Especificação e Implementação de Protocolos de Interação entre Agentes para a Plataforma COMPOR

GLAUBER VINÍCIUS VENTURA DE MELO FERREIRA[1]
HYGGO OLIVEIRA DE ALMEIDA[2]
ANGELO PERKUSICH[2]
EVANDRO DE BARROS COSTA[3]

UFCG - Universidade Federal de Campina Grande
DSC - Departamento de Sistemas e Computação
Caixa Postal: 10.106 - CEP 58.109-970 - Campina Grande (PB)
[1]glauber@dsc.ufcg.edu.br

UFCG - Universidade Federal de Campina Grande
DEE - Departamento de Engenharia Elétrica
Caixa Postal: 10.106 - CEP 58.109-970 - Campina Grande (PB)
[2]{hyggo,perkusic}@dee.ufcg.edu.br

UFAL - Universidade Federal de Alagoas
TCI - Departamento de Tecnologia da Informação
Campus A. C. Simões, BR 104 - Norte, Km 97, Tabuleiro dos Martins - CEP 57072-970 - Maceió (AL)
[3]evandro@tci.ufal.br

Resumo. A abordagem multiagentes tem sido apontada como adequada para o desenvolvimento de sistemas complexos devido ao alto nível de abstração que oferece, sendo a interação entre os agentes sua característica mais importante. Dentro deste contexto, a infra-estrutura COMPOR para a construção de Sistemas Multiagentes utiliza a abordagem baseada em componentes para prover flexibilidade e reutilização no desenvolvimento de tais sistemas. Porém, esta infra-estrutura não implementa mecanismos que facilitem a definição de protocolos de interação entre os agentes do sistema. Neste artigo apresenta-se a definição e a implementação de mecanismos para interação entre agentes da infra-estrutura COMPOR, levando-se em conta protocolos e linguagens de comunicação com base nos padrões definidos por FIPA.

Palavras-Chave: Sistemas Multiagentes, Protocolos de interação, Linguagens de comunicação entre agentes

1 Introdução

A abordagem multiagentes tem sido apontada por muitos pesquisadores como adequada ao desenvolvimento de sistemas de software complexos devido ao seu alto nível de abstração, o que torna mais conveniente a análise e o projeto de tais sistemas [Zambonelli et al.(2004), Jennings(1999), Almeida et al.(2004), Jennings(2001)]. Alguns dos domínios de aplicação nos quais esta abordagem tem sido utilizada são sistemas tutores inteligentes cooperativos, negociação em comércio eletrônico, dentre outros [Costa et al. (2002), Maes et al. (1999), Nunes Filho et al. (2003)].

No contexto dessa abordagem, a interação entre os agentes é a característica principal [Ferber (1999)]. A maior parte dos esforços no desenvolvimento de Sistemas Multiagentes (SMA) é direcionada à forma de interação entre os agentes e às regras que incidem sobre estas interações, denominadas protocolos de interação. Freqüentemente essas interações ocorrem com base em linguagens de alto nível, as quais são conhecidas no contexto de SMA como Linguagens de Comunicação entre Agentes - ACL [1] [Gluz et al. (2003)].

Existem atualmente várias ferramentas para auxiliar o desenvolvimento de Sistemas Multiagentes. Essas ferramentas disponibilizam APIs para a criação dos agentes e plataformas para execução do sistema construído. Alguns exemplos dessas ferramentas são JADE

[1]ACL - *Agent Communication Langauge*

- *Java Agent DEvelopment Framework* [Tilab (2000)] e FIPA-OS [Nortel Networks (1999)].

Dentro deste contexto, está sendo desenvolvido pela Universidade Federal de Campina Grande e pela Universidade Federal de Alagoas um projeto de Engenharia de Software para Sistemas Multiagentes, denominado COMPOR [Almeida et al.(2004)]. A infra-estrutura do projeto COMPOR utiliza a abordagem baseada em componentes para prover flexibilidade e reutilização no desenvolvimento de sistemas multiagentes. Um dos módulos desta infra-estrutura, é um ambiente integrado voltado para o desenvolvimento de Sistemas Multiagentes, denominado COMPOR-E [2] [Almeida et al. (2003)].

