Mestrado Integrado em Engenharia Informática e Computação

Universidade do Porto

Faculdade de Engenharia

FEUP

Mercado Electrónico com Negociação Multi-atributo

Relatório Final

Agentes e Inteligência Artificial Distribuída
4º ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do Grupo:

Jorge Silva - ei09016@fe.up.pt
Tiago Mota - ei09068@fe.up.pt

Data de Entrega: 9 de Dezembro de 2012

Objectivo

Este documento representa o relatório final destinado ao trabalho número 2: "Mercado Electrónico com Negociação Multi-atributo", da Unidade Curricular de Agentes e Inteligência Artificial Distribuída.

Foi proposta a implementação de um sistema de Agentes Inteligentes capazes de tomar decisões respectivas à transacção de produtos, ou seja, capazes de aceitar ou recusar propostas provenientes de outros agentes na rede, de forma a ser possível concluir uma negociação, que seja satisfatória para ambos, tendo sempre em consideração os seus atributos e preferências.

Este sistema contará com agentes Comprador e Vendedor, que irão tentar encontrar agentes que desejem obter/vender produtos que sejam do seu interesse.

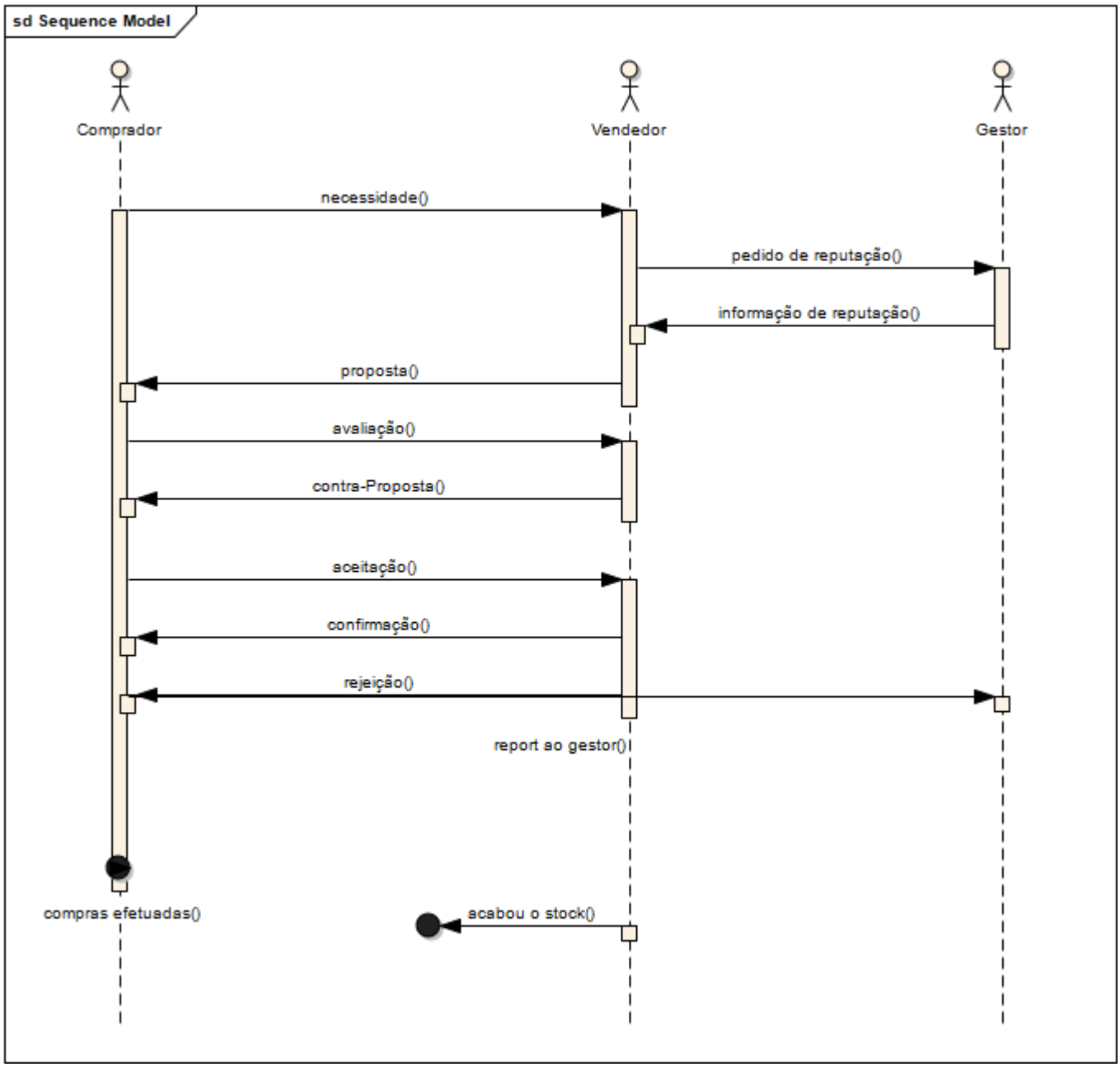
Em qualquer tipo de negócio, é importante que sejam seguidos processos de negociação para que a comunicação entre as duas partes interessadas, seja boa e correctamente transmitida, havendo tempo para que cada lado verifique a utilidade e necessidade de qualquer negócio em que esteja envolvido. Para que tal seja possível implementar, irá ser utilizada a plataforma JADE (Java Agent Development Framework), que será explicada no desenvolvimento deste documento.

