ЛАБОРАТОРНАЯ РАБОТА № 6
Использование стандартизированных протоколов взаимодействия агентов

Цель работы: научиться пользоваться стандартизированными организацией FIPA протоколами взаимодействия агентов.

6.1. Теоретические сведения

6.1.1. Протоколы взаимодействия агентов

FIPA определила набор стандартных протоколов взаимодействия, которые могут использоваться как шаблоны при построении разговоров агентов. Для каждого разговора платформа JADE различает 2 роли: *Инициатор* (агент, начинающий беседу) и *Респондент* (агент, участвующий в беседе как получатель запроса *Инициатора*). JADE предоставляет готовые классы поведения для двух указанных ролей в разговорах для большинства стандартизованных протоколов взаимодействия. Эти классы находятся в пакете jade.proto и реализуют набор методов работы с поведением агента на разных стадиях разговора.

Поведение Инициатора заканчивается при достижении любого заключительного состояния протокола взаимодействия. Все типы поведения Инициатора, кроме FipaRequestInitiatorBehaviour, позволяют одновременно общаться с несколькими респондентами, т. е. поддерживают режим 1:N.

6.1.2. Протоколы AchieveRE

Сообщение в языке FIPA-ACL представляет собой речевой акт и рассматривается как одно из действий, которое может выполнить агент. Для каждого речевого акта стандарты FIPA определяют предусловия их выполнимости и ожидаемый эффект действия, названный *рациональным эффектом*. Предусловия выполнимости – это условия, которые должны быть выполнены прежде чем агент сможет совершить речевой акт, т. е. отправить ACL-сообщение.

Согласно стандартам FIPA, отправивший сообщение агент не должен полагать по умолчанию, что рациональный эффект данного действия обязательно достигнут. Например, это может произойти вследствие того, что агент-приемник, являющийся автономным агентом, может проигнорировать полученное сообщение, исходя из своих целей. С учетом этого обстоятельства, речевое взаимодействие не сводится к отправке единственного сообщения, а реализуется в соответствии с протоколом, предполагающим, что агент-

отправитель инициирует общение и проверяет, был ли достигнут ожидаемый рациональный эффект.

Стандарты FIPA определяют множество протоколов взаимодействия, позволяющих инициатору проверить, был ли достигнут ожидаемый рациональный эффект речевого акта (Achieve Rational Effect). Наиболее важными из этих протоколов являются: FIPA-Request, FIPA-query, FIPA-Request-When, FIPA-recruiting, FIPA-brokering. Базовая структура этих протоколов одинакова, она представлена на рис. 6.1.

Респондент может в ответ на запрос Инициатора сообщить, что он не понял запроса (речевой акт **not-understood**) или отказывается от достижения рационального эффекта речевого акта (речевой акт **refuse**).

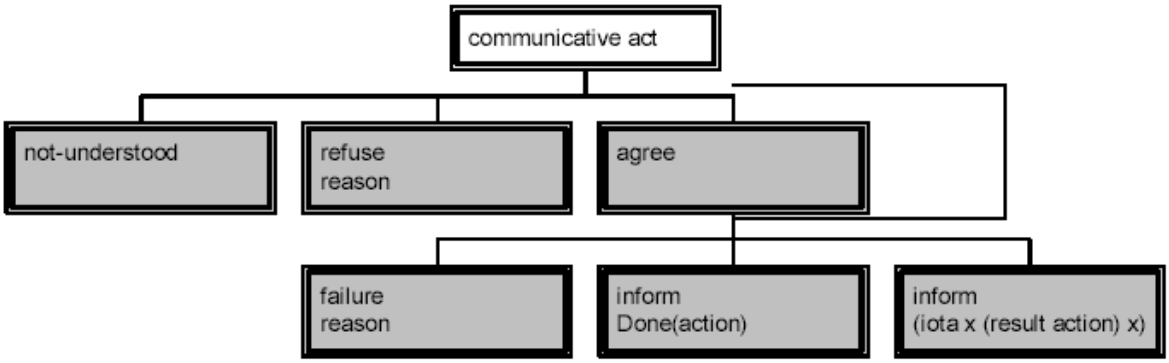

Рис. 6.1. Структура протоколов взаимодействия

В случае согласия выполнить (возможно, в будущем) запрос Инициатора, обеспечив тем самым рациональный эффект, Респондент отвечает речевым актом **agree**.