Neste artigo, apresenta-se a especificação e implementação de mecanismos que possibilitam a interação entre agentes do ambiente COMPOR-E, através de protocolos de interação e linguagens de comunicação. Mais especificamente, apresenta-se um componente para comunicação entre agentes através de ACL e uma ferramenta para a especificação de protocolos de interação com base nos padrões FIPA (*Foundation for Intelligent Physical Agents* [FIPA (1996)]).

As demais seções do artigo estão organizadas como descrito a seguir: na Seção 2 são apresentados conceitos relacionados a Sistemas Multiagentes; na Seção 3 é apresentado o componente desenvolvido para possibilitar a comunicação através de ACL; na Seção 4 é apresentada a ferramenta para especificação de protocolos de interação; e, finalmente, na Seção 5 são apresentadas as considerações finais deste trabalho.

2 Sistemas Multiagentes e o ambiente COMPOR-E

Um Sistema Multiagentes é composto por um grupo de agentes autônomos que interagem para alcançar seus objetivos [Lesser (1995)]. Devido à importância da abordagem multiagentes, existe uma entidade internacional que define padrões relacionados com esta área, denominada FIPA. Dentre as especificações padronizadas por FIPA, está a especificação de protocolos de interação entre agentes.

Linguagens de comunicação entre agentes

Os agentes de um SMA utilizam linguagens de comunicação entre agentes para expressar as mensagens trocadas entre os mesmos. Exemplos de ACLs são as linguagens KQML (*Knowledge Query and Manipulation Language*) [Finin et al. (1994)] e FIPA ACL [FIPA(2000d)]. A semântica das ACLs é baseada na Teoria dos Atos da Fala definidas em [Searle (1969)], onde as mensagens estão associadas a atos performativos que representam a vontade do agente sobre a informação contida na mensagem.

No Código 1 é apresentado um exemplo simples de uma mensagem FIPA ACL enviada por um agente *i* que requisita ao agente *j* a abertura de um arquivo. Nota-se na mensagem que a performativa utilizada é *request* e a linguagem utilizada no conteúdo é *Visual Basic* [Balena (1999)]. A especificação completa de FIPA ACL pode ser encontrada em [FIPA(2000d)] e a lista de atos performativos em [FIPA (2000c)].

Código 1: Estrutura básica de uma mensagem FIPA ACL

```
(request
    :sender (agent-identifier :name i)
    :receiver (set (agent-identifier :name j))
    :content "open \"db.txt\" for input"
    :language vb)
    :protocol fipa-request
```

Interação e protocolos de interação

Geralmente existem certos padrões na interação entre os agentes, tais como os tipos e a seqüência das mensagens trocadas. A esses padrões dá-se o nome de protocolos de interação.

FIPA define protocolos de interação para que um agente requisite uma tarefa a outro agente; protocolos para licitação, utilizados para selecionar o agente que melhor executa uma tarefa; entre outros.

Em cada protocolo de interação são definidos, entre outras coisas, os papéis que podem ser desempenhados pelos agentes, os tipos de mensagens que podem ser enviadas e recebidas em cada papel, além da seqüência em que essas mensagens são trocadas.

O ambiente COMPOR-E

COMPOR-E é um ambiente integrado para o desenvolvimento de Sistemas Multiagentes. Trata-se de um dos módulos da infra-estrutura COMPOR, que ainda provê um conjunto de diretrizes, técnicas e notações para o desenvolvimento de sistemas multiagentes. O ambiente é dividido nas seguintes ferramentas: ferramenta para a disponibilização de componentes, ferramenta para a construção e execução de agentes e ferramenta de monitoramento.

O projeto interno dos agentes do COMPOR-E é realizado através do Desenvolvimento Baseado em Componentes [Crnkovic (2001)], utilizando um modelo de componentes e o arcabouço de software que implementa as suas especificações [Almeida et al.(2004)]. Desta forma, as habilidades de cada agente são fornecidas pelos serviços dos seus componentes. Os componentes são

[2] Abreviação para COMPOR-*Environment*

dispostos no COMPOR-E em uma paleta de componentes, pertencente à ferramenta de disponibilização.

A ferramenta de construção de agentes possibilita que o desenvolvedor crie os agentes e defina os seus componentes. Um planejador para o agente é então implementado, definindo o seu *script* de execução. Uma vez criados os agentes, seus componentes e seu planejador, a ferramenta de execução é utilizada para inicializar os agentes. A partir deste momento, através da ferramenta de monitoramento são visualizadas as interações entre os agentes durante a execução.

