

Utilizando Padrões de Comunicação entre Agentes num Framework de Suporte à mobilidade em CLI

Flávio de Matos Rezende¹, Marco Antonio Costa Simões²

^{1,2} UNIT – Universidade Tiradentes
Av. Murilo Dantas, 300 – CEP 49032-490 – Aracaju(SE)

² FIB – Faculdade Integrada da Bahia
Rua Xingu, 179 – Jardim Atalia/STIEP – CEP 41770-130 – Salvador(BA)
flavio@jogosbrasil.pro.br, marcosimoes@fib.br

Abstract. *This paper presents .NET platform as an interesting alternative to support development of multiagents' systems. We finish presenting an application called MarketPlaceDotNet where two agents negotiate to buy/sell products. The agents communications was implemented based on KQML standard over Mobile Agent Framework .NET. This framework supports mobile intelligent agents over .NET using C# programming language.*

Keywords: *Multiagents' Systems; KQML; DotNET*

Resumo. *Este artigo apresenta a plataforma .NET, como uma promissora plataforma para o desenvolvimento de sistemas multiagentes, além de mostrar um aplicativo chamado MarketPlaceDotNET, na qual dois agentes comunicam-se em transações de compra e venda de produtos utilizando a tecnologia de comunicação entre agentes baseada no padrão de comunicação KQML (Knowledge Query and Manipulation Language) no Mobile Agent Framework .NET. Este é um framework de suporte a agentes móveis inteligentes, desenvolvido num ambiente CLI (Common Language Infrastructure) utilizado a linguagem C# na plataforma .NET*

PALAVRAS-CHAVES: *Sistemas Multi-agentes; KQML; DotNET*

1. Introdução

Nos últimos anos, temos sido testemunhas do interesse significativo e crescente no estudo e no desenvolvimento de aplicativos distribuídos. Este crescimento é fruto da demanda pelo acesso veloz às informações e a proliferação da Internet, o que tem causado maior preocupação com desafios como escalabilidade, desempenho, interoperabilidade, redução da complexidade, consultas dinâmicas, confiabilidade, e segurança. Estes problemas vêm abrindo maior espaço para o paradigma da arquitetura distribuída nos aplicativos *Web* frente à arquitetura cliente-servidor.

A inteligência Artificial distribuída (IAD) tem contribuído para uma importância cada vez mais abrangente da tecnologia de agentes na computação. Com os sistemas multiagentes é possível criar aplicações que executam tarefas de forma inteligente e

autônoma, com interações entre vários agentes contribuindo e comunicando-se para atingir um objetivo.

Essa idéia tem se tornado não só um tópico de pesquisa na área acadêmica, como também em aplicações comerciais e industriais. Esta preocupação tem fomentado um maior amadurecimento do conceito de sistemas multiagentes (SMA), além de ter como consequência um maior desenvolvimento de plataformas, metodologias e ferramentas para o desenvolvimento deste tipo de sistemas.

A comunicação entre agentes é o foco principal deste trabalho, pois muitos esforços na área lidam com o problema de estabelecer um bom meio de comunicação entre os agentes, utilizando protocolos de comunicação baseados em atos de fala, como FIPA Agent Communication Languages (FIPA ACL) – padrão proposto pela Foundation for Intelligent Physical Agents (FIPA, 1996) – e Knowledge Query and Manipulation Language (KQML) proposto por (FININ et al, 1993). Esta padronização da comunicação entre agentes permite que o conhecimento seja transferido de um agente para outro.

Este trabalho mostra bases teóricas introdutórias sobre agentes, sistemas multiagentes, comunicação entre agente e a plataforma .NET, como uma promissora plataforma para o desenvolvimento de sistemas multiagentes, na qual foi implementado o MarketPlaceDotNET, que é uma aplicação básica de comunicação entre agentes em KQML.

Nas seções seguintes discutiremos os conceitos de agente, sistemas multiagentes, agentes móveis. Em seguida apresentaremos uma descrição dos conceitos essenciais dos principais mecanismos e tecnologias de comunicação entre agentes. Logo após trataremos da plataforma .NET e do *Mobile Agent Framework .NET* (MAFDotNET) do trabalho (SIMÕES e SANTOS 2003) descrevendo uma breve introdução sobre a arquitetura e funcionamento, do *framework* e da plataforma. E por fim apresentaremos a principal contribuição deste trabalho que é a extensão do MAFDotNET para suportar um mecanismo básico de comunicação entre agentes baseado em KQML. Com este mecanismo será testada uma aplicação de negociação entre dois agentes, numa transação de compra e venda de produtos.

2. Agentes Inteligentes

Um conceito padrão universalmente aceito para definir o que seja um agente é algo que ainda não existe, tendo em vista as várias aplicabilidades e abordagens diferentes em inúmeros domínios de aplicação. O que há essencialmente é um consenso que autonomia é a principal característica de um agente.

