Acta Scientiarum 22(5):1339-1346, 2000.
ISSN 1415-6814.

Uma proposta de arquitetura de *software* baseada em agentes

Elisa Hatsue Moriya Huzita*, Hélio Marci de Oliveira e Jean Marcos Laine

Departamento de Informática, Universidade Estadual de Maringá, Av. Colombo, 5790, 87020-900, Maringá-Paraná, Brazil.
**Author for correspondence. e-mail: emhuzita@din.uem.br*

RESUMO. O desenvolvimento de *software* baseado em agentes tem sido objeto de estudo de diversos pesquisadores, que analisam a sua aplicação em sistemas distribuídos e buscam técnicas capazes de minimizar as dificuldades encontradas nesta nova abordagem. Neste contexto, este artigo tem por objetivo principal descrever uma proposta de arquitetura de *software* para sistemas multiagente, elaborada a partir de estudos realizados sobre agentes de *software*, sistemas multiagente e arquiteturas de *software* baseadas em agentes. São ainda apresentados alguns dos conceitos estudados, incluindo as principais falhas encontradas na construção de arquiteturas baseadas em agentes e dois modelos propostos por outros autores, que serviram de subsídio para a elaboração da proposta a ser apresentada.

Palavras-chave: agentes de *software*, arquiteturas de *software*, sistemas multiagente

ABSTRACT. A proposal of agent-based software architecture. Agent-based software development has been the object of several studies which analyze its application to distributed systems and look for techniques capable of minimizing the difficulties inherent to this new approach. A suggestion of software architecture for multi-agent systems has been provided. It has been elaborated from studies on software agents, multi-agent systems and agent-based software architectures. Some of these concepts have been given, including the main pitfalls found in the construction of agent-based architectures. Further, two models that served as subsidy for their elaboration have also been described.

Key words: software agents, software architectures, multi-agent systems

Um novo paradigma de desenvolvimento de software vem ganhando adeptos e despertando interesses e expectativas entre os pesquisadores de novas tecnologias, em especial os engenheiros de *software*. Este paradigma pode ser visto como uma evolução do desenvolvimento baseado em objetos e é caracterizado pela substituição destes por entidades autônomas denominadas agentes de *software*.

Novas aplicações têm surgido, apresentando características como distribuição dos dados e do processamento, que possibilitam a implementação utilizando agentes de *software*. Dentre essas aplicações, destacam-se algumas tarefas que podem ser delegadas aos agentes, como, por exemplo, gerenciamento de redes locais, distribuição de *software* por uma rede e *backup* de bancos de dados distribuídos (Knapik, 1998). Devido às características incorporadas aos agentes - como autonomia, mobilidade, reatividade e comunicação, por exemplo - é possível obter um *software* mais dinâmico que favoreça o paralelismo durante a execução das operações atribuídas aos agentes.

Contudo, as dificuldades encontradas na especificação de agentes têm mostrado a necessidade do desenvolvimento de técnicas apropriadas, que ofereçam, suporte para a representação destas características que diferenciam os agentes de outros software.

O desenvolvimento baseado em agentes precisa descrever os agentes e modelar as relações sociais entre os agentes, as interações dos agentes com os usuários e as características encontradas nos agentes.

Este trabalho apresenta uma proposta de arquitetura de software baseada em agentes e identifica algumas necessidades básicas que devem ser satisfeitas pela mesma. Tal arquitetura provê o suporte para a utilização de agentes na realização de tarefas e o gerenciamento de informações distribuídas. Os agentes definidos na arquitetura poderão estar distribuídos nos vários domínios ou áreas existentes na aplicação utilizando-se os diferentes recursos disponíveis.

Com base nos estudos realizados, é apresentada uma proposta de modelo de arquitetura de *software*

baseada em agentes e são descritas as relações existentes entre seus componentes. Nesta arquitetura os agentes são vistos como entidades de *software* que perseguem seus objetivos, podendo se comunicar com outros agentes inseridos no ambiente e utilizar os recursos disponíveis.

