Санкт-Петербургский государственный университет

Математико-механический факультет

Кафедра системного программирования

Яськов Сергей Андреевич

Мультиагентная система поддержки кооперативных покупок

Выпускная работа бакалавра

Допущен к защите
Зав. кафедрой:
д. ф.- м. н., проф. А. Н. Терехов

Научный руководитель:
к. ф.-м. н., доцент Д. Ю. Бугайченко

Рецензент:
к. ф.-м. н., доцент И. П. Соловьев

Санкт-Петербург
2013

Saint Petersburg State University

Faculty of Mathematics and Mechanics

System engineering department

Yaskov Sergey

Cooperative purchasing support multi-agent system

Graduate paper

Admitted for defence.
Head of the chair:
A. Terehov

Scientific supervisor:
D. Bugaychenko

Reviewer:
I. Soloviev

Saint Petersburg
2013

Оглавление

Введение....4
Постановка задачи....5
Обзор предметной области....6
Обзор технологий....8
  Фреймворк JADE....8
  BDI-архитектура....9
  Фреймворк BDI4JADE....10
Архитектура системы....11
  Агент-клиент....12
    Поиск предложения....13
    Формирование коалиций....13
    Отслеживание коалиций....16
  Агент-продавец....18
  Агент-посредник....19
Особенности реализация системы....20
  Передача объектов со взаимными ссылками....20
  BDI-интерпретатор в BDI4JADE....21
Заключение....22
Список литературы....23

Введение

Кооперативные покупки — принцип организации покупок, при котором несколько лиц объединяются в группу для приобретения товаров непосредственно от поставщика или производителя по оптовым ценам.

Основным их преимуществом является цена товаров. Товар приобретается непосредственно у поставщика за счет этого отсутствует торговая наценка магазина: при кооперативных покупках в цену не закладывается расходы продавца на персонал, аренду помещений, а так же прибыль продавца. Оптовая цена может отличатся от розничной до 300%.

Ещё одно серьёзное достоинство — широкий ассортимент товаров. Выбор по каталогам не ограничен ассортиментом магазина и его наличием в месте проживания. Часто на сайтах кооперативных покупок предлагается закупка товаров с зарубежных магазинов или напрямую от производителей товаров, не представленных в России.

Однако сегодня для организации кооперативных покупок люди ищут друг друга сами и сами договариваются. Для этого, как правило, используются различные форумы [1] и группы в социальных сетях [2]. Они обладают весьма посредственными средствами поиска предложений и коалиций а так же неудобными инструментами для отслеживания состояния закупки.

Наша цель предложить некоторую систему, которая позволила бы облегчить организацию таких процессов, сохранив при этом отсутствие центрального управляющего звена.

Задача отличается высокой степенью децентрализованности и динамичности, которые вызваны природой предметной области: ведь мы имеем дело с самоорганизующейся группой людей, а не с некоторой организацией, проводящей коммерческую деятельность. Поэтому предлагается применить мультиагентный подход для решения этой задачи [3, 4].

Постановка задачи

В ходе данной квалификационной работы необходимо:

1. Провести анализ предметной области и потребностей потенциальных пользователей кооперативных покупок;
2. Разработать модель системы поддержки кооперативных покупок;
3. Реализовать прототип системы поддержки кооперативных покупок на основе мультиагентного подхода.

Обзор предметной области

Сегодня сценарий кооперативных покупок выглядит приблизительно следующим образом:

1. Появляется «продавец», изъявляющий готовность совершить оптовую закупку, часто за рубежом. Он очерчивает примерную область того, что будет покупать, дает ссылки на сайты соответствующих брендов или магазинов, анонсирует минимальный объем, при котором поездка состоится, дату и прочие параметры.
2. Появляются «покупатели», каждый из которых формулирует, что и в каком количестве он хочет купить.
3. Поставщик говорит сумму к оплате, а покупатель платит. При этом платежные системы используются редко, чаще всего оплата совершается через внесение денег на банковскую карту, после чего поставщику предоставляется номер платежного поручения или копия чека.
4. Совершается поездка и закупка. При большом количестве заказов совершается несколько поездок. Купленные товары не всегда в точности соответствуют заказам: чего-то может не найтись, что-то подорожать и т. п. Покупателям сообщается о полученных товарах и денежной разнице.
5. Далее начинается развозка. Иногда она организуется автомобилем, который посещает по расписанию определенные точки в определенное время (для крупногабаритных вещей не адресно, а, как правило, около метро), но чаще — самовывозом и по принципу «передам по пути».

