Программные агенты в ERP системах

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

***Аннотация*—Настоящая работа посвящена рассмотрению использования программных агентов на уровне предприятий в корпоративных информационных системах. Мы рассматриваем общие модели и назначение программных агентов, приводим их классификацию. Вторая часть работы посвящена применению программных агентов в ERP системах или, более точно, вопросам проектирования корпоративных систем на базе моделей программных агентов. Именно новая модель для корпоративной системы является целью проекта, в рамках которого была выполнена данная работа.**

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

I. Введение

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 [1].

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.

Другие градации вводят и дополнительные термины, такие, например, как, например, интеллектуальные агенты [3]. Среди интеллектуальных функций выделяют, в первую очередь, возможность обучения. Автономные агенты [4] характеризуются способностью выбирать задачи и приоритеты. Распределенные агенты, согласно названию, работают на физически различных компьютерах [5]. Мультиагентные системы состоят из агентов, которые для выполнения своих действий должны каким-либо образом общаться с подобными им компонентами [6]. В указанной работе приводится описание формальных механизмов для их спецификации. Мобильные программные агенты (термин возник еще до повсеместного распространения мобильного программного обеспечения и соответствующих средств разработки) обладают способностью перемещаться между машинами сети [7].

Естественно (и это необходимо повторять постоянно) определения в программировании достаточно условны, подвержены некоторой моде, то есть меняются со временем. Формальный спор, является ли какое-то программное обеспечение агентом или нет достаточно бессмысленный. Общая идея (согласие в определениях) 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 [8], актор – это автономный, интерактивный, исполняющий несколько функций объект, который обладает внутренним состоянием и участвует в информационном обмене.

Агенты должны как-то реагировать на контекст (учитывать контекст в своей работе). Важность учета контекста особо подчеркивается для агентов (это реально связано с автономностью). Вместе с тем, сегодня чаще используется термин контекстно-зависимых вычислений (context-aware computing) [9] или ambient mobile intelligence [10]. Согласно последним изысканиям, особенно относящимся к области Интернета Вещей, основой всего являются так называемые киберфизические системы [11]. Соответственно, программные агенты в современном воззрении могут включать и человека в качестве элемента принятия решения [12].

Остальная часть статьи структурирована следующим образом. В разделе II мы останавливаемся на классификации программных агентов. Раздел III посвящен средствам разработки для программных агентов. И раздел IV посвящен использованию программных агентов в ERP системах.

Статья получена 14 апреля 2016.

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

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

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

II. Классификация программных агентов

Традиционно, классификация есть важный момент в описании новых концепций. Применительно к программным агентам, одна из наиболее часто цитируемых работ по классификации есть статья [13]. В ней выделяется только четыре основных типа интеллектуальных программных агентов [14]:

- Агенты-покупатели или торговые боты
- Пользовательские или персональные агенты
- Агенты по мониторингу и наблюдению
- Агенты по добыче и анализу данных

Но здесь, видимо, нужно сделать скидку на время, когда эта классификация готовилась. По классификации авторов агенты-покупатели просматривают сетевые ресурсы с целью получения информации о товарах и услугах. С момента написания этих строк такого рода индексирующие роботы весьма развились и далеко выходят за рамки просто поисковых систем. Другим моментом является то, боты (агенты), во-первых, все больше могут базироваться на семантической информации, представляемой в сети [15]. Во-вторых, большое распространение получили программные интерфейсы (API), которые существуют для большинства популярных проектов. Это сводит работу агента к вызовам API (или определения callbacks, которые будут получать уведомления от API). Самый простой пример – сбор данных из социальных сетей.

Другим моментом является недавний тренд на непосредственное использование так называемых чат-ботов (Telegram, Twitter, Facebook Messenger, etc. [16]) для информационных задач.

Пользовательские или персональные агенты по данной классификации — это интеллектуальные агенты, которые действуют от имени пользователя. Типичные примеры – отправка данных, получение и автоматическая обработка данных. Если проводить параллели с информационными системами (ERP), то это, пожалуй, наиболее близкий пример. Агент делает то, что делал бы пользователь в интерфейсе ERP.