Para além disto, foi também proposta, a implementação de uma funcionalidade extra no sistema. Um gestor de mercado, algo que faça a manutenção da rede de Agentes, avaliando a reputação de cada um destes, que ira ser dada pelo cumprimento ou não de negócios a que se comprometem.

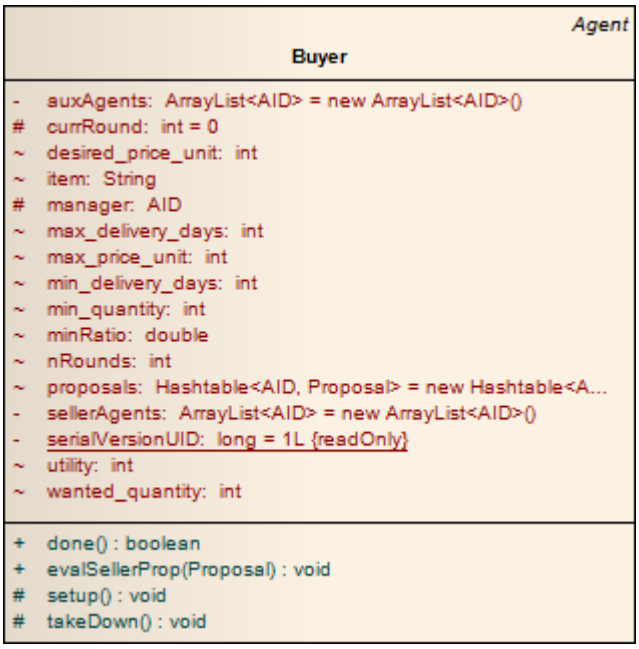
O objectivo para este projecto, passa então, pelo sucesso na implementação deste mercado e do seu processo de negócio. Dividindo isto em pequenos sub-objectivos, temos:

- Definição e construção dos Agentes Comprador e Vendedor, que contenham toda a informação relativa aos itens que desejam transaccionar.
- Comunicação eficaz entre os Agentes aplicando um processo de negócio eficaz, que terá troca de propostas e contra-propostas de negociação.
- Agentes com capacidade de tomar a decisão de finalizar um negócio, apoiando-se em toda a informação necessária relativa à utilidade e vantagens que determinado negocio, lhe trará.
- Criar um método ou função de avaliação de negócios de forma a um Agente saber se certo negócio lhe trará ou não vantagens e utilidade.
- Implementar um agente encarregue da gestão do mercado, que garanta informação sobre a reputação de todos os agentes presentes na rede. Isto irá permitir que os agentes possam decidir efectuar ou não alguma transacção, baseando-se no facto de o agente alvo tem ou não boa reputação no mercado.

Especificação



Comprador:



Comportamentos:

O agente Comprador conta com vários tipos de comportamentos, para as diferentes etapas de comunicação com a rede.

1 **GetManager - *SimpleBehaviour***

Com este comportamento, o agente Comprador irá receber uma mensagem (ACLMessage.*CFP*) com a identificação do agente Gestor da rede, de forma a poder aceder à informação sobre as reputações no mercado, assim como para enviar as actualizações dos suas transações bem sucedidas e falhadas.

2 **GetReputationFromManager - *CyclicBehaviour***

Após ter sido enviado um pedido de avaliação de determinado agente Vendedor, para o Gestor, por parte do agente Comprador, este recebe essa avaliação (ACLMessage.*QUERY_IF*) através deste comportamento, decidindo aqui se vai negociar com ele ou não, dependendo no valor dado ao atributo *ratio* por parte do Utilizador.

3 **“Publicitação de necessidade” - *TickerBehaviour***

Quando o agente comprador entra no sistema, inicia um TickerBehavior cuja função é enviar uma mensagem (ACLMessage.INFORM) para todos os agentes vendedores existentes, informando-os da sua necessidade de compra, este comportamento repete-se de tempo a tempo de

modo a entrar em comunicação com agentes vendedores que tenham entrado no mercado entretanto.

4 **EvaluateProposals - *CyclicBehaviour***

Este comportamento é responsável por receber propostas, gerar uma avaliação, que tem em conta as suas próprias preferências, para a proposta recebida e envia-la para o agente vendedor em causa (ACLMessage.*PROPOSE*). Quando é recebida uma proposta cuja utilidade avaliada é suficiente para satisfazer as especificações do Comprador este comportamento envia uma mensagem de aceitação para o vendedor em causa (ACLMessage.*ACCEPT_PROPOSAL*), a fim de terminar a negociação.

5 **TradeConfirm - *CyclicBehaviour***

De forma a um agente Comprador saber se a sua transacção foi efectuada com sucesso ou não, necessita de receber uma confirmação do Vendedor, sendo que este comportamento faz o tratamento deste tipo de mensagem, enviando de seguida ao agente vendedor em causa, uma "confirmação da confirmação".