Este artigo está estruturado na seguinte forma: na seção 2 são apresentadas algumas definições de agentes e de sistemas multiagente e descritas as principais características pertinentes aos agentes. Conceitos e características de sistemas distribuídos são vistos na seção 3. Na seção 4 são descritos, resumidamente, conceitos sobre arquiteturas de *software*, é apresentada uma classificação dos modelos de arquiteturas segundo a inteligência artificial tradicional (Fernándes, 1998) e são descritos ainda dois modelos de arquiteturas estudados para a elaboração da proposta (Dale, 1997 e Flores, 2000). Por fim, na seção 5 é descrita a proposta de um modelo de arquitetura de *software* utilizando agentes, incluindo as notações utilizadas e seus componentes.

Agentes de *software*

"Agentes inteligentes são entidades de *software* que realizam um conjunto de operações em benefício de um usuário ou de outro programa com algum grau de independência ou autonomia, fazendo uso de algum conhecimento ou representação de objetivos ou desejos do usuário" (Atkinson, 1996).

Os agentes podem apresentar características inteligentes, como capacidade de raciocínio e aprendizagem. Contudo, para dotar um agente com tais capacidades faz-se necessário um estudo mais aprofundado de técnicas de inteligência artificial que ofereçam o suporte adequado. Desta forma, neste trabalho são priorizadas outras características, discutidas na seção 2.1, presentes nos agentes de *software* que permitem identificá-los como tais.

"Um agente é um sistema computacional encapsulado que está situado em algum ambiente, e que é capaz de agir autônoma e flexivelmente naquele ambiente para alcançar seus objetivos de projeto" (Wooldridge, 1997).

Segundo (Ferber, 1999), um agente é uma entidade física ou virtual que possui uma representação parcial de seu ambiente, sendo capaz de perceber e atuar sobre o mesmo e oferecer serviços através de seus objetivos e habilidades. Uma entidade física é alguma coisa que atua sobre o mundo real, por exemplo, um robô ou um carro. Já um componente de *software* é uma entidade virtual sem existência física no mundo real. Para alcançar seus objetivos os agentes podem se comunicar e acessar os recursos disponíveis no ambiente.

Conforme pode ser visto nas definições apresentadas nestes parágrafos, pode-se perceber que, apesar das várias tentativas de diversos pesquisadores, das mais diferentes instituições de pesquisa, ainda não existe uma definição de agentes universalmente aceita.

Neste trabalho, um agente de *software* é considerado como sendo uma entidade de *software* capaz de executar tarefas de forma autônoma, reagindo aos estímulos do ambiente no qual está inserido, tomando suas próprias iniciativas e interagindo com outros agentes.

Quanto à sua mobilidade, um agente pode ser classificado como estático ou móvel. Um agente móvel possui a capacidade de se transportar através de uma rede de computadores visitando as diversas áreas definidas na aplicação. Pode-se definir área como sendo um ambiente limitado composto por recursos, de *hardware* ou *software*, e por agentes que executam as tarefas definidas em seus objetivos (Flores, 2000).

O conjunto de várias entidades autônomas (agentes) em uma mesma área, podendo se comunicar a fim de realizar tarefas que atendam a seus objetivos, constitui um sistema denominado sistema multiagente.

Características dos agentes de *software*. Em geral, os agentes de *software* apresentam algumas características básicas que permitem diferenciá-los dos programas de *software* convencionais. Dentre as várias características citadas em Brenner (1998), podem-se destacar as seguintes:

- **Reatividade:** é a capacidade que os agentes possuem de reagir a eventuais mudanças ocorridas no ambiente, respondendo através de ações estabelecidas em seus objetivos.
- **Proatividade:** permite aos agentes não apenas reagir a estímulos, mas também tomar iniciativas que visem alcançar seus objetivos.
- **Autonomia:** esta é a principal diferença entre um agente e um programa de *software* comum. Os agentes possuem autonomia para resolver suas tarefas sem necessitar da interferência do usuário, tomando suas próprias decisões.
- **Mobilidade:** esta capacidade é de grande importância para os agentes, pois permite que estes se desloquem através de uma rede de computadores em busca de recursos pertencentes a outras áreas, necessários à execução de suas tarefas, ou ainda, que eles

busquem ajuda de outros agentes, estabelecendo um ambiente de cooperação.
- **Comunicação:** em cada área podem existir vários agentes, constituindo-se assim, como já foi mencionado, um sistema multiagente. Para que seja possível estabelecer um ambiente de cooperação, é preciso utilizar uma linguagem comum que ofereça suporte à troca de mensagens entre os agentes. Dessa forma, tarefas complexas podem ser divididas entre os agentes, diminuindo o tempo de resposta do sistema para o usuário.
- **Benevolência:** um agente não pode realizar serviços para outros agentes, que o coloquem em conflito com seus próprios objetivos.

Sistemas distribuídos

O termo sistema distribuído originou-se em 1960 com o desenvolvimento de computadores multiusuário e das redes de computadores. Foi estimulado pelo desenvolvimento das *workstations* de baixos custos, das redes locais e do sistema operacional UNIX em 1970.

"Sistemas distribuídos consistem em uma coleção de computadores autônomos ligados por uma rede, buscando a coordenação das atividades eficientemente, e propiciando o compartilhamento de recursos" (Coulouris, 1998).

A seguir são apresentadas as características-chave dos sistemas distribuídos (Coulouris, 1998):
- **Recursos compartilhados:** existem diferentes tipos de recursos que podem ser compartilhados através de uma rede de computadores. Estes recursos podem ser: componentes de *hardware*, como discos e impressoras, ou de *software*, como arquivos, banco de dados e outros objetos de dados.
- **Concorrência:** quando vários processos estão sendo executados em um único computador diz-se que eles são concorrentes. Em um sistema distribuído, a presença de múltiplos usuários é uma fonte de solicitações concorrentes aos seus recursos compartilhados.
- **Escalabilidade:** um sistema distribuído é capaz de funcionar eficientemente em diversas escalas. Ele pode ser composto por apenas duas *workstations* e um servidor de arquivos ou até centenas delas e muitos servidores de arquivos, de impressão e outros. Esta característica tenta garantir que o sistema e a aplicação não necessitem de mudanças quando a escala do sistema aumentar.
- **Tolerância a falhas:** algumas vezes acontecem falhas nos sistemas que são indesejáveis e, muitas vezes, inesperadas. Quando ocorrem falhas, em *hardware* ou *software*, os programas podem produzir resultados incorretos ou eles podem parar antes de ter completado a atividade que vem sendo realizada. O projeto de sistemas de computadores tolerantes a falhas é baseado em duas abordagens: redundância de *hardware* (uso de componentes redundantes ou em excesso) e restabelecimento de *software* (programas que recuperam as falhas ocorridas).
- **Transparência:** transparência é definida como a propriedade de ocultar do usuário e do programador a forma como os componentes, hardware e software, estarão distribuídos dentro do sistema, de modo que o sistema seja percebido como se fosse centralizado, não exigindo esforços maiores para o usuário.
- **Abertura:** é a característica que determina se um sistema pode ser extendido em relação ao *hardware* e também ao *software*. A abertura, em um sistema distribuído, é determinada principalmente pelo grau em que novos recursos compartilhados podem ser adicionados e tornados disponíveis para uso pelos programas clientes. A interface (especificação e documentação) dos componentes de um sistema é tornada pública.

Arquiteturas de *software*

Uma arquitetura de *software* envolve a descrição de elementos com os quais os sistemas são construídos, as interações entre estes elementos, os padrões que guiam suas composições e as regras sobre estes padrões (Shaw, 1996). Geralmente um sistema é definido em termos de uma coleção de componentes e das interações entre estes componentes. As interações, neste nível de projeto, podem ser simples como uma chamada de procedimento ou um acesso a uma variável compartilhada.

