POSC/EIA/57672/2004

Electronic Institutions providing Automatic Contracting for Virtual Organizations

RELATÓRIO FINAL DE BOLSA DE INVESTIGAÇÃO

(15/09/2005 - 15/04/2007)

Rui Jorge Canelhas Bastos Neves

15 de Abril de 2007

ÍNDICE

1. INTRODUÇÃO 5
2. ARQUITECTURA 7
Tabela 1 – Descrição dos packages existentes no projecto 10
3. FUNCIONAMENTO 11
3.1. Iniciar a Instituição Electrónica 11
3.2. Enterprise Agent 13
3.2.1. Supply 14
3.2.2. Request Goods 16
3.2.3. Negotiations 17
3.2.4. Contracts 19
3.2.5. Enviar e receber mensagens 20
3.2.6. Ontologia 21
3.3. Negotiation Mediator 23
3.3.1. Negotiation Handler 25
3.4. Ontology Mapping Agent 28
3.5. Notary 29
4. IMPLEMENTAÇÃO 30
4.1. FERRAMENTAS UTILIZADAS 30
4.1.1. JADE 30
4.1.1.1. FIPA 30
4.1.1.2. ACL 31
4.2. Instituição Electrónica 31
4.3. Enterprise Agents 33
4.3.1. ExternalAgent 33
4.3.2. EnterpriseAgent 34
4.3.3. QEnterpriseAgent 35
4.3.4. AskOntoMapAgent 36
4.4. Negotiation Mediator 37
4.4.1. AgentifiedService 37
4.4.2. NegotiationMediator 37
4.5. Negotiation Handler 38
4.5.1. NegotiationHandler 38
4.5.2. QFNegotiationHandler 39
4.6. Ontology Mapping Agent 40
4.7. Notary 42
4.8. Protocolos de Comunicação 43
4.8.1. Enteprise Agent → Negotiation Mediator 43
4.8.2. Negotiation Mediator (→ Negotiation Handler) → Enterprise Agent(s) 44
4.8.3. Enterprise Agent → Ontology Mapping Agent (→ Negotiation Handler) 45
4.8.4. Negotiation Mediator → Notary → Enterprise Agent(s) 46
4.9. Classes e Objectos auxiliares 47
4.9.1. Proposal 47
4.9.2. ProposalSet 47
4.9.3. Good e Comp 47
5. CONCLUSÕES 49
6. REFERÊNCIAS 50

ANEXO I – DTD de configuração geral da plataforma**.....51**
ANEXO II – DTD de configuração da ontologia para componentes e atributos**.....52**
ANEXO III – DTD de configuração da ontologia para produtos **.....53**
ANEXO IV – DTD de configuração das preferências de um agente**.....54**

Índice de Figuras

Fig.1. – Serviços de uma Instituição Electrónica [1] 6
Fig.2 – Diagrama de Classes 8
Fig.3 – Electronic Institution GUI 12
Fig.4 – GUI pertencente a um Enterprise Agent 13
Fig.5 – Janela para configuração de preferências de atributos 14
Fig.6 – GUI para configuração de aquisição de produtos, num Enterprise Agent 16
Fig.7 – Separador com informação sobre as mensagens trocadas na negociação 18
Fig.8 – Janela para visualização de informação relativa a mensagens ACL 19
Fig.9 – Separador do GUI com os contratos referentes às negociações decorridas 20
Fig.10 – Zona comum do GUI de um Enterprise Agent 21
Fig.11 – GUI de um Negotiation Mediator 24
Fig.12 – GUI de um Negotiation Handler no decorrer de uma negociação 26
Fig.13 – Gráfico com os valores de "utility" na negociação do componente "frontseat" 27
Fig.14 – Gráficos com os valores propostos para cada atributo de um determinado componente ao longo da negociação 27
Fig.15 – Diagrama de comunicação entre um *Enterprise Agent* e um *Negotiation Mediator* 43
Fig.16 – Diagrama de comunicação entre um *Negotiation Mediator*, um *Negotiation Handler* por si criado e os vários *Enterprise Agent* envolvidos numa negociação 44
Fig.17 – Diagrama de comunicação entre um *Enterprise Agent*, um *Ontology Mapping Agent* e um *Negotiation Handler* 45
Fig.18 – Diagrama de comunicação entre um *Negotiation Mediator*, *Notary* e um ou vários *Enterprise Agents* envolvidos num contrato 46

1. INTRODUÇÃO

Interacções entre membros de uma sociedade são reguladas por instituições. Estas instituições definem regras, especificando o que é permitido ou proibido aos indivíduos e sob que condições. Uma Instituição Electrónica será então o equivalente electrónico de uma dessas instituições, impondo regras aos seus membros electrónicos (agentes). Em particular, a Instituição Electrónica irá regular a interacção entre membros envolvidos em transacções comerciais, providenciando um ambiente onde interacções reguladas entre agentes podem tomar lugar.

Este projecto consistiu no desenvolvimento de uma dessas Instituições Electrónicas [2], que fornecesse determinados serviços [4] através de um sistema de multi-agentes. Estes serviços podem ser adicionados à plataforma sem dificuldade, registando o agente que os fornece na Instituição Electrónica. Outros agentes externos podem então tornar-se membros da Instituição, registando-se, e fazer uso de qualquer dos serviços fornecidos, sob determinadas condições. Neste caso, podemos referir o serviço de negociação [6], sob o qual o trabalho efectuado incidiu em larga escala, além dos serviços de ontologia [3] ou de monitorização de contratos [5], também implementados no decorrer do projecto.

Pretendeu desenvolver-se então uma plataforma para facilitar/automatizar processos para a formação de organizações virtuais, que através de contratos formalizam acordos de colaboração entre várias empresas. Especificamente, uma plataforma em Java que permitisse que múltiplos agentes inteligentes negociassem produtos entre si. Após uma negociação com sucesso, estabelecer-se-ão contratos passíveis de serem devidamente monitorizados. Cada agente poderá ter a sua própria ontologia [7], havendo na Instituição Electrónica um serviço que procederá à tradução entre as várias existentes.

Este relatório pretende descrever o trabalho efectuado pelo bolseiro Rui Jorge Neves durante todo o projecto, além de especificar o modo de utilização da aplicação no seu estado actual. O programa de trabalhos teve os seguintes pontos:

1. Estudo das ferramentas a utilizar na implementação do projecto;
2. Integração no conceito de Instituição Electrónica e compreensão dos objectivos do projecto;

3. Implementação da arquitectura base da Instituição Electrónica com vista à integração de serviços;
4. Implementação computacional de linguagem para representação dos contratos;
5. Integração do serviço de negociação de contratos;
6. Especificação e implementação de serviços para a execução de contratos;
7. Exploração do funcionamento da Instituição Electrónica: serviços de reputação e emergência de normas;
8. Implementação de serviços de ontologia;
9. Avaliação da plataforma com casos de estudo;
10. Escrita de Relatório.

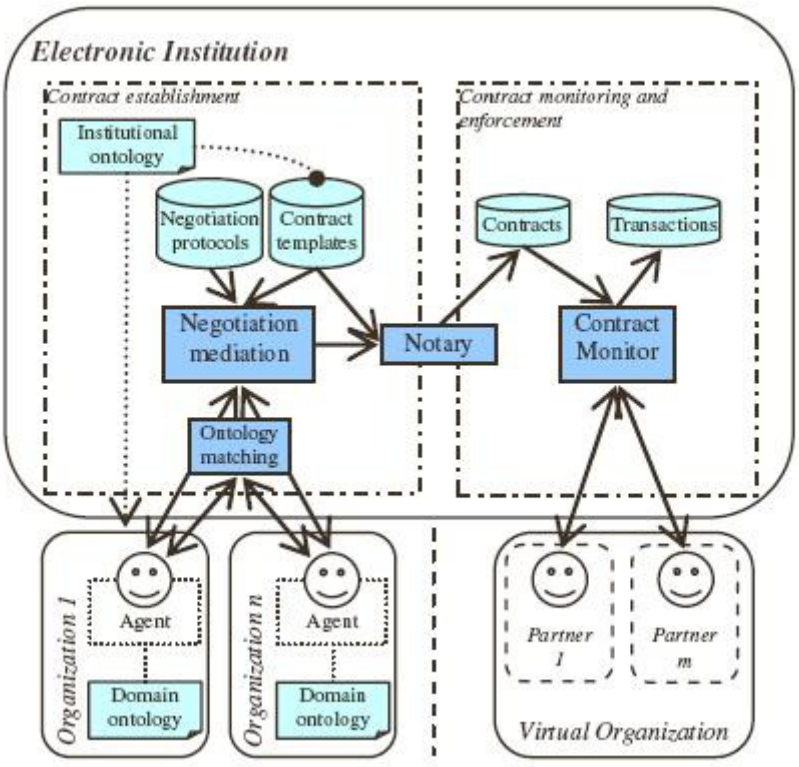

Fig.1. – Serviços de uma Instituição Electrónica [1]

Neste relatório iremos de seguida descrever a arquitectura da aplicação, após o que será descrito o funcionamento do sistema do ponto de vista do utilizador. Finalmente, será explicada a implementação de cada um dos agentes e serviços disponíveis.

2. ARQUITECTURA

Para melhor compreendermos a arquitectura da aplicação é preciso saber quais os objectivos do sistema a implementar. A Instituição Electrónica pretende facilitar/automatizar processos para a formação de organizações virtuais, que através de contratos formalizam acordos de colaboração entre várias empresas, representadas por agentes com diferentes estratégias e objectivos.

Existe um agente principal que tenta gerir toda a plataforma instituição electrónica, podendo múltiplos agentes registarem-se nela, e solicitarem outros agentes que proporcionam uma variedade de serviços distintos.

2.1 O funcionamento da Instituição Electrónica

A filosofia da aplicação consistirá em ter vários agentes representando empresas (e registados como tal) a fornecerem diversos componentes. Em determinada altura, um agente poderá proceder ao pedido de negociação para certo produto. Um produto será composto por vários componentes, cada um com determinados atributos (como preço, quantidade, etc). O pedido será então encaminhado para um mediador, que tratará de o reenviar para todos os agentes fornecedores (possivelmente incluindo o próprio agente que iniciou o pedido, pois poderá também operar como fornecedor). Os fornecedores, ao receberem os pedidos iniciais, verificam se fornecem o dito componente (ou mais que um) e, em caso positivo, formulam uma proposta inicial para o mediador analisar. No caso de não reconhecerem o componente pedido, podem solicitar o serviço de um agente de ontologia, caso exista, para tentarem uma tradução do componente pedido para algum existente na sua lista.

O mediador, após esperar algum tempo para receber as propostas iniciais dos fornecedores interessados, irá compará-las e analisar as preferências do agente consumidor, seleccionando a melhor e comentando os valores oferecidos em todas. Estes comentários serão enviados de volta aos respectivos agentes, para que possam reformular as suas propostas com base neles e na sua própria aprendizagem da situação. Este sistema de propostas e contra-propostas continua em acção até que alguma proposta atinja um valor considerado bom pelo mediador ou até que o número de rondas definido inicialmente para a negociação seja ultrapassado.

Caso o número de rondas seja ultrapassado e nenhuma proposta seja considerada satisfatória, a negociação termina nesse momento. No caso de existirem propostas

satisfatórias para todos os componentes requeridos, é formulado um contrato entre os agentes fornecedores e o agente consumidor. Nele estão especificados os termos da negociação e as condições que terão que ser cumpridas. Isto é conseguido recorrendo ao agente notário, que também pede à instituição electrónica que monitorize o dito contrato.

2.2 Os agentes e suas classes

Cada agente participante na instituição electrónica pode pertencer a diferentes classes, conforme a sua função. Vejamos agora o diagrama dessas classes, que mostra também as relações entre cada uma..