Estratégias:

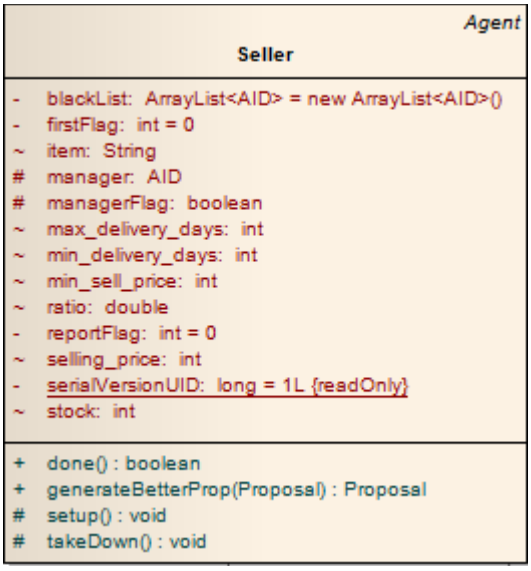
Inicialmente o agente comprador envia uma mensagem de necessidade para todos os agentes vendedores, informando-os do produto que quer adquirir, bem como a devida quantidade. Esta mensagem será enviada de tempo a tempo, de modo a notificar agentes vendedores que tenham entrado no sistema entretanto.

Os agentes vendedores interessados enviam uma proposta inicial, baseada nas suas preferências para o comprador, publicador da necessidade.

Após a recepção de uma proposta de um vendedor, o agente comprador devolve ao mesmo uma avaliação da dita proposta, contendo uma classificação de "bom", "suficiente" ou "mau" para cada atributo, dada com base na proximidade dos valores da proposta aos especificados pelo utilizador do agente comprador. Este processo repete-se para cada vendedor, sendo as respostas à necessidade inícios para um dado número de rondas de negociação compostas por avaliações e contra-propostas.

Quando uma proposta vinda do vendedor cuja utilidade avaliada é suficiente para satisfazer as especificações do comprador, é enviada uma mensagem de acordo ao vendedor, que se for respondida positivamente fecha o negócio. Caso chegue ao fim do número de rondas de negociação especificado pelo utilizador e não haja acordo mútuo entre os agentes, não é efectuada compra. Se isto se verificar, o *ratio* do Comprador na rede irá decrescer em 5%.

Vendedor:



Comportamentos:

O agente Vendedor, também necessita de diferentes tipos de comportamentos para as diferentes fases de comunicação existentes na rede.

1 GetManager - *SimpleBehaviour*

Com este comportamento, o agente Vendedor irá receber uma mensagem (ACLMessage.*CFP*) com a identificação do agente Gestor da rede, de forma a poder aceder à informação sobre as reputações no mercado, assim como para enviar as actualizações dos suas transações bem sucedidas e falhadas.

2 GetReputationFromManager - *CyclicBehaviour*

Após ter sido enviado um pedido de avaliação de determinado agente Comprador, para o Gestor, por parte do agente Vendedor, este recebe essa avaliação (ACLMessage.*QUERY_REF*) através deste comportamento, decidindo aqui se vai negociar com ele ou não, dependendo no valor dado ao atributo *ratio* por parte do Utilizador.

3 ReportToManager - *CyclicBehaviour*

Através deste comportamento, o agente Vendedor irá reportar ao agente Gestor as actualizações dos seus números de vendas bem sucedidas e falhadas.

4 FirstProposal - *CyclicBehaviour*

Assim que o agente Vendedor recebe a necessidade de um Comprador, caso tenha o item em causa para venda, envia a primeira proposta (ACLMessage.*PROPOSE*) baseada nas suas preferências, caso contrário envia uma mensagem a informar isso mesmo (ACLMessage.*REFUSE*).

5 GenerateCounterProps - *CyclicBehaviour*

Comportamento responsável por receber a avaliação de propostas de agentes compradores e utilizá-las como base para gerar contra-propostas e enviá-las para o agente comprador em questão.

Através da utilização de vários atributos, a geração de contra propostas é feita com base na avaliação dada pelo Comprador sobre cada um dos atributos enviados na proposta inicial. Usando isso, a função *generateBetterProp()* faz:

```
public Proposal generateBetterProp(Proposal p) {
    Proposal newProp = new Proposal();
    newProp = p;

    if(p.delivery_days_eval != "bom"){
        if(p.delivery_days > min_delivery_days){
            newProp.delivery_days = (int) Math.floor((p.delivery_days - min_delivery_days)*0.5);
        }
    }
    else{
        newProp.delivery_days = p.delivery_days;
    }

    if(stock < p.quantity){
        newProp.quantity = stock;
    }
    else{
        newProp.quantity = p.quantity;
    }

    if(p.proposed_value_eval != "bom"){
        float badpu = p.proposed_value/p.quantity;
        if(badpu > min_sell_price){
            newProp.proposed_value = (float) ((badpu-(badpu-min_sell_price)*0.5) * newProp.quantity);
        }
    }
    else{
        newProp.proposed_value = p.proposed_value;
    }
```

Com a proposta nova gerada, esta é enviada para o agente Comprador avaliador, de forma a que este reveja a sua avaliação e decida se a proposta já é boa o suficiente para ir de encontro com as suas preferências.

6 GetCancelFromBuyer - *CyclicBehaviour*

Se a negociação estiver a levar um número de rondas elevado, que ultrapasse o número definido pelo utilizador do agente Comprador, então será enviado um pedido de cancelamento para o Vendedor, onde através deste comportamento notifica o utilizador do sucedido (ACLMessage.*CANCEL*).

7 ConfirmDeal - *CyclicBehaviour*

Este comportamento irá permitir o envio de confirmação de venda efectuada ao comprador, efectuando depois a verificação de existência de stock para continuar na rede, pois caso não tenha, termina.

