Interacção e Contratação em Instituições Electrónicas

Universidade do Porto

Faculdade de Engenharia

FEUP

RELATÓRIO FINAL DE BOLSA DE INVESTIGAÇÃO

(01/06/2007 - 31/05/2008)

Rui Jorge Canelhas Bastos Neves

31 de Maio de 2008

ÍNDICE

Índice de Figuras

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 [2] irá regular a interacção entre membros envolvidos em transacções comerciais, providenciando um ambiente onde interacções reguladas entre agentes podem tomar lugar.

Este projecto pretendeu dar continuidade ao trabalho desenvolvido no âmbito do projecto FCT POSC/EIA/57672/2004 [6].

Assim, o trabalho desenvolvido centrou-se na continuação do desenvolvimento e implementação da plataforma de software que constitui a *Instituição Electrónica*. Especificamente, pretendeu envidar-se esforços no sentido de clarificar e documentar os protocolos de interacção entre os diversos intervenientes (agentes exteriores e institucionais), no sentido de tornar a plataforma concordante com as especificações da FIPA. A definição e utilização de ontologias apropriadas também foram uma preocupação.

Para além disso, a articulação dos serviços institucionais de contratação e o funcionamento de agentes com papéis institucionais foi também alvo de desenvolvimento.

Um tópico que ainda não estava explorado na plataforma e que também foi alvo de estudo e implementação diz respeito à segurança. Assim, foram estudados e implementados mecanismos de autenticação e autorização de agentes e de integridade e confidencialidade de mensagens (com base no *add-on* de segurança da plataforma JADE).

Pretendeu-se ainda especificar e testar cenários de aplicação da plataforma para formação e monitorização de contratos entre parceiros. Neste capítulo, foram ainda desenvolvidos mecanismos de simulação que permitissem testar convenientemente a plataforma.

O presente 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. Protocolos de interacção na Instituição Electrónica: levantamento do ponto da situação; clarificação de procedimentos;
2. Protocolos de interacção na Instituição Electrónica: adaptação para conformidade FIPA; utilização de ontologias JADE;
3. Serviços e papéis na Instituição Electrónica: clarificação da articulação entre os diversos intervenientes; resolução de lacunas;
4. Papéis na Instituição Electrónica: implementação de agentes tipo;
5. Segurança na Instituição Electrónica: estudo do *add-on* de segurança do JADE;
6. Cenários de aplicação da Instituição Electrónica: escolha e especificação de cenário(s);
7. Segurança na Instituição Electrónica: integração de mecanismos na plataforma;
8. Cenários de aplicação da Instituição Electrónica: desenvolvimento de mecanismos automáticos de simulação;
9. Cenários de aplicação da Instituição Electrónica: teste; avaliação de resultados
10. Escrita de Relatório.

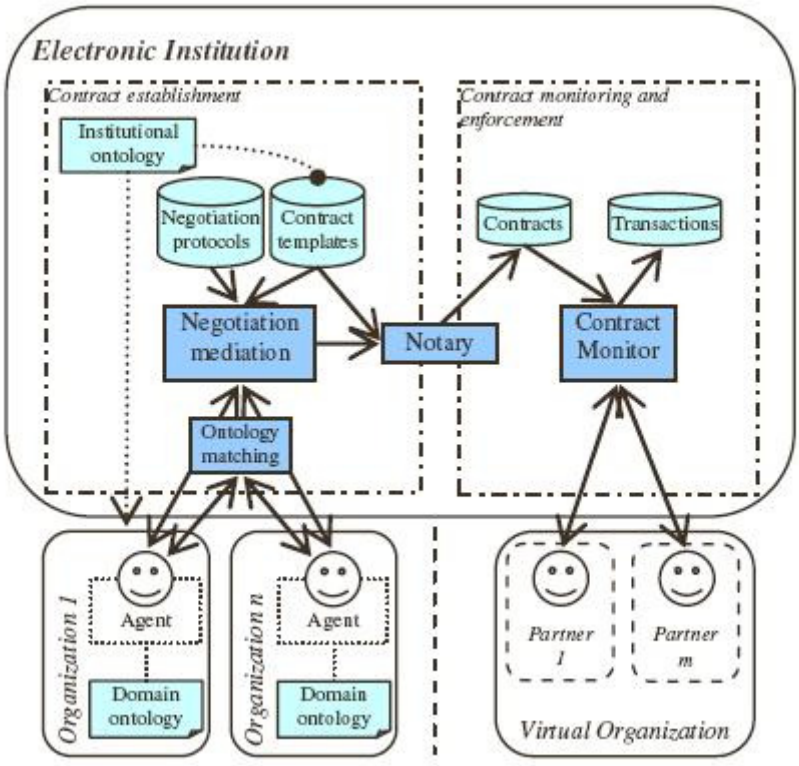

Figura 1 – Serviços de uma Instituição Electrónica [1]

Na figura 1 podemos ver um diagrama representativo de uma Instituição Electrónica e dos serviços que proporciona. Iremos de seguida descrever resumidamente a arquitectura da aplicação e as alterações significativas da versão anterior, após o que será descrito também o funcionamento do sistema do ponto de vista do utilizador. Finalmente, será explicada a implementação de cada um dos agentes e serviços disponíveis e será também dedicado um capítulo ao estudo dos mecanismos de simulação efectuados.

2. ARQUITECTURA

Para melhor compreendermos a arquitectura da aplicação implementada é preciso saber quais os objectivos a atingir. 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 A 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 [6] 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 [3], caso exista, para tentarem uma tradução do componente pedido para algum existente na sua lista.

Segue-se então uma negociação em várias rondas entre eles, de modo a serem determinados os vencedores (para cada componente) que irão então formar uma empresa virtual, com a assistência da Instituição Electrónica.

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.

Daqui para a frente esse contrato será monitorizado pela Instituição Electrónica e cada vez que uma obrigação ou condição for ou não cumprida, os agentes empresa envolvidos serão notificados por um dos serviços da plataforma, sendo que essa violação ou cumprimento ficará registada. Deste modo, cada agente poderá recorrer ao serviço de reputação existente na Instituição Electrónica, ou simplesmente ao seu registo interno de confiança, sabendo de antemão qual a dita reputação de cada agente possivelmente envolvido em negociações e contratos consigo mesmo. Isto permite a cada agente a possibilidade de tomar uma decisão mais consciente na escolha futura de parceiros de negócio.

Existem ainda outros agentes na Instituição Electrónica, que representam diversos papéis informativos ou auxiliares às operações passíveis de decorrerem na plataforma. Entre eles encontram-se os seguintes agentes:

- Bank, que guarda e actualiza informação financeira sobre cada agente empresa existente, desde balanço de contas até movimentos efectuados (particularmente entre dois agentes empresa).
- Delivery-Tracker, com informação sobre envio e recepção de itens entre agentes, de acordo com os contratos estabelecidos.
- Messenger, com o registo de mensagens trocadas entre agentes, relacionadas também com algum contrato em execução.

2.2 Classes

Podemos ter uma visão mais geral e abrangente de toda a plataforma a partir do diagrama de classes visível na figura 2.

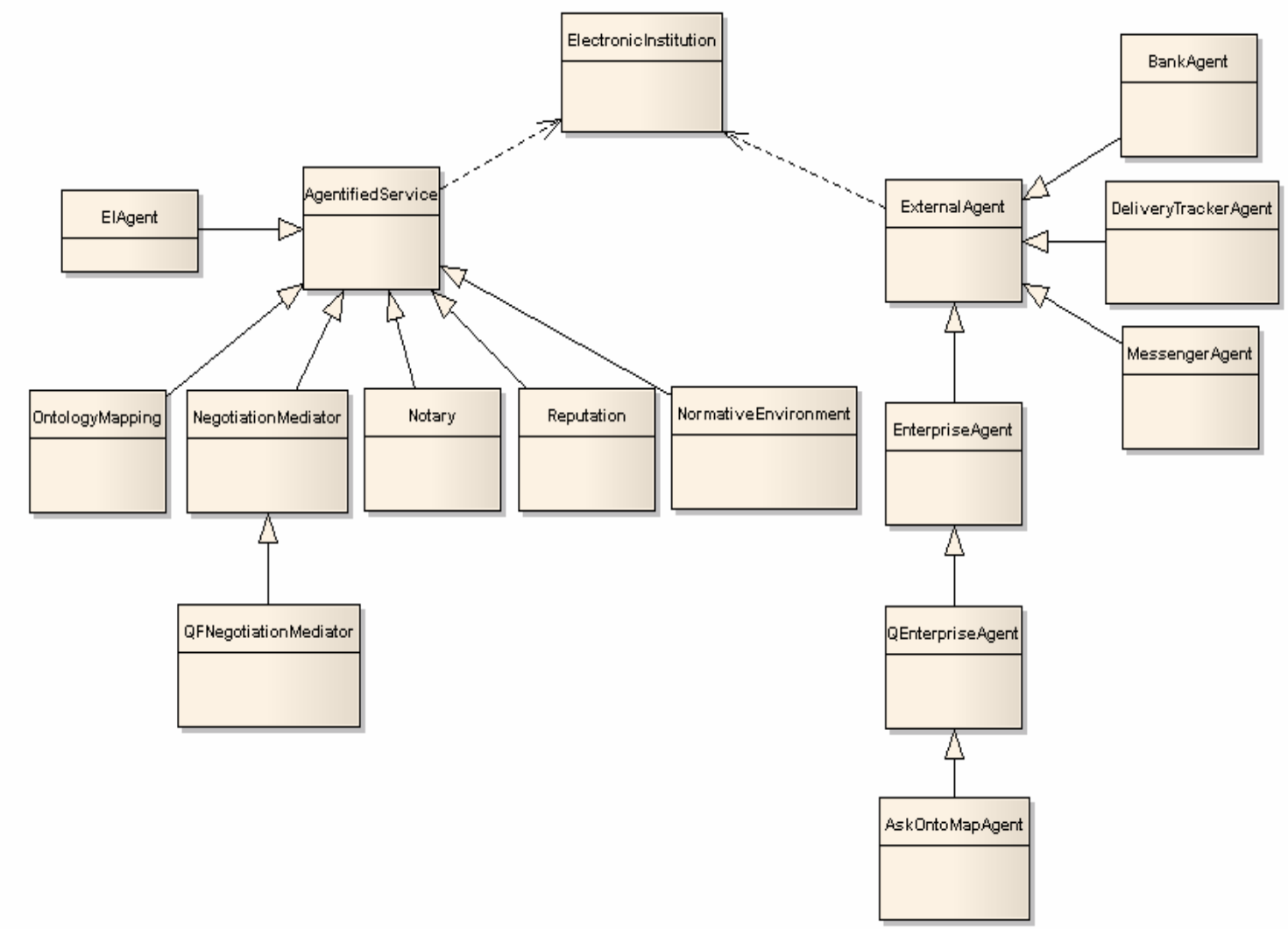

Figura 2 – Diagrama das principais classes presentes na plataforma

Como se pode ver no diagrama (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 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.
- `Reputation`, para agentes que forneçam serviços relacionados com reputação de agentes.
- `NormativeEnvironment`, para o agente representante do ambiente normativo.
- `EnterpriseAgent`, para agentes que forneçam e/ou pretendam adquirir produtos a outros;

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`, `OntologyMapping`, `Reputation` e `NormativeEnvironment`. 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. Para além destes foram também criados alguns agentes sobre a classe `AskOntoMapAgent`, que permitem automatizar o seu comportamento após a formação de alguma empresa virtual, fruto de negociações anteriores. Estes agentes facilitam a simulação destes comportamentos e permitem configurar as suas respostas de acordo com o que se pretende testar. Nomeadamente nos agentes criados da classe

`StochasticAutomaticEnterpriseAgent`, que nos permite configurar a probabilidade de falha de obrigações contratuais.

Outro agente de extrema importância para o correcto funcionamento da aplicação é o representado pela classe `NegotiationMediator`, que permite lidar com múltiplas negociações, atendendo também à escalabilidade e performance.

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

Todas estas classes descritas até ao momento 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 papéis
`ei.contract`	classes que tratam da parte da aplicação relacionada com contratos
`ei.normenv`	classes relacionadas com o agente que representa o Normative Environment
`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.onto.*`	definição das ontologias para os vários tipos e protocolos de comunicação existentes
`ei.service.notary`	agente que providencia serviços de notário e respectivas subclasses.
`ei.service.reputation`	relativo ao agente que lida com as reputações de todos os envolvidos em negociações

Tabela 1 – Descrição dos packages existentes no projecto

3. FUNCIONAMENTO

Nesta secção será descrito o modo de funcionamento de toda a plataforma na sua condição mais recente, do ponto de vista do utilizador.