Агенты по мониторингу и наблюдению используются для наблюдения за объектами и передачи информации на оборудование. Здесь термин агент давно употребляется, например, в SNMP мониторинге [17]. Применительно к информационным системам агенты могли бы, например, отслеживать складские запасы. Такой функционал может быть, как включен в общий интерфейс ERP (supply chain system), так и оформлен в виде автономного сервиса. В принципе, автономизация подобного рода сервисов имеет смысл с ростом приложений Интернета Вещей (Internet of Things – IoT). В силу их разнообразия и отсутствия единых стандартов, они обречены быть независимыми от монолитных ERP систем. Как нам кажется, именно так называемый devops [18] является движущей силой внедрения программных агентов или микросервисов (см. ниже). Devops (development и operations) – это подход к разработке программного обеспечения, нацеленный на активное взаимодействие и интеграцию специалистов по разработке и специалистов по сопровождению и эксплуатации. Он предполагает тесную связь и взаимозависимость разработки и эксплуатации программного обеспечения. Это подход для быстрого создания и обновления программных продуктов. Если нет точных моделей (как в IoT), то и обновлять систему придется чаще. А это проще сделать для немонолитного решения – меньше объем обновлений, проще тестировать изменения. Отметим также, что такая декомпозиция никак не отрицает наличия общих баз данных, например. “Хранитель” базы данных – это один из агентов, операции с этой базой – другие агенты.

Последними в этой классификации следуют агенты по добыче и анализу данных. Нам представляется, что в современных условиях это должно быть отнесено к первому пункту.

Отметим также еще одно очевидное различие в именованиях и “стандартных” понятиях. В современной терминологии, программы (сервисы), которые обрабатывают данные из нескольких источников (а именно это и характерно для агентов) чаще всего называют мэшапами. Все современные телекоммуникационные сервисы (Telecom 2.0), а также сервисы Интернета Вещей – это именно мэшапы [19].

III. Средства разработки для программных агентов

В этом разделе мы хотели бы остановиться на программных инструментах (фрейворках) для построения агентов.

Во-первых, это, конечно, спецификация FIPA (Foundation for Intelligent, Physical Agents) [20]. Спецификация описывает абстрактную архитектуру для системы программных агентов (рисунок 1).

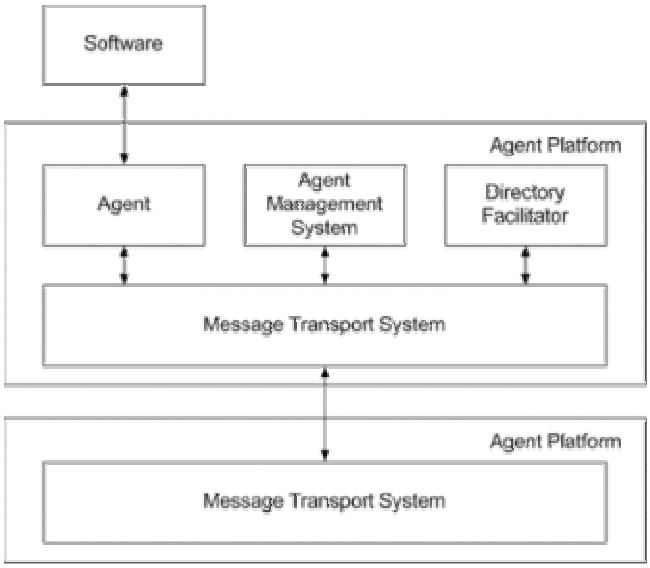

Рис. 1 FIPA

Основная идея платформы – выделение общих уровней, отделение коммуникаций (транспорта) от бизнес-логики.

Как конкретный пример реализации этой спецификации можно назвать JADE (Java Agent

Development) – Open Source пакет для создания программных агентов [21]. 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 2.

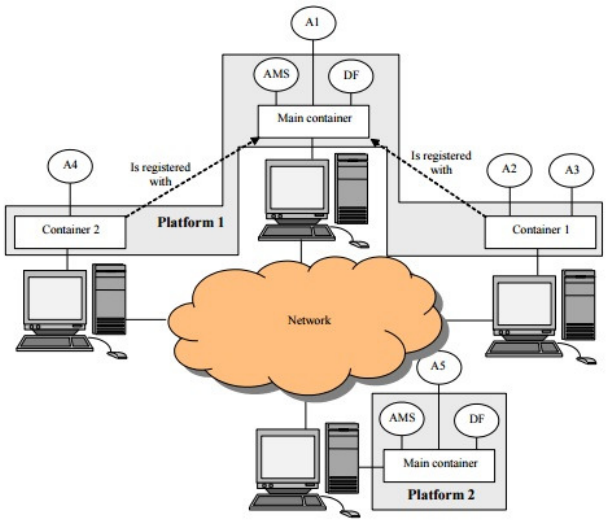

Рис. 2. Контейнеры JADE

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 3).