После выполнения действия Респондент должен сообщить о результате с помощью речевого акта **inform** (информирование), если рациональный эффект достигнут, а в противном случае – с помощью речевого акта **failure** (неудача).

Среда JADE содержит несколько программных классов AchieveREInitiator/Responder, поддерживающих реализацию данных протоколов взаимодействия агентов. Рассмотрим эти классы более подробно.

6.1.3. Класс AchieveREInitiator

Класс AchieveREInitiator соответствует агентам, инициирующим взаимодействие. Экземпляр этого класса может быть создан следующим вызовом конструктора:

```
new AchieveREInitiator(this, msg),
```

где this – объект, представляющий агента; msg – экземпляр класса ACLMessage.

В момент вызова конструктора объект ACLMessage может быть определен не полностью. Метод prepareRequests() может быть переопределен таким образом, чтобы возвращать полный список ACL-сообщений (вектор объектов ACLMessage), которые будут посланы.

Все методы можно переопределить в дочернем классе.

Для обработки полученных сообщений могут использоваться следующие 3 метода:

– *handleOutOfSequence* обрабатывает все неожиданно полученные сообщения, имеющие корректные значения полей conversation-id или in-reply-to;

– *handleAllResponses* обрабатывает все полученные первые ответы (not-understood, refuse, agree) и для каждого полученного ответа вызывает соответствующий его типу сообщения метод *handleNotUnderstood/Refuse/Agree*. Переопределение этого метода наиболее эффективно в случае разговора 1:N;

– *handleAllResultNotifications* обрабатывает все полученные вторые ответы (failure, inform) и для каждого полученного ответа вызывает соответствующий его типу сообщения метод *handleFailure/Inform*. Переопределение этого метода наиболее эффективно в случае разговора 1:N.

Информация об ACL-сообщениях, с которыми работает агент, может быть получена вызовом метода getDataStore(). Для указания конкретных требуемых результатов используются различные ключи:

– *getDataStore().get(ALL_RESPONSES_KEY)* возвращает вектор объектов ACLMessage со всеми первыми ответами (not-understood, refuse, agree);

– *getDataStore().get(ALL_RESULT_NOTIFICATIONS_KEY)* возвращает вектор объектов ACLMessage со всеми вторыми ответами (failure, inform);

– *getDataStore().get(REQUEST_KEY)* возвращает объект ACLMessage, являющийся параметром конструктора класса;

– *getDataStore().get (ALL_REQUESTS_KEY)* возвращает вектор объектов ACLMessage, возвращаемых методом prepareRequests.

Для реализации временно′го контроля за выполнением протокола в поле reply-by конструктора ACLMessage указывается значение тайм-аута.

6.1.4. Класс SimpleAchieveREInitiator

Данный класс соответствует упрощенному варианту реализации роли инициатора. Главное отличие SimpleAcheveREInitiator от AchieveREInitiator состоит в том, что данная версия протокола не позволяет программисту регистрировать прикладные поведения как методы состояний протокола. В данном варианте поддерживается только режим взаимодействия 1:1. Если программист задает в ACLMessage больше одного получателя, сообщение будет отослано только первому получателю.

6.1.5. Класс AchieveREResponder

Данный класс соответствует роли респондента. Конструктору должен быть передан в качестве аргумента шаблон, определяющий тип получаемого ACL-сообщения (объекта ACLMessage).

Шаблон сообщения может быть создан с использованием метода createMessageTemplate. Кроме того, при необходимости могут использоваться более частные шаблоны, в таких случаях для каждого возможного агента-отправителя необходимо создать отдельные экземпляры класса.