3.1. Iniciar a Instituição Electrónica

Para que se possa iniciar a plataforma, será necessário definir previamente um ficheiro de configuração (formato xml) com a informação necessária sobre quais os agentes e serviços que irão correr, entre outras configurações mais específicas.

Este ficheiro indicará o tipo de ficheiros a lançar, especificando a sua classe e possíveis argumentos. Um exemplo deste ficheiro pode ser visto na figura 3.

Neste ficheiro, podemos configurar os seguintes aspectos da plataforma:

- Segurança – especificação de quais os serviços de segurança a lançar inicialmente, que serão melhor descritos à frente.
- Agentes/Serviços – descrição da classe para os agentes a serem lançados, definindo alguns parâmetros necessários ao seu correcto funcionamento

Cada um dos serviços disponibilizados pela Instituição Electrónica terá associado alguns parâmetros específicos que será necessário ter em conta. Todos os possíveis agentes terão um nome e um campo descritivo da classe que representa o agente e do tipo de serviço que proporcionam. Para além desta informação, poderão também ter algum parâmetro específico do seu tipo:

- **Negotiation Mediator Agent**
  - O ficheiro xsd utilizado para formar os contratos.
- **Ontology Mapping Agent**
  - O endereço e porto utilizados pelo servidor de WordNet.
- **Notary Agent**
  - O directório onde serão guardados os contratos.

```
l version="1.0" encoding="UTF-8"?>
CTYPE ei-config SYSTEM "ei-config.dtd">
config>
    <jade-parameters>
        <parameter>java.security.policy=D:\rneves\rational projs\Electronic Institution Java Project\config_rneves\policy.txt</parameter
        <parameter>jade.security.authentication.logincallback=CmdLine</parameter>
        <parameter>jade.security.authentication.loginmodule=NT</parameter>
        <parameter>java.security.auth.login.config=D:\rneves\rational projs\Electronic Institution Java Project\config_rneves\jaas.conf<
        <parameter>owner=rneves:rneves</parameter>
        <parameter>imtp=jade.imtp.rmi.RMISSLIMTPManager</parameter>
    </jade-parameters>
    <institutional-agents>
        <agent name="negmed">
            <class>ei.service.negotiation.qnegotiation.QFNegotiationMediator</class>
            <argument>service=IS-negotiation-mediator</argument>
            <argument>contract_xsd=D:\rneves\rational projs\Electronic Institution Java Project\contracts\contract.xsd</argument>
        </agent>
        <agent name="osa">
            <class>ei.service.ontology.OntologyMapping</class>
            <argument>service=IS-ontology-mapping</argument>
            <argument>wordnet_host=127.0.0.1</argument>
            <argument>wordnet_port=6180</argument>
            <argument>wordnet_file=D:\rneves\rational projs\Electronic Institution Java Project\config_rneves\WordNetSimilarity.dat<
        </agent>
        <agent name="notary">
            <class>ei.service.notary.Notary</class>
            <argument>service=IS-notary</argument>
            <argument>contract_dir=D:\rneves\rational projs\Electronic Institution Java Project\contracts</argument>
        </agent>
        <agent name="normenv">
            <class>ei.normenv.NormativeEnvironment</class>
            <argument>service=IS-normative-environment</argument>
            <argument>jess_file=D:\rneves\rational projs\Electronic Institution Java Project\ei.clp</argument>
        </agent>
        <agent name="reputation">
            <class>ei.service.reputation.Reputation</class>
            <argument>service=IS-reputation</argument>
            <argument>reputation_file=D:\rneves\rational projs\Electronic Institution Java Project\config_rneves\reputation.xml</arg
        </agent>

        <agent name="bank">
            <class>ei.agent.role.BankAgent</class>
            <argument>accounts_file=D:\rneves\rational projs\Electronic Institution Java Project\config_rneves\accounts.xml</argumen
        </agent>
        <agent name="delivery-tracker">
            <class>ei.agent.role.DeliveryTrackerAgent</class>
        </agent>
        <agent name="messenger">
            <class>ei.agent.role.MessengerAgent</class>
        </agent>

        <agent name="Request1Ont1">
            <class>ei.agent.enterpriseagent.StochasticAutomaticEnterpriseAgent</class>
            <argument>good=D:\rneves\rational projs\Electronic Institution Java Project\config_rneves\domotics_scenario\Request1Ont1
            <argument>external_contract_viewer=C:\Program Files\Mozilla Firefox\firefox.exe</argument>
            <argument>trust_file=D:\rneves\rational projs\Electronic Institution Java Project\config_rneves\domotics_scenario\trusts
            <argument>failure_probability=0.2</argument>
            <argument>min_response_time=0</argument>
            <argument>max_response_time=1</argument>
        </agent>
```

Figura 3 – Ficheiro de configuração da Instituição Electrónica

- **Reputation Agents**
    - O directório onde está a informação sobre as reputações de outros agentes.
- **Enterprise Agents**
    - Argumentos opcionais relativos aos componentes fornecidos ou pretendidos (formato OWL).
    - A aplicação utilizada para visualizar os contratos.
    - O ficheiro de reputação interno (trust) do agente.
- **Automatic Enterprise Agents**
    - A probabilidade de falhar em resposta a uma obrigação contratual.
    - O tempo máximo e mínimo que demora a responder a obrigações.

Dando como argumento este ficheiro de configuração (de que se pode ver o DTD no anexo I) a plataforma é então iniciada, mostrando uma janela inicial (figura 4) de onde se pode aceder aos diversos agentes em funcionamento.

Figura 4 – Ecrã inicial da plataforma da Instituição Electrónica

Neste ecrã inicial podemos seleccionar os qual o tipo de serviço ou papel que pretendemos ver, obtendo uma listagem de todos os agentes desse tipo disponíveis. A partir dessa lista é possível aceder aos interfaces de cada agente, caso possuam algum. Também poderemos abrir o RMA do JADE através do botão correspondente, ou fechar toda a plataforma.

3.2. Enterprise Agent

Com a plataforma já em funcionamento, poderemos aceder ao GUI de um dos agentes primordiais na Instituição Electrónica, o Enterprise Agent. Este tipo de agentes podem tirar partido dos serviços oferecidos e tanto podem fornecer como procurar adquirir componentes ou produtos.

Figura 5 – GUI para um Enterprise Agent – secção de fornecimento de componentes

O seu GUI pode ser visto na figura 5 e está dividido em múltiplas subsecções, correspondentes aos diversos tipos de informação disponível ao agente.

São estas secções as seguintes:

3.2.1. Supply

Todos os agentes representantes de empresas que pretendam disponibilizar os seus produtos para venda na plataforma terão que dar especial atenção a esta secção.

Na janela maior (ver figura 5) são visualizados os componentes fornecidos pelo respectivo agente, bem como quais os atributos de cada. Para cada um destes atributos é

configurada uma gama de valores preferencial (para o próprio agente) bem como um domínio para além do qual não serão aceites negociações. Estes valores podem também ser mostrados através de um duplo click no respectivo atributo (ver figura 6).

Figura 6 – Detalhes e preferências de um atributo

Para cada componente pode ser visto não só a sua classe principal como o seu tipo dentro dessa classe. Todos estes valores e configurações poderão ser carregados de um ficheiro OWL construído numa ferramenta própria para o efeito, neste caso o Protégé [??].

Para além disto é também possível restringir as negociações a agentes com uma reputação específica, tanto interna como da própria plataforma, através do menu apropriado, podendo também aceder aos CFPs recusados até ao momento, quer devido a provirem de agentes sem a reputação ou confiança necessária ou outra qualquer situação imprevista.

3.2.2. Request

A segunda secção do interface de um Enterprise Agent é em muitos pontos semelhante à primeira, uma vez que tem o objectivo oposto dessa. Aqui, tal como na anterior, pode ser visto um quadro com uma lista de componentes e atributos, e para cada atributo uma lista de preferências (ver figura 7). A diferença reside que esta será a

lista de componentes que o agente pretende adquirir, através dos serviços de negociação da plataforma.

Também aqui terá que ser carregado um ficheiro OWL com toda esta configuração detalhada, em conformidade com o pretendido. Pode configurar-se também a reputação mínima desejada aos intervenientes da futura negociação, bem como o tipo de contrato que se pretende efectuar no caso de sucesso. Finalmente, é possível configurar o número máximo de rondas que a negociação poderá atingir e iniciar a mesma.

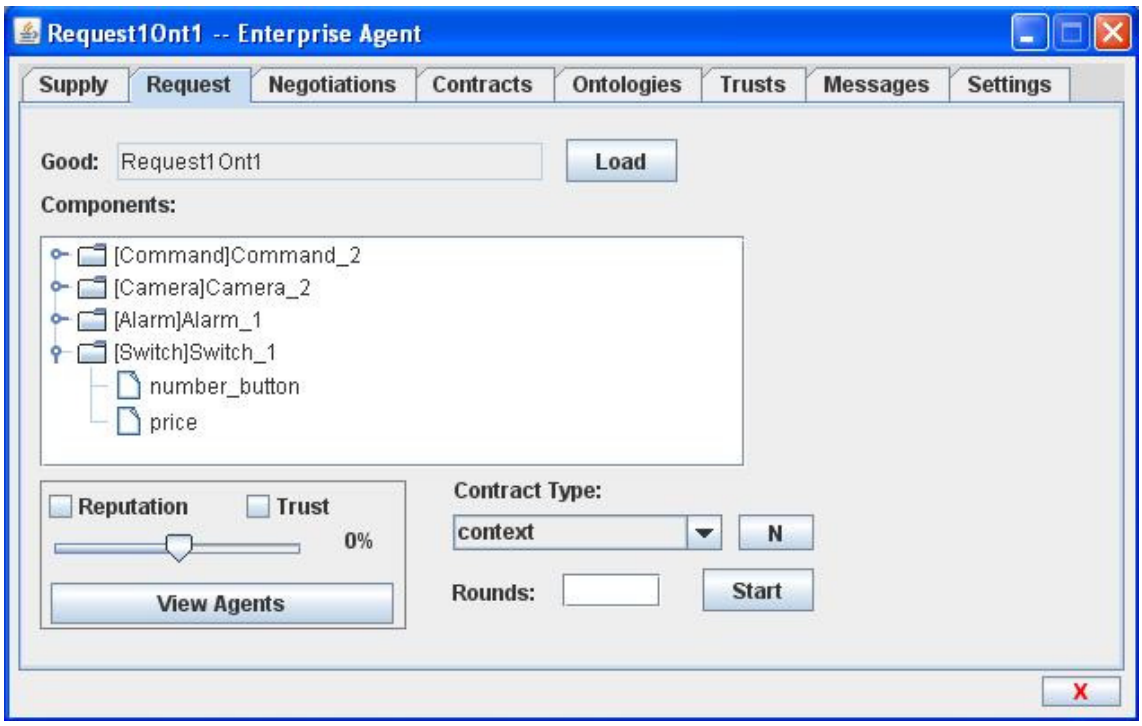

Figura 7 – GUI para um Enterprise Agent – secção de requisição de componentes

3.2.3. Negotiations

A secção seguinte serve como um histórico de informação sobre negociações decorridas. Aqui podemos ver quais as negociações em que o agente esteve envolvido, divididas pelo respectivo componente (ver figura 8).. Nessa informação é também visível o tipo de contrato proposto. Para cada um desses componentes temos então uma lista com todas as mensagens trocadas na negociação, onde através de um click na linha respectiva poderemos ver a mensagem correspondente em maior detalhe (figura 9).

Nesta janela poderemos então visualizar informação sobre os vários campos das mensagens trocadas, tais como:

- 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.