При этом на протяжении всей закупки люди ищут друг друга сами и сами друг с другом договариваются.

В этом им помогают различные форумы и группы в социальных сетях. Но они обладают достаточно убогими средствами поиска предложений и коалиций, а так же неудобными инструментами для отслеживания состояния закупки. Зачастую подобные инструменты отсутствуют вовсе.

Обзор технологий

В данном разделе рассмотрены использованные мультиагентные технологии, так как для разработки архитектуры системы был выбран именно мультиагентный подход.

Фреймворк JADE

JADE — фреймворк, предназначенный для разработки мультиагентных систем в соответствии со спецификациями FIPA.

Основными компонентами JADE являются:

- среда исполнения (контейнер) агентов;
- библиотека Java-классов для разработки агентов;
- набор инструментов для администрирования.

Контейнер агентов представляет собой Java-приложение. Каждый агент внутри контейнера — поток. Множество контейнеров составляют агентную платформу в рамках одного сервера. При этом система может включать в себя несколько платформ. В системе обязательно присутствует главный контейнер, содержащий ключевых агентов платформы – Directory Facilitator (DF, «жёлтые страницы») и Agent Management Service (AMS). Все взаимодействия между агентами осуществляется посредством пересылки сообщений языка FIPA ACL.

Библиотека Java-классов позволяет реализовывать агентов со сложным поведением, а также описывать онтологии для их взаимодействия.

Действия, которые совершает агент во время выполнения, описываются с помощью поведений. У каждого агента есть циклическая очередь поведений, которые последовательно исполняются.

Наиболее часто используются следующие типы поведений:

1. OneShotBehaviour — простое поведение. Исполняющееся ровно один раз, после чего удаляется из очереди;
2. CyclicBehaviour — также простое. Исполняющееся постоянно и циклично;
3. TickerBehaviour — простое циклическое поведение, исполняющееся с заданной периодичностью;
4. SequentialBehaviour — композитное поведение, т. е состоящее из поведений других типов (как простых, так и композитных). Данное поведение выполняет свои составные части строго по очереди;
5. FSMBehaviour — композитное поведение, конечный автомат поведений. Незаменимо в случае, если необходимо планировать выполнение поведений динамически.

Также JADE предоставляет набор инструментов для администрирования мультиагентных систем. В него входят такие инструменты, как Remote Management Agent (RMA), агент с графическим интерфейсом, позволяющий управлять платформой (удаление, создание агентов, обмен сообщениями). Кроме него, очень полезен инструмент Sniffer, позволяющий отслеживать обмен ACL-сообщениями в виде диаграммы последовательностей [5, 6].

BDI-архитектура

BDI (belief-desire-intention software model) — это программная модель разработки интеллектуальных агентов. Она состоит из следующих компонентов:

- Beliefs (убеждения) — представляют информационное состояние агента. Другими словами, это «представление» агента об окружающем его мире;
- Desires (желания) — представляют сценарии или ситуации, которые

агент хотел бы реализовать;

- Intentions (намерения) — те желания агента, достижения которых он запланировал, то есть выбрал план действий по их достижению. План — как правило некоторое поведение (простое либо составное) [7].

Фреймворк BDI4JADE

Один из наиболее широко распространённых способов реализации интеллектуальных агентов является BDI-архитектура.

Однако JADE её не реализует. Чтобы сохранить мощную функциональность JADE и при этом воспользоваться преимуществами BDI-архитектуры было решено использовать фреймворк BDI4JADE [8].

В отличие от аналогов (JACK, Jason, Jadex), BDI4JADE применяет Java, а не DSL[1] для определения агентов. Это позволяет пользоваться всеми достоинствами Java при реализации агентов.

Кроме того, являясь надстройкой над JADE, BDI4JADE позволяет пользоваться всеми преимуществами JADE.

Таким образом, выбранный фреймворк сочетает в себе достоинства Java, JADE и BDI-архитектуры.