As arquiteturas são representadas de forma abstrata como diagramas de caixas e linhas junto com uma descrição que explica o significado por trás de cada um dos símbolos.

De acordo com (Shaw, 1996), a relativa informalidade e o alto nível de abstração empregados no momento de descrever a arquitetura deixam a impressão de que o projeto arquitetural tem pouco

valor substancial para os engenheiros de *software*, porém, conforme explica (Shaw, 1996), apesar de a estrutura arquitetural ser abstrata, em relação aos detalhes de computação dos elementos envolvidos, os componentes da arquitetura provêem um esqueleto natural que possibilita uma melhor compreensão dos interesses em nível de sistema, como uma idéia geral dos fluxos, das interações, da estrutura de controle de execução e da escalabilidade.

Classificação das arquiteturas segundo a inteligência artificial. Segundo (Fernández, 1998), as arquiteturas de agentes podem ser classificadas de acordo com o tipo de processamento empregado na realização das atividades pertencentes ao sistema em deliberativas, reativas e híbridas.

Embora esta classificação se refira à arquitetura do agente, ela também pode ser utilizada para analisar as arquiteturas de sistemas baseados em agentes. Desta forma, se os agentes pertencentes a um sistema apresentam características deliberativas, pode-se considerar que a arquitetura deste sistema é deliberativa. Analogamente, caso existam agentes reativos ou ambos no mesmo sistema, a arquitetura deste sistema pode ser classificada como reativa ou híbrida, respectivamente.

Arquiteturas deliberativas. Este tipo de arquitetura segue a linha da inteligência artificial simbólica, baseada na hipótese dos sistemas de símbolos físicos. Estes sistemas são capazes de manipular estruturas simbólicas para exibir características inteligentes. Um dos problemas encontrados consiste em descrever os objetivos e os meios utilizados para alcançá-los e como traduzir o conhecimento ao nível simbólico.

Uma arquitetura deliberativa estabelece um estado inicial, um conjunto de operadores e um estado final como sendo o objetivo. Assim, o agente deliberativo terá que descobrir uma forma de alcançar o estado final, que é seu objetivo, partindo do estado inicial e utilizando os operadores definidos.

Arquiteturas reativas. Este tipo de arquitetura questiona a validade do modelo simbólico e propõe um novo modelo, que segue o modelo conducionista, baseado na relação estímulo e resposta. Dessa forma, esta arquitetura não apresenta um modelo do mundo simbólico como elemento central para o raciocínio dos agentes.

O funcionamento se resume em receber estímulos externos, processá-los e responder ao ambiente através de ações realizadas pelos agentes.

Arquiteturas híbridas. Esta arquitetura combina características dos dois modelos anteriores, ou seja, ela apresenta aspectos referentes à arquitetura reativa e à deliberativa. Portanto, existem módulos reativos que respondem aos estímulos recebidos do ambiente e módulos deliberativos que determinam quais ações devem ser feitas para que os objetivos do agente sejam satisfeitos.

Principais falhas/dificuldades encontradas na construção de arquiteturas baseadas em agentes. Devido, principalmente, ao fato de a tecnologia de agentes ser ainda um tema de pesquisa recente e à falta de técnicas capazes de oferecer o suporte adequado ao desenvolvimento baseado em agentes, muitas são as dificuldades encontradas por projetistas que trabalham com o desenvolvimento de sistemas multiagente. Algumas destas dificuldades são descritas em Wooldridge e Jennings (1999), podendo ser divididas nas seguintes classes de problemas:

- problemas relacionados aos conceitos envolvidos com a tecnologia de agentes e sua aplicação.
- dificuldades geralmente encontradas na análise e no projeto de sistemas multiagente.
- falhas que muitas vezes são cometidas durante a construção da arquitetura.