De acordo com (Russel & Norvig 1995) um agente é algo capaz de perceber seu ambiente através de sensores e agir sobre ele através de efetadores. Alguns pesquisadores como Michael Wooldridge (WOOLDRIDGE 2002) consideram um agente, um sistema de computador que está situado em algum ambiente e que é capaz de executar ações autônomas de forma flexível neste ambiente, a fim de satisfazer seus objetivos de projeto.

No contexto específico da Inteligência Artificial (IA) o ideal é um agente capaz de funcionar continuamente, “aprender” maximizando seu desempenho, tomar decisões a partir de situações diferentes, deduzindo um curso lógico de ações para atingir seus

objetivos. Concluindo, um agente deve possuir raciocínio. É o raciocínio que garante a autonomia do agente, pois é através dele que o agente toma suas próprias decisões baseando-se num conhecimento prévio e nas percepções do ambiente, elevando assim seu grau de autonomia.

Os processos de raciocínio em agentes inteligentes tem sido implementados através de mecanismos de inferência dedutiva ou indutiva (Russel & Norvig, 1995). A utilização desses mecanismos permite que o agente escolha as ações mais apropriadas a partir de suas percepções, de seu objetivo e de seu conhecimento

3.0 Sistemas Multiagentes

Seguramente a parte que nos faz, ainda mais únicos entre as espécies é a nossa habilidade social (WOOLDRIDGE 2002). Habilidade proveniente de como compreendemos o mundo e de como interagimos com ele e com nós mesmos. Tal compreensão inspirou a transição de sistemas artificiais tradicionais (monolíticos) em sistemas multiagentes (SMA).

O enfoque principal dos SMA é prover mecanismos para a criação de sistemas computacionais a partir de entidades de softwares autônomas, denominadas agentes, que interagem através de um ambiente compartilhado por todos os agentes de uma sociedade, e sobre o qual atuam, alterando seu estado (BORDINI et al 2001).

Para (WOOLDRIDGE 2002), em sistemas multiagentes podem ser formuladas as seguintes questões:

- Como a cooperação pode emergir numa sociedade de agentes *self-interested* (egoístas, no nosso caso, competitivos)?
- Que tipo de linguagem comum os agentes podem usar para comunicar as suas convicções e aspirações, tanto para as pessoas como para outros agentes?
- Como os agentes competitivos podem reconhecer quando as suas convicções, metas, ou ações conflitam. E como eles podem chegar num comum acordo sobre o assunto que interessa-os , sem recorrer a conflitos?
- Como agentes autônomos podem coordenar suas atividades para alcançar metas cooperativamente?

Estes desafios tratam de comunicação, ou seja, qual protocolo e linguagem os agentes usaram para comunicar-se com o seu ambiente e com os outros agentes?, da interação, como ocorrerá essa interação ?, como combinaram seus esforços?, da coordenação, ou seja, como efetivar a coordenação entre os agentes para atingirem seus objetivos.

Os sistemas multiagentes caracterizam-se por (HUHNS & LARRY, 2000):

- Prover uma infra-estrutura especificando protocolos de comunicação e interação;
- São geralmente abertos e não possuem apenas um projetista;
- Contêm agentes que são autônomos e distribuídos, e podem ser competitivos ou cooperativos.

Podemos concluir que SMA é uma área dentro da IAD que trata de aspectos relativos à computação distribuída em sistemas inteligentes disponibilizando mecanismos de interação, coordenação e comunicação. Tendo como base teórica, metáforas em relação ao comportamento humano individual e em sociedade.

3.1 Agentes Móveis

A idéia de agentes móveis, a princípio, é bem simples. Eles visam substituir a (Remote Procedure Call) RPC da arquitetura (Cliente Servidor) CS como um modo para processos comunicarem-se na rede. Na arquitetura CS a idéia é que um processo possa invocar um procedimento (método) em outro processo que fica situado remotamente. Já com agentes móveis essa interação entre processos é feita localmente, causando um uso mais eficiente dos recursos da rede e diminuindo a sobrecarga de trabalho no servidor.

Os Agentes móveis podem ser considerados como processos computacionais que estão livres das restrições impostas por modelos de comunicação como CS ou qualquer outro baseado em mobilidade de código apresentado na seção anterior. Pode-se dizer que modelos CS assumem que o importante é migrar os dados para o local onde se encontra o programa que irá processá-los, enquanto que o paradigma dos agentes móveis assume que o ideal é migrar o programa para o local onde se localizam os dados como apresenta a figura 3.1 (MACEDO et al, 2001).