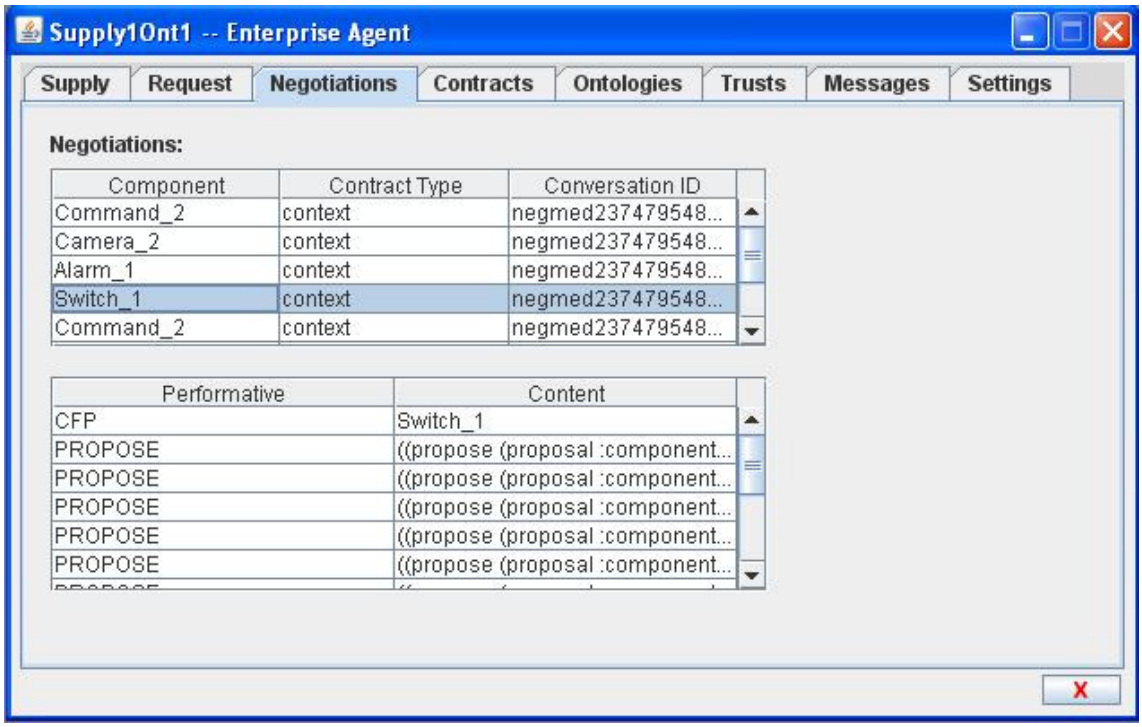

Figura 8 - GUI para um Enterprise Agent – informação sobre negociações

Figura 9 – Detalhes sobre uma mensagem ACL

3.2.4. Contracts

Numa perspectiva em tudo semelhante à do GUI anterior, nesta secção poderemos ver informação sobre os contratos em que o agente foi envolvido.

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, como violações ou obrigações. Seleccionando a mensagem pretendida, é possível com um duplo click no botão do rato aceder a informação sobre ela, de forma equivalente à secção das negociações.

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

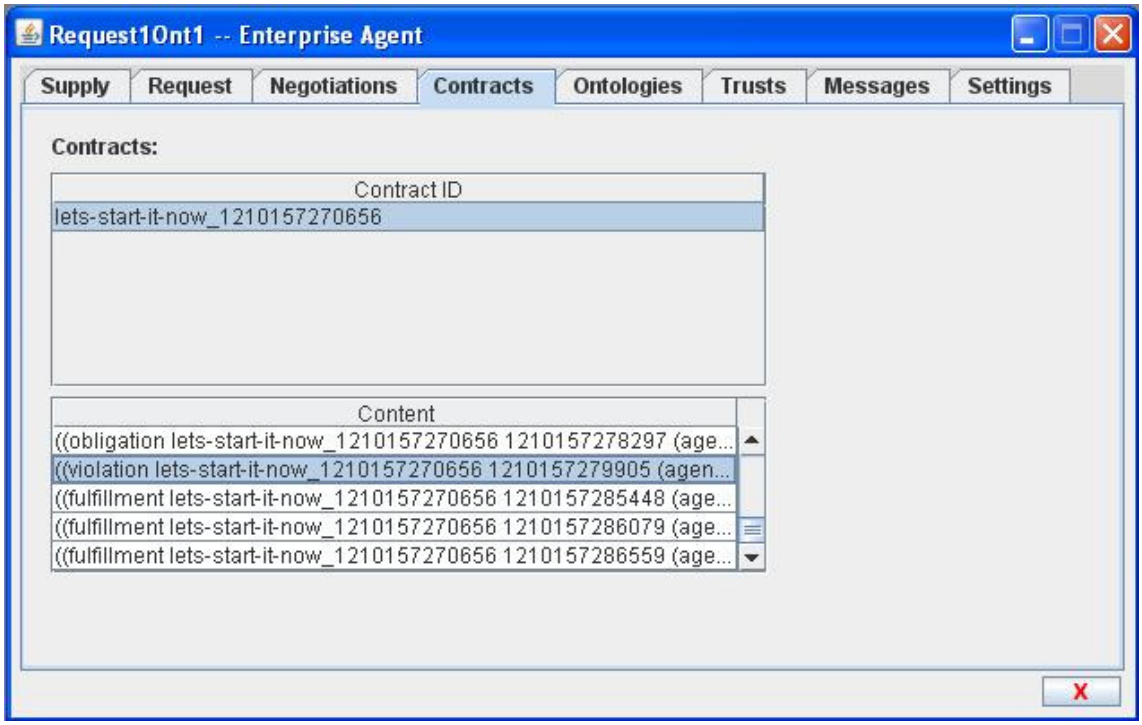

Figura 10 - GUI para um Enterprise Agent – informação sobre contractos

3.2.5. Ontologies

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 OWL. Estes ficheiros possuem informação sobre todos os componentes (e produtos) que um determinado agente pode fornecer.

Uma destas ontologias podem apresentar diferenças, quer na sua nomenclatura como nos seus atributos, de uma de outro agente com quem se pretenda negociar.

Um dos serviços disponibilizados pela Instituição Electrónica é precisamente o de tradução entre ontologias, de modo a possibilitar que dois agentes que peçam e forneçam respectivamente o mesmo componente, embora com nomes diferentes, não sejam impedidos de negociar. Assim sendo, quando um agente recebe um pedido de um componente que não possua, pede ao serviço de ontologia que lhe tente dar a correspondência entre o componente pedido e algum dos fornecidos por si.

É para se poder visualizar esta informação que serve a presente secção do GUI (ver figura 11). Aqui são detalhados numa lista os componentes pedidos que não foram reconhecidos mas cujo serviço de ontologia conseguiu obter correspondência para um dos fornecidos, bem como informação sobre essa mesma tradução. Pode ver-se qual a tradução efectuada para o nome do componente, bem como para cada um dos atributos.

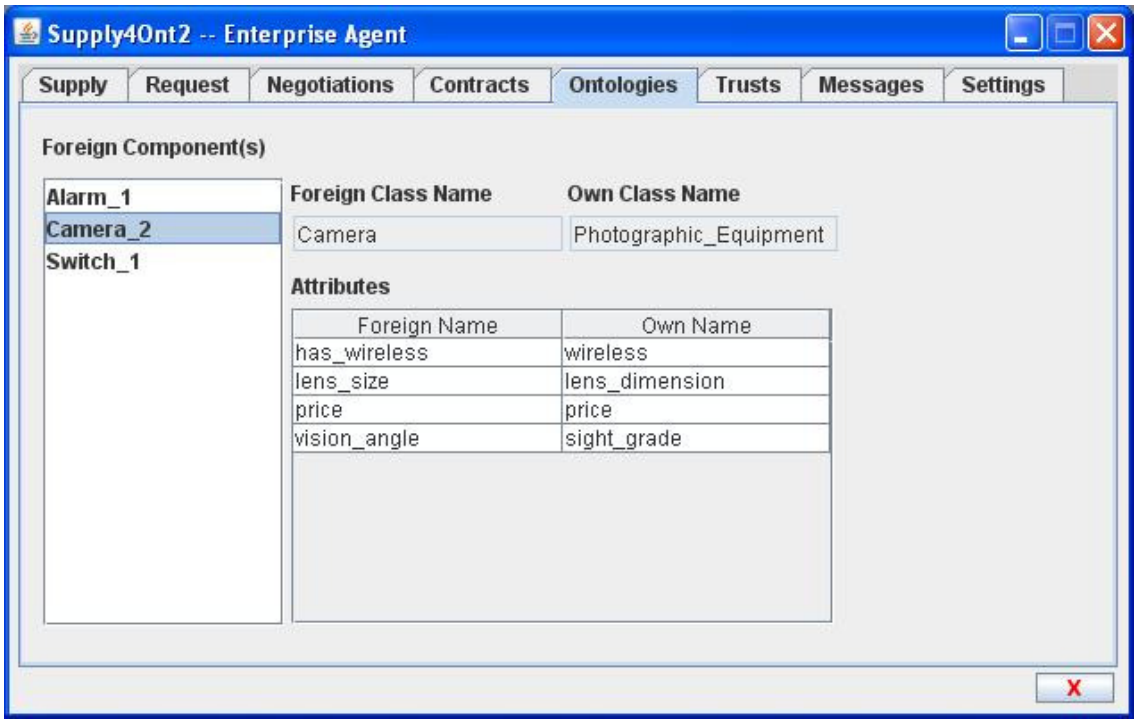

Figura 11 - GUI para um Enterprise Agent – informação sobre ontologias

3.2.6. Trusts

Um dos tipos de informação significativamente útil no domínio das negociações anónimas é a noção de confiança. Como tal, existe no GUI de um Enterprise Agent uma secção dedicada a este tema (ver figura 12).

Nesta secção pode ver-se uma listagem dos vários agentes que já participaram em negociações com o agente em questão. Para cada um destes agentes, temos acesso a informação sobre a suas quebras ou cumprimento dos contratos. São visíveis o número de obrigações cumpridas e violadas, bem como uma percentagem que corresponde à confiança depositada nesse agente. Toda esta informação está guardada num ficheiro XML (ou possivelmente em vários, para vários ambientes) que pode ser carregado ou gravado através dos botões respectivos. É também possível definir um rácio entre recompensa ou penalidade. Este rácio implica que o valor dado ao cumprimento de uma obrigação pode não ser o mesmo do dado à violação de uma condição contratual e depende de agente para agente.

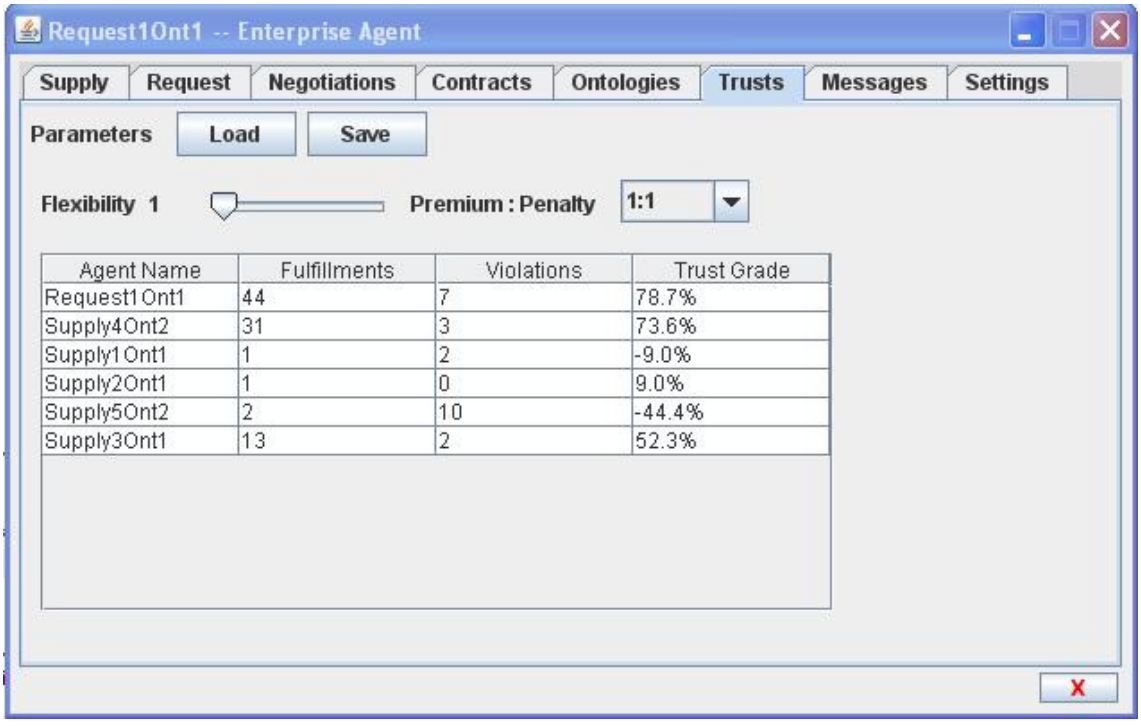

Figura 12 - GUI para um Enterprise Agent – informação sobre confiança

3.2.7. Messages

Neste local do GUI, é possível receber e enviar manualmente mensagens para qualquer agente registado na plataforma (ver figura 13). Primeiro temos um quadro em que serão mostradas todas as mensagens recebidas que não se enquadrem nos outros quadros (de negociações, contratos, etc). 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.