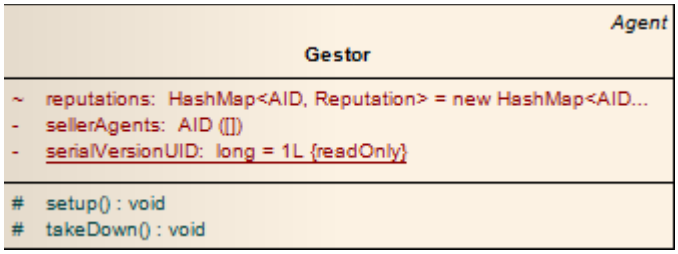
Estratégias:

Inicialmente o agente vendedor espera por mensagens de agentes compradores a publicitar necessidade de um dado item. Caso possua para venda, o item em questão, responde ao comprador, enviando uma proposta com quantidade pedida por ele e com as suas próprias preferências, caso contrário responde negativamente à mensagem.

Após a recepção da avaliação por parte do cliente da proposta enviada, o agente vendedor gera uma contra-proposta baseada nessa avaliação e nos seus limites especificados e envia-a para o comprador.

Quando recebe uma mensagem de intenção de compra é verificado o stock existente e enviada uma mensagem de aceitação se existir stock para efectuar negócio, caso contrário será enviada uma mensagem a informar que não se efectuará o negócio. Por cada negócio não efectuado com sucesso, o *ratio* do agente na rede irá diminuir de acordo com a seguinte fórmula: $n^{\underline{o}} sucessos \div n^{\underline{o}}\ total\ vendas$.

Gestor:



Comportamentos:

Agente responsável pela gestão da rede, ou seja, por manter o registo da reputação de cada agente na rede, permitindo aos agentes Vendedores e Compradores, aceitar ou recusar negociação com determinado agente, através da reputação deste na rede.

1 "Identifica-se na Rede" - *TickerBehaviour*

De tempo a tempo, envia para os agentes da rede a sua identificação, de modo a estes poderem pedir e enviar informações ao Gestor, durante a sua presença na rede (ACLMessage.*CFP*).

2 "Impressão da lista de Reputações" - *TickerBehaviour*

Por cada 3 segundos imprime na consola do Gestor a lista de reputações dos agentes presentes na rede.

3 EvaluateAgentsOnNetwork - *CyclicBehaviour*

Através destes comportamento o agente Gestor, irá receber mensagens de actualização dos números de vendas/compras (ACLMessage.*INFORM_REF*) dos agentes da rede, para que possa manter o registo sempre correcto.

4 RequestSellerRep - *CyclicBehaviour*

Ao longo do tempo, o agente Gestor irá ser abordado e questionado sobre a reputação de determinados agentes na rede, tendo de a enviar de seguida para o agente emissor do pedido. Este comportamento garante esta funcionalidade, separando pedidos de agentes Vendedores e Compradores.

Estratégias:

Ao iniciar a sua presença na rede, o agente Gestor fica à espera da entrada de agentes nesta. Depois, assim que a comunicação entre eles começa, irão necessitar de informação que o Gestor possuí, nomeadamente a reputação dos agentes. Recebe estes pedidos e responde aos agentes questionadores com a informação que eles necessitam.

Para além disto garante a actualização do *ratio* de cada agente, através de relatórios enviados pelos mesmos com informação nova, relativa às suas transações bem e mal sucedidas.

Desenvolvimento

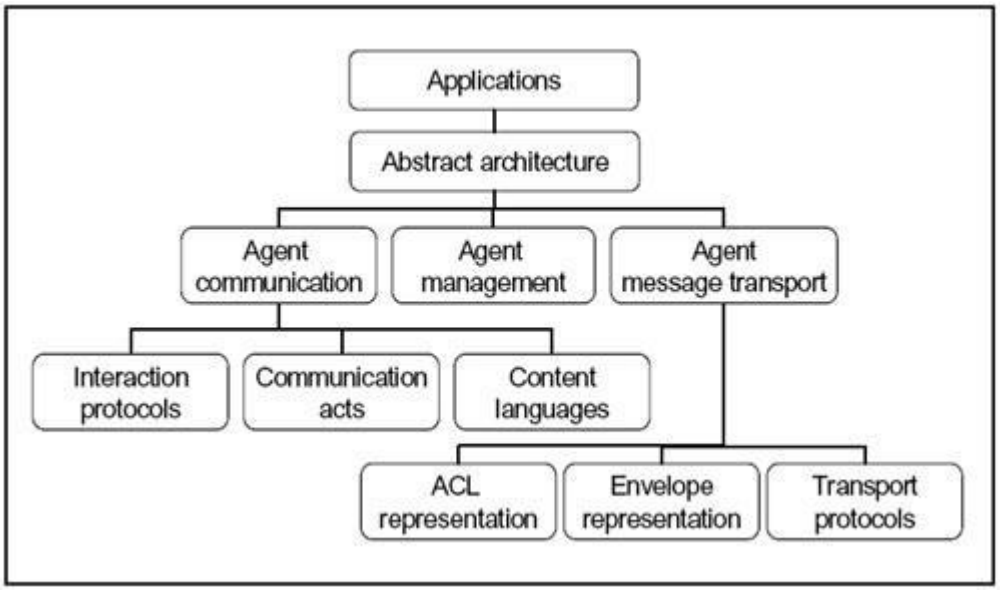
Plataforma de desenvolvimento

A plataforma que utilizamos para o desenvolvimento deste trabalho prático foi JADE (Java Agent Development Framework). JADE é um ambiente para desenvolvimento de aplicações baseado em agentes, tendo por base a linguagem Java.