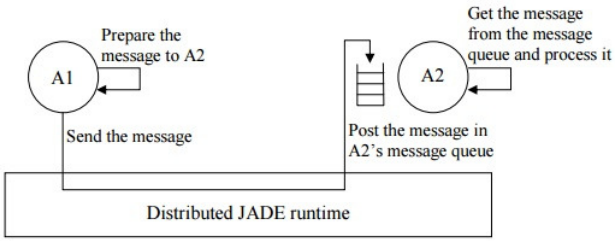

Рис. 3. Обмен сообщениями в JADE

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).

На рисунке 4 показана организация каталога агентов:

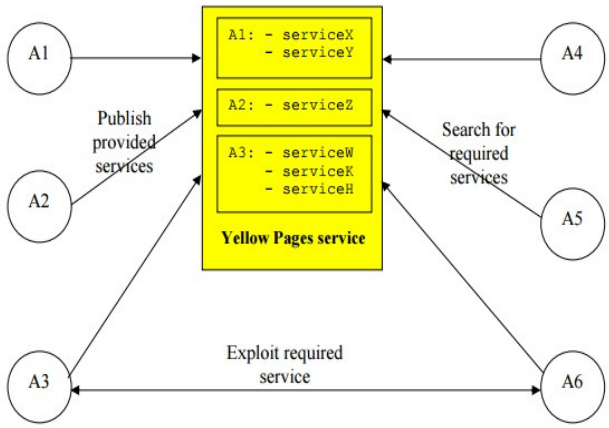

Рис. 4. Каталог в JADE

В целом, система JADE выглядит весьма активно развивающейся. Поиск в Google Scholar показывает свыше 1000 статей, упоминающих JADE с 2014 года.

Описанию платформ для проектирования, разработки и сопровождения систем на базе программных агентов посвящено довольно много работ [22-24]. Последняя из работ [24] содержит, пожалуй, самый большой список инструментов. Большой список инструментов приводится в материалах исследовательской группы The Network for Computational Modeling for SocioEcological Science (CoMSES Net) [25]. Компиляция различных списков для инструментальных средств приводится также и в Википедии [26].

Общие вопросы, которые в той или иной степени решают все средства разработки для программных агентов следующие:

- идентификация и поиск агентов
- жизненный цикл агентов, включая вопросы инсталляции и возможного перемещения

- планирование исполнения,
- приоритеты и синхронизация
- взаимодействие агентов друг с другом
- 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 компонента, отвечающая за взаимодействие.

IV. Программные агенты и ERP системы

По нашему мнению, проектирование информационных систем (ERP, например) имеет следующие основные цели:

- обеспечить гибкость информационной системы и увеличить скорость создания нового функционала;
- избавиться от монолитного кода и облегчить (удешевить, ускорить) обновления системы;
- обеспечить создание и запуск в эксплуатацию нового функционала, модели для которого неизвестны на данный момент, но, возможно, будут созданы позднее из имеющихся базовых элементов;
- переложить на независимые компоненты часть функций (действий), которые до этого выполнялись человеком.

На рисунке 5 представлена обобщенная модель информационной системы [27]

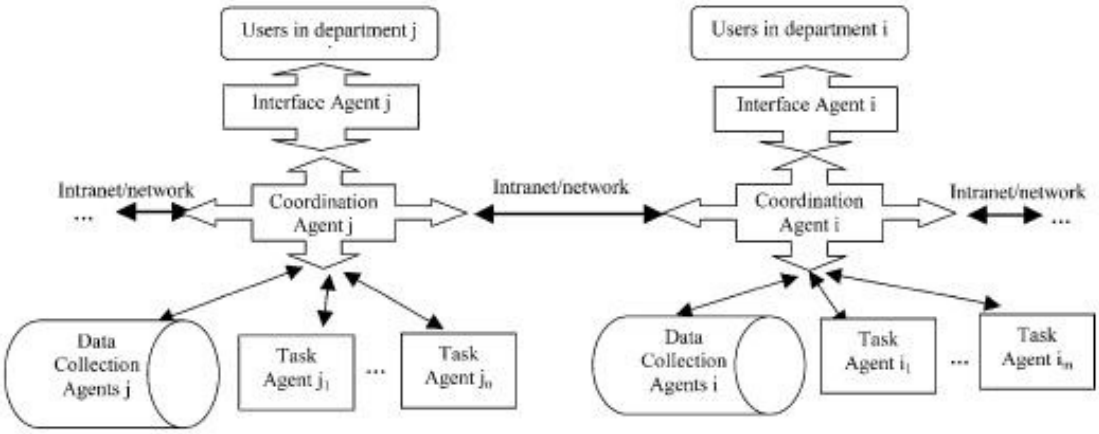

Рис. 5 Мульти-агентная информационная система

Функционал ERP 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.