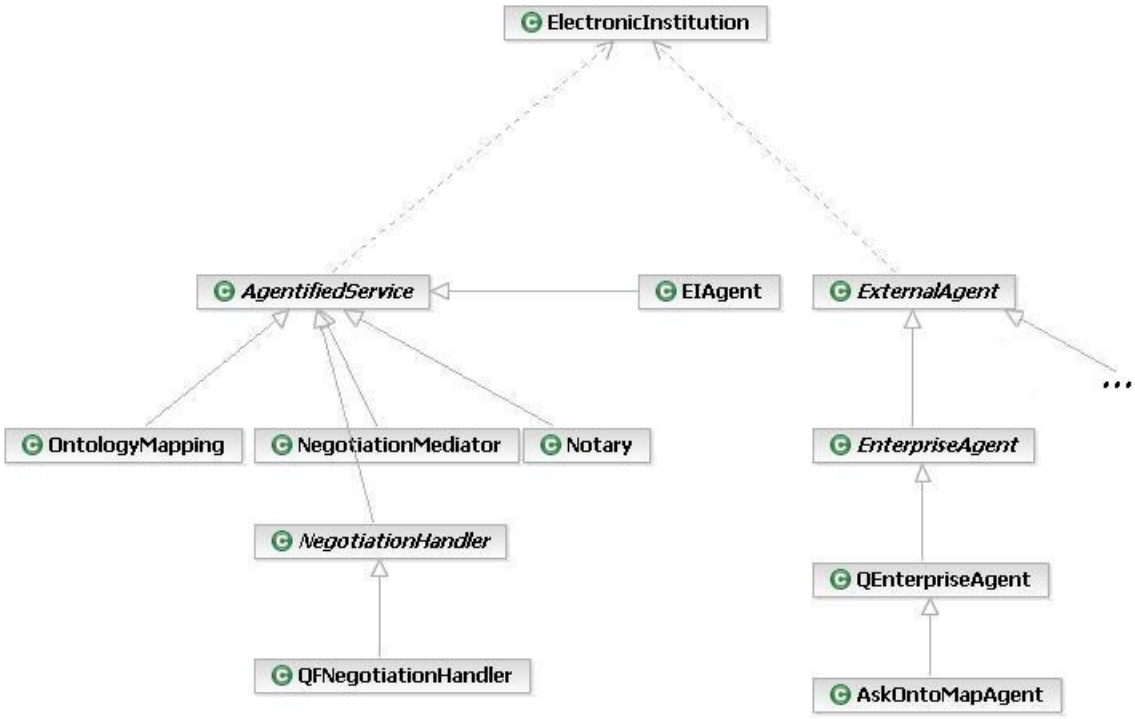

Fig.2 – Diagrama de Classes

Como se pode ver no diagrama de classes (ver Figura 2), existe uma classe principal `ElectronicInstitution` que permite lançar os vários tipos de agentes disponíveis. Os agentes podem ser dos seguintes tipos:

- `EIAgent`, para o agente principal, que monitoriza toda a plataforma;
- `OntologyMapping`, para agentes que forneçam serviços de ontologia;
- `NegotiationMediator`, para agentes que medeiem negociações entre agentes do tipo `EnterpriseAgent`;

- `Notary`, para agentes que forneçam serviços relacionados com contratos e notário.
- `EnterpriseAgent`, para agentes que forneçam e/ou pretendam adquirir produtos a outros;
- Poderão também existir outros agentes externos, com diferentes papéis, que não serão abrangidos neste relatório (ex: Bank, Messenger,...)

A classe `AgentifiedService` representa os serviços providenciados por diferentes agentes. É estendida por todos os agentes que providenciam serviços institucionais (`OntologyMapping`, `NegotiationMediator`, `Notary`, etc).

Ao mesmo nível temos a classe `ExternalAgent`, que é também estendida por agentes externos à instituição electrónica mas que interajam com ela (`EnterpriseAgent`, etc).

Abaixo destas encontram-se as classes com implementações específicas de agentes: `EnterpriseAgent`, `NegotiationMediator`, `Notary` e `OntologyMapping`. Cada uma delas representa um agente de determinado tipo, como explicado acima.

Entre estas encontra-se, como visto em cima, a classe `EnterpriseAgent`, que representa agentes empresa. Esta classe é abstracta, permitindo que algumas das suas características sejam re-implementadas sem dificuldade. Existem duas classes que aplicam esse princípio. A primeira é `QEnterpriseAgent`, que representa o protocolo de negociação e a estratégia de negociação (com aprendizagem) utilizadas pelo agente. Num nível ainda mais refinado temos a classe `AskOntoMapAgent`, que estende `QEnterpriseAgent` para utilizar, quando necessário, os serviços de ontologia disponíveis, ou seja, sempre que receber um pedido de algum componente que não conheça.

Outro agente de extrema importância para o correcto funcionamento da aplicação é o representado pela classe `NegotiationHandler`. Estes agentes são criados pelo `NegotiationMediator` para lidar com novas negociações, por uma questão de escalabilidade e performance.

Existem também, para todos os agentes que possam necessitar de intervenção humana, classes que representam os respectivos interfaces gráficos.

As classes estão agrupadas em packages com a seguinte estrutura:

Package	*Descrição*
ei	classe principal `ElectronicInstitution` e algumas ferramentas e utilitários
ei.agent	classes genéricas relacionadas com os agentes
ei.agent.enterpriseagent	mais específica, para tudo o que estiver relacionado com os agentes do tipo `EnterpriseAgent`
ei.agent.enterpriseagent.qnegotiation	para as classes relacionadas com o algoritmo de negociação com aprendizagem
ei.agent.gui	classes genéricas de interface gráfico
ei.agent.role	classes para os agentes que interpretam determinados "*roles*"
ei.contract	classes que tratam da parte da aplicação relacionada com contratos
ei.service	classes genéricas relacionadas com serviços disponibilizados pela aplicação
ei.service.negotiation	classes relacionadas com a negociação propriamente dita, do lado do mediador
ei.service.negotiation.qnegotiation	protocolo de negociação e *feedback* a propostas
ei.service.ontology	classes relacionadas com serviços de ontologia
ei.service.notary	agente que providencia serviços de notário e respectivas subclasses.

Tabela 1 – Descrição dos packages existentes no projecto

3. FUNCIONAMENTO

Iremos agora ver o funcionamento da aplicação do ponto de vista do utilizador. Este funcionamento será explicado isolando os diferentes agentes da Instituição Electrónica, para uma mais fácil compreensão da sua utilização.

3.1. Iniciar a Instituição Electrónica

Antes de iniciar a aplicação, será necessário definir um ficheiro de configuração no formato xml, que virá com as instruções sobre que agentes e serviços lançar. Este ficheiro indicará os agentes a lançar, especificando a sua classe e possíveis argumentos como se pode ver no seguinte exemplo (ver DTD no anexo I):

```
<institutional-agents>
        <agent name="ei-agent">
                <class>ei.agent.EIAgent</class>
                <argument>service=electronic-institution</argument>
        </agent>
        <agent name="negmed">
                <class>ei.service.negotiation.NegotiationMediator</class>

                <argument>handler_class=ei.service.negotiation.qnegotiation.QFNegotiationHandle
                </argument>
                <argument>service=IS-negotiation-mediator</argument>
                <argument>contract_xsd=D:\rneves\rational projs\EI Code\contract.xsd</argument>
        </agent>
        <agent name="starter">
                <class>ei.agent.enterpriseagent.AskOntoMapAgent</class>
                <argument>CA_ontology_file=ontology_ca.xml</argument>
                <argument>G_ontology_file=ontology_g.xml</argument>
                <argument>config_file_request=mk_agent.xml</argument>
                <argument>external_contract_viewer=C:\Program_Files\Mozilla
                Firefox\firefox.exe</argument>
        </agent>
</institutional-agents>
```

Após o preenchimento do ficheiro de configuração é possível lançar a aplicação, através da classe `ElectronicInstitution` levando o ficheiro de configuração

como argumento (`java ei.ElectronicInstitution eiconfig.xml`) que nos abrirá o seguinte ecrã:

Fig.3 – Electronic Institution GUI

Nesta janela (fig.3) existe uma caixa onde podemos escolher quais os agentes que providenciam os serviços seleccionados, que serão mostrados numa lista directamente em baixo. De forma semelhante podem ser vistos os agentes externos, escolhendo o tipo de papel ("*role*") que desempenham e visualizando em baixo a lista de todos os agentes desse tipo activos. É de realçar que os agentes activos não precisam de estar todos no mesmo computador, podendo estar distribuídos por diferentes máquinas.

No exemplo da figura 3 pode ver-se que existe um agente de serviços de ontologia chamado "osa" e em baixo três agentes com o papel de *enterprise-agent* activos, possivelmente a fornecer ou procurar componentes.

O GUI de qualquer um dos agentes activos será mostrado, se existir, ao fazer um duplo click com o rato no agente.

O botão "Shut Down" pode ser usado para encerrar toda a plataforma, tal como o nome indica.

Como foi visto, a Instituição Electrónica é constituída por dois grandes grupos de agentes: os que fornecem serviços e os que desempenham papéis.

No caso dos serviços serão apresentados três tipos diferentes de agentes:

- *negotiation-mediator*, para mediar negociações;
- *ontology-mapping*, para serviços de ontologia;
- *notary*, para serviços relacionados com contratos.

Quanto aos papéis existentes, neste relatório debruçar-nos-emos apenas nos *enterprise-agents*.

Vejamos então o funcionamento destes diversos agentes, tendo por base as classes da plataforma que implementam os serviços e papéis respectivos.

3.2. Enterprise Agent

Com a plataforma já em funcionamento, e após o lançamento de um *Enterprise Agent*, uma das tarefas primordiais será configurá-lo, o que é possível a partir do seu GUI. Este tipo de agentes tanto podem fornecer como procurar adquirir componentes ou produtos.

Começamos pelo caso de agentes fornecedores de componentes. Para configurar um agente para que forneça os componentes desejados é necessário abrir o GUI do agente, que terá o aspecto visto na figura 4.

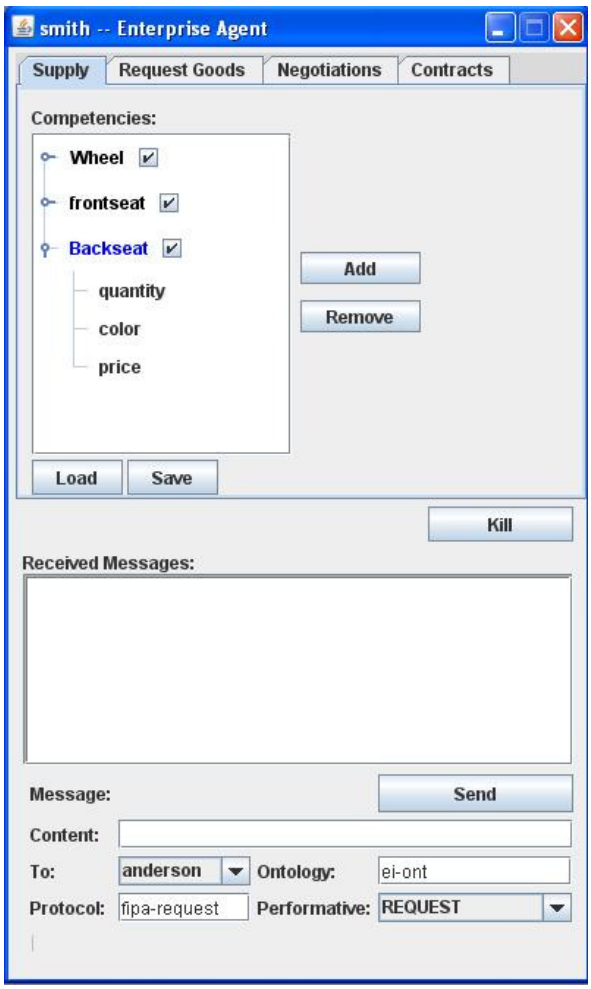

Fig.4 – GUI pertencente a um Enterprise Agent

Este interface gráfico, para além de uma área comum em baixo para recepção e envio "manual" de mensagens entre agentes, está dividido em quatro separadores:

- supply – para configurar o fornecimento de componentes;
- request goods – para iniciar o processo de composição de produtos;
- negotiations – com informação sobre as negociações levadas a cabo;
- contracts – com informação sobre os contratos derivados das negociações

Vejamos agora uma explicação mais detalhada sobre o funcionamento de cada um destes separadores.

3.2.1. Supply

No caso de desejarmos configurar o fornecimento de componentes teremos que adicioná-los à lista de competências. Para tal basta utilizar o botão "Add" e seleccionar o componente numa lista. Após adicionar o componente ele será mostrado na lista, com os respectivos atributos, como pode ser visto no exemplo da figura 4. É depois necessário configurar os valores dos atributos para que possam ser formuladas propostas. Para proceder a essa configuração basta fazer um duplo click com o rato em cima do atributo desejado, o que abrirá uma nova janela.

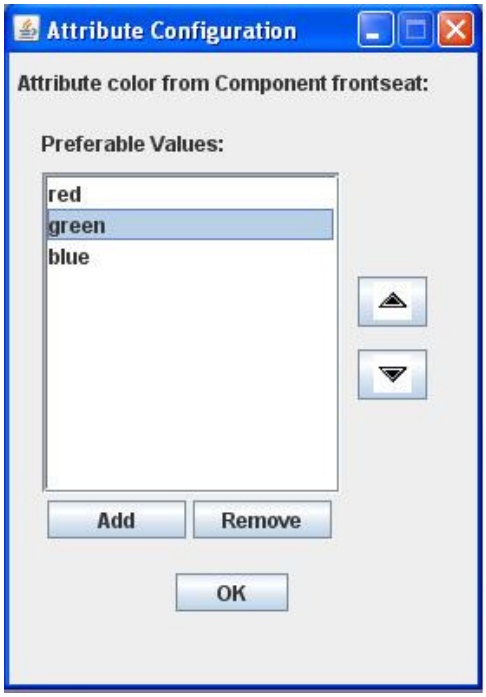

Fig.5 – Janela para configuração de preferências de atributos

Nesta janela (fig.4) poderá ser vista uma lista com os valores preferenciais para o atributo em causa. Existe também um botão "Add" e "Remove" que permite adicionar novos valores ou removê-los da lista. Após a inserção dos valores desejados é possível alterar a ordem de importância para cada um, recorrendo aos botões com as setas localizados à direita da lista. Isto assume particular importância no caso de valores discretos, em que apenas existe uma lista com a ordem de preferência do agente para cada um. Existe também a possibilidade de configurar intervalos de valores, dependendo do tipo do atributo. Nesse caso é suficiente inserir um único valor na lista, que representará o intervalo de valores preferencial para esse atributo, entre o valor mais desejado e o limite para o qual aceitará propostas (p.exemplo: 1000->800 para o caso dum atributo para o qual é desejado um valor alto, ou 1->100 no caso inverso). Após finalizarmos a inserção dos valores preferenciais para um determinado atributo e de os ordenarmos convenientemente bastará carregar em "ok" para retornar ao ecrã anterior.

