов можно упомянуть Redwood's Enterprise Process Automation [25], UIPath Process Automation [26]. Рисунок 9 иллюстрирует автоматическую подготовку счета в UIPath

Рис.9 Счет в UIPath

Схожие задачи решает Kofax Robotic Process Automation [27]. Основная идея – автоматизации повторяющихся действий. При этом системы могут работать как с ERP инструментами (SAP, Oracle), так и с веб-приложениями.

Robotic Process Automation - сравнительно новая область, которая, активно развивается. Из других решений можно отметить EnableSoft ERP Automation [28], MacroScheduler [29], AutoMate [30], Automation Anywhere [31]. Рисунок 10 иллюстрирует задачи AutoMate.

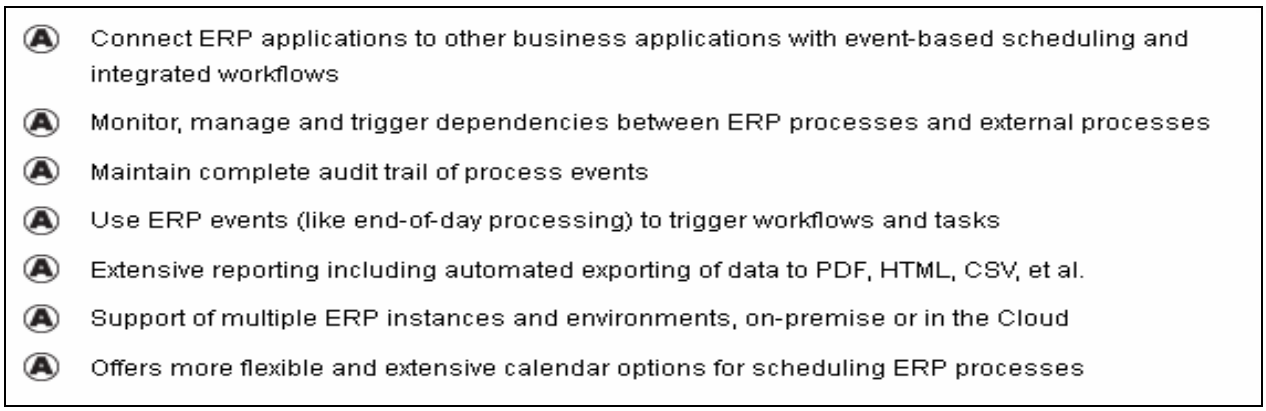

Рис. 10. AutoMate [30].

Одним из направлений, которое может быть привязано к понятию агента или информационного робота, являются, на сегодняшний день, так называемые чат-боты. Это приложения в коммуникационных средах (например, в мессенджерах), которые способны поддерживать диалоги с пользователями. Такой диалог включает как элементы понимания запроса, так и какие-то действия, которые такой бот должен выполнять, чтобы подготовить ответ пользователю. На рисунке 11 изображена карта продуктов из данной области [32]. На этой карте изображены четыре типа таких ботов:

Employee-Bots – боты для конкретных приложений (областей);

General User Interfaces – боты для поддержки общих интерфейсов по взаимодействию. Например, приложение Siri на iPhone попадает в эту категорию;

Bots Contractors – боты с возможностью конфигурации (настройки) для специфических областей;

Bots Factories – мета-инструменты для создания ботов.