1 DSL — Domain-specific Language, например XML.

Архитектура системы

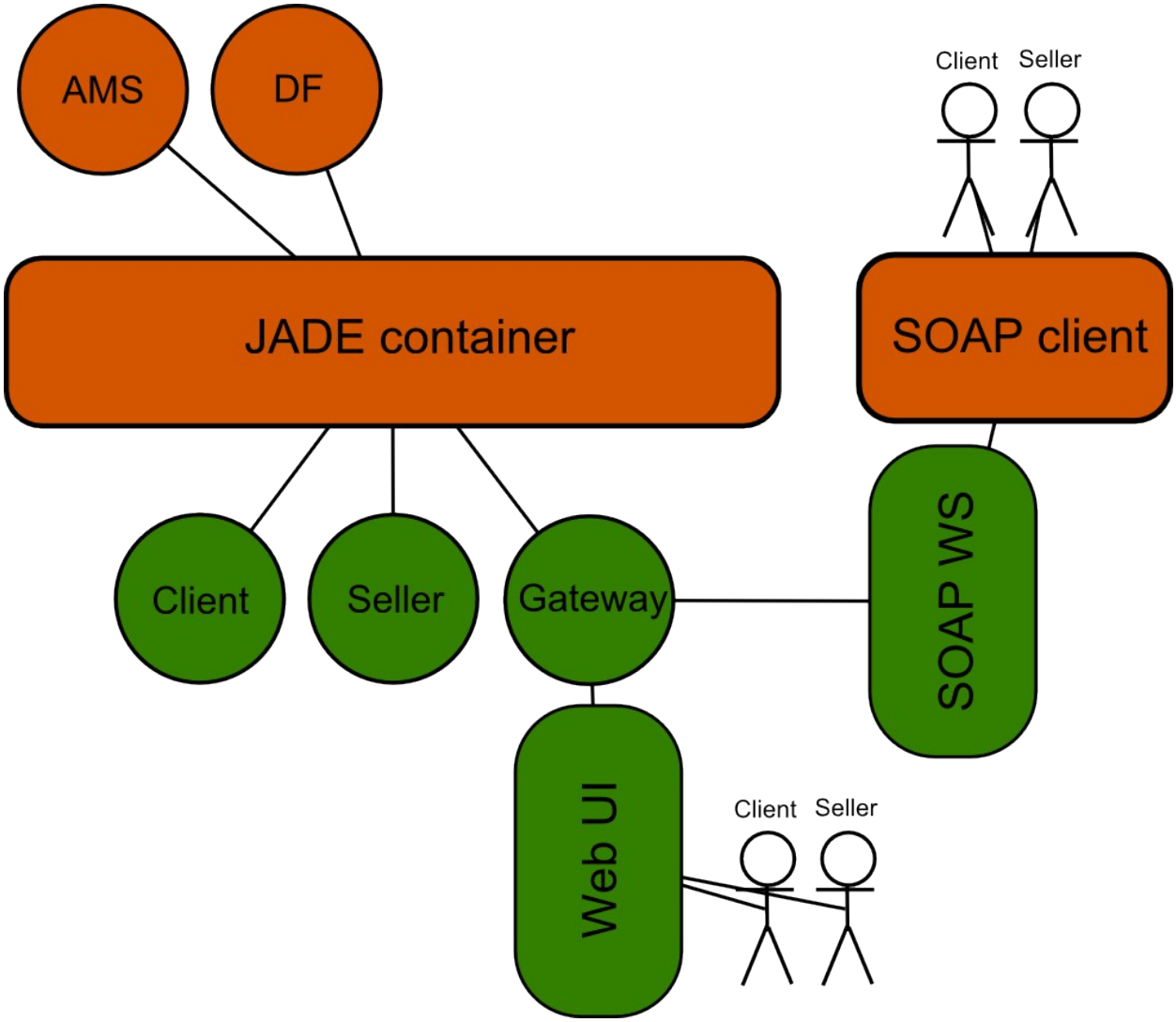

Рисунок 1: Архитектура мультиагентной системы поддержки кооперативных покупок

В ходе квалификационной работы была разработана архитектура системы поддержки кооперативных покупок.

Система мультиагентная, то есть представляет собой множество взаимодействующих интеллектуальных агентов. Агенты принадлежат к различным классам.

Помимо контейнера агентов, JADE предоставляет некоторые готовые агенты-службы. Из них в ходе работы использовались два:

- AMS — agent management system. Служба, позволяющая добавлять и

удалять агентов.