3 Componente para comunicação utilizando FIPA ACL

Nesta seção é apresentado o componente para comunicação ACL para o ambiente COMPOR-E, provendo mecanismos que possibilitam a interação entre agentes utilizando uma ACL e o cumprimento de protocolos de interação por parte dos agentes.

Este componente implementa o envio e o recebimento de mensagens ACL, possibilitando que os agentes possam enviar e receber tais mensagens; o filtro das mensagens enviadas, baseando-se no protocolo no qual a mensagem está inserida e repassando para o agente apenas as mensagens para as quais foi programado para receber, evitando processamento desnecessário; e a determinação dos protocolos seguidos pelos agentes, através de um mecanismo que possibilita a definição dos papéis desempenhados por cada um dos agentes do sistema.

Devido às diferenças existentes na estrutura das mensagens FIPA ACL e KQML, não foi possível criar uma especificação única para as duas linguagens. Diante desse problema, decidiu-se especificar o componente para a linguagem FIPA ACL, já que é padronizada por FIPA e utilizada amplamente no desenvolvimento de Sistemas Multiagentes [Gluz et al. (2003)].

Envio e recebimento de mensagens ACL

O envio e recebimento de mensagens ACL no componente de comunicação é implementado através dos serviços `sendACLMessage` e `receiveACLMessage`. Entretanto, esses dois serviços dependem de algum serviço que realize a comunicação via rede entre dois agentes. Neste trabalho foi utilizado um componente de comunicação RMI [3] já disponível na biblioteca de componentes do ambiente COMPOR-E, que provê esses dois serviços [Paes (2003)].

[3]RMI - *Remote Method Invocation* ou Invocação de Método Remoto, tecnologia utilizada para implementação de aplicações distribuídas em Java.

Para implementar os serviços `sendACLMessage` e `receiveACLMessage` é necessária a utilização de um tipo de dado *ACLMessage* que encapsula os atributos das mensagens trocadas entre os agentes. Na biblioteca de classes da ferramenta JADE - *Java Agent DEvelopment Framework* [Tilab (2000)], está disponível a classe *ACLMessage*, que provê uma implementação de mensagens ACL de acordo com *FIPA ACL Message Structure Specification* [FIPA(2000d)]. Decidiu-se, então, reutilizar esta classe, focando assim o desenvolvimento do componente na lógica envolvida com os protocolos.

Filtro das mensagens enviadas

Para implementar a funcionalidade de filtro das mensagens enviadas é necessário armazenar informações relativas à estrutura dos protocolos de interação. Para isso foi utilizado um arquivo no formato XML, que armazena as informações dos protocolos utilizando a estrutura ilustrada na Figura 1.

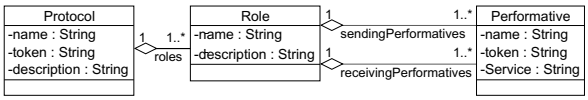


Figura 1: Diagrama de classes para os protocolos de interação

O atributo *token* na classe *Protocol* identifica o protocolo de interação, sendo utilizado no atributo *protocol* da mensagem FIPA ACL, apresentada no Código 1. O componente de comunicação faz referência ao arquivo XML através de um atributo de inicialização, denominado *protocolsFileName*. Na Figura 2 é apresentada a estrutura do arquivo XML, ilustrando um trecho da especificação do protocolo de interação *FIPA Request* [FIPA (2000a)].

Definição dos protocolos seguidos pelos agentes

Para que se possa identificar que mensagens devem ser repassadas para cada agente, é necessário saber quais os papéis desempenhados pelos agentes que seguem um determinado protocolo de interação. Para isso, é utilizado um atributo de inicialização do componente - *roles*.