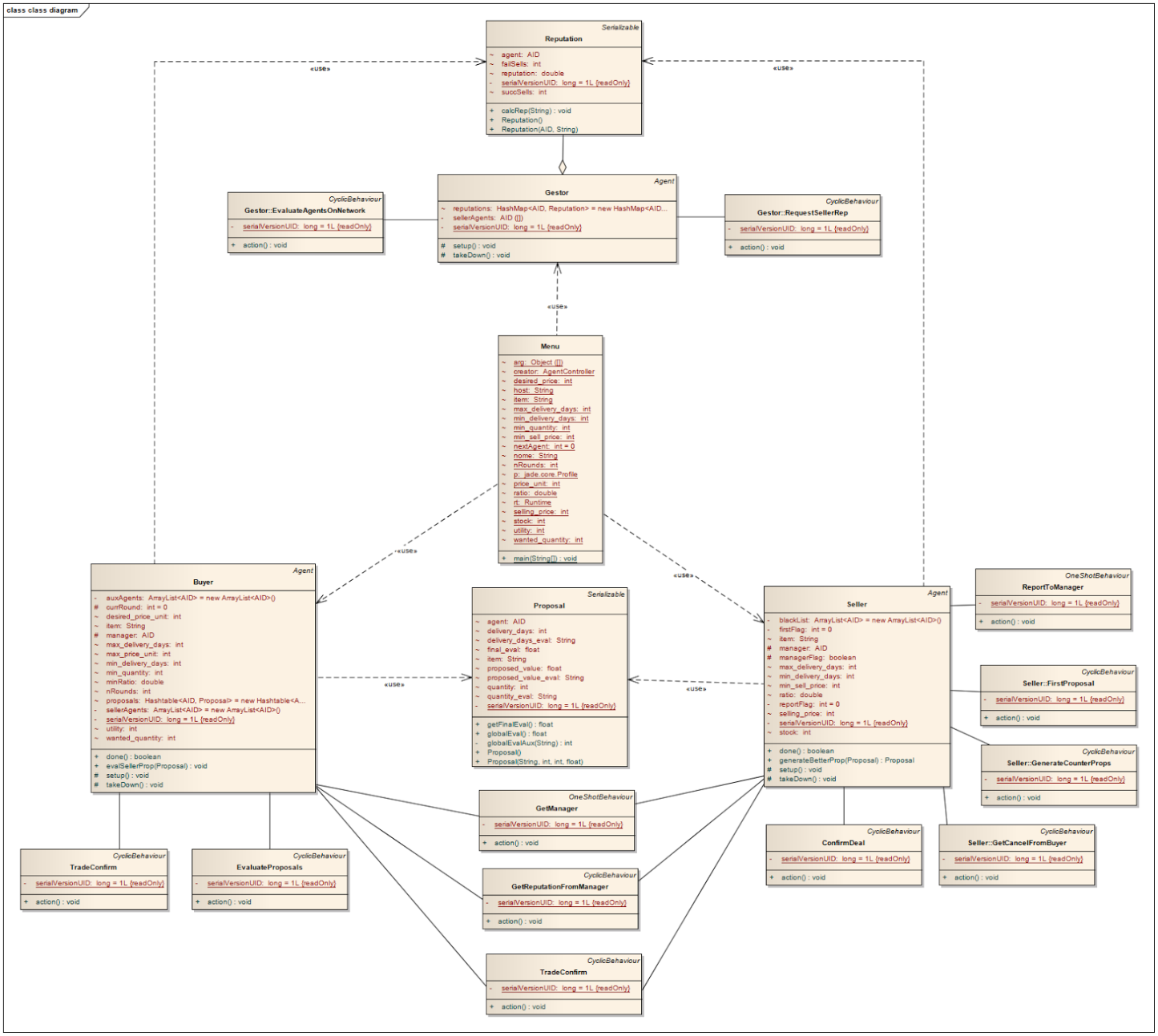
Este suporta a mobilidade intra-plataforma, permite a gestão de vários *containers* e possui diversas ferramentas que simplificam a administração e desenvolvimento de aplicações baseadas em sistemas multi-agente.

O principal objectivo do JADE é simplificar e facilitar o desenvolvimento de sistemas multi-agentes garantindo um padrão de interoperabilidade entre sistemas multi-agentes através de um abrangente conjunto de agentes de serviços de sistema, os quais tanto facilitam como possibilitam a comunicação entre agentes, de acordo com as especificações da FIPA (Foundation for Intelligent Physical Agents).

No que respeita ao ambiente de desenvolvimento, o trabalho prático foi desenvolvido no sistema operativo Windows 8 e teve como IDE o Eclipse Juno.



Estrutura da Aplicação



Detalhes relevantes da Implementação

Ao longo do desenvolvimento do projecto alguns detalhes foram surgindo, que o grupo achou importante referir:

- Antes de mais, a tecnologia do JADE não é nada má, e na nossa opinião consegue ser bastante útil para o mais diverso tipo de questões, como o projecto que por nós foi desenvolvido. No entanto, não é possível utilizar qualquer tipo *Debug* para testar a aplicação, já que ao usá-lo, todos os tempos de sincronização irão ser manualmente alterados por nós, assim que a aplicação para em determinada instrução com *breakpoint*. Isto fez com que não conseguissemos perceber o porque de determinados erros ao longo da implementação que levaram o dobro ou triplo do tempo a serem resolvidos.