- DF — directory facilitator. Служба «жёлтых страниц». Позволяет агентам искать друг друга по имени и прочим атрибутам, указанным при регистрации агента в службе «жёлтых страниц» [9].

В ходе работы были созданы три основные класса агентов:

- Client — агент-клиент, представляющий интересы покупателей в системе;
- Seller — агент-продавец, представляющий интересы продавцов. Продавец в данном случае не магазин или организация, а человек, берущийся совершить кооперативную покупку и ищущий желающих в ней поучаствовать.
- Gateway — агент-посредник. Позволяет платформе JADE взаимодействовать с «внешним миром».

«Внешним миром» в данном случае является:

- Web-интерфейс пользователя — JSF-приложение, предоставляющее клиентам и продавцам графический интерфейс для взаимодействия с системой.
- SOAP Web-служба — CXF-приложение, позволяющее данной системе взаимодействовать с любыми другими службами (например, логистическими системами), поддерживающими протокол SOAP.

Далее подробнее описаны классы созданных в ходе квалификационной работы агентов.

Агент-клиент

Покупатель, как правило, хочет приобрести товар по минимально возможной цене в кратчайшие сроки, затрачивая при этом минимум усилий. Агент-клиент призван предоставить покупателю инструменты поиска максимально

выгодного предложения, поиска наиболее быстро формирующихся коалиций для такого предложения и отслеживания изменений в коалициях, соответствующих этому предложению. Эти инструменты в совокупности способны исполнить вышеуказанные желания пользователя. Далее подробнее о каждом из этих инструментов.

Поиск предложения

Чтобы пользователь смог найти интересующее его предложение, реализован механизм тегов. Реализованное облако тегов не обладает иерархией. С одной стороны, это уменьшает точность поиска, но, с другой стороны, позволяет пользователям самим поддерживать теги в актуальном состоянии, что согласуется с отсутствием в данной системе центрального управляющего звена или, говоря иначе, с тем, что пользователи в данном случае представляют собой самоорганизующуюся группу людей.

Агент-покупатель опрашивает агентов-продавцов и формирует в своих представлениях список предложений. Этот список сортируется одновременно по степени соответствия поисковому запросу пользователя и по цене. Если для какого-то предложения множество его тегов вообще не пересекается с множеством тегов запроса, то такое предложение вообще не попадёт в список.

Формирование коалиций

Предположим, пользователь выбрал одно из предложений и нажал на кнопку «Купить». Агент-покупатель при этом получит от агента-посредника соответствующую команду с идентификатором выбранного пользователем предложения.