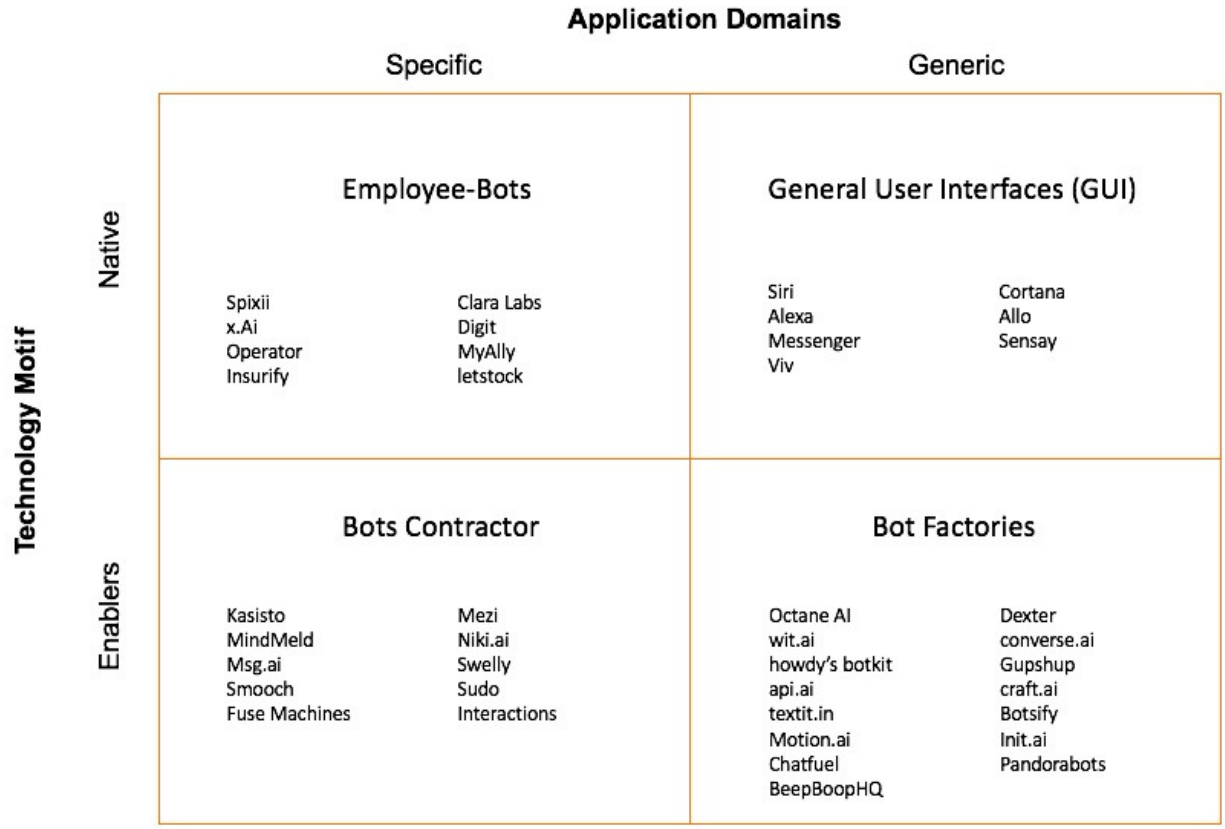

Рис. 11. Чат-боты и инструменты [32]

Классификатор самих чат-ботов по типам реализации приведен на рисунке 12.

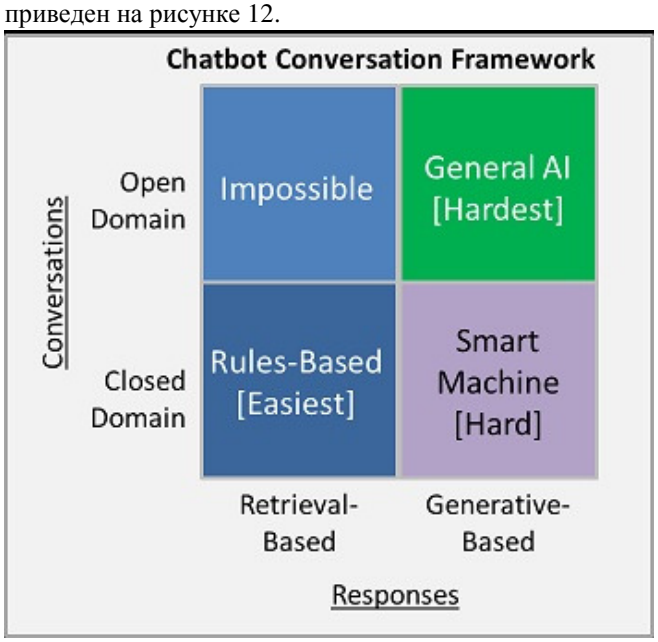

Рис. 12. Типы чат-ботов

Retrieval-based – боты, которые содержат в той или иной форме предопределенные ответы и лишь подбирают подходящий разными алгоритмами.
Generative-based – это боты, которые реально создают (генерируют) ответы.