Dentre as falhas encontradas no processo de construção da arquitetura do sistema multiagente, destacam-se:

- Muitos projetistas, mesmo não dispondo de conhecimento ou experiência suficientes, tentam desenvolver suas próprias arquiteturas de sistemas multi-agente sem considerar modelos já propostos por outros autores, que geralmente já passaram por várias análises, estando, portanto, mais consolidados.
- Ao projetar uma arquitetura, não se deve acreditar que ela seja genérica. Uma arquitetura desenvolvida para um domínio normalmente precisa sofrer algumas alterações antes de ser aplicada em um outro.

Portanto, é aconselhável que, ao elaborar uma arquitetura de *software* baseada em agentes, o projetista considere outros modelos validados, aproveitando experiências anteriores. Além disso, é interessante que a classe de problemas à qual esta arquitetura se refere esteja previamente identificada.

Modelos de arquiteturas estudados. Para a elaboração do modelo de arquitetura baseado em agentes de *software* foram analisadas duas arquiteturas: uma primeira, proposta por Dale (1997) e a outra por Flores, (2000).

A arquitetura de Dale (1997) contém tanto agentes móveis quanto estáticos, cada um desempenhando suas tarefas específicas. Na Figura 1, que representa a arquitetura de Dale (1997), está representado somente o agente móvel; no entanto, ele pode ser considerado um agente estático quando permanece fixo em um domínio.

Os ambientes contendo agentes, recursos e usuários são chamados de domínio por Dale (1997), enquanto Flores (2000) denomina-os de área.

A arquitetura desenvolvida por Dale é constituída basicamente pelos componentes: *User, User Interface Agent, Mobile Agent, Domain Agent, Resource Agent, Resource e Gateway Agent*. Uma breve descrição das funcionalidades de cada um destes componentes é realizada a seguir:

- ***Domain Agent*:** é um agente estático que supervisiona e coordena as atividades que podem ocorrer dentro de seu domínio. Exemplos destas atividades são: prover serviços de migração aos agentes móveis que desejam migrar para outros domínios, mediar o acesso aos recursos existentes em seu domínio, publicar os tipos de recursos e os agentes que existem no seu domínio.
- ***Resource Agent*:** é um agente estático que controla o acesso dos agentes aos recursos do domínio através das verificações de permissões concedidas aos agentes para cada recurso. Algumas de suas atividades são: prover uma descrição dos tipos de serviços que são oferecidos pelos recursos, permitir ou não o acesso ao recurso, de acordo com as permissões de cada agente, entre outras.
- ***Mobile Agent*:** é um componente da arquitetura que pode migrar entre domínios. Ele determina para onde migrar através de um consulta ao *Domain Agent* e se comunica não só com os *Resource Agents*, mas também com outros *Mobile Agents*.
- ***User Interface Agent*:** é um agente que reside dentro do domínio e faz a interface entre o usuário e os agentes existentes no sistema. Ele organiza e faz um pré-processamento das informações que são retornadas pelos *Mobile Agents* para informá-las aos usuários.
- ***Gateway Agent*:** é um agente estático que provê entradas e saídas de um domínio. Dessa forma, ele pode prover um *firewall* de modo a proteger as informações trocadas entre os domínios e controlar a migração dos agentes móveis entre as diversas áreas.
- ***Resource*:** podem ser componentes de *hardware*, como discos e impressoras, ou recursos de *software*, como arquivos e banco de dados.
- ***User*:** são usuários que estarão interagindo com a aplicação e solicitando serviços.

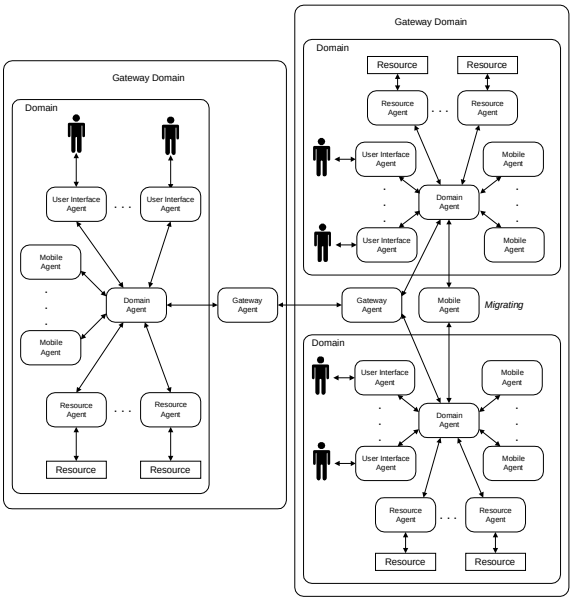


Figura 1. Arquitetura proposta por Dale Dale (1997)