Depois da configuração bem sucedida de todos os atributos de um componente, é então possível disponibilizá-lo a outros agentes. Para esse efeito, bastará marcar a caixa que se encontra em frente do nome do dito componente. A partir daí, o agente poderá começar a responder com propostas a pedidos de aquisição desse componente.

É também possível a qualquer momento, desde que não se encontre em negociação na altura, parar o fornecimento de determinado componente, desmarcando a caixa respectiva, quer para alterar a configuração dos atributos quer para proceder à sua remoção da lista. Para remover o componente é apenas necessário seleccioná-lo e pressionar o botão "Remove".

No caso de se utilizarem configurações semelhantes ou idênticas regularmente, é possível gravá-las num ficheiro xml, para posterior utilização. Para tal utiliza-se o botão "Save", escolhendo o nome e local para o ficheiro xml. Mais tarde poderemos então carregar em "Load" para utilizarmos uma configuração já definida. Pode ver-se o DTD utilizado neste tipo de ficheiros no anexo IV.

Este agente poderá ser lançado com vários argumentos (alguns obrigatórios) através do ficheiro de configuração inicial, explicado no início desta secção. Para além da classe do agente, é também obrigatório definir o ficheiro de ontologia para produtos e para componentes (explicado mais à frente). No caso de querermos utilizar uma configuração gravada previamente podemos definir o argumento opcional

config_file_supply e passar-lhe o nome do ficheiro onde a configuração está guardada. Nesse caso o agente é carregado já com a configuração desejada.

Existe também o argumento "external_contract_viewer" onde podemos definir o nome do programa a utilizar para visualizar os contratos negociados até ao momento.

Pode ver-se uma configuração de um agente deste tipo no exemplo mostrado a seguir:

```
<agent name="anderson">
        <class>ei.agent.enterpriseagent.AskOntoMapAgent</class>
        <argument>CA_ontology_file=ontology_ca2.xml</argument>
        <argument>G_ontology_file=ontology_g.xml</argument>
        <argument>config_file_supply=ent_agent2.xml</argument>
        <argument>external_contract_viewer=C:\Program_Files\Mozilla_Firefox\firefox.exe</argum
ent>
</agent>
```

3.2.2. Request Goods

Passemos agora ao caso da configuração de um Enterprise Agent que pretenda compor um produto (composto por componentes a obter de outros parceiros). Essa configuração é definida no segundo separador, que pode ser visto em baixo na figura 6.

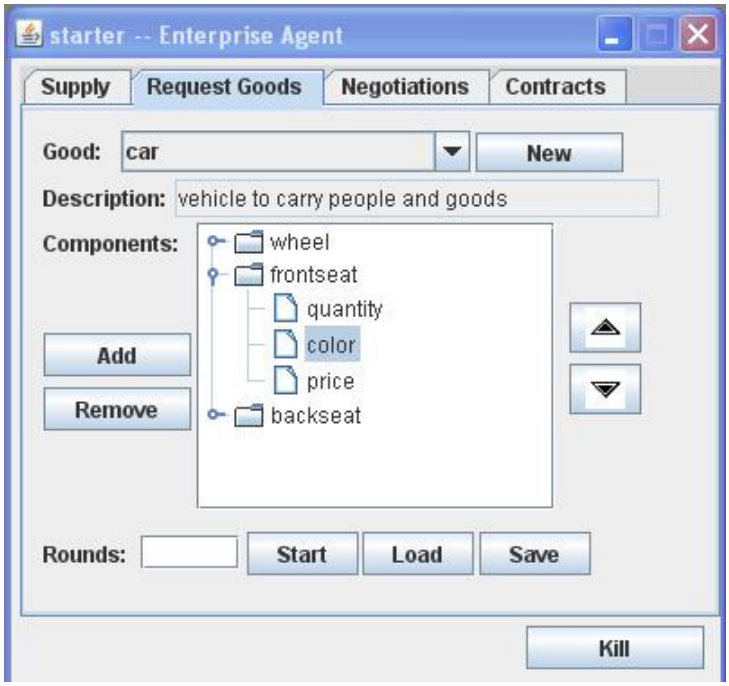

Fig.6 – GUI para configuração de aquisição de produtos, num Enterprise Agent

Neste caso, para a sua configuração, será primeiro necessário definir o nome do produto, ou escolher algum já existente na ontologia. Para isso poderemos escolher de uma lista no topo da janela o produto pretendido, ou criar um novo carregando em "New" e definindo o seu nome. Em baixo apresenta-se a descrição desse produto, e logo a seguir temos uma lista semelhante à do fornecimento, com os componentes que fazem parte do produto desejado. Caso seja um novo produto, proceder-se-á então à adição de novos componentes, de maneira idêntica à da etapa do fornecimento. Os botões "Add" e "Remove" permitem então adicionar ou remover componentes ao produto em questão. Quando todos os componentes estiverem na lista, podemos então configurar os atributos, também de maneira idêntica ao caso anterior. Existe apenas uma diferença nesta configuração, que passa pela importância dada a cada atributo. É possível ordenar os atributos pela ordem de importância que damos a cada um, recorrendo aos botões localizados à direita da lista. Isto poderá afectar a negociação, visto que dois agentes negociando o mesmo produto poderão dar importâncias diferentes aos mesmos atributos.

Quando todas as configurações estiverem concluídas podemos então, tal como no caso do fornecimento de componentes, gravá-las num ficheiro xml para posterior utilização, através dos botões "Save" e "Load". Ao lado destes, existe um outro botão, "Start", que permitirá que a busca e negociação pela formação do produto pretendido se inicie. Isto não poderá acontecer sem antes se definir o número de rondas máximo para essa negociação, para que não se dêem negociações infinitas no caso de as preferências de um agente estarem demasiado afastadas do pretendido pelo outro.

Também é possível aqui, definir um argumento no ficheiro de configuração da plataforma, para lançar o agente já com uma configuração pré-definida. Para esse efeito, utilizar-se-á o argumento "config_file_request" no ficheiro de configuração.

3.2.3. Negotiations

Após todas as configurações estarem concluídas e a negociação iniciada, os agentes começarão a trocar mensagens entre eles até atingirem um consenso ou o número de rondas máximo ser ultrapassado, o que nos leva ao terceiro separador do GUI.

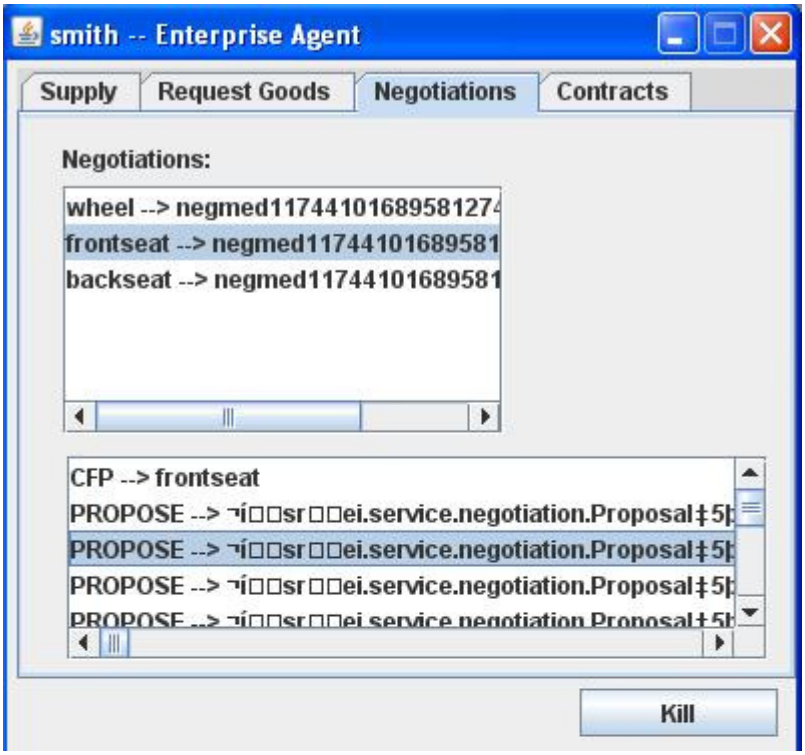

Fig.7 – Separador com informação sobre as mensagens trocadas na negociação

Neste local do GUI (fig.7), poderemos ver informação sobre as mensagens trocadas no decorrer das negociações efectuadas (ou ainda em curso).

Num primeiro quadro poderemos visualizar quais os componentes negociados e qual o agente que está a mediar (ou mediou) essa negociação. Seleccionando um componente aparecerão, no quadro de baixo (fig.7), todas as mensagens referentes a essa negociação. Estas são distinguidas pelo tipo (performative) e conteúdo e têm, normalmente, a sequência normal duma negociação, desde o pedido inicial (CFP), passando pelas várias propostas e contra-propostas, até à mensagem final de aceitação (ou recusa). Para estudar alguma mensagem em maior detalhe bastará um duplo click na mensagem em questão, o que trará para o ecrã uma nova janela com o seguinte aspecto:

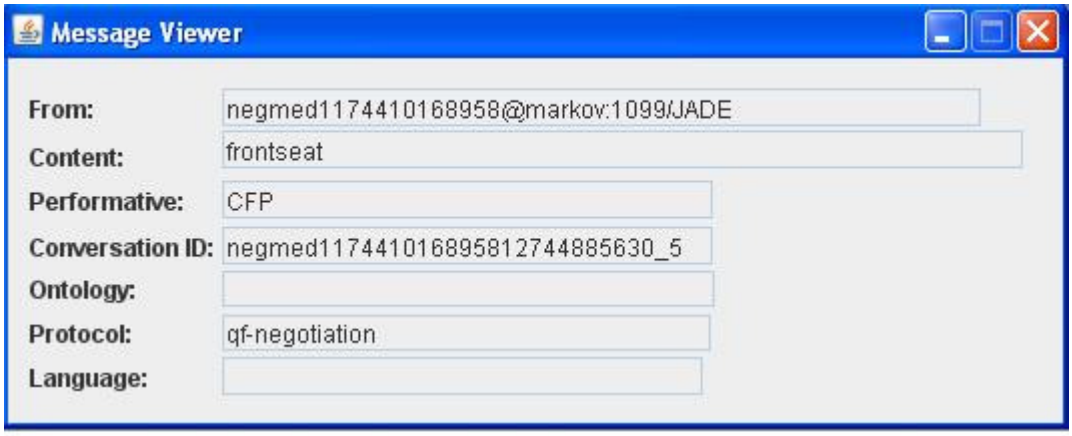

Fig.8 – Janela para visualização de informação relativa a mensagens ACL

Nesta janela poderemos visualizar informação sobre os vários campos das mensagens trocadas. Esses campos serão:

- o remetente, ou seja, de quem provem a mensagem em causa;
- o conteúdo da mensagem;
- o tipo de mensagem (performative);
- o ID de conversação;
- a ontologia, caso esteja definida;
- o protocolo utilizado;
- a linguagem.

3.2.4. Contracts

Depois de uma negociação bem sucedida é formulado um contrato entre os agentes envolvidos, a ser monitorizado pela Instituição Electrónica. Esses contratos poderão ser vistos no último separador deste GUI, mostrado na figura 9.

Chegamos então ao último separador deste GUI, que retrata a informação relativa aos contratos decorrentes da negociação entre os diversos agentes.

Neste local, de forma semelhante ao separador das negociações, existem dois quadros de informação. Em cima aparece uma listagem de todos os contratos envolvendo o agente em questão. Seleccionando um deles, aparecerão no quadro em baixo, todas as mensagens trocadas relativas a esse contrato. Seleccionando a mensagem pretendida, é possível com um duplo click no botão do rato aceder a

informação sobre ela, através da janela mostrada na figura 8, tal como no caso do separador “Negotiations”.