Por exemplo, caso o agente que possui o componente implementado desempenhe o papel *Initiator* definido no protocolo *FIPA Request*, o desenvolvedor utilizará o ambiente COMPOR-E para definir os atributos de inicialização com os seguintes valores:

- *protocolsFileName* - localização do arquivo XML com a especificação dos protocolos de interação;

```
<Protocols>
  <Protocol  name="FIPA Request Interaction Protocol"
             token="fipa-request"
             description="Possibilita que um agente requisite outro para a
                          execução de uma determinada ação">

    <Role    name="Initiator"
             description="Papel desempenhado pelo agente que
                          solicita a execução da requisição">
      <Send>
        <Performative name="request"
                      description="Enviada para iniciar a conversação"/>
      </Send>
      <Receive>
        <Performative   name="refuse"
                        service="refusedRequest"
                        description="Recebida quando o Participant
                                     recusa a requisição"/>
        <Performative   name="agree"
                        service="agreedRequest"
                        description="Recebida quando o Participant
                                     aceita a requisição"/>
        ...
        Mais Performativas
        ...
      </Receive>
    </Role>
    ...
    Mais Papéis
    ...
  </Protocol>
  ...
  Mais Protocolos
  ...
</Protocols>
```

Figura 2: Arquivo XML com um trecho da especificação do protocolo *FIPA Request*

- *roles* - " fipa-request;Initiator ".

A partir dessas duas informações configuradas, o componente implementado tem acesso à toda a estrutura dos protocolos disponíveis, como também conhece quais papéis de interação o agente desempenha, possibilitando a filtragem das mensagens recebidas. O componente implementado foi disponibilizado na paleta de componentes do ambiente COMPOR-E para que possa ser utilizado por outros desenvolvedores.

Exemplo de utilização do componente

Para descrever a utilização do componente implementado, apresenta-se a seguir um exemplo de interação no qual um agente requisita a execução de uma multiplicação de números inteiros provida por outro agente. Para este exemplo, utiliza-se o protocolo de interação *FIPA Request*, que possibilita que um agente requisite outro agente para a execução de uma determinada ação, definindo o papel *Initiator* para o agente que faz a requisição e *Participant* para aquele que recebe a requisição. Na Figura 3 é ilustrada a interação entre os agentes neste exemplo. Como ilustrado nesta figura, o *Agente B* tem a habilidade de multiplicar, através do componente denominado *MULT*. Os dois agentes também podem se comunicar via rede utilizando o componente *RMI*. Como a interação entre eles será feita utilizando uma linguagem de comunicação entre agentes, ambos possuem o componente implementado para prover comunicação através de *FIPA ACL*.

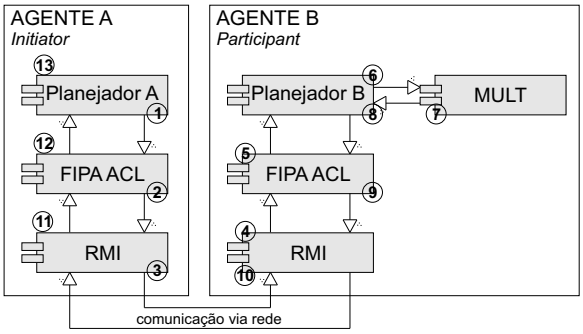


Figura 3: Exemplo de interação entre agentes seguindo o protocolo *FIPA Request* e utilizando o componente de comunicação FIPA ACL

Para que o *Agente A* requisite a execução da multiplicação ao *Agente B* utilizando FIPA ACL, é necessário que o *Planejador A* implemente o código de envio da mensagem utilizando o serviço `sendACLMessage` provido pelo componente de comunicação ACL. É necessário também definir os valores para os atributos de inicialização do componente FIPA ACL dos *Agentes A* e *B*. Para o *Agente A* o atributo *roles* terá o valor "fipa-request;Initiator" e para o *Agente B* o valor "fipa-request;Participant". Para os dois agentes, o valor do atributo de inicialização *protocolsFileName* é a localização do arquivo XML com a especificação do protocolo de interação *FIPA-Request*.

Uma vez implementado o código e configurados os componentes FIPA ACL, os agentes podem ser executados. A seguir é descrito o caminho percorrido pela mensagem enviada até a execução da multiplicação requisitada no *Agente B*.

1. O *Planejador A* cria a mensagem FIPA ACL e o serviço `sendACLMessage` é invocado, passando a mensagem criada como parâmetro.

2. Quando essa mensagem é recebida pelo componente FIPA ACL, o serviço `sendACLMessage` é executado. O serviço `sendService` é então invocado em cada um dos *receivers* da mensagem, para que a mensagem possa ser enviada pela rede. Neste exemplo, apenas o *Agente B* recebe a mensagem.