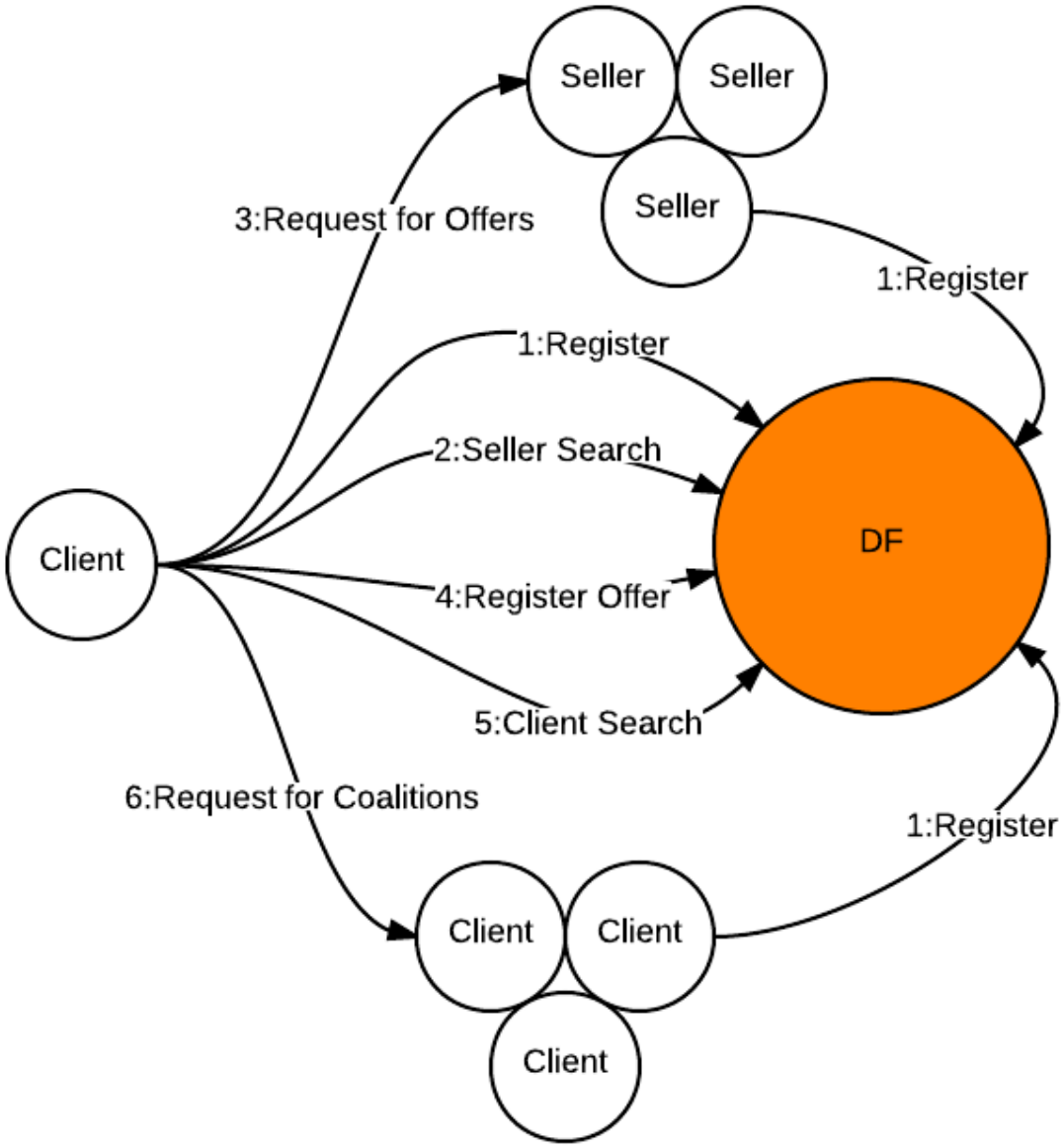

Рисунок 2: Поиск и формирование коалиций

Поскольку речь идёт об оптовых закупках, агент-покупатель, после регистрации у себя предложения, должен найти «единомышленников», которые желают приобрести данное предложение, иначе говоря, зарегистрировали его у себя ранее.

На рисунке 2 представлена общая схема процесса регистрации предложения (говоря точнее, регистрация желания пользователя-покупателя ответить на предложение) и последующих процессов поиска и формирования коалиций. Для этого агентом-покупателем и другими участниками кооперативных покупок осуществляются следующие действия:

1. Register. Мы подразумеваем, что все агенты-покупатели (в том числе выбранный нами в качестве рассматриваемого) и все агенты-продавцы обязательно регистрируются в жёлтых страницах;
2. Seller Search. Выделенный агент-покупатель запрашивает у жёлтых страниц список агентов-продавцов и сохраняет его у себя;
3. Request for Offers. Выделенный агент-покупатель запрашивает у агентов-продавцов список предложений и находит в этом списке то предложение, которое он в данный момент обрабатывает (используя полученный от пользователя уникальный идентификатор);
4. Register Offer. Выделенный агент-покупатель сообщает жёлтым страницам, что с этого момента его интересует предложение с данным уникальным идентификатором. Кроме этого, он сохраняет это предложение в своих представлениях;
5. Client Search. Выделенный агент-покупатель запрашивает у жёлтых страниц список тех агентов-покупателей, которые также «интересуются» этим предложением, или, говоря иначе, уже когда-то ранее выполнили шаг 3. Полученный список также сохраняется в представлениях;
6. Request for Coalitions. Агент-покупатель формирует у себя список коалиций, запрашивая их у тех агентов-покупателей, которые были получены на шаге 5. Из этого списка выбирается коалиция, которая будет закрыта быстрее остальных. Прогноз скорости закрытия осуществляется на основании даты появления этой коалиции, текущей даты и текущего количества её участников (см. рис. 3). Если ни одной коалиции найдено не было, агент-покупатель создаёт новую. Создание новой коалиции, как и присоединение к уже существующей, заключается в сохранении в представлениях агента факта принадлежности этого агента к коалиции, т. е. добавлении в