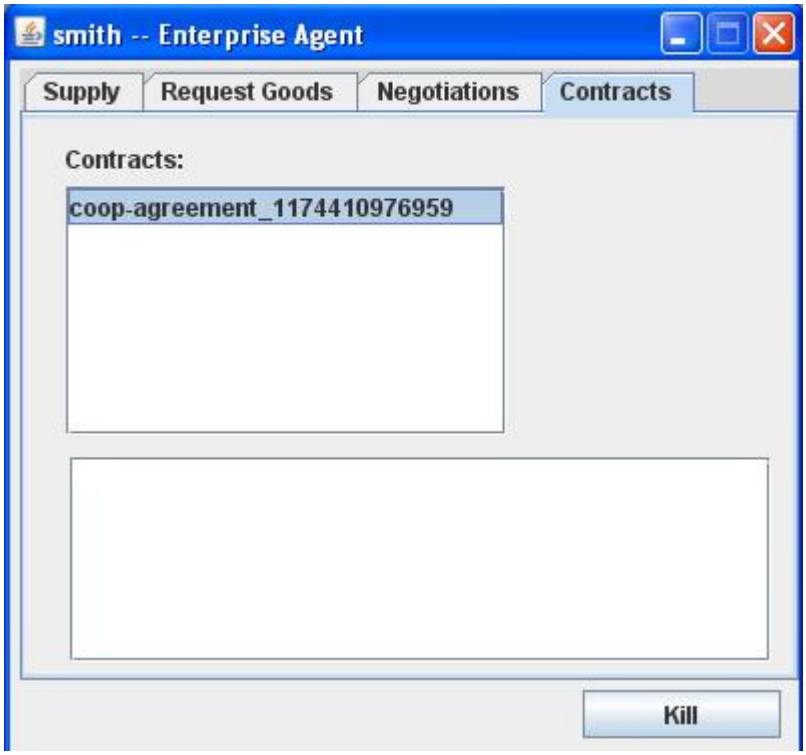

Fig.9 – Separador do GUI com os contratos referentes às negociações decorridas

Caso se pretenda aceder ao ficheiro xml que representa o contrato, é possível fazer um duplo click no dito contrato, através da lista, sendo o mesmo visualizado pelo programa dado como argumento no ficheiro de configuração da plataforma, caso exista.

3.2.5. Enviar e receber mensagens

Para que a explicação sobre o funcionamento do GUI de um Enterprise Agent esteja completa, é preciso fazer referência à zona comum a todos os separadores, mostrada na figura 10.

Neste local do GUI, é possível receber e enviar manualmente mensagens para qualquer agente registado na plataforma. Primeiro temos um quadro em que serão mostradas todas as mensagens recebidas que não se enquadrem nos outros quadros de negociações ou de contratos. Em baixo podemos criar uma mensagem para posterior

envio para outro agente, definindo o conteúdo, a ontologia e protocolo utilizados e a performativa. Além disto, teremos também que escolher de uma lista de agentes presentes na Instituição Electrónica, para qual deles será a mensagem enviada. Depois bastará carregar em "Send" e a mensagem seguirá para o destino.

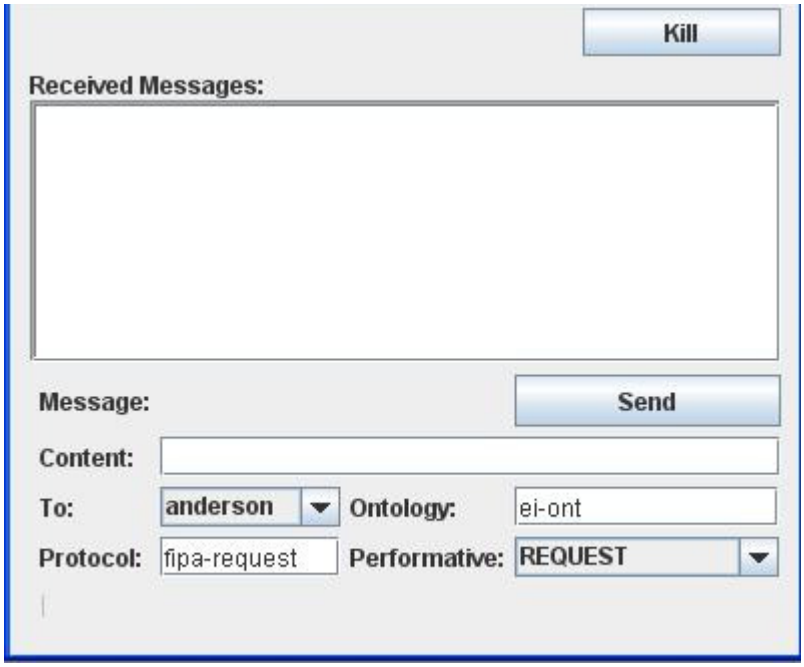

Fig.10 – Zona comum do GUI de um Enterprise Agent

Para além de todas estas configurações existe um outro botão, "Kill", que como o nome indica, encerrará o agente em questão. Isto é necessário visto que o simples fecho da janela apenas esconde o GUI, não afectando o agente.

3.2.6. Ontologia

Uma ontologia é um modelo de dados que representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre estes.

Todos os Enterprise Agents possuem a sua própria ontologia, representada por um ficheiro xml. Estes ficheiros possuem informação sobre todos os componentes (e produtos) que um determinado agente pode fornecer.

Tal como visto acima, estes ficheiros de ontologia têm que ser carregados com o agente, para que tal informação esteja disponível. Isto faz-se adicionando os argumentos "CA_ontology_file" e "G_ontology_file" ao ficheiro de configuração, especificando aí quais os ficheiros utilizados para cada uma das ontologias.

Embora ambos os ficheiros tenham uma estrutura semelhante, o primeiro lida com os componentes existentes e seus atributos, enquanto que o segundo trata apenas dos componentes que compõem os produtos compostos já definidos. O DTD para um ficheiro de ontologia relativo a componentes e atributos encontra-se no anexo II.

Este DTD especifica que os ficheiros deste tipo irão ter o esquema visto no exemplo a seguir:

```
<ontology-ca>
        <component id="wheel">
                <description>wheel</description>
                <attribute id="quantity">
                        <type id="integer"></type>
                        <domain value="1-->10000"></domain>
                </attribute>
                <attribute id="diameter">
                        <type id="real"></type>
                        <domain value="1-->10000"></domain>
                </attribute>
                <attribute id="price">
                        <type id="real"></type>
                        <domain value="1-->10000"></domain>
                </attribute>
        </component>
        <component id="front_seat">
                <description>driver and accompanying seats</description>
                <attribute id="quantity">
                        <type id="integer"></type>
                        <domain value="1-->10000"></domain>
                </attribute>
                <attribute id="color">
                        <type id="string"></type>
                        <domain value="red"></domain>
                        <domain value="green"></domain>
                        <domain value="blue"></domain>
                </attribute>
                <attribute id="price">
                        <type id="real"></type>
                        <domain value="1-->10000"></domain>
                </attribute>
        <ontology-ca>
</component>
```

O outro ficheiro possível refere-se a produtos e os componentes que os compõem. Apenas é utilizado para a criação do produto por negociação; o fornecimento de componentes não implica qualquer agrupamento. O DTD pode ser visto no anexo III. O que corresponde a um ficheiro com a estrutura do exemplo seguinte:

```
<ontology-g>
        <good id="car">
                <description>vehicle to carry people and goods</description>
                <component-name id="wheel"></component-name>
                <component-name id="front_seat"></component-name>
                <component-name id="back_seat"></component-name>
        </good>
        <good id="coat">
                <description>sleeved outer garment</description>
                <component-name id="tissue"></component-name>
                <component-name id="button"></component-name>
                <component-name id="sewing machine"></component-name>
        </good>
</ontology-g>
```

Estes ficheiros serão então carregados pelo agente, para que tenham acesso a todas as informações sobre os produtos e componentes sobre os quais poderão efectuar negociações.

3.3. Negotiation Mediator

Para que todos os *Enterprise Agents* existentes possam efectuar negociações bem sucedidas é necessário que exista um outro tipo de agente na Instituição Electrónica, para as mediar. A esse agente dá-se o nome de *Negotiation Mediator* e o seu objectivo é encontrar parceiros para negócios através de pedidos vindos de outros agentes.

O funcionamento de um *Negotiation Mediator* pode ser explicado de forma simples. Ao ser iniciado ficará a aguardar pedidos de agentes para a formação de empresas virtuais com vista à criação de produtos, constituídos por diferentes componentes. Assim que recebe um pedido, cria um agente da classe *Negotiation Handler* (ou sua subclasse), agente este que irá realmente mediar as negociações

necessárias, e fica a aguardar por novos pedidos. Deste modo a aplicação nunca entupirá ao receber pedidos simultâneos.

Para configurar um agente *Negotiation Mediator* para que seja lançado ao iniciar a Instituição Electrónica, é necessário preencher o ficheiro de configuração, como se pode ver no exemplo seguinte:

```
<agent name="negmed">
        <class>ei.service.negotiation.NegotiationMediator</class>
        <argument>handler_class=ei.service.negotiation.qnegotiation
                                .QFNegotiationHandler</argument>
        <argument>service=IS-negotiation-mediator</argument>
        <argument>contract_xsd=D:\rneves\rational_projs\EI_Code
                                \contract.xsd</argument>
</agent>
```

Aqui se define o seu nome, classe e tipo, além da classe dos *Negotiation Handlers* criados por si e do ficheiro xsd a utilizar para os contratos.

É possível definir qual a classe dos *Negotiation Handlers* pois em cada uma se implementam determinadas estratégias para lidar com as propostas recebidas. No caso deste projecto, é retornado um feedback qualitativo mas poderão existir outras implementações com resultados distintos bastando para isso aplicar outra classe de *Handlers*.

Quanto ao ficheiro xsd, servirá como especificação para os diferentes tipos de contrato que virão a existir, mas não foi alvo do trabalho aqui relatado.

Apesar de muito simples, o mediador também possui um GUI, como se pode ver na seguinte figura:

Fig.11 – GUI de um Negotiation Mediator

Neste GUI podem ver-se os *Negotiation Handlers* activos no momento e, com um duplo click, aceder aos seus próprios GUIs.

Como já foi dito, assim que recebe um pedido de algum produto, o *Negotiation Mediator* cria um novo agente do tipo *Negotiation Handler* e reencaminha-lhe esse pedido. A partir desse momento, quem lidará com todo o processo de negociação será esse novo agente, como explicado a seguir.

3.3.1. Negotiation Handler

O *Negotiation Handler*, ao receber um pedido para um qualquer produto, enviará mensagens para todos os *Enterprise Agents* existentes, procurando quais os que fornecem os componentes constituintes desse produto. Esses agentes, ao receberem as ditas mensagens, formularão uma nova proposta para cada componente em questão, que será de seguida enviada para o *Negotiation Handler*. Este, após recepção de todas as propostas (ou recusas, no caso dos agentes que não fornecem nenhum componente necessário) de todos os agentes interessados, analisa-as e comenta-as, reenviando-as para os respectivos agentes. A partir daí, aguarda de novo as novas e previsivelmente alteradas propostas e reproduz o comportamento anterior. Isto continua até se esgotarem as rondas definidas pelo agente que iniciou a negociação, ou até o *Negotiation Handler* determinar que alguma das propostas apresenta valores suficientemente bons para uma aceitação imediata. No final, uma mensagem com os resultados da negociação é enviada para o *Negotiation Mediator* (e depois desse para o *Enterprise Agent* que iniciou a negociação), sendo que após ter cumprido a sua função o agente *Negotiation Handler* ir-se-á auto-eliminar.

Em qualquer altura da negociação explicada em cima será possível aceder ao GUI de um *Negotiation Handler* activo, que terá o aspecto visível na figura 12.

Fig.12 – GUI de um Negotiation Handler no decorrer de uma negociação

Este GUI consiste numa tabela com os vários componentes requisitados pelo *Enterprise Agent* iniciador da negociação. Nesta tabela podem ver-se as seguintes informações:

- o nome dos componentes em negociação;
- o número de agentes que se encontra no momento a negociar o seu fornecimento;
- o nome do agente que está a vencer a negociação (não significa que a proposta seja considerada aceitável, apenas que é melhor que a dos outros agentes envolvidos);
- o número de rondas total e a ronda actual para esse componente;
- a utilidade da melhor proposta actual, que é um valor calculado com base nos valores dos atributos oferecidos pelo agente fornecedor e nas preferências do agente iniciador.

Além desta informação, é possível aceder a gráficos com informação mais detalhada sobre cada componente em negociação, com um duplo click do rato em cima do componente pretendido. Isto irá abrir a seguinte janela:

Como se pode ver na figura 12, este gráfico mostra os valores de utilidade que as propostas de cada agente envolvido obtiveram, para cada ronda da negociação.

Caso se queira obter informação ainda mais específica sobre as propostas para um determinado componente, basta fazer um duplo click em cima do gráfico, o que mostrará informação mais detalhada sobre os valores propostos para cada atributo do componente em questão.