3. O componente RMI do *Agente A* envia a mensagem pela rede, através da invocação do serviço `receiveService` do componente RMI presente no *Agente B*.

4. Quando a mensagem é recebida pelo *Agente B*, através do seu componente RMI, é invocado o serviço `receiveACLMessage` implementado pelo componente FIPA ACL, passando a mensagem recebida como parâmetro.

5. O serviço `receiveACLMessage` é executado no componente FIPA ACL.

   (a) Verifica-se inicialmente que a mensagem recebida pertence ao protocolo *fipa-request*, estando associada à performativa *request*, de acordo com os atributos contidos na mensagem.

   (b) A partir desses valores e com as informações de configuração do componente, confirma-se que o *Agente B* implementa o protocolo *FIPA Request* desempenhando o papel de *Participant*, no qual está definido o recebimento de mensagens com a performativa *request*.

   (c) Após essa confirmação, o componente recupera do arquivo XML as informações referentes à performativa *request* do papel *Participant* do Protocolo *FIPA Request*. Então, invoca o serviço relacionado a esta performativa: `processRequest`.

6. O *Planejador B* implementa `processRequest`. Ao receber a mensagem, interpreta o seu conteúdo (atributo *content*) e identifica que deve ser invocado o serviço `multiply` do componente MULT.

7. O serviço `multiply` do componente MULT é executado e retorna o valor da multiplicação requisitada que é devolvido para o *Planejador B*.

8. O *Planejador B* ao receber o resultado da multiplicação, cria uma mensagem FIPA ACL com a performativa *inform* e com o conteúdo igual ao resultado da multiplicação. Então, o serviço *sendACLMessage* é invocado.

9-11. O passo 9 é análogo ao passo 2, o passo 10 análogo ao passo 3 e o passo 11 análogo ao passo 4.

12. O serviço `receiveACLMessage` é executado no componente FIPA ACL. Então, de acordo com os atributos da mensagem recebida, é verificado que o *Agente A* implementa o protocolo *FIPA Request* desempenhando o papel *Initiator*. Após essa verificação, o componente recupera do arquivo XML as informações referentes à performativa *inform* do papel *Initiator* do protocolo *FIPA Request*. Por fim, o componente invoca o serviço relacionado com esta performativa, que neste caso é o serviço `informResultRequest`.

13. Ao receber a mensagem, o *Planejador A* recupera o conteúdo, obtendo o resultado da multiplicação que foi solicitada.

O componente desenvolvido possibilita ao desenvolvedor implementar a comunicação entre os agentes utilizando uma linguagem de comunicação entre agentes. Permite também que as mensagens sejam filtradas, repassando para os agentes apenas as mensagens que estes estão aptos a receber.

4 Ferramenta para especificação de protocolos de interação

A implementação do componente de comunicação apresentada na Seção 3 não provê um bom suporte para a especificação dos protocolos de interação, já que para isso o desenvolvedor precisa editar um arquivo XML, tornado-se uma tarefa complexa na medida em que o número de protocolos de interação aumenta.

Para resolver este problema foi implementada uma ferramenta para gerenciar a especificação e edição de protocolos de interação. Para isso, as informações dos protocolos são obtidas a partir de uma interface gráfica com formulários, que posteriormente são armazenadas em um arquivo XML.

Na Figura 4 tem-se a tela inicial da ferramenta, exibindo as informações presentes no arquivo XML de especificação de protocolos. A seguir são detalhados os itens numerados da figura.