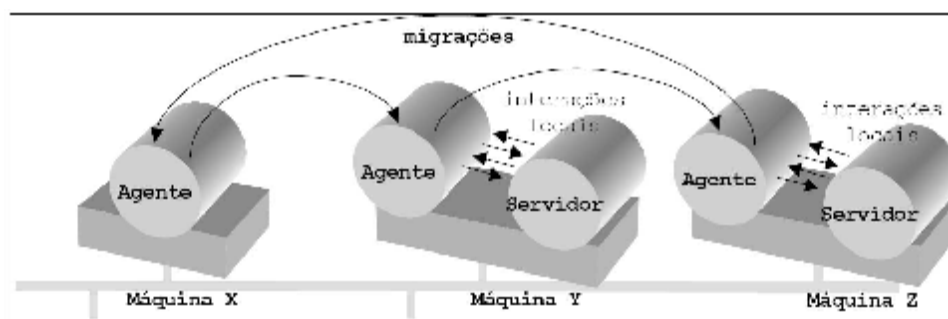


Figura 3.1 Modelo de comunicação baseado em agentes móveis.

4. Comunicação entre agentes

Como em qualquer sociedade a comunicação entre os membros é de fundamental importância. Isso não seria diferente numa sociedade de agentes na qual estes se comunicam na busca de atingir seus objetivos em um ambiente distribuído.

A comunicação é o caminho natural para haver interação, cooperação e competição (negociação) entre agentes, em sistemas de IAD. Pois como mencionado anteriormente um agente deve ter a habilidade de perceber (receber mensagens) e agir (enviar mensagens) de maneira efetiva e eficiente em seu ambiente. Necessitando assim de uma linguagem e protocolo de comunicação que sejam compreendidos por todos neste ambiente.

4.1 Atos de Fala

A teoria *Speech Acts* (atos de fala) foi abordada por filósofos como John Austin (AUSTIN 1962) e por John Searle (SEARLE 1969) que estende o trabalho de Austin. O estudo desses filósofos a respeito dos atos de fala, tem servido de inspiração em estudos relativos a criação de linguagens e protocolos de comunicação entre agentes.

De acordo com (AUSTIN 1962) os atos de fala podem ser distribuídos em três aspectos:

- Ato locucionário: é uma declaração proveniente de um meio físico (Por exemplo o ato fonético, ou a escrita)
- Ato ilocucionário: é a intenção ou compreensão associada à declaração do locutor
- Ato perlocucionário: é a ação resultante ou efeito da locução.

Estes atos podem ser caracterizados como enunciados que produzem efeitos nos sentimentos, pensamentos ou ações dos ouvintes, e ainda que podem ser realizados com a intenção de produzir tais reações. Atos desse tipo podem ser, por exemplo: respostas, perguntas, requisições, declarações, ou seja, atos que não podem ser diferenciados como o simples verdadeiro e falso, ou que deixam dúvidas quando ao seu propósito. A esse atos ou sentenças Austin chamou de performativas.

4.2 KQML

(FININ et al, 1993) propuseram a linguagem de comunicação entre agentes: *Knowledge Query and Manipulation Language* (KQML). Esta foi patrocinada pela *U.S. Defense Advanced Research Project Agency* (DARPA), e desenvolvida no projeto *Knowledge Sharing Effort* (KSE) na *University of Maryland Baltimore County* na cidade Baltimore USA.

A KQML é baseada na teoria dos atos de fala. Esta linguagem adiciona contexto (força ilocucionária, ou seja, intencionalidade) a cada mensagem disponibilizada em seu conjunto pré-definido. Cada uma delas é chamada de performativa, as quais definem as operações possíveis de serem executadas pelos agentes.

A linguagem KQML é também um protocolo para manipular troca de informações e conhecimento entre os agentes. Ela é independente de linguagem interna (Pascal, Prolog, Lisp, C#, Java, etc), independente de protocolo de comunicação (TCP/IP, SMTP, HTTP, etc), e de ontologia assumida pelo conteúdo.

Um dos critérios utilizados na criação da KQML segundo (Finin et al, 1994) era produzir uma linguagem que suportasse a grande variedade de arquiteturas de agentes, implementados em diferentes linguagens de programação e paradigmas. Para isso foi utilizado um conjunto pequeno de performativas, as quais os agentes utilizariam para descrever os meta dados que especificaria as informações requeridas e capacidades.

Todas as performativas da KQML giram em torno das bases de conhecimentos, ou seja, as trocas de mensagens estão normalmente associadas com as crenças ou objetivos contidos na base do agente, de forma que sua implementação não deve necessariamente ser estruturada como uma base de conhecimento. Ela pode usar outra estrutura de dados, preferencialmente persistente, para representar as crenças e os objetivos que os agentes perseguem. Dessa maneira, pode-se dizer que cada agente gerencia uma base de conhecimento virtual (VKB) (FARACO & GAUTHIER, 1998) na qual para (WOOLDRIDGE, 2002) é de fundamental importância para a compreensão das performativas.