O segundo modelo estudado, mostrado na Figura 2, foi proposto por Flores (2000). Esta arquitetura contém os seguintes componentes: *Local Area Coordinator, Yellow Pages, Agent, Cooperation Domain* e *Cooperation Domain Server*. Assim como no modelo proposto por Dale (1997), as funcionalidades de cada um destes componentes serão descritas a seguir:

- ***Local Area Coordinator*:** ele atua como um representante da área, coordenando a comunicação entre os agentes e os recursos disponíveis e implementando a política definida pelo dono do recurso.
- ***Yellow Pages*:** publica serviços oferecidos pelos agentes, que se encontram dentro de sua área, para que outros agentes tenham condições de localizá-los e propor um ambiente de colaboração na execução das tarefas. Este ambiente só é obtido se os agentes tiverem propósitos comuns.
- ***Agent*:** ele se registra junto ao *Local Area Coordinator*, pode participar do *Cooperation Domain*, estabelecendo comunicação com

demais agentes, e pode oferecer e requisitar serviços a outros agentes.

- ***Cooperation Domain*:** ambiente onde os agentes podem se comunicar;
- ***Cooperation Domain Server*:** tem a função de gerenciar o *Cooperation Domain*, coordenando o processo de comunicação entre os agentes.

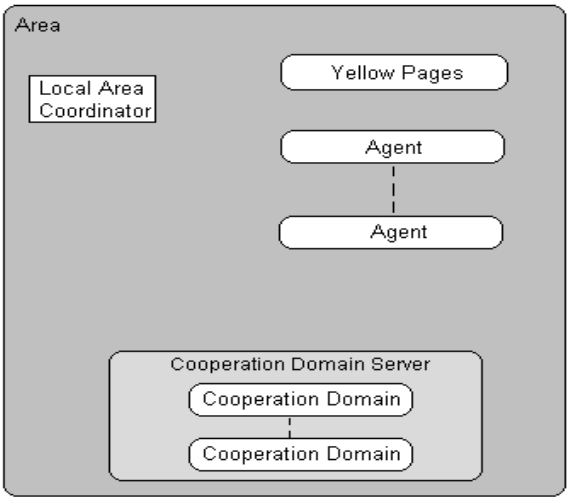


Figura 2. Arquitetura proposta por Flores (2000)

Proposta de um modelo de arquitetura de *software* baseada em agentes

A partir dos modelos estudados, os seguintes requisitos inerentes à arquitetura foram identificados:

- identificar e organizar os recursos disponíveis em cada área.
- estabelecer uma forma de controle de acesso aos recursos de cada ambiente.
- permitir que os agentes possam localizar outros agentes e propor a colaboração e delegação de tarefas.
- proporcionar meios aos usuários de se comunicar com os agentes e solicitar tarefas.
- manter uma relação ou possível comunicação entre áreas a fim de saber quais recursos e serviços existem e são oferecidos em cada uma delas.

Com base nestes requisitos e considerando as particularidades (falhas/dificuldades) a serem observadas na elaboração de arquiteturas baseadas em agentes, foi desenvolvida uma proposta de um modelo de arquitetura de software para uso em aplicações distribuídas que utilizem sistemas multiagente (Figura 3).