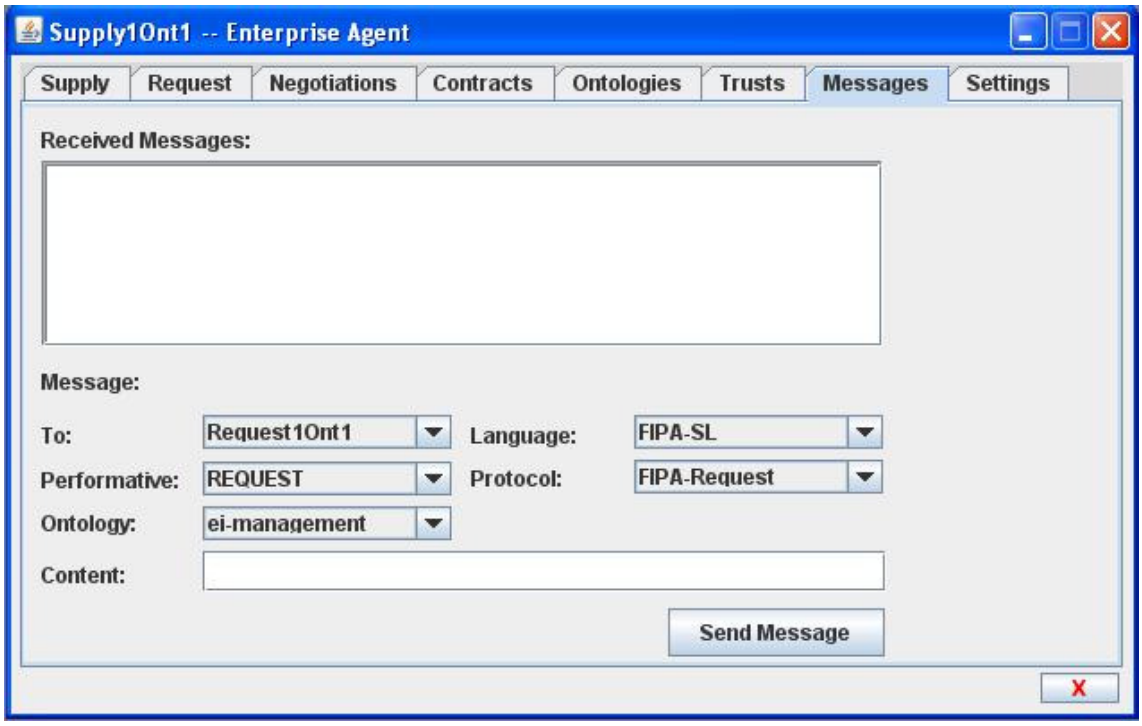

Figura 13 - GUI para um Enterprise Agent – enviar mensagens manualmente

3.2.8. Settings

A secção final só será apresentada caso o agente seja do tipo StochasticAutomaticEnterpriseAgent, pois é uma extensão ao GUI de um EnterpriseAgent normal, que permite configurar alguns parâmetros necessários à

simulação da plataforma em situações mais realistas (ver figura 14). São então configuráveis três atributos:

- Probabilidade de Falha – este parâmetros controla a percentagem com que o agente falha em responder a obrigações contratuais, o que significa que quanto mais alta for, pior reputação terá o agente ao fim de algum tempo.
- Tempo Máximo de Resposta – aqui poderemos configurar o tempo máximo que o agente demora a responder a essas obrigações, numa percentagem que vai desde a resposta imediata até ao prazo limite imposto nessa clausula contratual.
- Tempo Mínimo de Resposta – tal como o anterior, este parâmetro permite configurar o tempo, mínimo neste caso, que o agente demora a responder a obrigações. O seu tempo de resposta efectivo será um valor aleatório escolhido entre as duas percentagens. Isto pretende dar mais flexibilidade à simulação da realidade, visto que nesses casos nem todos os agentes responderão automaticamente, sendo possível que alguns requeiram intervenção humana, sujeita a atrasos portanto.

Figura 14 - GUI para um Enterprise Agent – configurações para agentes automáticos

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, atende-o embora nunca deixando de aguardar por novos pedidos. Deste modo a plataforma não entupirá ao receber múltiplos pedidos simultâneos.

Este agente, 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 proposta inicial para cada componente em questão, que será de seguida enviada de volta para o *Negotiation Mediator*. Este, após recepção de todas as propostas (ou recusas, que também serão possíveis em vários casos) 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 se determinar que alguma das propostas apresenta valores suficientemente apreciados pelo agente "cliente" para uma aceitação imediata. No final, uma mensagem com os resultados da negociação e o contrato formulado com base neles é enviada para o *Enterprise Agent* que iniciou a negociação, bem como para todos os agentes fornecedores envolvidos na negociação. Este contrato terá que ser assinado por todos e enviado para o agente notário, passando depois a ser monitorizado de forma a garantir que as condições estipuladas são cumpridas com rigor.

Tal como no caso anterior, também o Negotiation Mediator possui o seu próprio GUI (ver figura 15).

Figura 15 - GUI para um Negotiation Mediator – ecrã principal

Neste interface gráfico vemos os identificadores das negociações a decorrer, bem como o estado de cada uma delas. Através de um duplo click em cima de qualquer delas será mostrada uma nova janela com informação mais detalhada dessa negociação (ver figura 16).

Figura 16 - GUI para um Negotiation Mediator – detalhes de uma negociação

Esta janela 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 em negociação, disputando 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, através de um duplo click em cima do componente pretendido. Esta acção fará aparecer uma nova janela que consistirá num gráfico demonstrando o valor da utilidade das propostas de cada agente envolvido na negociação, em cada ronda (ver figura 17).

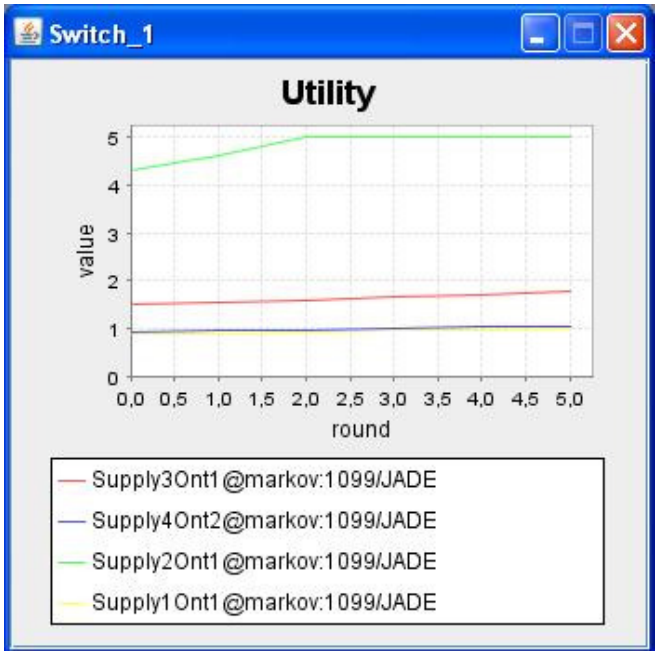

Figura 17 - GUI para um Negotiation Mediator – valores de utilidade por agente

Toda esta informação ajudará a compreender o estado e evolução de uma negociação mas é ainda possível obter gráficos detalhando a evolução das propostas para cada atributo específico (ver figura 18), em vez da visão geral apresentada nos gráficos de utilidade. Um duplo click sobre o gráfico de utilidade irá alterar a janela para que apresente os ditos gráficos, sendo que novo duplo click percorrerá o sentido inverso.

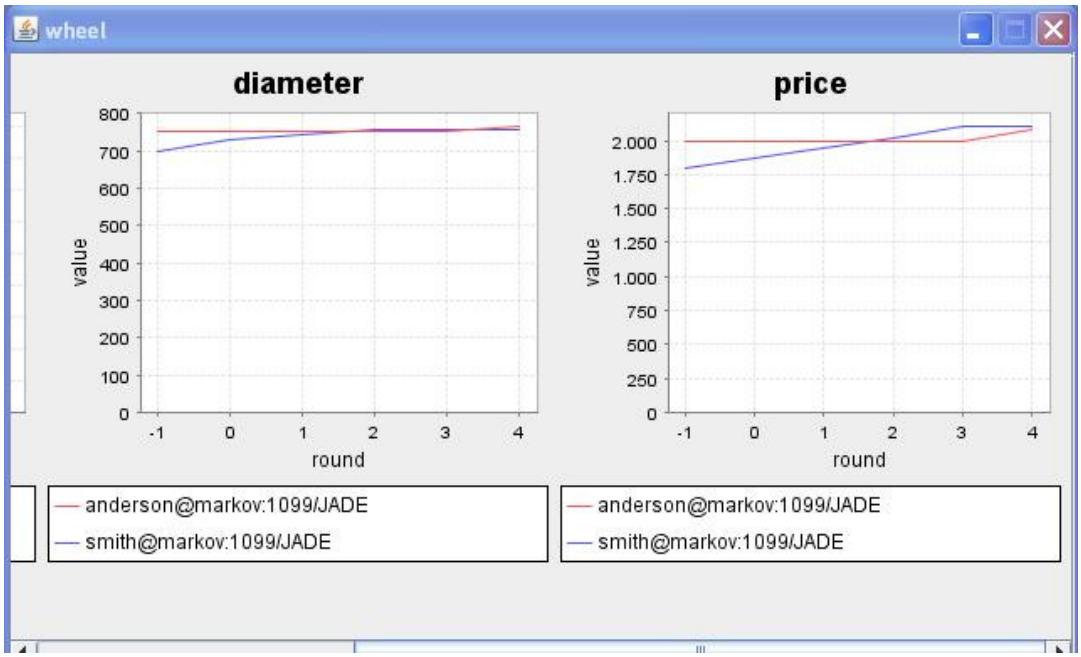

Figura 18 - GUI para um Negotiation Mediator – gráficos por atributo

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.

3.4. Ontology Service Agent

Um dos agentes pertencente à Instituição Electrónica é o *Ontology Service Agent*, que faculta serviços de ontologia a agentes que o requisitem.

O *Ontology Service Agent* funciona apenas no início de uma negociação, caso seja necessário. Quando um *Negotiation Mediator* envia um pedido acerca de determinado componente, e algum agente fornecedor verifica que não o tem disponível, pode pedir ao *Ontology Service 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 Service Agent* é então, descobrir a correspondência entre componentes com nomes diferentes (e possivelmente atributos diferentes), através da sua comparação, e dos seus atributos.

Este agente não tem nenhum GUI, até porque o seu funcionamento não tem necessidade de intervenção humana.

É apenas necessário configurar, no ficheiro de arranque da plataforma, tal como descrito em cima, 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 Service 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 Agent

Outro agente que falta referir e que, em oposição ao *Ontology Service 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, pedir as respectivas assinaturas ou enviar ao Normative Environment Agent os ditos contratos para monitorização.

Este agente funciona de maneira automática, sem qualquer GUI ou intervenção do utilizador, salvo a configuração inicial, tal como nos casos anteriores.

3.6. Normative Environment Agent

Este agente representa o ambiente normativo da Instituição Electrónica. O ambiente normativo contém uma série de normas e mecanismos para monitorizar e aplicá-las. Estes mecanismos e normas estão implementados através do Jess[??].

Os contratos são representados de forma declarativa, incluindo normas contratuais monitorizadas por si.

O Normative Environment Agent funciona, tal como os agentes anteriores, sem qualquer tipo de GUI, e não será foco de muita atenção neste relatório, estando fora do seu âmbito e do plano de trabalhos associado.

3.7. Bank Agent

Para além dos agentes que representam serviços na Instituição Electrónica, foram desenvolvidos novos agentes de particular importância às operações passíveis de existirem na plataforma e melhor descritos como representando papéis. Entre eles encontra-se o Bank Agent que, tal como o nome indica, representa um banco na Instituição Electrónica.

Este agente possui interface com o utilizador (ver figura 19), onde se pode verificar o balanço das contas de cada agente registado na plataforma. Para cada um destes é também possível aceder a um menu onde podemos ver alguns detalhes sobre os movimentos de conta dos respectivos agentes (ver figura 20). Entre estes detalhes encontram-se o AID do agente que iniciou o movimento, bem como do seu receptor, para além da quantidade movida.