Conceitualmente uma mensagem KQML é dividida em três camadas (Figura 4.1):

- **Conteúdo:** É nessa camada que é transportado o conteúdo da mensagem. Esse conteúdo pode ser representado por uma expressão em alguma linguagem que possa encapsular a informação ou conhecimento a ser transmitido. Pode conter qualquer tipo de conteúdo, e este representado em qualquer linguagem, pois as implementações da KQML ignoram o significado desse conteúdo;
- **Comunicação:** Essa camada é representada por parâmetros de identificadores únicos, com os quais sejam possíveis informa qual o emissor e receptor da mensagem;
- **Mensagem :** Essa é o ponto chave da KQML .Essa camada diz respeito aos tipos de mensagens que são enviadas entre os agentes durante a comunicação, isto é, qual performativa deve ser usada no momento de transmitir o conteúdo. Determinando assim os tipos de interação que um agente pode ter com outro e definido que operação deve ser executada com o conteúdo.

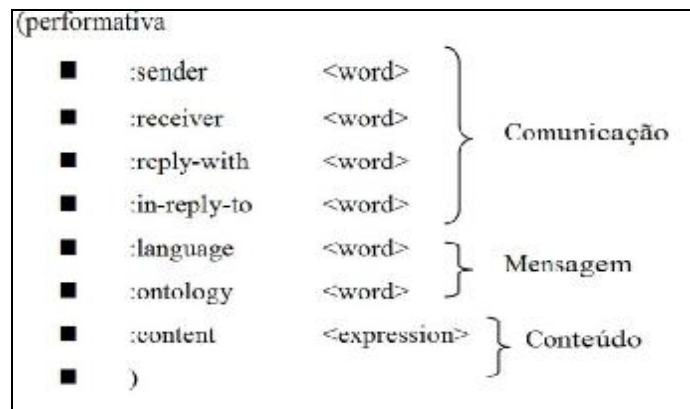


Figura 4.1 Camadas de uma mensagem padrão KQML.

(Finin et al, 1994) menciona basicamente dois tipos de comunicação entre agentes: a comunicação direta (Figura 4.2) e mediada através de facilitadores (Figura 4.3). Esta arquitetura baseada em facilitadores permite que sociedades de agentes funcionem de forma aberta , ou seja, que agentes entrem e saiam da sociedade. Se um agente quiser entra em uma sociedade basta avisar o facilitador a respeito de suas habilidades e qual o seu endereço físico na rede. Através de consultas ao facilitador, todo agente pode localizar outros agentes com os quais seja interessante interagir (BORDINI et al, 2001).

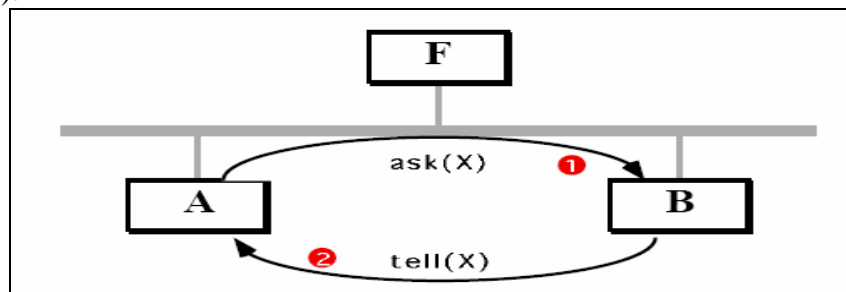


Figura 4.2 Protocolo usado quando A conhece B e sabe que B pode responder sobre X (Finin et al, 1994).

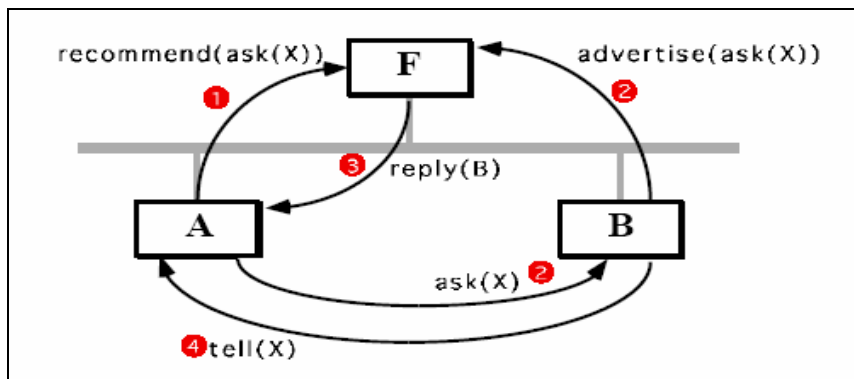


Figura 4.3 Protocolo utilizando performativas de facilitação (Finin et al, 1994).