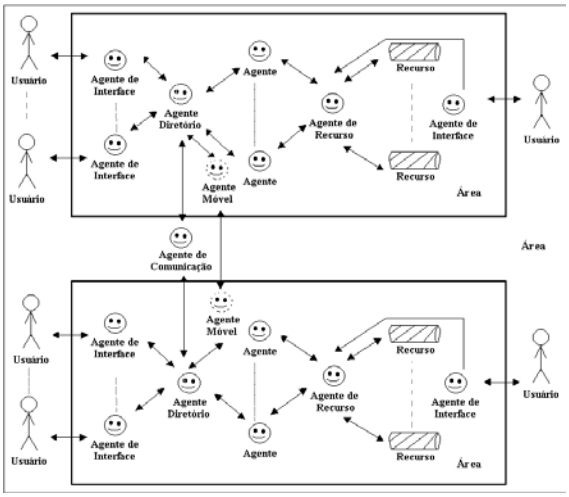


Figura 3. Modelo de arquitetura de software baseada em agentes

O modelo proposto apresenta características reativas (vide Classificação das arquiteturas segundo a inteligência artificial), pois os agentes inseridos nesta arquitetura respondem a estímulos enviados pelos usuários, através de solicitações de tarefas ao sistema.

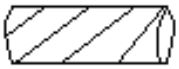
Os seguintes componentes fazem parte do modelo elaborado:

- **Área:** região limitada que organiza os recursos, *hardware* ou *software*, existentes no sistema. Este componente satisfaz o primeiro requisito.
- **Agente de recurso:** estabelece uma forma de controlar o acesso dos agentes aos recursos disponíveis em cada área. Estas medidas são necessárias por questões de segurança e integridade dos dados existentes nestas áreas. Assim, o segundo requisito é satisfeito.
- **Agente diretório:** responsável por divulgar os serviços oferecidos por cada agente, bem como localizar os agentes responsáveis por cada funcionalidade do sistema. Com isso, o terceiro requisito é garantido.
- **Agente de interface:** como o próprio nome já diz, este agente faz a interface entre o usuário e o sistema, recebendo solicitações de tarefas dos usuários e as transmitindo aos agentes designados para executá-las. Este agente atende ao quarto requisito.
- **Agente de comunicação:** sua função é interligar as áreas e proporcionar um canal de comunicação entre elas, de modo que um Agente Diretório tenha conhecimento dos tipos de serviços e recursos existentes nas demais áreas. Isso satisfaz o quinto requisito.

Notações utilizadas na arquitetura proposta. Para a elaboração da arquitetura proposta foi preciso definir algumas notações para representar os componentes de software existentes em um sistema e os relacionamentos entre eles. Cada uma destas notações está listada a seguir, com uma descrição de suas funcionalidades.


Representa a figura de um agente estático, responsável por realizar alguma tarefa dentro do sistema, de modo reativo ou pró-ativo. Estes agentes podem se comunicar entre si, durante a realização de determinadas tarefas, estabelecendo, assim, um ambiente de cooperação, ou eles podem se comunicar com o usuário. Para alcançar seus objetivos um agente talvez precise acessar determinados recursos disponíveis na área em que ele se encontra ou até mesmo em áreas remotas.


Simboliza um usuário do sistema. Este usuário pode interagir com o mesmo e solicitar tarefas aos agentes através de requisições enviadas ao agente de interface.


Representa um recurso do sistema, hardware ou software, que será utilizado pelos agentes durante a execução de suas funcionalidades. Este recurso pode ser, por exemplo, um banco de dados ou uma impressora.


Representa a figura de um agente móvel. Além de possuir as mesmas funcionalidades de um agente estático, ele pode se mover pela rede em busca de recursos ou de outras funcionalidades oferecidas pelos agentes.


Esta seta representa a ocorrência de troca de mensagens entre os componentes relacionados. Pode-se perceber que estas trocas de mensagens acontecem entre agentes ou entre agentes e usuários do sistema.

Considerações sobre a utilização do modelo proposto. A adoção de uma arquitetura de *software* pode auxiliar no processo de desenvolvimento de sistemas multiagente, permitindo que seja definido, através de sua estrutura, um esqueleto da forma como o sistema está organizado, os componentes existentes em cada área e seus possíveis relacionamentos. Além disso, uma análise da mesma pode ajudar a identificar o fluxo de mensagens trocadas pelos seus componentes.