соответствующий список записи вида (**coalitionId**, **offerId**, **count**), где **coalitionId** — идентификатор коалиции, он генерируется при её создании; **offerId** — идентификатор предложения, на которое ориентированна данная коалиция (коалиция может быть ориентирована только на одно предложение, хотя для одного предложения может быть сразу несколько коалиций); **count** — количество единиц товара данного предложения, которое приобретает данный агент в данной коалиции.

$$t_{\text{before closing}} = \frac{N_{\text{closing}} - N_{\text{current}}}{\frac{N_{\text{current}}}{t_{\text{current}} - t_{\text{creation}}}}$$

Рисунок 3: Оценка времени до закрытия коалиции

Отслеживание коалиций

Число участников коалиции растёт. В конце концов это приведёт к тому, что коалиция будет окончательно сформирована и закрыта. Коалиция считается сформированной, если суммарное число единиц товара, покупаемое всеми её участниками сравняется (или превысит) количество единиц товара, заявленное в предложении агента-продавца (т. е. минимальный объём закупки, при котором она вообще состоится). Агент-продавец, владелец предложения, должен быть оповещён о закрытии коалиции и проинформирован о её составе. Проинформированный таким образом агент-продавец немедленно оповещает пользователя-продавца через агента-посредника.

Поскольку центрального управляющего звена у коалиции нет, обязанность на отслеживание момента закрытия коалиции ложится на её участников, а точнее на последнего агента-покупателя, добавленного в неё.