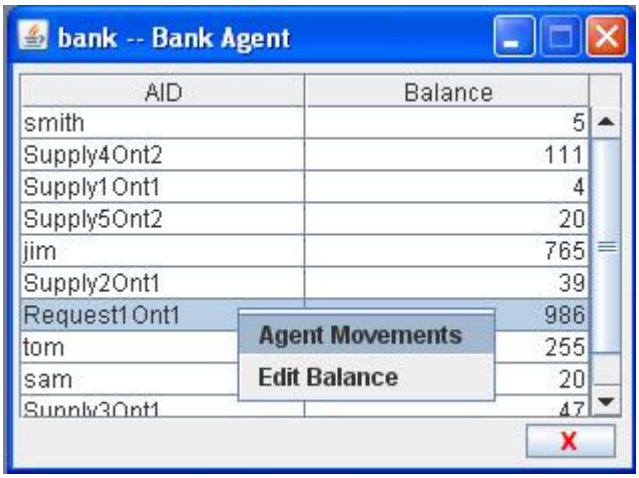

Figura 19 - GUI para um Bank Agent

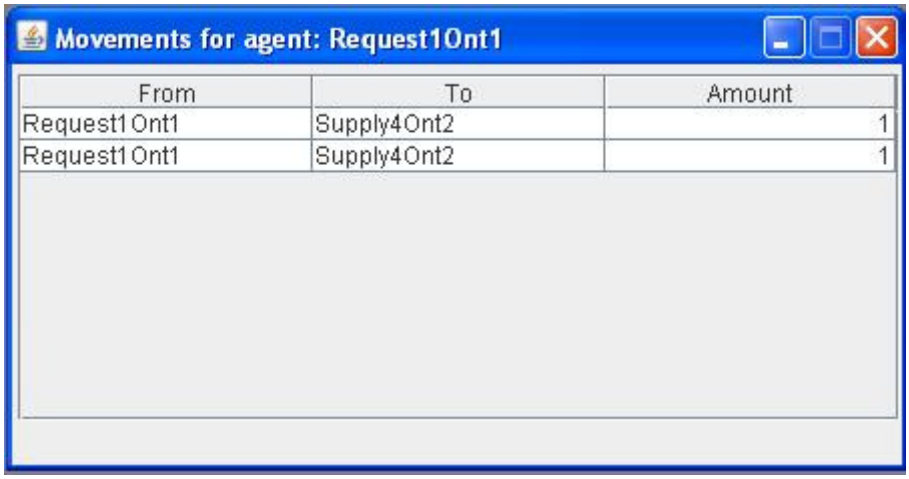

Figura 20 - GUI para um Bank Agent – movimentos de conta de um agente

A outra opção do menu contextual do GUI do Bank Agent permite editar manualmente a conta de um qualquer agente (ver figura 21). Esta operação será útil numa vertente de simulação e teste da plataforma, visto que numa situação normal este agente funcionará sem intervenção humana.

Figura 21 - GUI para um Bank Agent - edição de conta

3.8. Delivery Tracker e Messenger Agent

Os outros dois agentes representantes de papéis na Instituição Electrónica são o Delivery Tracker Agent e Messenger Agent. Serão aqui tratados em conjunto pois apresentam algumas semelhanças características.

O primeiro observa e regista todas as transacções entre agentes, efectuadas sobre contratos prévios. O seu interface com o utilizador (ver figura 22) apresenta uma listagem com o nome dos agentes remetentes e receptores da transacção, o contexto (tipo de contrato) e o item que foi transaccionado, bem como a respectiva quantidade.

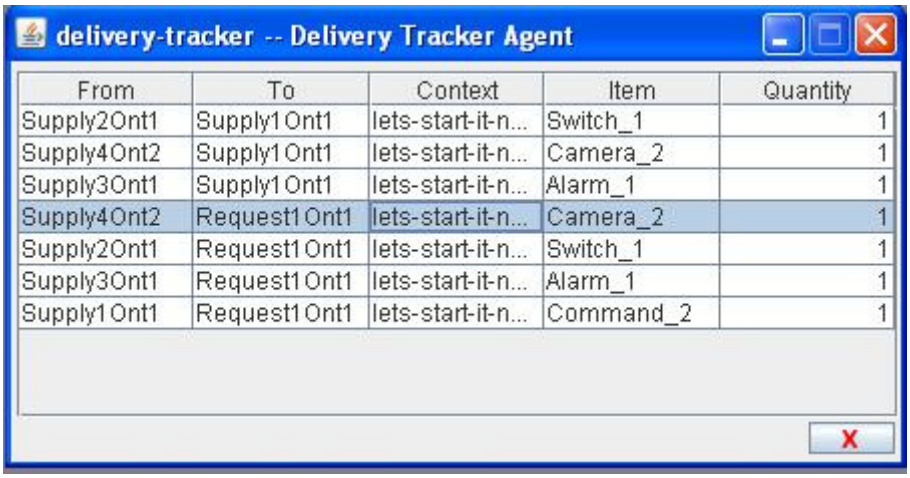

Figura 22 - GUI para um Delivery Tracker Agent

O segundo destes agentes, o Messenger Agent, apresenta uma listagem semelhante, embora relativa apenas a mensagens trocadas, num GUI em tudo semelhante ao da figura 22.

3.9. Segurança

Também houve o objectivo de estudar os mecanismos de segurança associados ao JADE, através do seu add-on de segurança (JADE-S) e integrá-los na já existente Instituição Electrónica, permitindo que os agentes pertencentes aos serviços disponibilizados estejam autenticados e tenham permissões bem definidas e que as mensagens trocadas entre eles sejam enviadas devidamente encriptadas e assinadas, permitindo assim uma muito maior segurança em qualquer operação relacionada com os serviços oferecidos pela plataforma.

A plataforma actual da Instituição Electrónica pretende facilitar/automatizar processos para a formação de organizações virtuais, que através de contratos que formalizam acordos de colaboração entre várias empresas, representadas por agentes com diferentes estratégias e objectivos.

Para isso disponibiliza uma série de serviços, desde gestão de negociações e contratos a serviços de ontologia, entre outros.

O trabalho desenvolvido consistiu em implementar os mecanismos de segurança do add-on de segurança do JADE (JADE Security) nesta plataforma da Instituição Electrónica, de modo a assegurar que a gestão da mesma e toda a troca de mensagens associada não sofrerá intervenções maliciosas externas ao sistema. A implementação consistiu em três fases distintas:

- Introdução de mecanismos de autenticação e permissões;
- Encriptação de todas as mensagens trocadas;
- Assinatura de todas as mensagens trocadas, com especial relevância para as que envolvem contratos;
- Assinatura dos próprios contratos.

3.9.1. Mecanismos de Autenticação e Permissões

A plataforma disponibiliza agora um serviço de autenticação e permissões. Exige, portanto, um nome de utilizador e uma palavra passe para que lhe seja possível aceder. Isto permite garantir que apenas utilizadores autorizados podem iniciar ou aceder à plataforma. Além disso, também implica que cada utilizador apenas possa efectuar determinadas operações de acordo com as permissões estabelecidas para o mesmo, permitindo estabelecer níveis de acesso diferentes para diferentes utilizadores. Por exemplo, enquanto que um utilizador representante de uma empresa apenas poderá criar os seus próprios agentes e enviar mensagens aos outros, um administrador da plataforma já poderá criar ou matar qualquer agente, suspender ou resumir a sua actividade, além de também enviar mensagens para qualquer um, obviamente.

3.9.2. Encriptação de Mensagens

Todas as comunicações estão agora protegidas por encriptação, evitando assim que, caso sejam adquiridas por entidades exteriores ao sistema, possam ser lidas correctamente. Uma mensagem ACL é composta por duas partes, o “envelope”, que

contém informação relacionada com o seu transporte e a "payload" informação que realmente se quer transmitir (conteúdo, ontologia, protocolo,...). A encriptação aplica-se a esta última informação, tal como a assinatura como veremos à frente. Caso ocorra algum problema com a encriptação (ou assinatura) da mensagem, ela é eliminada e é enviada uma mensagem de performativa "FAILURE" para o remetente.

Sendo a encriptação uma operação que consome algum tempo e recursos, não é aplicada em mensagens entre agentes da mesma plataforma.

3.9.3. Assinatura de Mensagens

Essas mesmas mensagens são agora também assinadas para que todos os agentes ao recebê-las tenham garantias da sua proveniência, sabendo assim que não foram alteradas durante o seu percurso. Isto será feito através dum mecanismo normal de chave pública/privada automático, existente no JADE Security.

Caso ocorra algum problema com a assinatura (ou encriptação) da mensagem, ela é eliminada e é enviada uma mensagem de performativa "FAILURE" para o remetente. A implicação imediata disto é que de cada vez que um agente recebe uma mensagem assinada, sabe que a assinatura é válida (ou seja, não tem que verificar a assinatura outra vez).

Um dos tipos de mensagens em que a segurança tem mais relevância é o que envia os contratos criados no final de negociações bem sucedidas. Neste caso é também a própria mensagem a ser assinada e não o objecto que representa o contrato, visto que o JADE-Security apenas permite a assinatura de mensagens ACL.

3.9.4. Assinatura de Contratos

Os contratos resultantes de negociações bem sucedidas são também eles próprios assinados, embora não o sejam através da utilização do JADE-Security.

Isto é feito através do Java Security, no qual o próprio JADE Security é baseado, através do método de chave pública/privada.

O contrato é assinado utilizando a chave privada do agente respectivo e enviado para o agente notário, que tem como tarefa verificar essa assinatura. Para isso pede ao primeiro agente a sua chave pública e verifica se o objecto continua intacto.

4. IMPLEMENTAÇÃO

Na secção anterior foi descrito o funcionamento do ponto de vista do utilizador quer da Instituição Electrónica no geral, tal como para cada um dos agentes envolvidos na Instituição Electrónica. Passamos 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 Eclipse como IDE, utilizado para a programação e desenho 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 efectuar (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 qual a acção a tomar aquando da sua recepção.

4.2. Instituição Electrónica

A Instituição Electrónica, na forma da sua classe principal, limita-se a lançar os agentes, caberá depois 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.
- *IS-normative-environment,* para o ambiente normativo.

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

De modo a simplificar todo o processo de um Enterprise Agent utilizam-se várias classes distintas na sua implementação, 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. Cada uma destas classes permite especificar um dos níveis do comportamento dos agentes, desde o funcionamento básico à estratégia de negociação utilizada.

Existem portanto quatro 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.
- *StochasticAutomaticEnterpriseAgent,* que permite automatizar o comportamento dos agentes para utilizar em simulações e testes da plataforma, através de comportamentos semi-aleatórios baseados em parâmetros previamente configurados.

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

A classe abstracta EnterpriseAgent 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 na 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 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 serviço 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 alguma possibilidade de serem os equivalentes, nos casos em que haja a necessidade de tradução entre agentes com ontologias distintas.

Caso o agente em causa pretenda 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 receberá 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 protocolo *QF Negotiaton*. 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 criada dando, a 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, nem o seu domínio, obviamente. 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 um comentário 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: “excellent”, “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 “excellent” 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.

4.3.5. StochasticAutomaticEnterpriseAgent

Esta classe irá implementar os já falados comportamentos automáticos por cima de um Enterprise Agent, permitindo assim testar a plataforma através de simulações aleatórias de possíveis comportamentos mais ou menos realistas de um agente no mundo real.

Este tipo de agente e a sua classe pode ser examinada em maior detalhe no capítulo 5.

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. 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, inicia o seu tratamento, através do

envio de mensagens para os Enterprise Agents disponíveis, continuando à escuta de novos pedidos. Desta forma não bloqueará caso seja alvo de múltiplos pedidos simultâneos, melhorando significativamente a sua fiabilidade.

Depois, aguarda durante um determinado período de tempo por propostas (ou recusas em negociar) vindas desses agentes. No fim desse tempo, verifica se recebeu alguma proposta e, em caso negativo aborta a negociação, informando os agentes respectivos, caso já esteja algum envolvido na negociação, para além do inicial. 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 Negotiation *Mediator*.

Este sistema continua até terminarem as rondas definidas pelo agente que iniciou o pedido, ou até o *Negotiation Mediator* 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.

Em seguida, e no caso de sucesso, obtém um contrato de acordo com os resultados da negociação, que enviará para todos os agentes envolvidos para validação e respectiva assinatura, incluindo 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.