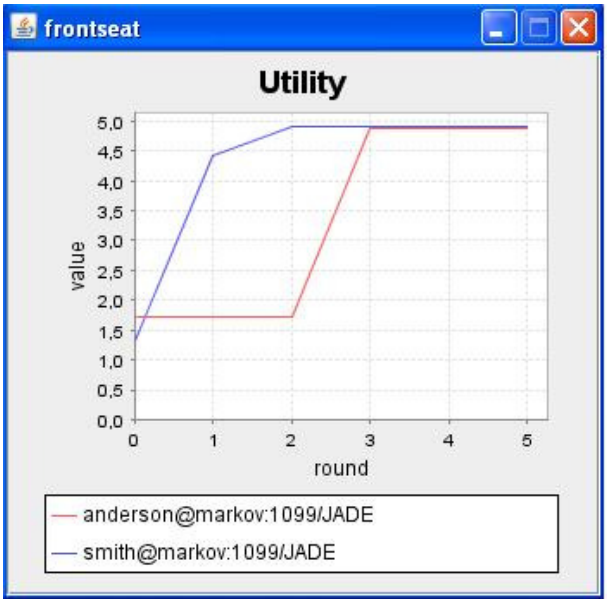

Fig.13 – Gráfico com os valores de "utility" na negociação do componente "frontseat"

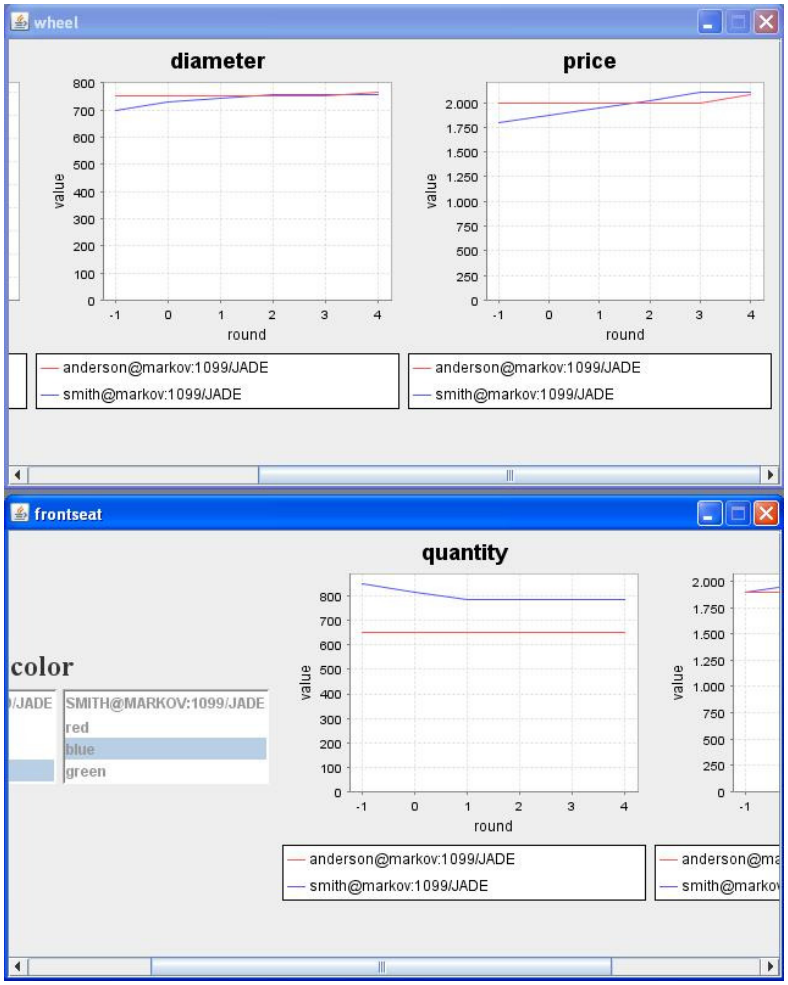

Fig.14 – Gráficos com os valores propostos para cada atributo de um determinado componente ao longo da negociação

Nestes gráficos mais detalhados, podem-se ver os valores propostos para cada atributo, ao longo das rondas, de cada agente envolvido na negociação do componente seleccionado.

Existe também o caso de os valores serem strings em vez de valores numéricos. Quando isso acontece, em vez da representação gráfica normal, será mostrado uma lista com os valores que cada agente pode oferecer, aparecendo seleccionada a opção corrente. Este caso pode ser parcialmente visto no exemplo de cima, na figura 14, para o atributo "color".

Para além dos *Enterprise Agents* e *Negotiation Mediator* e *Handler*, existem mais dois agentes, que participam respectivamente na pré e pós-negociação.

3.4. Ontology Mapping Agent

Um dos agentes que poderá estar activo na Instituição Electrónica é o *Ontology Mapping Agent*, que faculta serviços de ontologia a agentes que o requisitem.

O *Ontology Mapping Agent* funciona apenas no início de uma negociação, caso seja necessário. Quando um *Negotiation Handler* envia um pedido de determinado componente, e algum agente fornecedor verifica que não o tem disponível, pode pedir ao *Ontology Mapping Agent* que tente descobrir uma correspondência entre o componente pedido e algum que ele forneça. Isto poderá acontecer porque nem sempre a ontologia do agente que pede os componentes é igual à do agente que os fornece, podendo o mesmo componente existir em ambos com nomes diferentes.

A função do *Ontology Mapping Agent* é então, descobrir a correspondência entre componentes com nomes diferentes, através da sua comparação, e dos seus atributos.

Este agente não tem nenhum GUI, sendo o seu funcionamento completamente automatizado; no entanto, tal como os demais agentes vistos até ao momento, para que o seu lançamento na plataforma seja bem sucedido é necessário configurá-lo devidamente através do ficheiro para o efeito. Neste caso, a configuração de argumentos será semelhante ao exemplo seguinte:

```
<agent name="osa">
        <class>ei.service.ontology.OntologyMapping</class>
        <argument>service=IS-ontology-mapping</argument>
        <argument>host=127.0.0.1</argument>
```

```
        <argument>port=6180</argument>
        <argument>wordnet_file=WordNetSimilarity.dat</argument>
</agent>
```

Aqui se configurará o nome do agente, a sua classe e tipo de serviço, tal como nos restantes, mas também o IP e a porta do servidor *wordnet* utilizado. O *wordnet* é uma base de dados que contém relações semânticas e lexicais entre palavras, utilizada pelo *Ontology Mapping Agent*, ao procurar relações entre palavras. É também necessário especificar o nome do ficheiro onde estarão guardadas as pesquisas anteriores do *wordnet*, para facilitar futuras buscas.

3.5. Notary

O outro agente que falta referir e que, em oposição ao *Ontology Mapping Agent*, funciona na parte final da negociação, é o agente notário.

A sua função é, no caso de negociações que terminem com sucesso, validar e registar contratos e pedir à Instituição Electrónica que os monitorize.

Este agente funciona de maneira automática, sem qualquer GUI ou intervenção do utilizador, salvo a configuração dos argumentos, tal como nos outros, da qual temos um exemplo a seguir:

```
<agent name="notary">
        <class>ei.service.notary.Notary</class>
        <argument>service=IS-notary</argument>
</agent>
```

Nesta configuração apenas é necessário definir o nome do agente, a sua classe e o tipo de serviço que fornece, tal como no resto dos agentes vistos anteriormente.

4. IMPLEMENTAÇÃO

Na secção anterior foi explicado o funcionamento do ponto de vista do utilizador de cada um dos agentes envolvidos na Instituição Electrónica. Passemos agora a aprofundar o funcionamento de cada um deles, especificando as características e métodos utilizados para a sua implementação e os protocolos utilizados nas comunicações entre os diversos agentes.

4.1. FERRAMENTAS UTILIZADAS

Esta plataforma foi inteiramente implementada em Java, recorrendo ao software IBM Rational, baseado no Eclipse, utilizado para a programação e desenho UML de toda a aplicação. Foi utilizada a plataforma JADE para lidar com todo o sistema de multi-agentes.

4.1.1. JADE

O JADE [9] (**J**ava **A**gent **DE**velopment Framework) é uma plataforma de desenvolvimento de aplicações, para interoperabilidade entre sistemas multi-agentes, totalmente implementado em Java. Implementa um conjunto de serviços de sistema, os quais tanto facilitam como possibilitam a comunicação entre agentes, de acordo com as especificações da FIPA (**F**oundation for **I**ntelligent **P**hysical **A**gents): serviço de nomes e páginas amarelas, transporte de mensagens, serviços de codificação e descodificação de mensagens além de uma biblioteca de protocolos de interacção.

O JADE lida com todos os aspectos independentes das aplicações, tais como transporte, codificação e interpretação de mensagens e o ciclo de vida dos agentes.

4.1.1.1. FIPA

A FIPA é uma instituição que tem como missão desenvolver e estabelecer padrões para interacções entre agentes e sistemas baseados em múltiplos agentes. Produz várias especificações para esse efeito, das quais uma das mais importantes é o

ACL (Agent Communication Language), utilizada nesta aplicação para troca de mensagens.

Especifica também uma série de protocolos de comunicação, que permitem aos agentes fazer ou responder a determinados pedidos. Um desses protocolos é o FIPA-Request, no qual são baseados quase todos os protocolos utilizados nesta aplicação.

4.1.1.2. ACL

No JADE, toda esta troca de mensagens entre agentes é feita utilizando ACL (**A**gent **C**ommunication **L**anguage), que é uma linguagem que permite aos agentes que comuniquem entre si através de mensagens (communicative acts), de acordo com as especificações da FIPA [8]. Consiste num conjunto de tipos de mensagem e descrições dos efeitos da mensagem sobre os agentes que a enviam e a recebem. A sua estrutura é a seguinte:

```
(performative
        :sender   <valor>
        :receiver <valor>
        :content <valor>
        :language <valor>
        :ontology <valor>
        :conversation-id <valor>
        ...)
```

A performativa será o tipo de mensagem e pode ser desde um pedido de informação ou uma proposta até uma recusa ou aceitação, no caso de respostas. Depois poderemos especificar o receptor da mensagem, o conteúdo, linguagem, ontologia, etc, aumentando a informação enviada para que o receptor saiba sempre o que tem que fazer com a mensagem.

4.2. Instituição Electrónica

Para lançar a plataforma é preciso configurá-la através do recurso a um ficheiro xml, especificando quais os agentes a lançar. O DTD associado a esse ficheiro pode ser visto no anexo I.

Esse ficheiro especificará os agentes a lançar inicialmente na plataforma. Para cada agente, indica-se o nome, a classe que o implementa, e opcionalmente alguns argumentos específicos de cada agente. Tipicamente, um agente irá estender a classe `AgentifiedService` ou `ExternalAgent`, disponíveis na API do projecto.

A Instituição Electrónica limita-se a lançar os agentes, caberá a cada um registar-se no DF (*Directory Facilitator*) do JADE e outras tarefas possivelmente necessárias ao seu funcionamento. Os agentes que estendam a classe `AgentifiedService` serão automaticamente registados no DF, utilizando-se o argumento "service=...".

Os serviços que podem ser providenciados por agentes que participem na Instituição Electrónica estão definidos na classe `ElectronicInstitution`, sendo eles:

- *electronic-institution*, de uso exclusivo para o agente que encarna a própria Instituição Electrónica;
- *IS-ontology-mapping*, para serviços de ontologia;
- *IS-negotiation-mediator*, para mediadores de negociações;
- *IS-notary*, para agentes notário.

Existe também uma definição dos "roles" que podem ser tomados por outros agentes, ditos externos (e que tipicamente estenderão a classe *ExternalAgent*), que são os seguintes:

- *IR-bank*;
- *IR-delivery-tracker*;
- *IR-messenger*;
- *IR-enterprise-agent*.

Após o lançamento destes agentes, a plataforma mais não faz que os monitorizar, sendo o funcionamento de cada um independente do resto da Instituição Electrónica. Passemos então à implementação dos diversos agentes envolvidos.

4.3. Enterprise Agents

O funcionamento de um *Enterprise Agent* será dos mais complexos de todo o sistema. No entanto, de modo a simplificar todo o processo, foi separado em três classes diferentes, de modo a torná-lo o mais modular possível, para que a re-implementação de alguma das suas características principais se torne o mais simples possível.

Existem portanto três classes diferentes, cada uma a implementar os seguintes comportamentos:

- *Enterprise Agent*, que representa uma empresa fornecendo um conjunto de componentes, podendo participar em negociações como iniciador ou como respondente, e podendo estabelecer contratos;
- *QEnterpriseAgent,* que negoceia e analisa as propostas recebidas com base num protocolo de negociação específico (neste caso o QF-negotiation protocol), para além de também tratar de uma possível aprendizagem inteligente do agente;
- *AskOntoMapAgent,* que implementa um agente que utiliza, quando necessário, os serviços de ontologia disponíveis, podendo assim previsivelmente negociar alguns componentes que não possua na sua lista.

4.3.1. ExternalAgent

A própria classe *EnterpriseAgent* é uma extensão de *ExternalAgent*. Esta classe serve para construir agentes que interajam com o ambiente da Instituição Electrónica. Nela estão já incluídos uma série de métodos que permitem capturar os argumentos passados ao agente no seu lançamento, registar-se automaticamente no DF (ou vice-versa, aquando do seu término) ou mostrar o GUI, caso o agente em questão o possua.

4.3.2. EnterpriseAgent

Esta é uma classe abstracta, que representa agentes que tanto podem fornecer produtos como adquiri-los a outros do mesmo tipo, através de uma negociação. Pode ser estendida para lidar com protocolos de negociação específicos.

Ao iniciar, pode carregar um ficheiro de configuração de preferências e define qual o ficheiro de ontologia a utilizar.

Caso esteja apenas a fornecer componentes, o agente fica a aguardar até algum pedido chegar. Os pedidos virão em forma de uma mensagem CFP, da parte de algum mediador envolvido em negociação. Neste caso irá verificar se o componente pedido existe na sua lista e, em caso positivo, criará uma nova proposta inicial, que enviará ao mediador da negociação. O método para criar esta proposta inicial é abstracto, pois será implementado em `QEnterpriseAgent`, que é a classe que lida especificamente com a criação de propostas. Caso não possua o componente em questão, responderá ao CFP com um REFUSE, não participando mais nessa negociação.