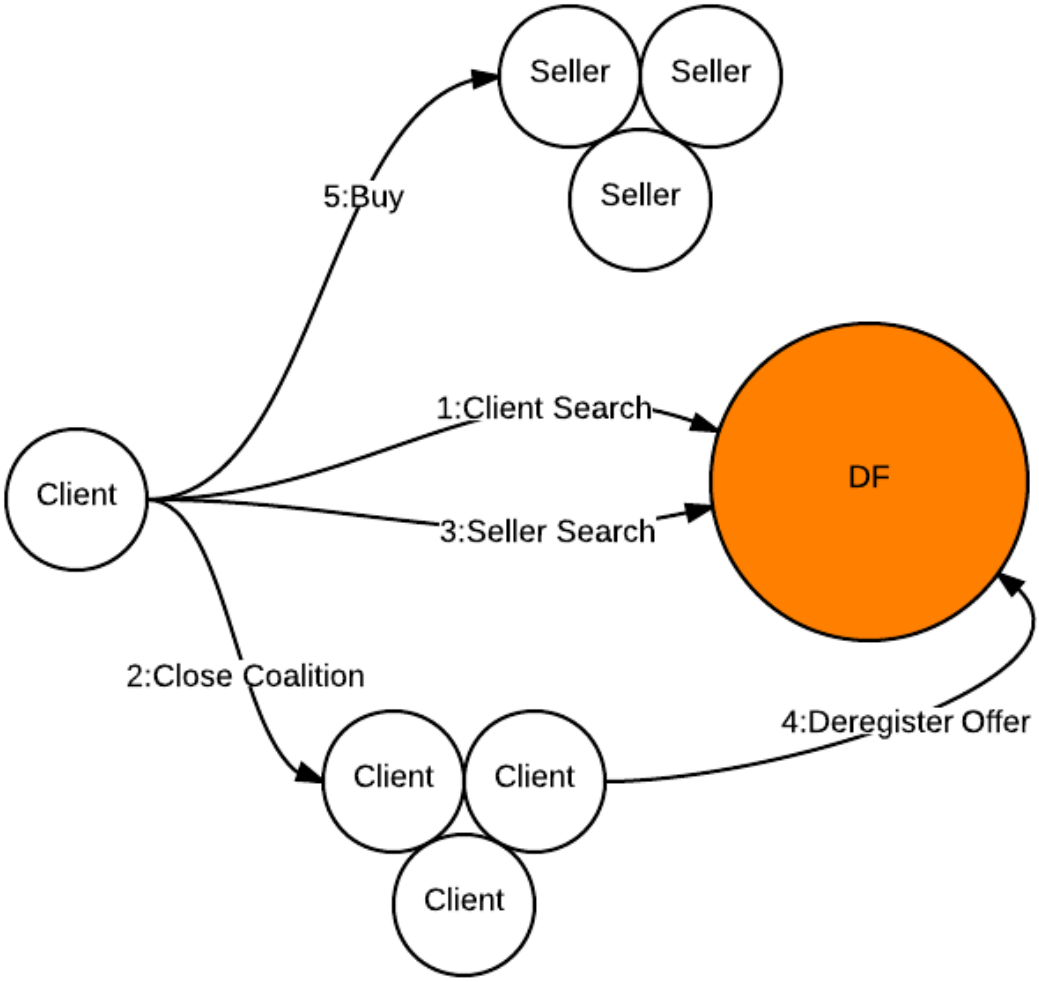

Рисунок 4: Отслеживание коалиций

На рисунке 4 представлена общая схема процесса отслеживания момента закрытия коалиции. Для этого агентом-клиентом и другими участниками кооперативных покупок осуществляются следующие действия (предполагается, что к этому моменту регистрация агентов и предложений в жёлтых страницах уже была проведена):

1. Client Search. Агент-клиент составляет список клиентов, используя DF. В список включаются клиенты, приобретающие то же предложение, что и рассматриваемое в контексте отслеживаемой коалиции;
2. Close Coalition. Агент-клиент оповещает клиентов из списка о том, что группа набрана и должна быть закрыта. В этом же диалоге клиенты присылают в ответ свои имена-идентификаторы и количество покупаемых единиц товара;

3. Seller Search. Агент-клиент обновляет список агентов-продавцов, используя DF;
4. Deregister Offer. Другие клиенты тем временем оповещают DF, что данное предложение (говоря точнее, приложение с данным идентификатором) ими больше не рассматривается как покупаемое;
5. Buy. Агент-клиент к этому моменту уже собрал необходимую информацию о закрываемой группе. На этом шаге он отправляет её соответствующему агенту-продавцу: «коалиция с такими-то участниками и таким-то количеством единиц товара закрывается и можно совершать оптовую закупку».

Агент-продавец

Предположим, появился некий пользователь-продавец, желающий приобрести товары оптом. Другими словами, продавец собрался создать предложение.

Продавец должен обладать возможностью разместить своё предложение в системе и обеспечить к нему доступ пользователей-покупателей, чтобы они имели возможность ответить на предложение. Кроме того, продавец должен быть оповещён о том, что набралось необходимое количество покупателей и можно совершать оптовую закупку.

Чтобы предоставить пользователю-продавцу вышеуказанные возможности, мной был реализован агент-продавец.

У этого агента есть три основные задачи. Первая заключается в управлении предложениями пользователя: добавление, удаление и редактирование предложений. Вторая — в предоставлении своих предложений по требованию агентов-покупателей. Третья задача заключается в оповещении пользователя-продавца о том, что необходимое количество желающих откликнуться на предложение достигнуто и можно совершать закупку.

Агент-посредник

Для того, чтобы платформа JADE могла взаимодействовать с внешним миром, необходимо реализовать агента-посредника, принимающего команды от внешних по отношению к платформе модулей системы (например, Web-интерфейса пользователя или Web-сервисов) и переадресующего их нужным агентам платформы. В обязанности агента-посредника входит также получение ответов от агентов платформы и возвращение этих ответов пользователю.

Фреймворк JADE предоставляет класс GatewayAgent. Агент, наследующий этот класс и реализующий необходимые абстрактные методы, автоматически становится агентом-посредником. После регистрации агента-посредника в платформе, он начинает принимать и переадресовывать запросы от внешних по отношению к платформе модулей системы.

Мной был реализован агент MediatorAgent, наследующий абстрактный класс GatewayAgent и обеспечивающий получение команд пользователя, делегирование их выполнения соответствующим агентам платформы и отправку результатов выполнения обратно пользователю, если это необходимо.

Особенности реализация системы

В ходе квалификационной работы мной был реализован прототип мультиагентной системы поддержки кооперативных покупок.

Далее рассмотрены некоторые проблемы, решённые в процессе реализации прототипа.