Este agente também responde a pedidos do agente de serviços de ontologia, semelhante ao encontrado nos *Enterprise Agents*. Recebe pedidos (REQUEST) acerca dos detalhes dos componentes fornecidos e responde de forma apropriada equivalente à já descrita na situação anterior.

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. QFNegotiationMediator

Esta classe representa um *Negotiation Mediator* que utiliza o protocolo de negociação QF. Este protocolo 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 "excellent".

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, "excellent", será apenas utilizado quando o valor oferecido for superior ao valor esperado pelo *Negotiation Mediator*, 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 Service Agent

Este agente, que fornece mais um dos serviços disponíveis 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 Service 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 Service 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 Mediator. 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 [7], 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 Service 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 notário 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. BankAgent

Este agente representa um banco registando dados sobre os balanços de contas de agentes empresa registados na plataforma e os seus movimentos. Implementa protocolos para possibilitar transacções financeiras entre os vários agentes. Todos os registos são gravados num ficheiro especificado na configuração inicial para que estejam disponíveis sem falhas em futuras ocasiões.

4.9. DeliveryTrackerAgent

A classe representante do Delivery Tracker Agent serve um papel semelhante ao do banco mas relacionada com transacções de bens materiais (ainda que virtualmente). Aqui são registadas todas as transacções que ocorram derivadas de algum contrato formado anteriormente. Estas transacções ocorrem automaticamente em qualquer Automatic Enterprise Agent (a versão de simulações de um Enterprise Agent normal) a fim de cumprir obrigações contratuais existentes.

4.10. Messenger

O agente Messenger apenas implementa um protocolo para troca de todo o tipo de mensagens relativas a contratos, desde que não se insiram nas tipologias de transacções financeira ou de bens materiais.

4.11. NormativeEnvironment

A classe NormativeEnvironment representa, como seria de esperar, um agente monitorizador do ambiente normativo da Instituição Electrónica. Este tipo de agente não será explicado em maior pormenor visto que se encontra fora do âmbito da Bolsa de Investigação.

4.12. Segurança

Os sistemas multi-agente, quando executados num ambiente aberto exigem um grande nível de segurança a todos os níveis. A Instituição Electrónica aqui falada, pretendendo para além de negociações e outros serviços, a criação e monitorização de contratos, é um exemplo perfeito deste nível de segurança necessário.

O JADE Security (versão 3.5) permite a protecção de um sistema multi-agente construído sobre o JADE, dificultando e dissuadindo intervenções maliciosas externas.

Todos os agentes num destes sistemas são pertença de alguém devidamente identificado, e com permissão para executar apenas determinadas acções, definidas previamente pelo administrador da plataforma. Cada agente contém também uma chave privada e pública para poder encriptar e assinar mensagens.

Uma limitação conhecida do JADE Security é a inexistência de permissões relativas a agentes móveis, ou seja, não contempla ainda suporte ao serviço de mobilidade de agentes do JADE. Como consequência, e de maneira a continuar seguro, as plataformas que utilizem este add-on não devem ter o serviço de mobilidade activo ao lançar o JADE.

Outra limitação no âmbito deste trabalho é a inexistência da possibilidade de assinar os objectos transportados nas mensagens para além das próprias mensagens, o que fez com que fosse necessário utilizar o próprio Java Security, para assinatura dos objectos que representam contratos.

Descrevemos agora em mais detalhe a implementação efectuada no que à segurança diz respeito. Em conformidade com a arquitectura do JADE baseada no conceito de serviços, cada um lidando com um aspecto distinto, o suporte à segurança é

implementado como um conjunto de serviços do JADE. Existem quatro serviços relativos à segurança

4.12.1. Autenticação

O serviço principal (jade.core.security.SecurityService), gere os mecanismos de autenticação mas também providencia funcionalidades comuns a todos os serviços de segurança, como a gestão de pares de chaves entre outros

Este mecanismo de autorização é baseado no JAAS (Java Authentication and Authorization Service) que providencia vários módulos para efectivar o login: ***NT, Unix e Kerberos***. O módulo ***Unix*** e ***NT*** são específicos para os respectivos sistemas operativos, fazendo uso da identidade do utilizador retirada da sessão corrente do próprio sistema operativo. O módulo ***Kerberos*** é independente do sistema mas requer configurações específicas para utilizar.

A plataforma utiliza o módulo NT, que é especifico para o sistema operativo Windows, no qual está a correr no momento, podendo no entanto ser configurada para utilizar outro, caso seja necessário corrê-la noutro sistema operativo.

O método de introdução no sistema da password e nome de utilizador também pode ser configurado entre uma janela para introdução manual até à introdução desses dados no ficheiro de configuração. Está a ser utilizada a segunda opção, pois a primeira será mais indicada para plataformas com muitos utilizadores distintos (no mesmo computador), o que não é o caso nesta plataforma.

Este mecanismo é configurado no ficheiro de configuração principal da Instituição Electrónica (eiconfig.xml). Aí poderemos definir:

- O ficheiro de configuração de permissões a utilizar;
- O método de login;
- O módulo de login a utilizar, entre os descritos acima;
- O ficheiro de configuração do JAAS;
- O login e password a utilizar;

Isto pode ser configurado com a sintaxe vista no exemplo da figura 2, que mostra um exemplo da parte relacionada com a segurança do ficheiro de configuração da plataforma da Instituição Electrónica.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ei-config SYSTEM "ei-config.dtd">
<ei-config>
<jade-parameters>
        <parameter>java.security.policy=policy.txt</parameter>
        <parameter>jade.security.authentication.logincallback=CmdLine</parameter>
        <parameter>jade.security.authentication.loginmodule=NT</parameter>
        <parameter>java.security.auth.login.config=jaas.conf</parameter>
        <parameter>owner=rneves:rneves</parameter>
        <parameter>imtp=jade.imtp.rmi.RMISSLIMTPManager</parameter>
</jade-parameters>
```

Figura 23 – Exemplo do ficheiro de configuração (parâmetros de segurança)

4.12.2. Permissões

O segundo serviço destes quatro tem a seu cargo a verificação de que os agentes que estão a efectuar determinadas operações têm realmente autorização para o fazer.

Para além da óbvia vantagem, em termos de segurança, de que apenas utilizadores autorizados possam aceder à plataforma, o mecanismo de autenticação faz com que a plataforma passe a permitir múltiplos utilizadores, onde cada um passa a ser "dono" dos agentes ou containers que cria. Isto faz com que haja a possibilidade de definir diferentes níveis de acesso, através das ditas permissões, para diferentes utilizadores. Estes utilizadores ficam então a poder, ou não, efectuar determinadas operações desde criar novos agentes ou eliminá-los até algo tão simples como apenas poder enviar mensagens a determinados agentes. As regras que definem estas permissões estão localizadas num ficheiro (policy.txt), que segue a sintaxe normal do JAAS (Java Authentication & Authorization Service), mas que permite uma maior flexibilidade num sistema baseado em agentes distribuídos.

Estes parâmetros também podem ser configurados no ficheiro de configuração da plataforma (eiconfig.xml), nomeadamente ao nível da selecção do local do ficheiro "policy.txt" visto em cima. Um exemplo desta sintaxe pode também ser visto na figura 2, no tópico anterior.

Este ficheiro tem não só a descrição das permissões de cada agente, mas também as permissões sobre os próprios ficheiros e código do programa, permitindo (ou não) que a plataforma corra. Um exemplo dum ficheiro deste tipo pode ser visto na figura 3

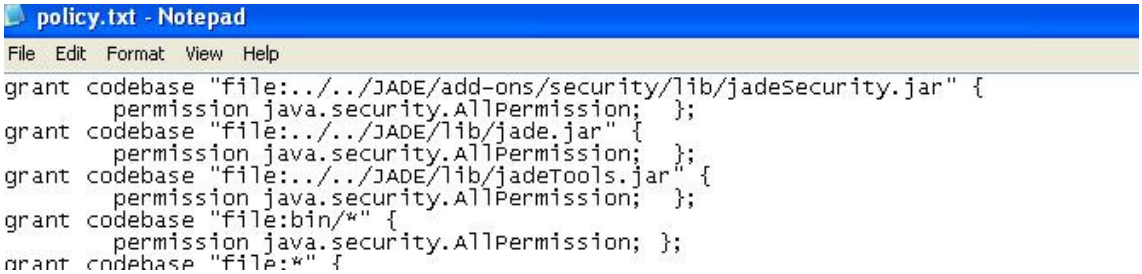

Figura 24 – exemplo de um ficheiro policy.txt para definição de permissões

Começa-se por dar permissões aos próprios ficheiros (tanto do JADE como do resto do projecto respectivo), através da sintaxe vista no exemplo, especificando o ficheiro em questão e a permissão pretendida (casos de "all","read","write",etc)

Depois de todos os ficheiros terem a permissão respectiva, pode fazer-se o mesmo para os utilizadores da plataforma. Um exemplo disto pode ser visto na figura 4, para um utilizador "rui", com permissões para criar e matar agentes, enviar mensagens e até modificar ou registar agentes ao nível do AMS.