Desta forma, uma arquitetura de *software* constitui-se em um poderoso artefato, resultante do processo de engenharia de *software*, cuja aplicação na construção de sistemas multiagente pode melhorar a compreensão das interações entre os agentes, além de auxiliar no entendimento do papel de cada agente dentro do sistema. Contudo, para que esta arquitetura ofereça o suporte adequado ao desenvolvimento baseado em agentes, ela deve satisfazer alguns requisitos.

Neste intuito, a proposta de arquitetura apresentada neste artigo procura cumprir tais requisitos e oferecer suporte ao desenvolvimento de aplicações distribuídas que utilizem sistemas multiagente. Comparando-a com os modelos estudados, pode-se destacar algumas vantagens.

A sobrecarga de funções executadas pelo *Domain Agent*, conforme proposto por Dale (1997), foi minimizada na proposta apresentada neste trabalho pela inclusão dos agentes Diretório e de Recurso. Desta forma, espera-se aumentar o paralelismo entre as atividades que serão executadas pelos agentes, pois um número maior de agentes poderá estar recebendo respostas destes dois agentes em um mesmo período de tempo.

Para validar a proposta descrita neste artigo, esta arquitetura está sendo aplicada no desenvolvimento de um sistema-exemplo destinado ao controle de eventos científicos.

Referências bibliográficas

Atkinson, B. *Developing Intelligent Agents for Distributed Systems.* New York: McGraw-Hill, 1998. p. 3.

Brenner, W.; Zarnekow, R.; Wittig, H. *Intelligent Software Agents*. New York: Springer, 1998.

Coulouris, G. *Distributed Systems* - Concepts and Design. England: Addison-Wesley, 1998.

Dale, J. *A Mobile Agent Architecture for Distributed Information Management*. United Kingdom, 1997. (Doctoral Thesis in Artificial Intelligence) - Faculty of Engineering and Applied Science, University of Southampton.

Ferber, J. *Multi-Agent Systems* - An Introduction to Distributed Artificial Intelligence. New York : Addison-Wesley, 1999.

Fernández, C.A.I. *Definición de uma Metodología para el desarrollo de sistemas multiagente*. Madrid, 1998. (Doctoral Thesis in Artificial Intelligence) - Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid.

Fernández, C.A.I.; Garijo, M. presented as position paper at FIRST SIG MEETING, September 24 & 25, Belgium: Brussels, 1998. Disponível em: http://www.cs.vu.nl/~treur/SIG1.ciglesias.html. Acesso: set./2000.

Flores, R. An Architecture for Modeling Internet-based Collaborative Agent Systems. In: WORKSHOP ON INFRASTRUCTURE FOR SCALABLE AGENTS SYSTEMS, FOURTH INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS (AGENTS'2000), 2000, Barcelona, *Proceedings*... Barcelona, 2000.

Grosse, A. presented as position paper at FIRST SIG MEETING, September 24 & 25, 1998, Belgium: Brussels, Belgium. Disponível em: http://www.cs.vu.nl/~treur/SIG1.agrosse.html. Acesso em: set./2000.

Knapik, M.; Johnson, J. Agent Applications. In: Knapik, M.; Johnson, J. *Developing Ingelligent Agents for Distributed Systems*. New York: McGraw-Hill, 1998. p. 317-340.

Shaw, M.; Garlan, D. *Software Architecture:* Perspectives on an Emerging Discipline. [S.l]: Prentice-Hall, 1996.

Wooldridge, M.N.R. Agent-Oriented Software Engineering. In: INTERNATIONAL CONFERENCE ON INDUSTRIAL AND ENGENEERING APPLICATIONS OF I.A., 12., 1999, Cairo, *Proceedings*..., Cairo, 1999. p. 4-10.

Received on August 17, 2000.

Accepted on November 24, 2000.