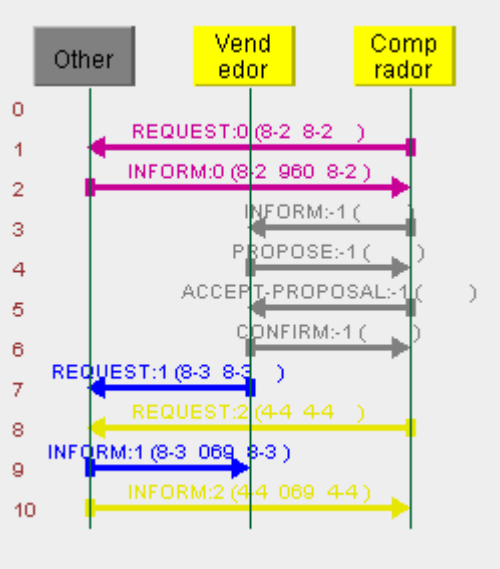
- Agora mais relacionado com a aplicação em si, em todo o processo de negociação as mensagens que são trocadas na rede, têm associadas *Performatives* diferentes que não seguem, de nenhum modo, alguma tipo de padrão, tendo sido usadas genericamente, apenas com o intuito de diferenciar as mensagens presentes na rede. A única excepção a isto é o uso das *Performatives* PROPOSE, REFUSE, CONFIRM e ACCEPT_PROPOSAL.

- De modo a facilitar o grupo de trabalho na fase de testar a aplicação, foi implementado um *parser* simples, que irá importar de um ficheiro de texto, agentes comprador e vendedor, excluindo assim a necessidade de inserir um a um pelo Menu textual disponível. Assim é também possível testar a aplicação rapidamente.

Experiências

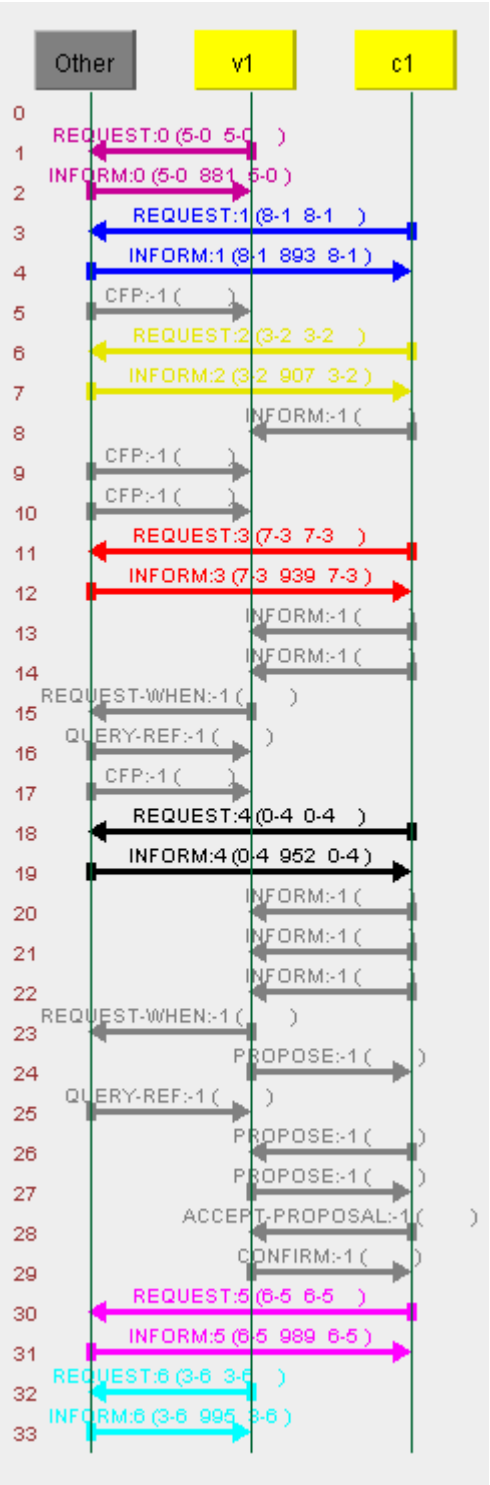
Experiência 1

Exemplo simples de inicialização de agentes e aceitação imediata de uma proposta.



Experiência 2

Exemplo de execução em que um cliente e um vendedor chegam a acordo após negociação utilizando propostas do vendedor e devida avaliação por parte do cliente.