Здесь есть свои модели. Есть средства разработки. Эта область пока не привязана непосредственно к промышленным информационным системам. В первую очередь, тут все завязано на общение на естественном языке, но именно здесь есть свои модели и искусственный интеллект (в отличие от фреймворков для агентов).

С точки зрения проектирования системы, принятие подхода микро-сервисов естественным образом означает, что в качестве первого шага эти самые микро-сервисы необходимо выявить и описать. Поэтому направление работ в плане построения “new ERP” вполне однозначно. Это декомпозиция задач управления и отчетности. Не объединение их, не поиск общей модели (и общего интерфейса), а наоборот, именно декомпозиция.

Классический подход к такому проектированию – это так называемый Domain-Driven Design [33]. Предметно-ориентированное (проблемно-ориентированное) проектирование - это набор принципов и схем, помогающих разработчикам создавать простые системы объектов. В основе его лежит создание программных абстракций - моделей предметных областей. Практически - это набор правил, которые позволяют принимать правильные проектные решения. Хорошее представление об этом подходе дает, например, презентация [34]. Самый большой смысл здесь в том, что здесь проектирование начинается с проблемы, а не с технических средств, которыми, безусловно, являются агенты. Проблема – первична. Именно из решаемой проблемы должна вытекать необходимость программных агентов, а не наоборот.

Библиография

[1] Намиот Д. Е., Сухомлин В. А., Шаргалин С. П. Программные агенты в ERP системах //International Journal of Open Information Technologies. – 2016. – Т. 4. – №. 6. – С. 49-54.

[2] Bradshaw J. M. Software agents. – MIT press, 1997.

[3] Franklin S., Graesser A. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents //Intelligent agents III agent theories, architectures, and languages. – Springer Berlin Heidelberg, 1996. – С. 21-35.

[4] Coen M. H. Building brains for rooms: Designing distributed software agents //AAAI/IAAI. – 1997. – Т. 1997. – С. 971-977.

[5] Bauer B., Müller J. P., Odell J. Agent UML: A formalism for specifying multiagent software systems //International journal of software engineering and knowledge engineering. – 2001. – Т. 11. – №. 03. – С. 207-230.

[6] Pham V. A., Karmouch A. Mobile software agents: an overview //Communications Magazine, IEEE. – 1998. – Т. 36. – №. 7. – С. 26-37.

[7] Hewitt C. Viewing control structures as patterns of passing messages //Artificial intelligence. – 1977. – Т. 8. – №. 3. – С. 323-364.

[8] Namiot D., Sneps-Sneppe M. On micro-services architecture //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 9. – С. 24-27.

[9] Namiot D., Sneps-Sneppe M. Context-aware data discovery //Intelligence in Next Generation Networks (ICIN), 2012 16th International Conference on. – IEEE, 2012. – С. 134-141.

[10] Sneps-Sneppe M., Namiot D. On Physical Web models //arXiv preprint arXiv:1602.00841. – 2016.

[11] Куприяновский В. П., Намиот Д. Е., Синягов С. А. Киберфизические системы как основа цифровой экономики //International Journal of Open Information Technologies. – 2016. – Т. 4. – №. 2.- С. 18-25.

[12] Rajkumar R. R. et al. Cyber-physical systems: the next computing revolution //Proceedings of the 47th Design Automation Conference. – ACM, 2010. – С. 731-736.