5. Mobile Agent Framework .NET

5.1 .NET Framework

O *DotNET Framework* (MICROSOFT, 2002a) é um ambiente novo para a criação, instalação e execução de aplicações.. Ele oferece suporte a várias linguagens de programação como C++ , *COBOL*, *Perl*, *Python*, *Smalltalk* dentre outras. Tem como principal mecanismo de execução o *Common Language Runtime* (CLR) que manipula alocação de memória , detecção de erros e interação com os serviços do sistema operacional.

O *DoTNET* tem a *Common Language Infrastructure* (CLI) (ECMA, 2001a) (ISO 2003a) como a especificação dos seus principais serviços. Ela implementa a tecnologia que permite que um aplicativo seja desenvolvido em diversas linguagens de programação e executados num mesmo ambiente de execução. Além de ser suportada em diversos sistemas operacionais (sistemas *Win32*, *FreeBSD*, *MAC OS X* e em fase de migração, o *Linux*), o que é de fundamental importância em aplicações distribuídas, como os sistemas multiagentes.

A principal linguagem da plataforma *.NET* é a C#, esta é uma linguagem simples, derivada da C e C++, bastante similar a Java e totalmente orientada a objetos suportando conceitos como herança, encapsulamento, polimorfismo, indexadores, e outros. Foi padronizada pela *International Organization for Standardization* (ISO) em 2003 e pela *European Carton Makers Association* (ECMA) em 2001.

5.2 Mobile Agent Framework.NET (MafDotNET)

O *Mobile Agent Framework .NET* (SIMÕES & SANTOS 2003) é um *framework* de suporte a agentes móveis inteligentes, baseado numa solução de (BIGUS 2001). Através deste é possível implementar agentes com características de autonomia, reatividade, mobilidade, comunicação e raciocínio. O MafDotNET foi implementado no ambiente de execução *Microsoft .Net Framework* que implementa o padrão CLI.

O MafDotNET possui um sistema de agentes (SA) que dá suporte à mobilidade dos agentes. Também possui um mecanismo de comunicação baseado em eventos. Qualquer agente que deseje receber mensagens de outro deverá assinar o evento *OnAgentEvent* para ser notificado das mensagens daquele agente. O SA sempre assinará

este evento para todos os agentes locais, de forma que poderá repassar as mensagens para outros SAs remotos, funcionando como uma espécie de roteador de mensagens. Desta forma o SA junto com o MafDotNET implementa um mecanismo de eventos distribuídos(SIMÕES & SANTOS 2003).

O SA é considerado como uma plataforma de agentes, ou seja, cada agente executará como uma *Thread* do processo SA. Assim, cada SA tem conhecimento de todos os agentes que estão residentes sob seu controle, de forma a efetuar o roteamento de mensagens para os mesmos e eliminar os agentes mortos, ou seja, aqueles que encerraram de forma autônoma o seu processamento(SIMÕES & SANTOS, 2003).

Foi sobre essa estrutura de eventos distribuídos que foi implementada a comunicação entre agentes em KQML no estudo de caso descrito na seção 6.

6. MarketPlaceDotNET

6.1 Visão geral da aplicação

O MarketPlaceDotNET é uma adaptação do *CIAgent MarketPlace* (BIGUS, 2001) em Java para DotNET. Esta é uma aplicação na qual dois agentes inteligentes, Agente Vendedor (*Seller*) e Agente Comprador (*Buyer*) comunicam-se através da linguagem KQML e competem através de transações de compra e venda de produtos um com o outro.

Esta aplicação possui um agente Facilitador, o qual media a comunicação entre o agente comprador e vendedor, tendo basicamente a mesma arquitetura de um sistema federado (Figura 6.1), se fosse incorporado um agente Facilitador em diversos SAs.