Após o envio da proposta inicial, o agente continuará a aguardar, até receber uma nova mensagem do mediador, com o feedback à sua proposta. Esse feedback será então analisado e, com base nele, criada uma nova proposta, que será novamente enviada ao mediador. Este método de análise é também abstracto, e será implementado na classe `QEnterpriseAgent`, pelas razões já explicadas.

Este sistema continuará até o mediador decidir aceitar ou recusar definitivamente a proposta, fechando a negociação para o componente em causa. Isto pode acontecer de duas formas: recebendo uma mensagem ACCEPT_PROPOSAL, que significa que este agente foi o vencedor para a negociação de determinado componente, especificado no conteúdo da mensagem; ou com uma mensagem REJECT_PROPOSAL, que significa que a negociação terminou, e que este agente não a venceu, terminando por aqui o seu envolvimento nos assuntos relativos ao componente em questão.

Este agente também responde a pedidos vindos do agente de serviços de ontologia. Recebe um REQUEST para comparar os preços dos seus componentes com o do componente a ser verificado. No final, envia um vector com todos os componentes que possui com preços relativamente semelhantes ao recebido inicialmente, como conteúdo de uma mensagem INFORM. Isto permite ao serviço de ontologia verificar

quais os componentes que tenham possibilidades de serem os mesmos nos casos em que haja a necessidade de tradução entre agentes com ontologias distintas.

No caso do agente em causa pretender adquirir algum produto, inicia ele próprio uma negociação. Neste caso o agente procura um mediador disponível e envia-lhe uma mensagem (REQUEST) com o produto que pretende adquirir especificado no seu conteúdo. Depois ficará a aguardar uma resposta desse mediador, relativamente ao resultado da negociação. Caso receba um INFORM, será sinal que a negociação foi bem sucedida, e no conteúdo dessa mensagem virá um contrato com as informações sobre se comprometeu a fornecer o quê e em que condições. Pode também receber um FAILURE, que significa que um ou mais componentes não conseguiram ser negociados com sucesso, ou um REFUSE, caso o mediador em questão não esteja disponível para mediar a negociação pretendida.

4.3.3. QEnterpriseAgent

A classe `QEnterpriseAgent` implementa um *Enterprise Agent* de acordo com o *QF negotiation protocol*. Este tipo de agentes utilizará uma estratégia de negociação Q, com aprendizagem, ao negociar algum componente.

Existem dois métodos, abstractos na classe `EnterpriseAgent`, que são implementados aqui, permitindo definir toda a estratégia negocial do agente.

Primeiro temos o método que cria a proposta inicial face a um pedido de algum componente. Esta proposta inicial é feita atribuindo, para cada atributo do componente, o valor mais próximo da preferência máxima do agente, que não ultrapasse o valor mínimo das preferências do agente que iniciou a negociação. No caso de serem utilizados valores discretos, é utilizado o valor preferível para este agente, sem restrições.

O segundo e mais complexo método é o que trata de analisar o feedback enviado sobre as propostas deste agente. Este feedback não passa de comentários sobre os valores oferecidos na proposta anterior. Esta informação é utilizada para construir uma nova proposta, além de servir para actualizar os valores do algoritmo de aprendizagem.

Os comentários para cada valor podem ser: “mantain”, “sufficient”, “bad” ou “very bad”. No caso de “bad” ou “very bad”, esse valor terá provavelmente que ser alterado para haver hipóteses de que a proposta seja aceite. “Sufficient” será, tal como o

nome indica, suficiente para que a proposta seja aceite mas, caso os agentes em competição pelo negócio de fornecimento do componente em causa ofereçam valores mais elevados, poderá revelar-se insuficiente. Neste caso o agente terá que decidir se valerá a pena subir o valor, ou arriscar perder o negócio. Isto poderá ser feito com base no algoritmo de aprendizagem, que poderá ter informação sobre negócios semelhantes passados e respectivos resultados. O comentário "mantain" será apenas utilizado para casos de propostas que não necessitem de alterar o valor em questão, sendo portanto já de muito bom nível.

A partir destes comentários os agentes decidirão se é necessário alterar o valor de algum dos atributos e, em caso positivo, qual a dimensão da alteração. Após estas alterações, a nova proposta é devolvida para posterior envio para o mediador.

4.3.4. AskOntoMapAgent

Como é possível que os diversos *Enterprise Agents* sejam implementados pelas empresas que representam, pode dar-se o caso de utilizarem ontologias diversas uns dos outros. Pode então acontecer que o agente receba um pedido de algum componente que não forneça, ou julgue que não forneça, pois não o encontra na sua lista. Neste caso é necessário que o agente verifique se não possuirá mesmo esse componente, embora com outro nome.

Para esse efeito, o agente irá contactar o serviço de ontologia e perguntar-lhe se algum dos componentes que possui pode ser o correspondente do componente pedido.

Esta implementação, relacionada com a ontologia dos agentes, é efectuada na classe `AskOntoMapAgent`. Aqui, o agente irá requisitar o serviço de ontologia quando necessário, respondendo depois ao CFP recebido, de acordo com a resposta ao seu pedido.

Mais concretamente, envia um pedido (REQUEST) ao *Ontology Mapping Agent* presente (caso exista), com o nome do componente no conteúdo.

Fica depois a aguardar uma resposta, que poderá chegar na forma de um FAILURE, caso nenhum componente corresponda ao pedido no CFP original, ou de um INFORM, caso encontre uma correspondência. Nesse caso devolverá o nome do componente que corresponde ao pedido e o nível de confiança que tem nessa

correspondência. Para além disso, também indicará qual a correspondência entre os atributos, caso a consigo encontrar, pois se o componente tem outro nome será possível que os atributos também o tenham.

A partir deste momento, todas as propostas que cheguem relativas a este componente serão automaticamente traduzidas para a correspondência encontrada anteriormente.

4.4. Negotiation Mediator

4.4.1. AgentifiedService

Tal como para os agentes externos como *Enterprise Agents* e afins, que são estendidos da classe *ExternalAgent,* todos os agentes fornecedores de serviços disponibilizados pela plataforma podem ser estendidos da classe *AgentifiedService*, criada para o efeito. Esta classe regista automaticamente o agente no DF com o serviço que providencia, além de responder a mensagens vindas da Instituição Electrónica, como por exemplo, pedidos para a visualização do interface gráfico do agente respectivo.

Um dos principais agentes fornecedores de serviços que faz uso desta classe é o *Negotiation Mediator*, do qual falaremos em seguida.

4.4.2. NegotiationMediator

A função deste agente é mediar negociações, mas não o faz directamente. Inicialmente, aguarda por pedidos (REQUEST) de *Enterprise Agents* acerca de produtos que pretendam adquirir. Estes produtos são compostos por vários componentes e, cada um destes componentes, possui determinados atributos. Os pedidos possuem preferências (sob a forma de valores ou intervalos de valores) para cada atributo de cada componente. Ao chegar um destes pedidos, e caso o mediador aceite a negociação, cria um novo agente do tipo *Negotiation Handler* (que será explicado em detalhe a seguir) para tratar dessa negociação, e reenvia-lhe o pedido original. Desta forma nunca ficará sobrecarregado com pedidos, melhorando significativamente a sua fiabilidade.

Após a criação do *Handler*, ficará a aguardar pelos resultados da negociação, que lhe serão enviados antes da terminação desse *Handler*. Este resultado chega na forma do conjunto de propostas vencedoras da negociação, que será em seguida transformado num contrato, antes de ser reenviado para o agente que deu início à negociação. Caso a negociação não termine de forma bem sucedida, será enviada a mensagem correspondente (FAILURE) para esse mesmo agente.

4.5. Negotiation Handler

Como foi visto na secção anterior, um *Negotiation Mediator* cria um *Negotiation Handler* para lidar com as várias negociações associadas ao pedido de um produto.

Este agente foi, à semelhança do caso dos *Enterprise Agents*, separado em duas classes, para que a re-implementação do protocolo de negociação utilizado não implique re-implementar também o protocolo de comunicação:

- *NegotiationHandler*, é a classe que trata de toda a comunicação entre os diversos agentes e o Handler;
- *QFNegotiationHandler*, que representa um *Handler* que utiliza o protocolo de negociação QF (Qualitative Feedback).

4.5.1. NegotiationHandler

Esta é uma classe abstracta que representa um agente que lida com uma negociação. Este *Handler*, após ser criado pelo *Negotiation Mediator* assim que recebe um pedido, irá procurar todos os *Enterprise Agents* registados no DF e enviar uma mensagem (CFP) requisitando cada um dos componentes pedidos, para todos os agentes encontrados.

Depois, aguarda durante um determinado período de tempo por propostas (ou recusas em negociar) vindas desses agentes. No fim desse tempo (pré-definido), verifica se recebeu alguma proposta e, em caso negativo aborta a negociação, enviando um FAILURE de volta para o *Negotiation Mediator*. No caso de ter recebido pelo menos uma proposta para cada componente, analisa-as e compara-as, comentando os valores

oferecidos e assinalando a melhor. Este feedback sobre a proposta é depois reenviado para o agente correspondente, que altera (ou não) a sua proposta inicial e torna a enviá-la para o *Handler*.

Este sistema continua até terminarem as rondas definidas pelo agente que iniciou o pedido, ou até o *Negotiation Handler* receber alguma proposta com valores acima de um mínimo pré-definido. Quando tal acontece, envia uma mensagem a terminar a negociação (REJECT_PROPOSAL) a todos os agentes envolvidos, exceptuando o agente com a proposta vencedora, que receberá um ACCEPT_PROPOSAL.

Após o envio destas mensagens, enviará uma última mensagem com os resultados para o *Negotiation Mediator* que o criou, seja um INFORM, no caso da negociação ter sido bem sucedida, ou um FAILURE, no caso oposto. Assim que todas as mensagens forem enviadas, este Handler irá auto-destruir-se, pois já não terá mais nenhuma função.

Um *Negotiation Handler* também permite responder a pedidos do agente de serviços de ontologia, semelhante ao encontrado nos *Enterprise Agents*. Recebe pedidos (REQUEST) acerca dos detalhes dos componentes fornecidos.

Esta informação é depois enviada de volta para o agente que a pediu, para que possa ser feita uma possivelmente necessária correspondência entre componentes.

4.5.2. QFNegotiationHandler

Esta classe representa um *Negotiation Handler* que utiliza o protocolo de negociação QF. Isto consiste em avaliar as propostas, determinar a melhor e providenciar feedback a todos os agentes em participação.

Já foi explicado como funciona a negociação, pelo lado dum *Enterprise Agent*; vejamos agora como são então classificadas as propostas.

É recebido um conjunto de propostas e a vencedora é inicialmente considerada a primeira. Depois, cada proposta seguinte será comparada com a actual vencedora e, caso seja superior, passará ela mesma a ser a proposta vencedora. Esta comparação é feita com base nos valores oferecidos em cada atributo, relativamente aos valores tidos como preferenciais pelo agente que iniciou a negociação. Se o conjunto dos valores, de acordo com a importância dada a cada um, for superior à actual melhor proposta, esta passará a ser a proposta vencedora para a ronda em causa.

Depois de todas as propostas serem comparadas e escolhida a vencedora, será atribuído um comentário ao valor oferecido para cada atributo do componente em negociação. Este comentário poderá ser: "sufficient", "bad", "very bad" ou "mantain".

No caso de o valor oferecido se encontrar abaixo ou muito abaixo das expectativas do agente, o comentário será, respectivamente, "bad" ou "very bad", que indicam que uma proposta que ofereça valores semelhantes para esse atributo não terá grandes perspectivas de ser aceite, a não ser que os valores dos outros atributos sejam suficientemente bons para o compensar, ou caso a importância dada a este atributo não seja muito elevada.

O comentário "sufficient" será colocado em valores que se encontrem perto dos desejados pelo *Handler*, e indica que novas propostas poderão ser aceites sem alterar este valor. No entanto, isto dependerá de não existirem propostas com valores superiores vindas de outros agentes. O único caso em que um comentário de "sufficient" será propício a uma não alteração do valor correspondente será caso a proposta tenha sido considerada vencedora na ronda em questão.

O último comentário, "mantain", será apenas utilizado quando o valor oferecido for superior ao valor esperado pelo *Negotiation Handler*, indicando ao agente que fez a proposta que, caso pretenda melhorá-la, deverá focar-se nos valores dos restantes atributos que fazem parte do componente em causa.

Após esta avaliação, o conjunto de propostas será devolvido já com o feedback correspondente para que seja reenviado para os respectivos *Enterprise Agents*.

4.6. Ontology Mapping Agent

Este agente, caso exista na Instituição Electrónica, poderá ser determinante para o sucesso de uma negociação.

Entre os múltiplos Enterprise Agents que possam estar activos na plataforma, poderá dar-se o caso de vários deles terem ontologias distintas. Isto quererá dizer que ao tentarem negociar qualquer componente entre eles, poderão atribuir-lhe um nome diferente. Normalmente, o resultado desta situação seria: o agente A pede ao mediador que lhe arranje fornecedores para o componente X. O mediador pergunta aos Enterprise Agents existentes se algum fornece o componente X. O agente B até contém o

componente X na sua lista, mas na sua ontologia tem o nome Y. Por este motivo o agente B informa o mediador que não pode participar na negociação porque não fornece o componente X. E assim se perde uma possível negociação bem sucedida.