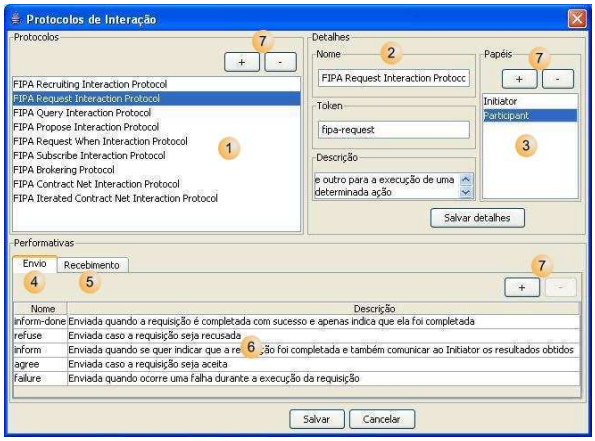


Figura 4: Ferramenta para especificação de protocolos de interação

1. Lista dos protocolos de interação disponíveis no arquivo XML de especificação de protocolos.

2. Informações de detalhes do protocolo de interação selecionado. Tem-se o nome, o *token* e a descrição do protocolo.

3. Lista dos papéis referentes ao protocolo de interação selecionado.

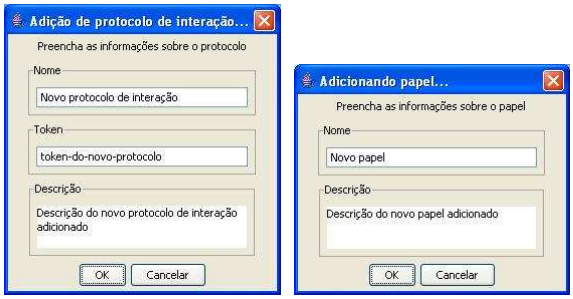
4. Lista de performativas de envio do papel selecionado.

5. Lista de performativas de recebimento do papel selecionado.

6. Informações sobre as performativas. Nas performativas de envio são exibidos o nome e a descrição e nas performativas de recebimento, além dessas duas informações, também é exibido o serviço referente à performativa.

7. Botões para adição e remoção de protocolos, papéis e performativas.

Na Figura 5(a) apresenta-se a tela com o formulário para adição de um novo protocolo de interação. As informações necessárias são o nome, a descrição e o *token* que identifica o protocolo, de acordo com a modelagem apresentada na Figura 1. Na Figura 5(b) tem-se a tela com o formulário para adição de um novo papel com o nome e a descrição do mesmo, também de acordo com a modelagem apresentada na Figura 1. As informações do protocolo e do papel de interação são armazenadas pela ferramenta no arquivo XML de especificação de protocolos.



(a) Protocolo de interação (b) Papel de interação

Figura 5: Adição de protocolos e papéis de interação

Esta ferramenta foi utilizada na definição de protocolos de interação padronizados pela FIPA. Dois protocolos foram especificados de forma completa: *FIPA Request* e *FIPA Contract Net* [FIPA (2000b)]. A escolha desses protocolos foi feita com base na representatividade destes dois em relação aos demais protocolos padronizados pela FIPA. O arquivo XML com a especificação dos protocolos, criado a partir da ferramenta, pode ser encontrado em [Ferreira (2004)].

A ferramenta desenvolvida possibilita a especificação e edição de protocolos de interação para o ambiente COMPOR-E. A ferramenta também pode ser utilizada como um repositório de protocolos de interação, auxiliando nas fases de análise e projeto de um Sistema Multiagentes, nas quais são definidas as interações entre os agentes.

5 Considerações finais

Neste artigo foram apresentadas a especificação e a implementação de mecanismos para permitir a interação entre agentes da infra-estrutura COMPOR, através de protocolos e linguagens de interação.

Foi descrita a implementação de um componente para possibilitar que a interação entre os agentes do sistema seja efetuada utilizando uma Linguagem de Comunicação entre Agentes. Este componente foi implementado possibilitando a utilização da linguagem FIPA ACL, uma linguagem de comunicação entre agentes padronizada pela FIPA. Além disso, foi apresentada uma ferramenta para facilitar a definição dos protocolos de interação entre agentes de acordo com os padrões definidos por FIPA.

Com os resultados obtidos neste trabalho, o desenvolvimento de sistemas utilizando o ambiente integrado COMPOR-E é facilitado, já que o conceito de interação está presente no ambiente, podendo ser utilizado na implementação dos agentes.

Referências

[Almeida et al. (2003)] Almeida, H. O.; Loureiro, E.; Vinícius, G.; Paes, R.; Perkusich, A. & Costa, E. Ambiente Integrado para o desenvolvimento de sistemas multi-agentes. In: *XVII Simpósio Brasileiro de Engenharia de Software - 10a Sessão de Ferramentas*, (10):55-60, 2003.

[Almeida et al.(2004)] Almeida, H. O.; Perkusich, A.; Costa, E. B. & Paes, R. B. COMPOR: a Methodology, a Component Model, a Component based Framework and Tools to Build Multiagent Systems. *CLEI Electronic Journal*, 7(1), 2004

[Balena (1999)] Balena, F. *Programming Microsoft Visual Basic 6.0*. Microsoft Press, 1999.