```
grant principal jade.security.Name "rui" {
        permission jade.security.PlatformPermission "", "create,kill";
        permission jade.security.ContainerPermission "", "create,kill";
        permission jade.security.AgentPermission "", "create,kill";
        permission jade.security.AgentPermission "", "suspend,resume";
        permission jade.security.AMSPermission   "", "register,deregister,modify";
        permission jade.security.MessagePermission "", "send-to";
};
```

Figura 25 – outro exemplo de um ficheiro policy.txt para definição de permissões

Existem dois tipos de ficheiros possíveis onde se podem distribuir as permissões pelos utilizadores. Um referente ao "Container" principal que define permissões alargadas a toda a plataforma (ex: os agentes criados pelo utilizador X podem matar os agentes criados pelo utilizador Y), e outros que se referem a "containers" específicos (ex: os agentes criados por X podem matar agentes criados por Y neste container) .

Cada utilizador pode ter então vários tipos de permissões, relativas a elementos diferentes do sistema:

- Toda a Plataforma, que permite que os utilizadores possam criar ou matar agentes dentro da plataforma;
- "Containers" específicos, com as características anteriores mas em relação a "containers" pertencentes ao próprio utilizador (ou a outro específico);

- Agentes específicos, dando autorização ao utilizador para matar ou criar agentes de um determinado "container" ou com um nome, classe ou dono particular, ou mesmo suspender ou resumir o funcionamento de um agente;
- Mensagens, que permitem que o utilizador envie mensagens a agentes pertencentes a outros utilizadores específicos ou com um determinado nome.

4.12.3. Encriptação de Mensagens

O serviço jade.core.security.encription.EncryptionService trata dos mecanismos relacionados com a encriptação de mensagens, e respectiva desencriptação no receptor.

Foi implementada a encriptação de todas as mensagens utilizadas nas comunicações entre agentes da plataforma. Neste caso, é possível configurar qual o algoritmo a usar e o tamanho da chave em bits de entre as escolhas seguintes (AES e 128 respectivamente, por defeito):

Parameter Name:	jade.security.AsymAlgorithm
Synopsis	The default asymmetric algorithm that is used to generate a key pair.
Values	*RSA (default value)* DSA …

Parameter Name:	jade.security.AsymKeySize
Synopsis	Sets the default key size for public/private keys.
Values	*512 (default value)* 1024 2048

Parameter Name:	jade.security.SymAlgorithm
Synopsis	Sets the default symmetric algorithm used when the message has to be encrypted
Values	*AES (default value)* Blowfish DES DESede TripleDES PBEWith<digest>And<encryption> e.g. PBEWithMD5AndDES PBEWith<prf>And<encryption> e.g. PBEWithHmacSHA1 AndDESede

Figura 26 – tabela de parâmetros de configuração de segurança e os seus valores possíveis

Como estes mecanismos diminuem a performance dos agentes, nenhuma mensagem é encriptada automaticamente, cabendo a cada agente pedir explicitamente que o pretende no código de envio da mensagem. Foi por isso necessário alterar todos os envios de mensagens existentes na Instituição Electrónica para que tal sucedesse.

No entanto, todo o processo de encriptação é automático, através do JADE-S, sendo apenas necessário indicar que se pretende que a mensagem seja encriptada ou verificar e desencriptar, no caso do receptor.

4.12.4. Assinatura de Mensagens

Finalmente, o último dos serviços de segurança, mas não menos importante, prende-se com a assinatura de mensagens, e é fornecido arrancando o serviço jade.core.security.signature.SignatureService. Este serviço trata de assinar as mensagens e de verificar a validade das assinaturas de mensagens recebidas.

Foi então implementado este sistema de assinatura de mensagens entre agentes presentes na Instituição Electrónica. Também neste caso é possível configurar o algoritmo a utilizar, de modo a atingir o balanço entre performance e segurança desejados, de entre as opções visíveis a seguir:

Parameter Name:	jade.security.SignAlgorithm
Synopsis	Sets the default algorithm that is used when messages have to be signed.
Values	SHA1withRSA MD5withRSA SHA1withDSA DSA …

Figura 27 – parâmetros para assinatura de mensagens e respectivos valores possíveis

A informação relacionada com esta segurança, ou seja, a assinatura (mas também o algoritmo ou a chave pública, no caso da encriptação) é colocada no envelope. Todo o processo de assinatura (e encriptação) é automático, sendo apenas necessário indicar que se pretende que a mensagem seja assinada ou verificar se o é, no caso do receptor.

A excepção a isto são os contratos, que são assinados com uma chave privada gerada por cada agente, e verificados aquando da sua recepção, utilizando as classes disponibilizadas pelo Java Security. Também neste caso é possível definir o algoritmo a utilizar, sendo que neste momento é utilizado o DSA 1024-bits.

4.13. Protocolos de Comunicação

Todo o sistema foi implementado utilizando JADE, que funciona com base na troca de mensagens ACL entre os vários agentes existentes. Desta maneira, todas as comunicações podem ser baseadas em protocolos pré-definidos, em que cada agente implementa os *initiator* ou *responder roles,* sendo que o primeiro permite iniciar conversações, aguardando depois pelas respostas e o segundo espera por mensagens iniciadoras de conversações, respondendo-lhes em seguida. Isto é efectuado, na sua maioria, através das classes do JADE *AchieveREInitiator* e *AchieveREResponder*, facilitando assim o seu desenvolvimento. Veremos agora quais os protocolos utilizados nas comunicações entre eles.

4.13.1. 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 28.

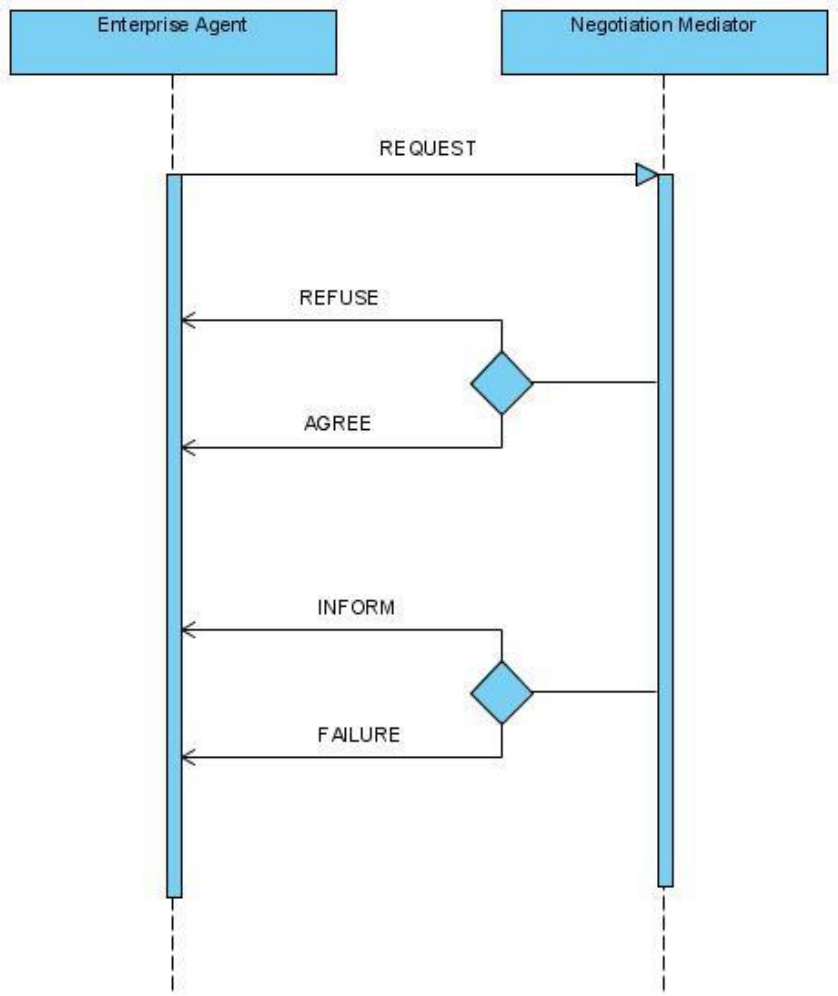

Fig.28 – Diagrama de comunicação entre um *Enterprise Agent* e um *Negotiation Mediator*

4.13.2. Negotiation Mediator → Enterprise Agent(s)

A comunicação da fase seguinte da negociação acontece depois do pedido ao *Negotiation Mediator*. O *Negotiation Mediator* 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 29.

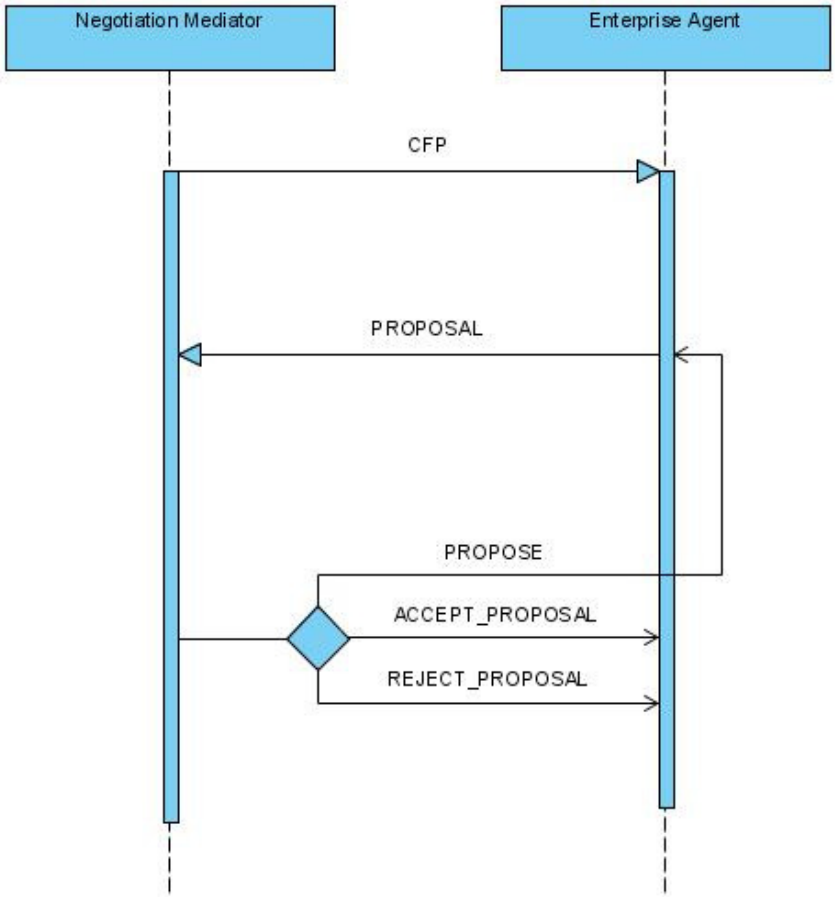

Fig.29 – Diagrama de comunicação entre um *Negotiation Mediator* e vários *Enterprise Agents*

4.13.3. Enterprise Agent → Ontology Mapping Agent (→ Negotiation Mediator)

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 30. É também feito um pedido ao Negotiation Mediator para que forneça alguma informação sobre os componentes em questão.

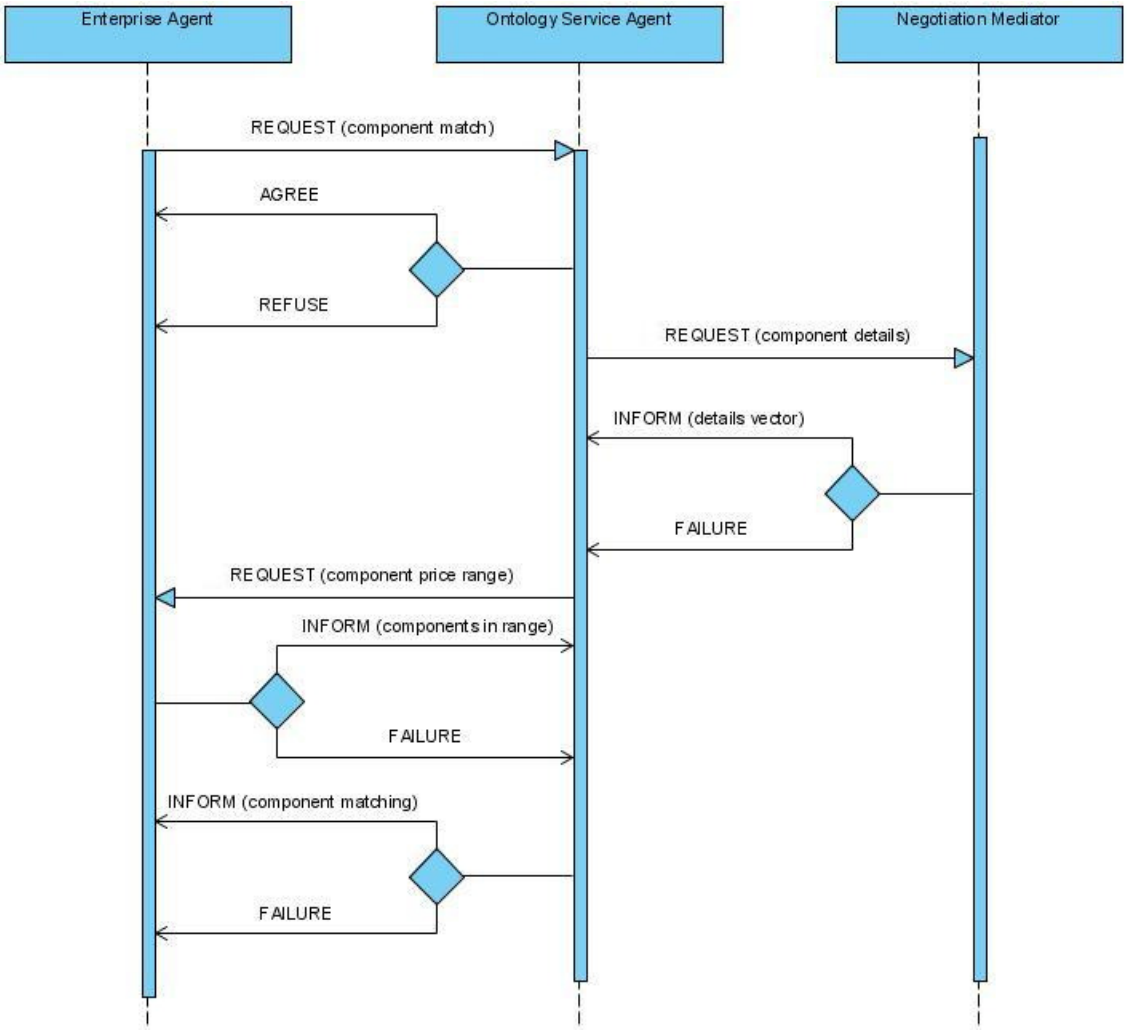

Fig.30 – Diagrama de comunicação entre um *Enterprise Agent, um Ontology Service Agent (* e um *Negotiation Mediator)*

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

Iremos 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 31.

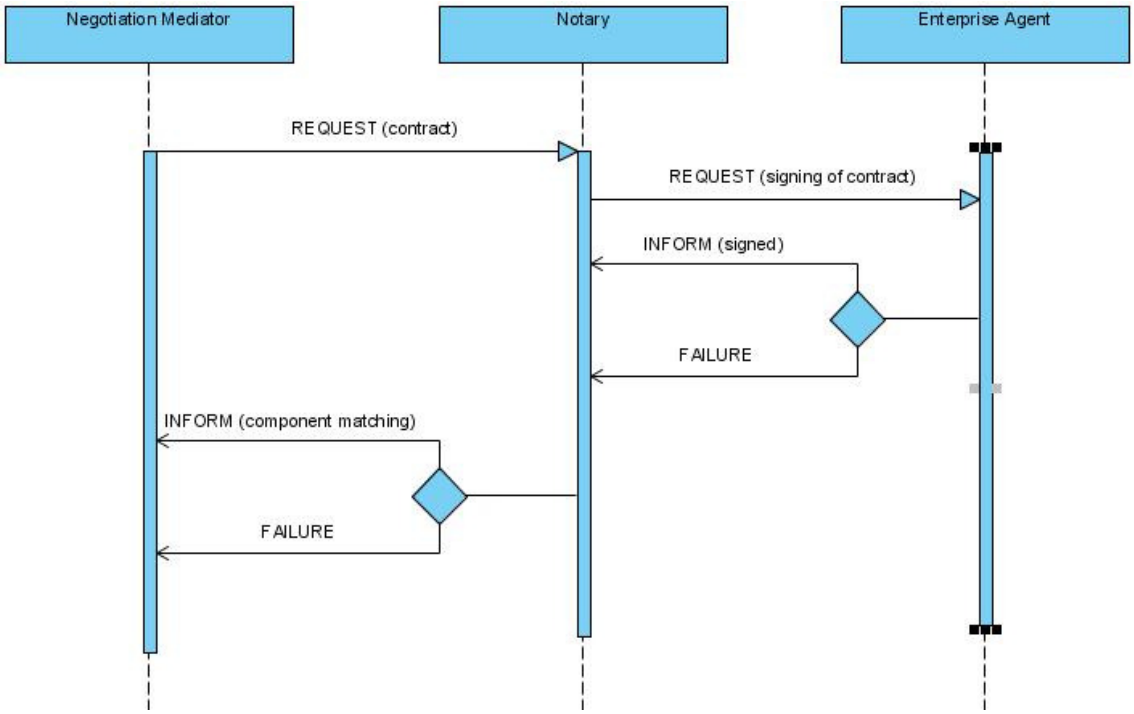

Fig.31 – Diagrama de comunicação entre um *Negotiation Mediator*, *Notary* e um ou vários *Enterprise Agents* envolvidos num contrato

4.13.5. Enterprise Agent → Reputation Agent