O Ontology Mapping Agent existe para evitar casos como o exemplo acima. E o seu funcionamento é simples, tal como explicado na secção 2.3: quando um agente recebe um pedido de fornecimento para um componente que não encontra na sua lista, esse agente pede ao Ontology Mapping Agent que verifique se o componente pretendido não estará mesmo na sua lista, sob outro nome, devido a serem utilizadas ontologias distintas.

Este agente lidará com os pedidos de Enterprise Agents. Estes pedidos chegarão sob a forma de uma mensagem ACL (REQUEST) contendo o nome do componente em questão.

Após a recepção do pedido, o agente tentará encontrar uma correspondência entre esse componente e algum componente que seja fornecido pelo agente que lhe dirigiu o pedido.

Para esse efeito irão ser pedidos os detalhes sobre o componente em causa ao Negotiation Handler. Além disso, haverá outro pedido ao Enterprise Agent acerca dos detalhes de todos os componentes que fornecidos cujos valores dos atributos preço estejam na vizinhança do valor de preço do componente pedido.

Assim que recebe a resposta, o agente comparará o componente pedido com todos os componentes descritos nela, tentando encontrar um que corresponda.

Caso o resultado da comparação seja superior a um valor mínimo, os componentes são considerados correspondentes, mediante um nível de confiança, baseado no valor atingido, e é enviada uma mensagem (INFORM) para o agente fornecedor indicando qual o componente que corresponde ao pedido, qual a correspondência entre atributos e qual o nível de confiança nessa correspondência. Este valor da confiança será também adicionado ao conteúdo da mensagem de resposta, para informação do Enterprise Agent, que depois decidirá se considera a informação válida com base nessa informação. Essa confiança poderá ser: fraca, moderada ou alta, sendo que para um nível de confiança alto não deverão existir dúvidas de que os componentes são os mesmos, enquanto de um nível de confiança fraco pode implicar que são parecidos mas não necessariamente iguais, ficando por isso ao critério do agente que requisitou a informação o que fazer em seguida.

No caso oposto, em que nenhum componente atinja o valor mínimo na comparação, a mensagem enviada (FAILURE) indicará isso mesmo, que nenhum dos componentes fornecidos aparenta ser igual ao componente pedido.

Como foi dito em cima, as comparações são feitas recorrendo ao algoritmo NGrams [3], e em seguida, ao *WordNet*. O *WordNet* é uma base de dados que contém relações semânticas e lexicais entre palavras. Nesta aplicação é utilizada recorrendo ao módulo de Perl, *WordNet::Similarity* que utiliza essa base de dados para calcular a similaridade semântica entre os nomes dos dois componentes. O script em Perl utilizado funciona como um servidor, que recebe um pedido de comparação do Ontology Mapping Agent e, após consultar a base de dados, retorna o resultado da comparação.

Os resultados obtidos pelo *WordNet* ficarão guardados num ficheiro para poupar tempo numa posterior utilização, visto que estas comparações poderão demorar largos minutos quando são efectuadas pela primeira vez.

4.7. Notary

O agente notary recebe pedidos de validação e registo de contratos, pedindo em seguida à Instituição Electrónica que os monitorize.

Este serviço tem três fases distintas:

Primeiro, fica a aguardar pedidos de registo de contratos (REQUESTs), informando depois o agente em questão se esse registo foi bem sucedido, através de um INFORM ou FAILURE.

De seguida, envia um pedido (REQUEST) a todos os agentes participantes numa determinada negociação com o contrato resultante, para que o assinem. Depois verificará se os contratos vêm todos devidamente assinados nas respostas (INFORMs).

E finalmente, envia o contrato assinado resultante das fases anteriores com um pedido (REQUEST) para que a Instituição Electrónica o monitorize. Depois fica a aguardar a resposta, que virá, como habitualmente, na forma de um INFORM ou FAILURE, consoante a disponibilidade da Instituição Electrónica.

4.8. Protocolos de Comunicação

Toda o sistema é implementado através da plataforma JADE, que funciona com base na troca de mensagens ACL entre os vários agentes existentes. Estas comunicações são baseadas em protocolos pré-definidos, sendo que cada agente implementa um *initiator* ou *responder role,* sendo que o primeiro permite iniciar conversações e esperar depois pelas respostas e o segundo aguarda por mensagens iniciadoras de conversações, respondendo-lhes em seguida. Isto é efectuado na maioria das vezes utilizando as classes do JADE *AchieveREInitiator* e *AchieveREResponder*, facilitando assim o seu desenvolvimento. Veremos agora quais os protocolos utilizados nas comunicações entre eles.

4.8.1. Enteprise Agent → Negotiation Mediator

Este procotolo é utilizado quando um *Enterprise Agent* pretende adquirir um produto, dirigindo esse pedido a um *Negotiation Mediator* disponível. A troca de mensagens ACL será então a que podemos ver em baixo, na figura 15.

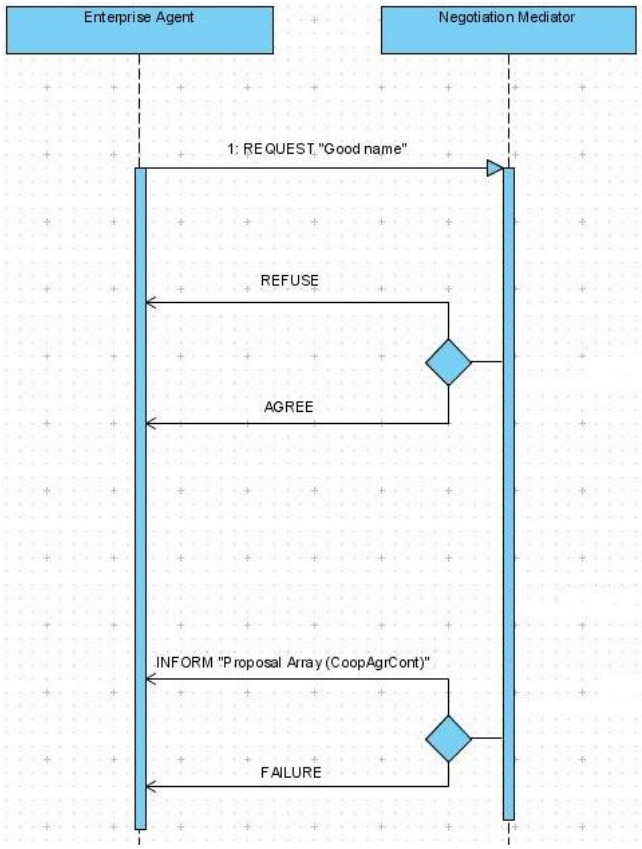

Fig.15 – Diagrama de comunicação entre um *Enterprise Agent* e um *Negotiation Mediator*

4.8.2. Negotiation Mediator (→ Negotiation Handler) → Enterprise Agent(s)

A comunicação da fase seguinte da negociação acontece depois do pedido ao *Negotiation Mediator*. Este cria então um *Negotiation Handler* para lidar com a negociação e reenvia-lhe o pedido. O *Negotiation Handler* envia então o CFP inicial aos *Enterprise Agents* existentes, aguarda por propostas, comenta-as e envia o feedback respectivo, durante um número pré-definido de rondas e, finalmente, aceita ou rejeita definitivamente cada uma delas. O diagrama com a troca de mensagens ACL entre agentes pode ser visto na figura 16.

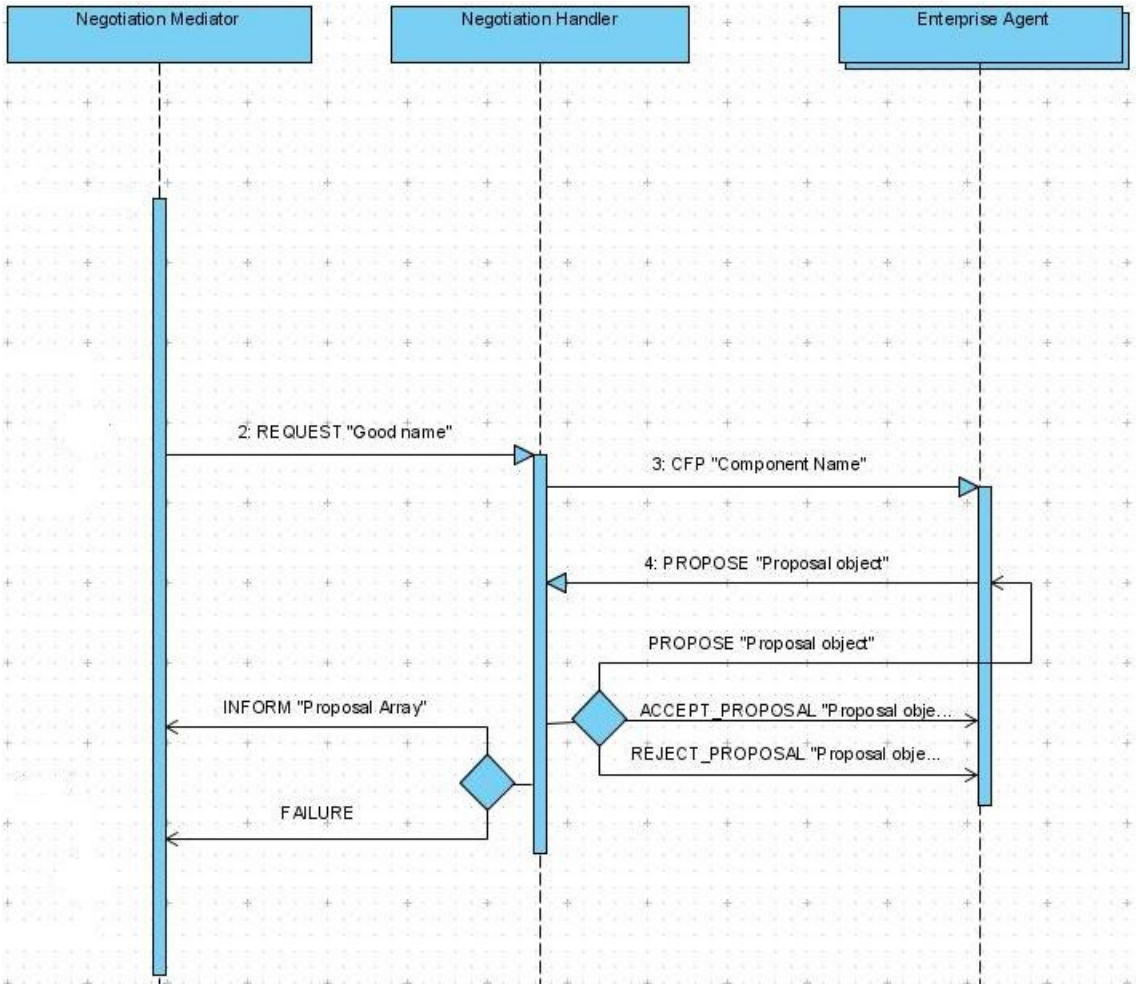

Fig.16 – Diagrama de comunicação entre um *Negotiation Mediator*, um *Negotiation Handler* por si criado e os vários *Enterprise Agent* envolvidos numa negociação

4.8.3. Enterprise Agent → Ontology Mapping Agent (→ Negotiation Handler)

Outro protocolo de comunicação existente é entre um *Enterprise Agent* e o *Ontology Mapping Agent*. O primeiro pede ao segundo que lhe encontre (ou não) alguma correspondência entre o nome de um componente pedido (por um outro agente) e a sua lista de componentes fornecidos. Esta troca de mensagens pode ser vista na figura 17.