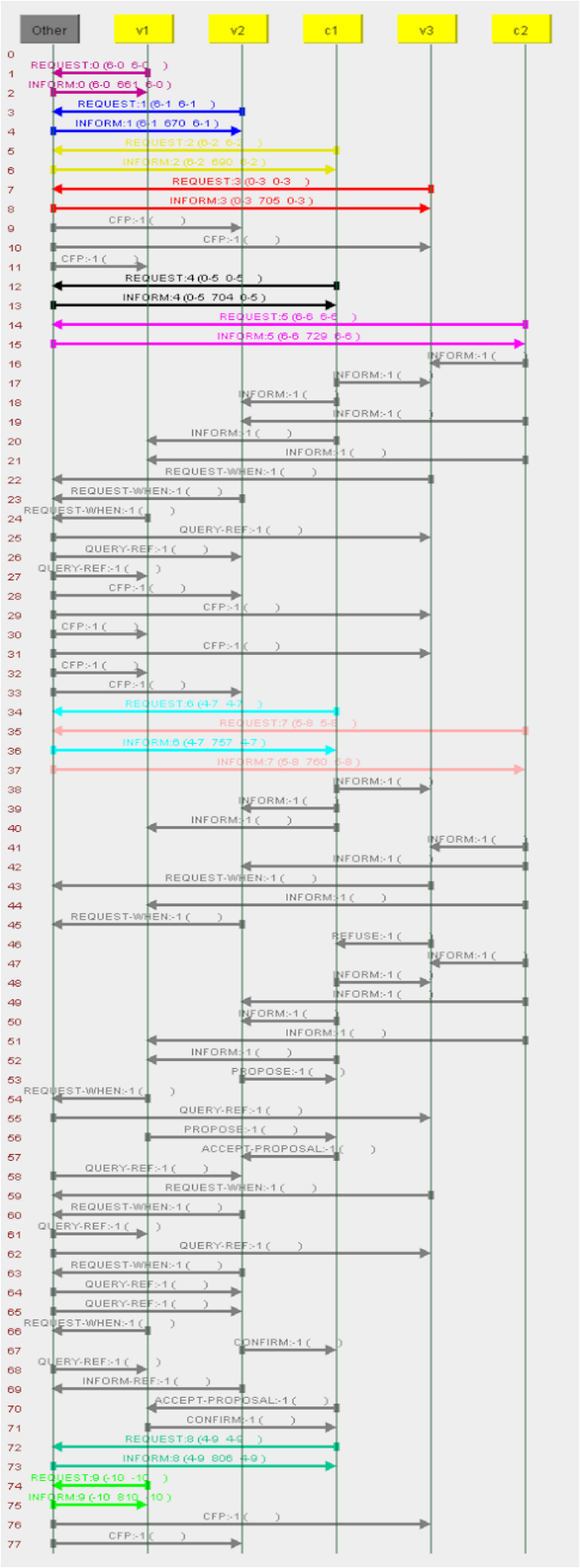


Output da consola:

```
Agente vendedor v1@172.30.31.219:1099/JADE criado!
Agente comprador c1@172.30.31.219:1099/JADE criado!
v1@172.30.31.219:1099/JADE identificou agente gestor: gestor@172.30.31.219:1099/JADE


-----------------> PROPOSTA RECEBIDA <------------------
item: bananas
quandidade: 100
dias de entrega: 10
preço: 600.0
utilidade: 66.666664
------------------------------------------------------------
Utilidade da proposta para mim (c1@172.30.31.219:1099/JADE): 66.666664


-----------------> PROPOSTA ENVIADA <------------------
item: bananas
quandidade: 100
dias de entrega: 2
preço: 450.0
------------------------------------------------------------


-----------------> PROPOSTA RECEBIDA <------------------
item: bananas
quandidade: 100
dias de entrega: 2
preço: 450.0
utilidade: 77.77778
------------------------------------------------------------
Utilidade da proposta para mim (c1@172.30.31.219:1099/JADE): 77.77778
Stock actual do Vendedor v1@172.30.31.219:1099/JADE: 0
Vendedor - v1@172.30.31.219:1099/JADE Produto vendido a: c1@172.30.31.219:1099/JADE

Agente comprador c1@172.30.31.219:1099/JADE a terminar.
Agente Vendedor v1@172.30.31.219:1099/JADE a terminar.
```

Experiência 3

Criada com o objectivo de observar o comportamento de vários agentes compradores e vendedores no sistema com necessidades diferentes ou concorrentes.



Output da consola:

```
Agente vendedor v1@172.30.31.219:1099/JADE criado!
Agente vendedor v2@172.30.31.219:1099/JADE criado!
Agente comprador c1@172.30.31.219:1099/JADE criado!
Agente vendedor v3@172.30.31.219:1099/JADE criado!
Agente comprador c2@172.30.31.219:1099/JADE criado!
v3@172.30.31.219:1099/JADE identificou agente gestor: gestor@172.30.31.219:1099/JADE
v2@172.30.31.219:1099/JADE identificou agente gestor: gestor@172.30.31.219:1099/JADE
v1@172.30.31.219:1099/JADE identificou agente gestor: gestor@172.30.31.219:1099/JADE


-----------------> PROPOSTA RECEBIDA <------------------
agente: c1@172.30.31.219:1099/JADE
item: item1
quandidade: 100
dias de entrega: 6
preço: 200.0
utilidade: 77.77778
------------------------------------------------------------
Stock actual do Vendedor v2@172.30.31.219:1099/JADE: 900


-----------------> PROPOSTA RECEBIDA <------------------
agente: c1@172.30.31.219:1099/JADE
item: item1
quandidade: 100
dias de entrega: 4
preço: 200.0
utilidade: 77.77778
------------------------------------------------------------
Stock actual do Vendedor v1@172.30.31.219:1099/JADE: 0
Vendedor - v2@172.30.31.219:1099/JADE Produto vendido a: c1@172.30.31.219:1099/JADE

Agente comprador c1@172.30.31.219:1099/JADE a terminar.
Agente Vendedor v1@172.30.31.219:1099/JADE a terminar.
```

Conclusões

Após a conclusão do projecto e analisando os resultados obtidos na secção anterior, é possível verificar o cumprimento dos objectivos do trabalho, tendo sido criada a simulação de um mercado de compra e venda multi-agente, baseado em negociações multi-atributo.