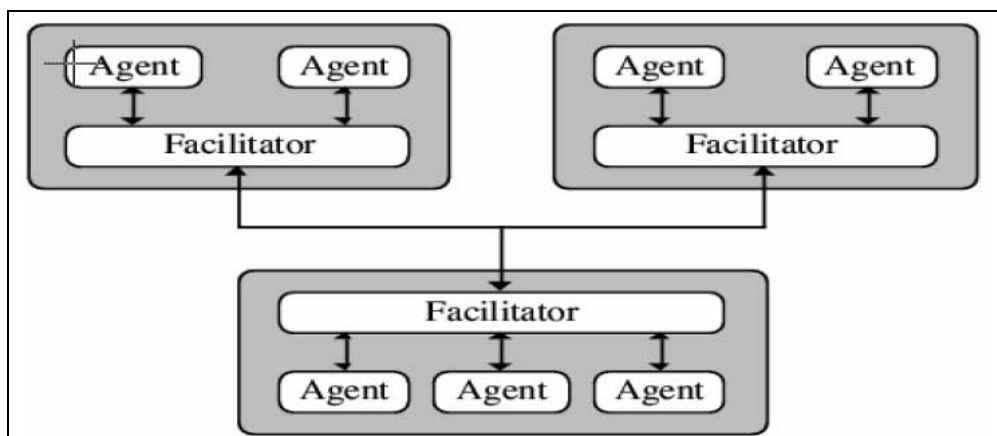


Figura 6.1 Comunicação por um sistema de federação de agentes.

Toda a comunicação do MarketPlaceDotNET é feita através de mensagens KQML, implementada como uma classe extensão do MafDotNET.

6.2 Estrutura da Aplicação

A MarketPlaceDotNET é composta por 6 classes:

- AgenteComprador.cs: Agente responsável pela compra de produtos contidas em sua lista de compras, possui a habilidade básica de negociar uma oferta de venda.

- AgenteVendedor.cs: Agente responsável pela venda de produtos contidos em seu estoque pessoal, possui a habilidade básica de negociar uma oferta de compra
- FacilitatorAgent.cs: Agente responsável por mediar e atender solicitações do agente comprador e vendedor.
- BaseDeNegocio.cs: Classe responsável em armazenar a base de negociação como última oferta, próxima oferta e preço atual.
- Oferta.cs :Classe responsável por armazenar dados de uma oferta tipo: quem enviou a oferta? Qual produto esta sendo negociado? Qual o seu preço e sua identificação ?
- MarketPlace.cs: é a aplicação. Esta intercepta as mensagens trocadas entre Facilitador, Agente Vendedor e Agente Comprador em duas áreas de texto distintas.

A tela de execução do MarketPlaceDotNET mostrando as ofertas dos agentes Comprador e Vendedor e o roteamento das mensagens pelo agente Facilitador pode ser vista na Figura 6.2.

6.3 Descrição do Funcionamento

O Agente Facilitador tem a capacidade de registrar qualquer agente que entre na sociedade através de método estático *register* (Figura 6.3).

Quando um agente, a exemplo o AgenteVendedor, quer entrar na sociedade, ou seja, no SA, o mesmo tem que registrar-se, no agente facilitador disparando um evento *Handler* e chamando o método *register* do Agente Facilitador conforme ilustrado na Figura 6.4).

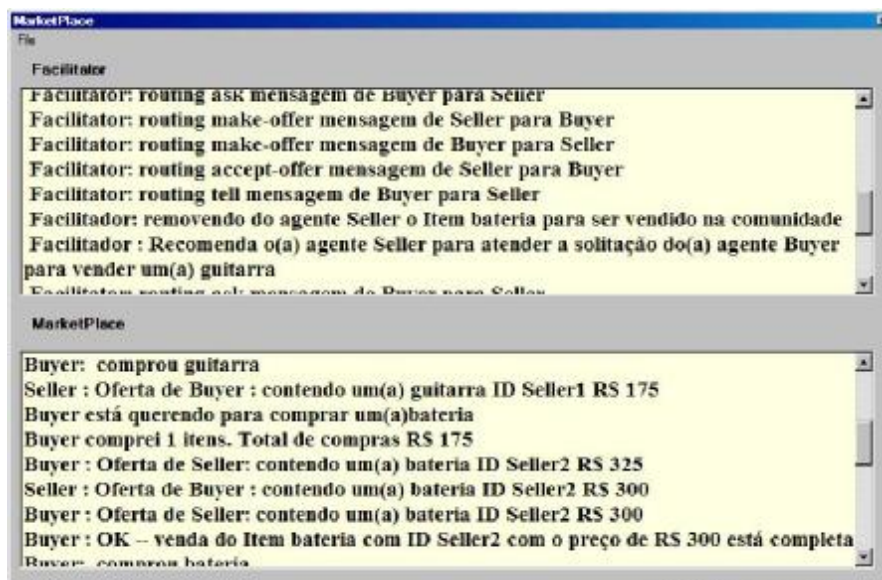


Figura 6.2 Aplicação MarketPlaceDotNET.

```

public static void register(MAFAgentEvent e)
{
    lock(typeof(FacilitatorAgent))
    {
        if (instance == null)
        {
            instance = new FacilitatorAgent();
        }
        instance.allAgents.Add(((MAFAgent) e.Source).Name, e.Source);
        ((MAFAgent) e.Source).OnAgentEvent += new MAFAgentEventHandler
            (instance.processMAFAgentEvent);
        instance.OnAgentEvent += ((MAFAgentEventHandler) e.Args.Argumento);
    }
}

```

Figura 6.3 Método *register* do FacilitatorAgent.

```

MAFAgentEventHandler f = new MAFAgentEventHandler(processMAFAgentEvent);

MAFAgentEvent e = new MAFAgentEvent(this, f);

FacilitatorAgent.register(e);

```

Figura 6.4 Forma de registrar-se no FacilitatorAgent.