Нам представляется, что хорошей моделью здесь является активно развиваемая в последнее время концепция микро-сервисов [28]. Согласно этой модели, система строится из набора отдельных сервисов, которые взаимодействуют посредством передачи сообщений. Отличие от сервисной архитектуры (SOA) именно в размерах. Слово микро (в некоторых работах уже и нано) как раз и подчеркивает размеры сервисов. Другое отличие состоит в том, что эти сервисы запускаются (и работают) независимо (та же самая автономность, о которой говорилось и для программных агентов). В SOA отдельные компоненты чаще всего являются элементами одной и той же платформы и автономностью (в смысле исполнения) не обладают.

Микро-сервисы, естественно, облегчают обновление функционала. Они позволяют разрабатывать и обновлять сервисы независимо. Реализации могут быть выполнены на разных языка программирования. Это может быть важно для удешевления процесса – нет необходимости подбирать каких-либо конкретных разработчиков.

Различия агентов и сервисов в таком подходе могут быть чисто терминологические. Например, в банковской системе уведомление пользователей по SMS о совершенных транзакциях – это агент или сервис? Это автономная компонента – то есть агент. Но что изменится, если это назовут сервисом? Сервис также может быть автономным. Такой функционал, как, например, индексация веб-сайтов в Интернет-приложениях чаще называют сервисом, хотя это по каноническим примерам (см. выше) – типичный агент (бот).

Платой за такую модель является усложнение взаимодействия. Обмен данными, скрытый внутри монолитной модели теперь должен быть явно описан. Если количество компонент (микро-сервисов) возрастает по сравнению с сервисной архитектурой, то,

усложняется и взаимодействие [29].

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 [30]. 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 [31].

Библиография

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

[2] 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.

[3] Brenner W., Zarnekow R., Wittig H. Intelligent software agents: foundations and applications. – Springer Science & Business Media, 2012.

[4] 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.

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

[6] 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.

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

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

[9] Dey A. K. Context-aware computing //Ubiquitous Computing Fundamentals. – 2010. – С. 321-352.

[10] Aarts E. et al. Into ambient intelligence. – Springer Berlin Heidelberg, 2006. – С. 1-16.

[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] Lin J., Sedigh S., Miller A. A semantic agent framework for cyber-physical systems //Semantic Agent Systems. – Springer Berlin Heidelberg, 2011. – С. 189-213.

[13] Cummings M., Haag S., McCubbrey D. Management information systems for the information age. – 2003.

[14] Программный агент https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B3%D0%B5%D0%BD%D1%82

[15] Ristoski P., Paulheim H. Semantic Web in data mining and knowledge discovery: A comprehensive survey //Web Semantics: Science, Services and Agents on the World Wide Web. – 2016.

[16] Namiot D. Twitter as a transport layer platform //Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT), 2015. – IEEE, 2015. – С. 46-51.

[17] Stallings W. SNMP, SNMPv2, SNMPv3, and RMON 1 and 2. – Addison-Wesley Longman Publishing Co., Inc., 1998.

[18] Httermann M. DevOps for developers. – Apress, 2012.

[19] Namiot D., Sneps-Sneppe M. On software standards for smart cities: API or DPI //ITU Kaleidoscope Academic Conference: Living in a converged world-Impossible without standards?, Proceedings of the 2014. – IEEE, 2014. – С. 169-174.

[20] FIPA http://www.fipa.org/specs/fipa00023/

[21] JADE http://jade.tilab.com/

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

[23] Macal, Charles, and Michael North. "Introductory tutorial: Agent-based modeling and simulation." Proceedings of the 2014 Winter Simulation Conference. IEEE Press, 2014.

[24] C. Nikolai and G. Madey, "Tools of the trade: A survey of various agent based modeling platforms," Journal of Artificial Societies and Social Simulation, vol. 12, 2009

[25] The Network for Computational Modeling for SocioEcological Science (CoMSES Net) https://www.openabm.org/page/modeling-platforms

[26] Comparison of agent-based modeling software https://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software

[27] 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.

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

[29] Newman S. Building Microservices. – " O'Reilly Media, Inc.", 2015.

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

[31] Domain Driven Design 101 http://www.slideshare.net/rdingwall/domain-driven-design-101

On Software Agents in ERP Systems

Dmitry Namiot, Vladimir Sukhomlin, Sergey Shargalin

Abstract— **This work is devoted to the considerations of the use of software agents at the enterprise level in corporate information systems. We consider general models and assignments for software agents, as well as provide their classification. The second part is devoted to the application of software agents in ERP systems or, more precisely, to the design of enterprise systems models based on software agents. This new model for corporate systems is the final goal of the project, which was carried out this work.**

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