A utilização da plataforma JADE facilitou imenso este trabalho, uma vez que são apenas necessárias instruções simples, para que dois agentes em máquinas diferentes se conectem e comuniquem entre si.

Com o desenvolvimento deste projecto os nossos conhecimentos em JADE, processos de negócio e comunicação inteligente, foram melhorados consideralvelmente. Fazendo a restrospectiva, um trabalho como este pode ser extremamente melhorado como se refere a seguir, mas o que foi implementado demonstra uma aplicação já com algum poder e confiança nas suas escolhas de negócios, o que nos leva a acreditar que os objectivos foram cumpridos.

Melhoramentos

Apesar de os agentes já efectuarem negócios, aumentando ou diminuindo os atributos das propostas com base em avaliações e nas suas preferências, é sempre possível melhorar um algoritmo deste género. Posto isto, um refinamento da função de avaliação de propostas do agente comprador pode muito bem ser considerado para implementação posterior.

Nunca é demais polir a interface com o utilizador, pelo que a implementação de uma interface gráfica criando uma plataforma de gestão simplificada de compra e venda de vários artigos poderia ser um upgrade significativo na interacção com o utilizador.

Recursos

Software:

- Enterprise Architect

- Enhancing the Supply Chain Performance by Integrating Simulated and Physical Agents into Organizational Information Systems, http://jasss.soc.surrey.ac.uk/9/4/1.html **(2.4 e 2.5)**
- Home page do Framework JADE. http://jade.tilab.com
- The Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org/about/index.html
- Using JADE Behaviours, http://www.iro.umontreal.ca/~vaucher/Agents/Jade/primer6.html#6.1

Bibliografia

- Michael Wooldridge; An introduction to multiagent systems. ISBN: 978-0-470-51946-2 (2nd Edition, 2009)

Elementos do Grupo

Ambos os elementos do grupo participaram activamente do desenvolvimento deste projecto tendo havido divisão de tarefas inicialmente, no entanto a maior parte do código foi implementado em reuniões de trabalho entre os dois elementos.

Apêndice

Manual do Utilizador:

Inicialmente, a execução do programa mostra na consola 2 opções, 1 para criar um agente gestor que define o Main-Container, permitindo a ligação de outros agentes a este. 2 para criação de agentes compradores ou vendedores:

```
Bem vindo!
Pretende iniciar um agente:
        1 - Gestor (Host)
        2 - Comprador/Vendedor (Client)
        0 - Sair
```

Esta criação pode ser feita manualmente ou através do carregamento de ficheiro de teste, contendo informação sobre um conjunto de agentes compradores/vendedores:

```
Insira IP do gestor:
172.30.31.219
Deseja comprar ou vender itens?
        1 - Comprador
        2 - Vendedor
        3 - Correr ficheiro de testes
```

Introdução manual de dados:

Comprador:

```
1
Introduza os seguintes dados:

        Nome do agente:


        Nome do item:


        Numero de rondas de negociacao maximo:


        Tempo de entrega maximo (em dias):


        Tempo de entrega razoavel (em dias):


        Quantidade pretendida:


        Quantidade minima:


        Preco por unidade desejado:


        Preco maximo que esta disposto a pagar por unidade:


        Utilidade minima (0-100):


        Minimo de reputacao que
        esta disposto a aceitar (0-1, Exemplo 0,5):
```

Vendedor:

```
2
Introduza os seguintes dados:

        Nome do agente:


        Nome do item:


        Tempo de entrega maximo (em dias):


        Tempo de entrega minimo (em dias):


        Stock:


        Preco de venda inicial:


        Preco min que esta disposto a vender:


        Minimo de reputacao que
        esta disposto a aceitar (0-1, Exemplo 0,5):
```

Os ficheiros de teste são compostos por informação deste género:

```
v!v1!item1!4!4!100!2!2!0.5
v!v2!item1!6!2!1000!2!1!0.6
c!c1!item1!4!3!100!100!2!2!50!0.5!10
v!v3!bolachas!6!3!50!3!1!0.7
c!c2!bolachas!4!3!40!40!2!2!50!0.6!4
```

Cada linha do ficheiro especifica um agente, tendo as suas características separadas por “!”.
O primeiro campo identifica o tipo de agente, o segundo o seu nome e o terceiro o item que pretende vender, quanto aos campos seguintes, no caso de ser vendedor são respectivamente o tempo standard de entrega, o tempo mínimo de entrega, o stock disponível, o preço standard de venda, o preço mínimo de venda e o valor de reputação mínima que exige dos agentes com que faz negócio, no caso de ser comprador estes campos serão respectivamente o numero de dias máximo de entrega, o número de dias de entrado esperados, a quantidade esperada, a quantidade mínima, o preço máximo por unidade, o preço desejado por unidade, a utilidade mínima de uma proposta para ser aceite, o valor de reputação mínima exigida para fazer negócio e o número limite de rondas de negócio.
É possível ao utilizador acompanhar o sucesso ou insucesso das negociações através da informação disponível na consola.