[13] Р. Д. Гимранов, М. И. Лугачев ПОДХОДЫ К ПОСТРОЕНИЮ ЦИФРОВОГО ПРЕДПРИЯТИЯ НА ОСНОВЕ ЭМЕРЖЕНТНОЙ СТРАТИФИКАЦИИ ИНФОРМАЦИОННЫХ СИСТЕМ // Вестник кибернетики. 2016. № 2. С. 165—168.

[14] Mesbahi N. et al. An Agent-Based Modeling for an Enterprise Resource Planning (ERP) //Advanced Approaches to Intelligent Information and Database Systems. – Springer International Publishing, 2014. – С. 225-234.

[15] Galante A. T. Intelligent Agent Technologies: The Work Horse of ERP E-Commerce //International Journal of Intelligence Science. – 2015. – Т. 5. – №. 04. – С. 173.

[16] Lea B. R., Gupta M. C., Yu W. B. A prototype multi-agent ERP system: an integrated architecture and a conceptual framework //Technovation. – 2005. – Т. 25. – №. 4. – С. 433-441.

[17] Kreps J. et al. Kafka: A distributed messaging system for log processing //Proceedings of the NetDB. – 2011. – С. 1-7.

[18] FIPA http://www.fipa.org/specs/fipa00023/ Retrieved: Dec, 2016

[19] JADE http://jade.tilab.com/ Retrieved: Dec, 2016

[20] JACK http://www.aosgrp.com/products/jack/ Retrieved: Dec, 2016

[21] Belief-Desire-Intention software model https://en.wikipedia.org/wiki/Belief%E2%80%93desire%E2%80%93intention_software_model Retrieved: Dec, 2016

[22] Broersen J., Dastani M., van der Torre L. Beliefs, obligations, intentions, and desires as components in an agent architecture //International Journal of Intelligent Systems. – 2005. – Т. 20. – №. 9. – С. 893-919.

[23] Kravari, Kalliopi, and Nick Bassiliades. "A survey of agent platforms." Journal of Artificial Societies and Social Simulation 18.1 (2015): 11.

[24] IF_THEN_THAN_THAT for business https://www.patchworks.co.uk/ifttt-review-how-can-it-help-your-business/ Retrieved: Jan, 2017

[25] Redwood's Enterprise Process Automation https://www.redwood.com/automation/sap-automation Received: Jan, 2017

[26] UIPath Process automation https://www.uipath.com/automate/sap-gui-automation Received: Jan, 2017

[27] Kofax Robotic Process Automation http://www.kofax.com Retrieved: Jan 2017

[28] EnableSoft ERP Automation http://www.enablesoft.com/rpa-industry-solutionserp-automation/ Received: Jan, 2017

[29] MacroScheduler https://www.mjtnet.com/index.htm Retrieved: Jan, 2017

[30] AutoMate http://www.networkautomation.com/sales/erp/ Retrieved: Jan, 2017

[31] Automation AnyWhere https://www.automationanywhere.com/ Retrieved: Jan, 2017

[32] Chat bots https://chatbotsmagazine.com/ai-and-speech-recognition-a-primer-for-chatbots-a63af042526a#.oeb42xwbs Retrieved: Dec, 2016

[33] Evans E. Domain-driven design: tackling complexity in the heart of software. – Addison-Wesley Professional, 2004.

[34] Domain Driven Design 101 http://www.slideshare.net/rdingwall/domain-driven-design-101 Retrieved: Dec, 2016

Information robots in enterprise management systems

Dmitry Namiot, Vladimir Sukhomlin, Eugene Starikov, Sergey Shargalin, Anatoly Stiapshin

Abstract— **The present work is devoted to the use of software agents (information robots) in corporate information systems. We consider the general pattern and purpose of information intermediaries, present their research opportunities and prospects. In the paper, we discuss the requirements for such kind of components. Also, our research is carried out a study of the existing agent software development environments and addresses the design of corporate information systems and enterprise management systems based on models of software agents. The final goal of the project, under which this work has been implemented, is just a new model for corporate information system.**

Keywords—**software agents, information robots, micro-services, ERP systems.**