6.4 Simulação no MarketPlaceDotNET

Como mencionado anteriormente MarketPlaceDotNET é uma aplicação na qual dois agentes comunicam-se em busca de seus objetivos. Estes dois agentes têm objetivos na qual uma mediação é de fundamental importância. O AgenteComprador tem como objetivo comprar os itens contidos em sua lista de compra, na qual contém duas guitarras e uma bateria. O AgenteVendedor tem como objetivo vender seus produtos em estoque, no qual contém duas guitarras e uma bateria.

A simulação realizada nesta aplicação esta baseada na estratégia de compra e venda de produtos pelos AgenteComprador e AgenteVendedor respectivamente. O preço da guitarra inicial contida na Base de Conhecimentos do AgenteVendedor é de 100 reais e da bateria 225 reais. Como todo vendedor, seu objetivo é ganhar e por isso ele acrescenta 100 reais em cada produto ao divulgar uma oferta com a performativa *make-offer*.

A negociação do produto pelo AgenteVendedor é feita com base no preço que o comprador oferecer em cima do preço anunciado de seu produto, como mostra

anteriormente a figura 6.1 na área de texto *MarketPlace*. Esta figura mostra o preço da bateria recebido pelo AgenteComprador do AgenteVendedor, cujo preço é 325 reais (225+100).

A negociação feita pelo AgenteComprador é de maneira análoga ao AgenteVendedor, só que neste caso, no preço do produto é feita uma “pechincha” de 25 reais. A qual é aceita pelo AgenteVendedor, a final o agente esta obtendo lucro na transação, como o AgenteComprador também esta ganhando 25 reais de “pechincha”, as negociações dos produtos foram todas feitas com sucesso como demonstra a figura 6.5.

```
Buyer:initialize()
Seller: inicialize()
Buyer está querendo para comprar um(a)guitarra
Buyer comprei 0 itens. Total de compras R$ 0
Buyer : Oferta de Seller: contendo um(a) guitarra ID Seller1 R$ 200
Seller : Oferta de Buyer : contendo um(a) guitarra ID Seller1 R$ 175
Buyer : Oferta de Seller: contendo um(a) guitarra ID Seller1 R$ 175]
Buyer : OK -- venda do Item guitarra com ID Seller1 com o preço de R$ 175 está
completa
Buyer: comprou guitarra
Buyer está querendo para comprar um(a)bateria
Buyer comprei 1 itens. Total de compras R$ 175
Seller : Oferta de Buyer : contendo um(a) guitarra ID Seller1 R$ 175
Buyer : Oferta de Seller: contendo um(a) bateria ID Seller2 R$ 325
Seller : Oferta de Buyer : contendo um(a) bateria ID Seller2 R$ 300
Buyer : Oferta de Seller: contendo um(a) bateria ID Seller2 R$ 300
Buyer : OK -- venda do Item bateria com ID Seller2 com o preço de R$ 300 está
completa
Buyer: comprou bateria
Seller : Oferta de Buyer : contendo um(a) bateria ID Seller2 R$ 300
Seller tem 1 item no Estoque.
Seller vendeu R$ 475.
Buyer está querendo para comprar um(a)guitarra
Buyer comprei 2 itens. Total de compras R$ 475
Buyer : Oferta de Seller: contendo um(a) guitarra ID Seller3 R$ 200
Seller : Oferta de Buyer : contendo um(a) guitarra ID Seller3 R$ 175
Buyer : Oferta de Seller: contendo um(a) guitarra ID Seller3 R$ 175
Buyer : OK -- venda do Item guitarra com ID Seller3 com o preço de R$ 175 está
completa
Buyer: comprou guitarra
Seller : Oferta de Buyer : contendo um(a) guitarra ID Seller3 R$ 175
Buyer comprei 3 itens. Total de compras R$ 650
```

Figura 6.5 Mensagens enviadas ao marketPlace pelos agentes

7. Conclusões

Sistemas Multiagentes é uma área que tem mostrado um grande potencial computacional, e por isso a comunicação entre os agentes desse sistema é de fundamental importância. Torna-se cada vez mais necessária a evolução de padrões de comunicação entre agentes como FIPA, que é uma iniciativa de padronização frente a outras tentativas como a KQML, pois contribuem para um aumento da interoperabilidade e da escalabilidade entre sistemas computacionais.