[Costa et al. (2002)] Costa, E. B.; Almeida, H. O.; Lima, E. F. A.; Nunes Filho, R. R. G.; Silva, K. S. & Assunção, F. M. A Cooperative Intelligent

Tutoring System: The case of Musical Harmony domain. In: *Proceedings of 2nd Mexican International Conference on Artificial Intelligence - MICAI'02*, (2313):367-376, 2002

[Crnkovic (2001)] Crnkovic, I. Component-based Software Engineering - New Challenges in Software Development. *Software Focus*, (4):127-133, 2001.

[Ferber (1999)] Ferber, J. *Multi-Agent Systems - An Introduction to Distributed Artificial Intelligence*. Addison Wesley Professional, 1999.

[Ferreira (2004)] Ferreira, G. V. Especificação e implementação de protocolos de interação para o ambiente COMPOR-E. *Trabalho de Conclusão de Curso do Departamento de Tecnologia da Informação da Universidade Federal de Alagoas*, 2004.

[Finin et al. (1994)] Finin, T; Fritzson, R.; McKay, D.; & McEntire, R. KQML as an Agent Communication Language. *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, 456-463, 1994.

[FIPA (1996)] FIPA. Foundation for Intelligent Physical Agents. *Acessado em Outubro de 2004. http://www.fipa.org*, 1996.

[FIPA (2000a)] FIPA. FIPA Request Interaction Protocol Specification. *Acessado em Outubro de 2004. http://www.fipa.org/specs/fipa00026/*, 2000.

[FIPA (2000b)] FIPA. FIPA Contract Net Interaction Protocol Specification. *Acessado em Outubro de 2004. http://www.fipa.org/specs/fipa00029/*, 2000.

[FIPA (2000c)] FIPA. FIPA Communicative Act Library Specification. *Acessado em Outubro de 2004. http://www.fipa.org/specs/fipa00037/*, 2000.

[FIPA(2000d)] FIPA. FIPA ACL Message Structure Specification. *Acessado em Outubro de 2004. http://www.fipa.org/specs/fipa00061/*, 2000.

[Gluz et al. (2003)] Gluz, J. C. & Viccari, R. M. Linguagens de Comunicação entre Agentes: Fundamentos Padrões e Perspectivas. In: *III Jornada de Mini-Cursos de Inteligência Artificial - SBC2003*, 53-102, 2003.

[Jennings(1999)] Jennings, N. R. Agent-Oriented Software Engineering. In: *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World : Multi-Agent System Engineering*, (1647):1-7, 1999

[Jennings(2001)] Jennings, N. R. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35-41, 2001

[Lesser (1995)] Lesser, V. Multiagent Systems: An Emerging Subdiscipline of AI. *ACM Computing Surveys*, (27):340-342, 1995.

[Maes et al. (1999)] Maes, P; Guttman, R. & Moukas, A. Agents that Buy and Sell: Transforming Commerce as We Know It. *Communications of the ACM*, 42(3):81-91, 1999

[Nortel Networks (1999)] Nortel Networks. FIPA-OS. *Acessado em Outubro de 2004.* http://www.nortelnetworks.com/products/ announcements/fipa/, 1999.

[Nunes Filho et al. (2003)] Nunes Filho, R. R. G.; Costa, E. B. & Almeida, H. O. An Architecture and A Decision-Making Model to Web-Based Electronic Commerce. In: *Proceedings of of I3E - 3rd IFIP conference on e-Commerce, e-Business, and e-Government*, 117-128, 2003

[Paes (2003)] Paes, R. B. Desenvolvimento de componentes para Sistemas Tutores Inteligentes baseados na arquitetura MATHEMA. *Trabalho de Conclusão de Curso do Departamento de Tecnologia da Informação da Universidade Federal de Alagoas*, 2003.

[Searle (1969)] Searle, J. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge, Cambridge University Press, 1969.

[Tilab (2000)] Tilab. Jade - Java Agent DEvelopment Framework. *Acessado em Outubro de 2004. http://sharon.cselt.it/projects/jade/*, 2000.

[Zambonelli et al.(2004)] Zambonelli, F. & Parunak, V. Towards a Paradigm Change in Computer Science and Software Engineering: a Synthesis. *The Knowledge Engineering Review*, 18(4):329-342, 2004