Класс может быть легко расширен. Метод prepareResponse вызывается, когда получено сообщение инициатора, и необходимо послать ответ (например, **agree**).

Метод prepareResultNotification вызывается, когда должен быть достигнут рациональный эффект (например, выполнено действие по протоколу FIPA-Request), нужно отправить инициатору заключительное сообщение (например, **inform(done)**).

Получение информации об обрабатываемых агентом ACL-сообщениях с использованием метода getDataStore() и ключей аналогично описанному выше. Возможны следующие варианты:

– *getDataStore().get(REQUEST_KEY)* возвращает объект ACLMessage, полученный от инициатора;

– *getDataStore().get(RESPONSE_KEY)* возвращает объект ACLMessage, посланный инициатору первым;

– *getDataStore().get(RESULT_NOTIFICATION_KEY)* возвращает объект ACLMessage, посланный инициатору вторым.

6.1.6. Класс SimpleAchiveREResponder

Данный класс соответствует упрощенному варианту реализации роли респондента. Главное отличие SimpleAchiveREResponder от AchiveREResponder состоит в том, что данная версия протокола не позволяет программисту регистрировать прикладные *Поведения* как методы состояний протокола.

6.1.7. Пример реализации протокола

Протоколы взаимодействия агентов в соответствии со стандартами FIPA могут быть достаточно просто реализованы в среде JADE с использованием описанных выше классов.

Следующий пример показывает, как добавить агенту поведение инициатора в протоколе FIPA-Request:

```
ACLMessage request =
                    new ACLMessage(ACLMessage.REQUEST);
request.setProtocol(FIPAProtocolNames.FIPA_REQUEST);
request.addReceiver(new AID("receiver",
                                      AID.ISLOCALNAME));
myAgent.addBehaviour(new AchieveREInitiator(myAgent,
                                                request)
{ protected void handleInform(ACLMessage inform)
  {
    System.out.println("Protocol finished. Rational
                        Effect achieved. Received the
                        following message: "+inform);
  }
});
```

Следующий пример показывает, как добавить агенту поведение респондента в протоколе FIPA-Request:

```
MessageTemplate mt =
                AchieveREResponder.createMessageTemplate
                       (FIPAProtocolNames.FIPAREQUEST);
myAgent.addBehaviour(new AchieveREResponder
                                           (myAgent, mt)
{
  protected ACLMessage prepareResultNotification
```

```
                (ACLMessage request, ACLMessage response)
  {
    System.out.println("Responder has received the
                        following message: " +request);
    ACLMessage informDone = request.createReply();
    informDone.setPerformative(ACLMessage.INFORM);
    informDone.setContent("inform done");
    return informDone;
  }
});
```

6.1.8. Протокол FIPA-Contract-Net

Протокол контрактной сети (FIPA-Contract-Net) позволяет инициатору посылать предложение о выполнении некоторого действия нескольким респондентам, оценивать поступившие от них предложения и выбирать наиболее предпочтительный вариант (или отклонять все).

Базовая структура протокола представлена на рис. 6.2. Детальное описание протокола контрактной сети содержится в спецификациях FIPA [4].