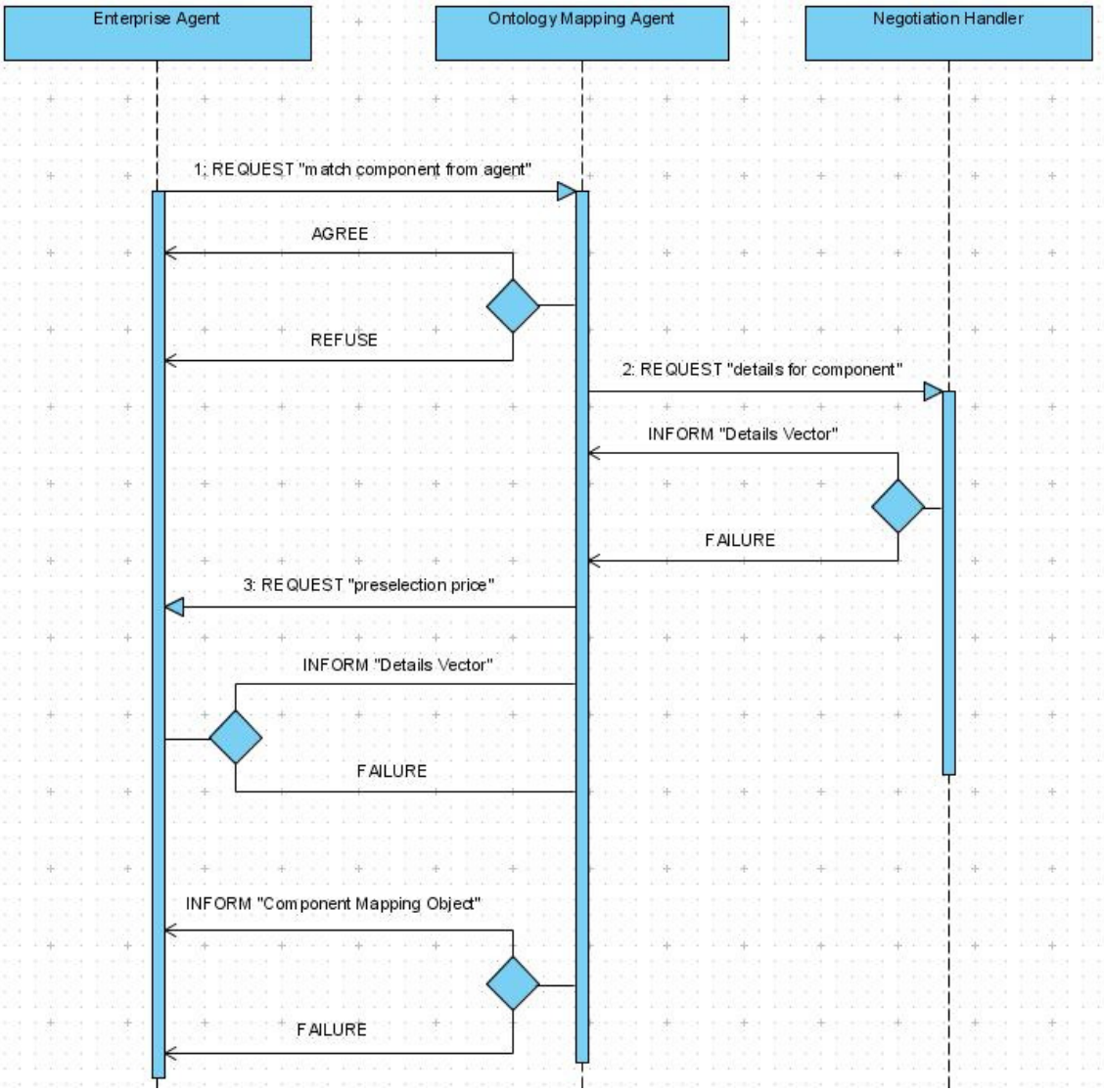

Fig.17 – Diagrama de comunicação entre um *Enterprise Agent*, um *Ontology Mapping Agent* e um *Negotiation Handler*

4.8.4. Negotiation Mediator → Notary → Enterprise Agent(s)

Resta agora descrever o protocolo de comunicação utilizado no registo e assinatura de contratos por parte do agente mediador, notário e fornecedores envolvidos. O *Negotiation Mediator* envia um pedido ao notário com o contrato resultante de uma determinada negociação, e este envia-o a cada um dos *Enterprise Agents* participantes para que o assinem. Caso todos os participantes o assinem devidamente, é enviada essa informação ao mediador, significando que o contrato está registado e pronto a ser monitorizado. Este sistema pode ser visto na figura 18.

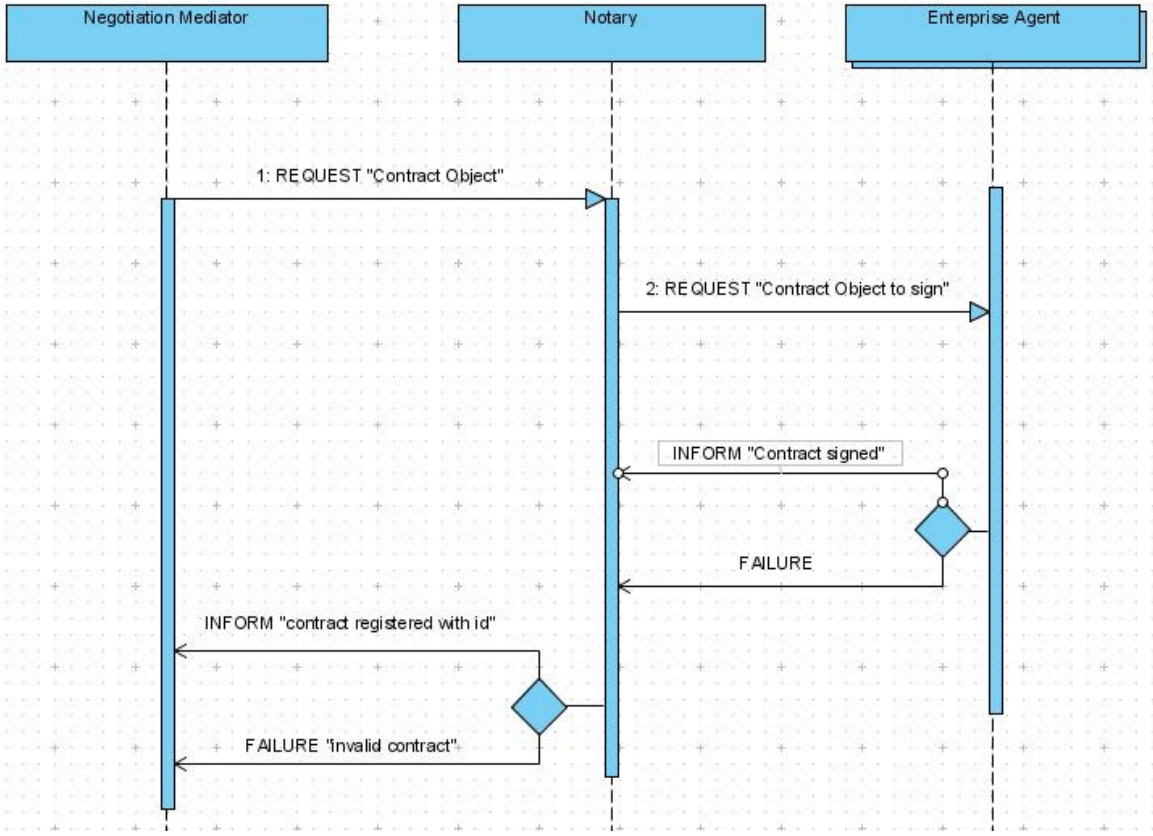

Fig.18 – Diagrama de comunicação entre um *Negotiation Mediator*, *Notary* e um ou vários *Enterprise Agents* envolvidos num contrato

4.9. Classes e Objectos auxiliares

4.9.1. Proposal

Toda esta troca de mensagens entre agentes não serviria de nada se no seu conteúdo não fossem os objectos de negociação, as propostas. Estas propostas são representadas por objectos Java e incluem o nome do componente em questão, o AID do agente que a enviou e uma subclasse que representa os atributos pertencentes a esse componente e os valores oferecidos para cada um deles. Nesta subclasse estão definidos os nomes dos atributos, o tipo de cada um e o valor oferecido, além de uma indicação de se o valor preferencial é acima ou abaixo do oferecido.

Após a análise de uma destas propostas por parte do mediador, mais alguns campos serão preenchidos, sendo a proposta enviada de volta para o agente. Os campos a serem preenchidos são o valor da avaliação dada à proposta e uma indicação de se a proposta é a vencedora para a ronda actual. Para além destes, é também preenchido um campo para cada atributo, comentando o valor oferecido ("mantain", "sufficient", "bad" ou "very bad", como foi visto mais em cima).

4.9.2. ProposalSet

Para além do objecto que representa uma proposta individual, existe outro que representa um grupo de propostas, feitas numa determinada ronda de negociação: o *ProposalSet*. Nesta classe existe uma hashtable que tem todas as propostas enviadas na ronda actual, organizadas de acordo com o agente que a enviou. Tem também a informação de qual é a proposta vencedora dessa ronda e qual o nome do componente que está a ser negociado.

4.9.3. Good e Comp

Estas classes representam, respectivamente, os produtos e componentes a negociar. No caso de *Good,* a única informação contida em si é, o nome do produto, uma descrição e o array de componentes que o compõem (da classe *Comp*, obviamente).

Quanto à classe *Comp,* contém o nome do componente, uma descrição e um array com os atributos pertencentes a cada um. Para além disso, contém também uma subclasse que representa os atributos. Nesta subclasse existe informação sobre o nome do atributo e o seu tipo, o domínio que os valores podem tomar e um vector com os valores preferenciais para esse atributo.

5. CONCLUSÕES

Ao terminar o projecto podemos dizer que quase todos os objectivos deste projecto foram concluídos no tempo previsto.

A aplicação existente está agora implementada de maneira bem sucedida utilizando a plataforma JADE. Para além disso, e com vista a melhorar a expansibilidade, o sistema apresenta-se agora numa forma sensivelmente mais modular, de modo a permitir que a integração de novos agentes, serviços e funcionalidades seja o mais simples possível.

Foram também feitas alterações para aumentar a robustez do sistema. Através de várias medidas implementadas diminuiu-se a probabilidade de falha da aplicação.

O principal objectivo, de implementação de um serviço de negociação entre agentes inteligentes, foi terminado com sucesso.

Para além disso, também os serviços de ontologia foram implementados como pretendido, sendo que os agentes já não abdicam de negociações por não conhecerem o nome de um determinado componente. Este serviço poderá ainda ser melhor aplicado no futuro, implementando as ontologias dos diversos agentes como objectos, em vez de ficheiros xml.

Foi também implementado com sucesso todo o sistema de negociação e monitorização de contratos.

6. REFERÊNCIAS

1. Henrique Lopes Cardoso (2006). “Electronic Institution: an E-contracting Platform for Virtual Organizations”, in /Actas da 1ª Conferência de Metodologias de Investigação Científica (CoMIC’06)/, Programa de Doutoramento em Engenharia Informática, pp. 33-42, FEUP, 9 de Janeiro de 2006.

2. Henrique Lopes Cardoso, Eugénio Oliveira (2005). “Virtual Enterprise Normative Framework within Electronic Institutions”, in M.-P. Gleizes, A. Omicini & F. Zambonelli (eds.), /Engineering Societies in the Agents World V/, LNAI 3451, Springer, ISBN 3-540-27330-1, pp. 14-32.

3. Andeia Malucelli (2006). "Ontology-based Services for Agents Interoperability/, Tese de Doutoramento, Faculdade de Engenharia da Universidade do Porto.

4. Andreia Malucelli, Henrique Lopes Cardoso, Eugénio Oliveira (2006). “Enriching a MAS Environment with Institutional Services”, in D. Weyns, H. Van Dyke Parunak & F. Michel (eds.), /Environments for Multi-Agent Systems II/, LNAI 3830, Springer, ISBN: 3-540-32614-6, pp. 105-120.

5. Daniel Moura (2004). /Creating and Monitoring Contracts in Virtual Enterprises/, Technical Report, Faculdade de Engenharia da Universidade do Porto.

6. Ana Paula Rocha (2001). /Metodologias de Negociação em Sistemas Multi-Agentes para Empresas Virtuais/, Tese de Doutoramento, Faculdade de Engenharia da Universidade do Porto.

7. Bernd Schneiders (2007). /Implementation of an ontology mapping algorithm/, Relatório de Estágio, Faculdade de Engenharia da Universidade do Porto.

8. FIPA – The Foundation for Intelligent Physical Agents, http://www.fipa.org

9. JADE – Java Agent Development Framework, http://jade.tilab.com

ANEXO I

DTD de configuração geral da plataforma

```
<!ELEMENT institutional-agents    (agent+)>
<!ELEMENT agent                   (class,argument*)>
<!ELEMENT class                   (#PCDATA)>
<!ELEMENT argument                (#PCDATA)>

<!ATTLIST agent name CDATA #REQUIRED>
```

ANEXO II

DTD de configuração da Ontologia para componentes e atributos

```
<!ELEMENT ontology-ca (component+)>

<!ELEMENT component (description, attribute+)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT attribute (type, domain+)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT domain (#PCDATA)>

<!ATTLIST component id CDATA #REQUIRED>
<!ATTLIST attribute id CDATA #REQUIRED>
<!ATTLIST type id CDATA #REQUIRED>
<!ATTLIST domain value CDATA #REQUIRED>
```

ANEXO III

DTD de configuração da Ontologia para produtos

<!ELEMENT ontology-g (good+)>

<!ELEMENT good (description, component-name+)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT component-name (#PCDATA)>

<!ATTLIST good id CDATA #REQUIRED>
<!ATTLIST component-name id CDATA #REQUIRED>

ANEXO IV

DTD de configuração das preferências de um agente

<!ELEMENT agent-config (name, good?, component+)>

<!ELEMENT name (#PCDATA)>
<!ELEMENT good (#PCDATA)>
<!ELEMENT component (description, attribute+)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT attribute (domainValue*, preferableValue*)>
<!ELEMENT domainValue (#PCDATA)>
<!ELEMENT preferableValue (#PCDATA)>

<!ATTLIST name id CDATA #REQUIRED>
<!ATTLIST good id CDATA #REQUIRED>
<!ATTLIST component id CDATA #REQUIRED>
<!ATTLIST attribute id CDATA #REQUIRED>
<!ATTLIST domainValue value CDATA #REQUIRED>
<!ATTLIST preferableValue value CDATA #REQUIRED>