Передача объектов со взаимными ссылками

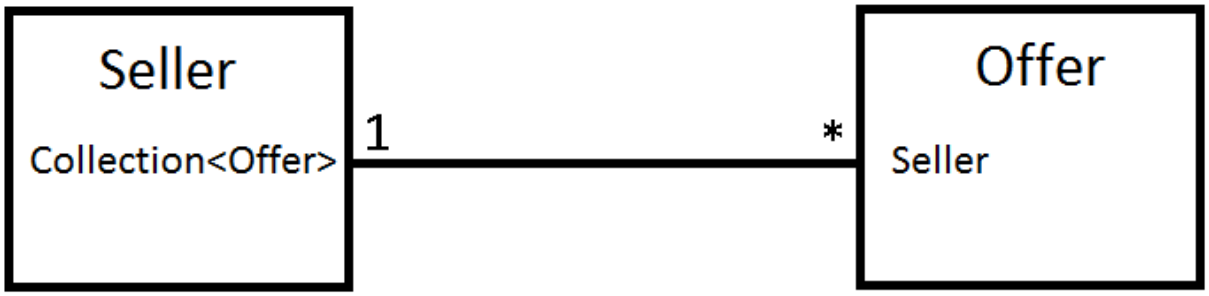

Рисунок 5: Пример взаимных ссылок

JADE позволяет передавать Java-объекты в ACL-сообщениях. Для этого он сериализует их в виде некоторого текстового представления и затем передаёт в теле ACL-сообщения.

Этот механизм удобен, а зачастую практически незаменим. Но есть у него и неприятная особенность: взаимные (рекурсивные, перекрёстные) ссылки не сериализуются.

Для примера рассмотрим ситуацию, проиллюстрированную на рисунке 5. Изначально модель предметной области прототипа системы была устроена так, что, к примеру, объект **Seller** содержал коллекцию своих предложений и, при этом, объект **Offer** содержал ссылку на своего владельца (на Seller'a). При попытке сериализовать средствами JADE объекты, устроенные таким образом, приложение вылетало с исключительной ситуацией StackOverflowError.

Для решения этой проблемы помимо модели предметной области также была

спроектирована коммуникационная модель, которая представляет собой «упрощённую» модель предметной области (без взаимный ссылок) и используется агентами для обмена сообщениями: сообщение содержит некоторый набор фактов, а агент при его получении «накладывает» этот набор фактов на свою модель предметной области.

BDI-интерпретатор в BDI4JADE

BDI-интерпретатор — бесконечное циклическое поведение, непрерывно анализирующее список желаний агента и превращающее эти желания в намерения, а так же удаляющее из списка желаний осуществлённые или неосуществимые желания. В нашем случае BDI-интерпретатор предоставлен фреймворком BDI4JADE.

В BDI4JADE интерпретатор изначально реализован в виде постоянно исполняющегося CyclicBehaviour, а это полностью загружает процессор.

Для решения проблемы в коде самого фреймворка BDI-интерпретатор был изменён: его CyclicBehaviour теперь блокируется до тех пор, пока в списке желаний агента не произойдут какие-либо изменения. Если изменений нет, интерпретатор простаивает и не загружает процессор.

Заключение

В процессе данной квалификационной работы выполнено следующее:

1. Произведён анализ предметной области и потребностей потенциальных пользователей кооперативных покупок;
2. Разработана модель системы поддержки кооперативных покупок;
3. Реализован прототип системы поддержки кооперативных покупок на основе мультиагентного подхода.

Список литературы

[1] Форум кооперативных закупок [Электронный ресурс]. – Режим доступа: http://sptovarov.ru/ (Дата обращения: 24.12.2012).

[2] Группа кооперативных закупок [Электронный ресурс]. – Режим доступа: http://vk.com/shopogolike (Дата обращения: 27.12.2012).

[3] Бугайченко Д. Ю., Соловьев И. П. Абстрактная архитектура интеллектуального агента и методы ее реализации // Системное программирование. – 2005. – С. 36-67.

[4] Michael Wooldridge. An Introduction to MultiAgent Systems. – 2002. – С. 365.

[5] Bellifemine F.L., Caire G., Greenwood D. Developing Multi-Agent Systems with JADE // Wiley Series in Agent Technology. – 2007. – С. 303.

[6] JADE фреймворк [Электронный ресурс]. – Режим доступа: http://jade.tilab.com/ (Дата обращения: 03.05.2013).

[7] Rao M. P., Georgeff. BDI-agents: From Theory to Practice // Proceedings of the First International Conference on Multiagent Systems. – 1995. – С. 14.

[8] Nunes I., Lucena C. J. P., Luck M. BDI4JADE: a BDI layer on top of JADE. – 2011. – С. 88-103.

[9] Спецификации FIPA [Электронный ресурс]. – Режим доступа: http://www.fipa.org/specs/ (Дата обращения: 03.05.2013).