Com o crescimento vigente de um mercado cada vez mais globalizado e competitivo, soluções corporativas como a da *Microsoft* com a plataforma *.NET*, têm se tornado uma realidade frente a soluções tradicionais como a plataforma Java da Sun, principalmente no que diz respeito ao suporte a várias linguagens de programação, contribuindo para um maior grau de integração de soluções de tecnologia da informação.

Esse trabalho procurou mostrar a plataforma *.NET*, como uma promissora plataforma para o desenvolvimento de sistemas multiagentes, na qual foi implementado o

MarketPlaceDotNET, que é uma aplicação básica de comunicação entre agentes em KQML.

A principal contribuição deste trabalho está na implementação da extensão do MAFDotNET para suportar uma aplicação padrão de comunicação entre agentes em KQML, o MarketPlaceDotNET, no qual a simulação foi realizada da maneira satisfatória fazendo com que os agentes, vendedor e comprador cumprissem seus objetivos.

Referências

- Austin, J. L. "How to Do Thing with Words". London. Oxford University Press, 1962.
- Bigus, Joseph P.& Bigus Jennifer. "Constructing Intelligent Agents Using Java". USA :John Wiley & Sons Inc, 2.nd , ed 2001.
- Bordini, R. H., Vieira, R. & Moreira, A. F. "Fundamentos de Sistemas Multiagentes". In : Anais do XXI Congresso da Sociedade Brasileira de Computação Fortaleza: UNIFOR, V II, p.3-41, 2001
- ECMA - European Carton Makers Association - Common Language Infrastructure (CLI) ECMA-335, 2001.
- ECMA - European Carton Makers Association C# programming language (C#) ECMA-334, 2001.
- Faraco, Rafael Ávila & Gauthier, Fernando Álvaro Ostuni. "Uma Arquitetura De Agentes Para Negociação Dentro Do Domínio Do Comércio ELETRÔNICO". Dissertação de Mestrando do Curso de Pós-Graduação em Engenharia De Produção. Florianópolis: Universidade Federal De Santa Catarina, 1998.
- Finin, T. Fritzson, R. Mckay, D. McEntire, R Weber, J. Wiederhold, G. Shapiro, S & Beck, C. "Specification of the KQML agent-communication language (plus example agent policies and architectures)". Draft, The DARPA Knowledge Sharing Initiative–External Interfaces Working Group. Disponível na Internet <<http://www.cs.umbc.edu/kqml/papers/>> ,1993.
- Finin, T. Fritzson, R. Mckay, D. & McEntire, R.(1994). "KQML as an Agent Communication Language". UMBC, Baltimore, Disponível na Internet: <<http://www.cs.umbc.edu/kqml>>, 1994.
- FIPA, 1996 Foundation for Intelligent physical Agents. Disponível na Internet: <<http://www.fipa.org>>, 1996 .Acesso : Novembro de 2003
- Foundation for Intelligent physical Agents. Disponível na Internet: <<http://www.fipa.org>>, 1996 .Acesso : Novembro de 2003
- ISO – International Organization for Standardization. Information Technology - Common Language Infrastructure (CLI) ISO/IEC 23271, 2003.
- ISO – International Organization for Standardization. Information Technology – C# programming language (C#) ISO/IEC 23270, 2003.
- Macedo, Hendrik Teixeira & Ramalho, Geber Lisboa. "Mobilidade, Autonomia e Distribuição em Agentes para gerenciamento Corporativo de Sistemas". Dissertação de

Mestrado do Curso de Ciência da Computação. Recife : Universidade Federal de Pernambuco, 2001.

Microsoft Corporation. Microsoft.NET Framework: Product Overview. Disponível na Internet: <<http://msdn.microsoft.com/netframework/productinfo/overview/default.asp> >, 2002

Microsoft Corporation. Shared Source Common Language Infrastructure 1.0 Release. Disponível na Internet: <<http://www.microsoft.com/downloads/details.aspx?FamilyId=3AIC93FA-7462-47D0-8E56-8DD34C6292F0&displaylang=en>>, 2002.

Russel, S. J. & Norvig, P. “Artificial Intelligence: A Modern Approach”. New Jersey : Prentice-Hall, 1995.

Searle, J. R. “Speech Acts: An Essay in the Philosophy of Language”. Cambridge University Press. Cambridge, 1969

Simões, Marco A. C.& Santos A. L. M. “Unificando Agentes Móveis Inteligentes e WBEM para o gerenciamento Corporativo de Sistemas”.Dissertação de Mestrado do Curso de Ciência da Computação. Recife : Universidade Federal de Pernambuco, 2003.

Wooldridge, Michael. An Introduction to Multiagent Systems England: John Wiley & Sons ,Inc, 2002.