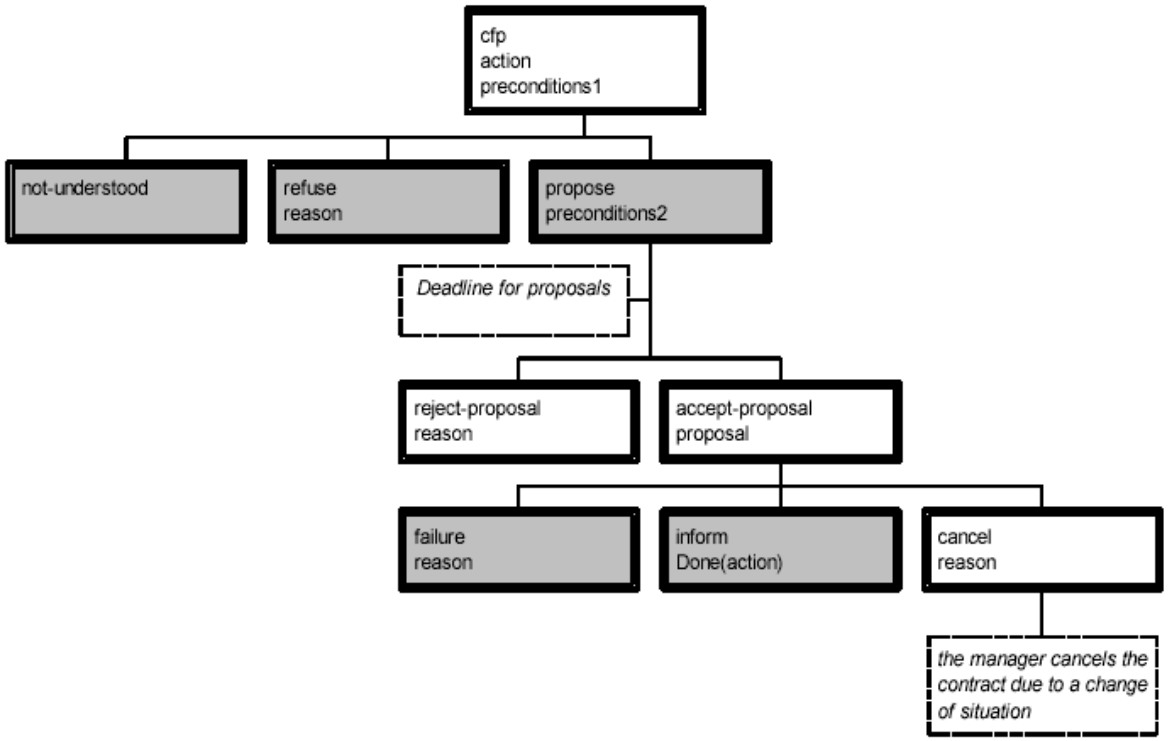

Рис. 6.2. Структура протокола FIPA-Contract-Net

Инициатор посылает запрос предложений, используя сообщение типа **cfp** (call for proposals). В этом сообщении указывается действие, которое требуется выполнить, и, если необходимо, условия его выполнения.

Респонденты могут ответить сообщением типа **propose** («Предложение»), содержащим предварительные условия (например, цену или время выполнения). Если респондент не может корректно интерпретировать полученный запрос, он посылает сообщение **not-understood** («Не понял»). В случае отклонения предложения респондент посылает сообщение **refuse** («Отказ»).

Получив от респондентов сообщения с предложениями, инициатор оценивает их, выбирает лучшее и отправляет соответствующему респонденту сообщение **accept-proposal** («Предложение принято»).

Респондент, получивший сообщение **accept-proposal** (предложение которого было принято), выполняет запрошенное действие и по его окончании отвечает сообщением **inform** с результатами действия или сообщением типа **failure** («Неудача»), если по какой-то причине не удалось выполнить запрошенное действие.

Инициатор может отменить взаимодействие по данному протоколу, послав сообщение **cancel** («Отмена»).

6.2. Порядок выполнения работы

1. Изучить тексты примеров, представленных в прил. 2.

2. Доработать код агента, созданного в Лабораторной работе № 4, в соответствии с вариантом задания (табл. 4.1).

6.3. Содержание отчета

1. Текст программы агента, выполняющего запросы. Пояснение работы этой программы.

2. Протокол консольных сообщений.

3. Выводы.

6.4. Вопросы для самоконтроля

1. Зачем нужны протоколы взаимодействия агентов?

2. Как создать и добавить агенту новое поведение на основе протокола?

3. Какие программные методы необходимо реализовать для агентов, выступающих в роли инициатора и респондента?

4. Какие протоколы стандартизированы FIPA? Кратко опишите последовательность действий агента (Инициатора и Респондента) для протокола.