Também o agente fornecedor do serviço de reputação possui o seu próprio protocolo de comunicação, do qual os agentes empresa podem tirar partido fazendo pedidos sobre reputações dos variados agentes com quem pensam entrar em negociação. Após este pedido o agente devolve a resposta, ou informa que falhou por algum motivo (ver figura 32).

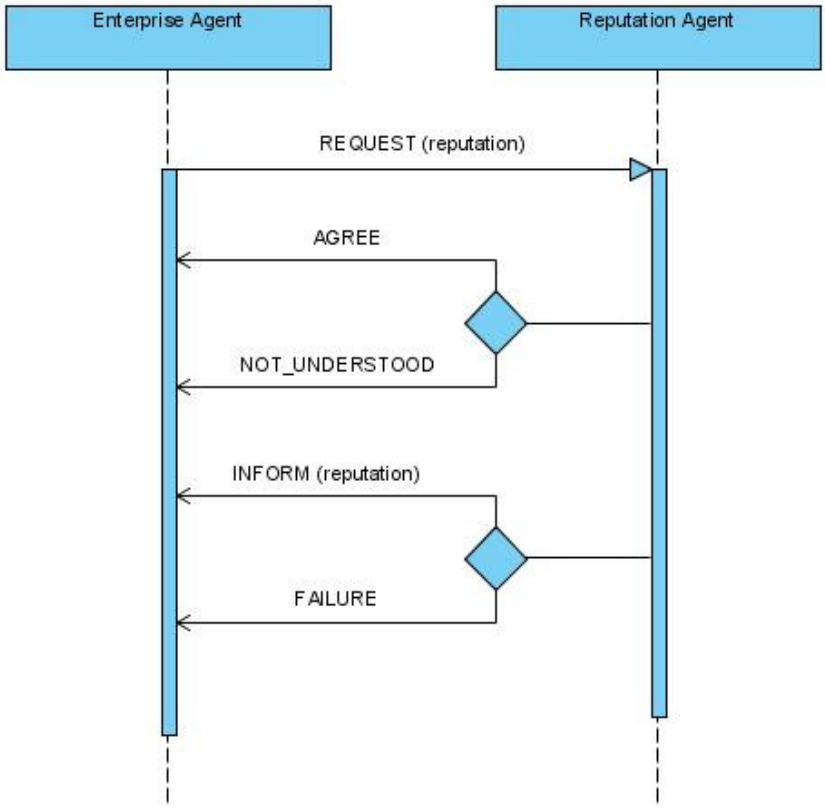

Fig.32 – Diagrama de comunicação entre um *Enterprise Agent* e um *Reputation Agent*

4.14. Ontologias no JADE

Cada agente comunica com os outros utilizando ontologias específicas implementadas através das facilidades para o efeito do JADE.

Estas ontologias correspondem ao tipo de informação que irá ser trocada, sendo criados objectos especiais para o efeito. Deste modo, cada agente pode apenas trocar informação padronizada e garante o reconhecimento do tipo de mensagem no destinatário, muito embora não possa garantir que o seu pedido irá ser atendido, obviamente. Cada ontologia define quais os objectos e o seu tipo passíveis de serem trocados numa comunicação desse tipo, bem como as suas quantidades ou obrigatoriedade.

4.15. Mecanismos de Simulação

Um dos objectivos deste projecto era o de desenvolver algum mecanismo que permitisse testar e simular a plataforma em diferentes condições de modo a permitir retirar algumas conclusões do processo.

Nesse sentido foi criado um tipo de Enterprise Agent automático no qual se pudessem configurar alguns parâmetros significativos, obtendo assim comportamentos aleatórios, mas que no entanto não fugissem muito de situações encontradas a qualquer momento no mundo real.

Este agente, da classe StochasticAutomaticEnterpriseAgent, permite então, como visto acima, configurar o tempo máximo e mínimo de resposta a obrigações contratuais, bem como a probabilidade de falhar essa resposta, violando o contrato em vigor.

Este tipo de agente permite observar com grande detalhe o comportamento do serviço de reputação em funcionamento, percebendo assim que tudo funciona da maneira mais correcta e pretendida.

Podemos ver agora um exemplo de teste efectuado à plataforma, que nas condições iniciais apresentava as percentagens de reputação de cada agente vistas na figura 33.

Agent Name	Fulfillments	Violations	Reputation Grade
Supply3Ont1	12	2	50.0%
Request1Ont1	40	7	76.7%
Supply5Ont2	2	10	-44.4%
Supply1Ont1	0	2	-16.6%
Supply4Ont2	30	3	72.9%

Figura 33 – Reputação dos agentes antes da experiência

É de ressaltar de imediato que os agentes focados foram o Supply1Ont1 e o Supply3Ont1. Ao primeiro foi-lhe dada uma probabilidade de falha de apenas 10%, enquanto que ao segundo agente foi dada uma probabilidade da falha de 85%, significativamente maior portanto.

Inicialmente o agente Supply1Ont1 tinha uma reputação de -16.6%, enquanto que o Supply3Ont1 já beneficiava de uma reputação de 50% (ver figura 33).

Com estes dados iniciais foram efectuadas em sequência 10 negociações sobre os mesmos componentes, todas bem sucedidas, sendo que ao fim dessas 10 negociações, ambos os agentes estavam agora com uma reputação de 16.6% (ver figura 34). Podemos então ver que o agente Supply3Ont1 efectuou 8 violações, o que fez descer a sua reputação em quase 35%, enquanto que o agente Supply1Ont1, que tinha apenas 10% de probabilidade de falha, conseguiu cumprir 6 obrigações, subindo assim a sua reputação de -16% para +16%, e colando-se ao outro.

Agent Name	Fulfillments	Violations	Reputation Grade
Supply3Ont1	12	10	16.6%
Request1Ont1	54	10	81.4%
Supply2Ont1	3	5	-16.6%
Supply5Ont2	2	10	-44.4%
Supply1Ont1	6	4	16.6%
Supply4Ont2	38	3	77.7%

Figura 34 – Reputação dos agentes após a experiência

Através desta experiência fica então simples de observar que o sistema automático de simulação de falha dos agentes funciona de forma correcta, pois tal como esperado, a reputação dos dois agentes foi afectada de forma inversa, em concordância com as suas respectivas probabilidades. Podemos ver um gráfico com o comportamento de alguns agentes durante esta experiência na figura 35.

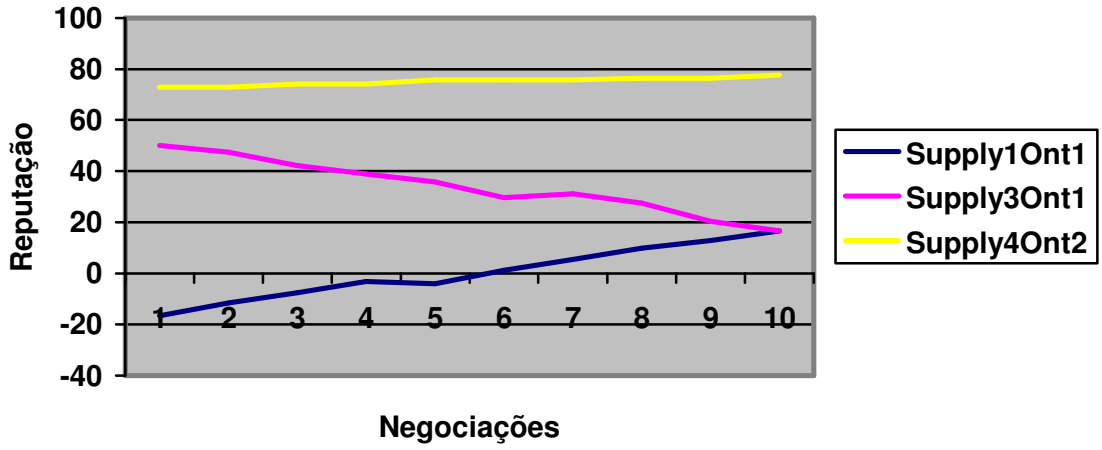

Figura 35 – Reputação dos agentes ao longo de várias negociações

5. CONCLUSÕES

Ao terminar o projecto podemos dizer que os objectivos foram concluídos de maneira satisfatória.

Foram aplicadas inúmeras melhorias sobre a versão anterior, estando agora todos os protocolos implementados de acordo com as especificações da FIPA, padronizando assim o sistema. Para além disso, todos os agentes comunicam agora através de ontologias do JADE, deixando de passar simples objectos entre si. Foi também retirado da equação o Negotiation Handler, um tipo de agente que era criado para lidar com cada negociação, através do mediador. Esta operação foi efectuada pois tornava-se redundante criar um agente novo para cada negociação quando a própria filosofia dos comportamentos de um agente proporcionava uma base muito apropriada para que as negociações fossem lidadas dentro do próprio mediador.

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

Deu-se também novo ênfase aos diversos papéis disponibilizados pela aplicação, entre os quais o de um agente representativo do banco e outro para monitorizar transacções de bens. Foram então criados estes agentes tipo com vista a assistência e monitorização da plataforma numa fase pós-negociação.

Todos os agentes tiveram uma interface com o utilizador desenhada, ou substancialmente melhorada, de modo a facilitar a interacção humana com a plataforma mas também permitindo uma maior transparência de tudo o que acontece em determinado momento.

Um enorme módulo inexistente anteriormente era o da segurança, que foi devidamente implementado, permitindo assim a encriptação e assinatura de mensagens e contratos, bem como um sistema de autenticação e permissões que afasta qualquer utilizador com intenções maliciosas.

Finalmente, foram também criados mecanismos de simulação, nomeadamente através de agentes automáticos, de modo a poder testar a aplicação em diversas situações e experimentar diversos tipos de comportamentos, avaliando de seguida os resultados. Isto permitiu-nos observar, entre outras coisas, que o serviço de reputação da plataforma funcionava da maneira pretendida.

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

4. JADE Security Guide, 28/02/05

5. JADE Programmer's Guide, 18/06/07

6. Rui Neves (2007), Relatório Final de Bolsa de Investigação, 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

```
<!ELEMENT ei-config                    (jade-parameters, institutional-agents)>
<!ELEMENT jade-parameters              (parameter*)>
<!ELEMENT parameter                    (#PCDATA)>
<!ELEMENT institutional-agents         (agent+)>
<!ELEMENT agent                        (class, argument*)>
<!ELEMENT class                        (#PCDATA)>
<!ELEMENT argument                     (#PCDATA)>
<!ATTLIST agent name                   CDATA #REQUIRED>
```

Figura 36 – DTD para o ficheiro de configuração da plataforma