

Aplicação da Arquitetura de Objetos Distribuídos em Sistemas Multi-Agentes

Jomi Fred Hübner^{*,1}

Jaime Simão Sichman^{†,1}

Jorge Risco Becerra²

Escola Politécnica da Universidade de São Paulo

¹Laboratório de Técnicas Inteligentes

²Laboratório de Sistemas Abertos

Av. Prof. Luciano Gualberto, nr. 158, 05508-900 São Paulo, SP

{jomi,jbecerra,jaime}@pcs.usp.br

<http://www.pcs.usp.br>

RESUMO

Na área de Inteligência Artificial Distribuída (IAD) tem-se despendido bastante esforço na construção de ambientes para execução de agentes. Paralelamente, a engenharia de software tem feito avanços na definição de *frameworks* para Sistema de Objetos Distribuídos (SOD). Neste contexto, este trabalho tem por objetivo avaliar a aplicação da arquitetura de objetos distribuídos no desenvolvimento de ambientes para a IAD, verificando primeiro a relação entre objetos e agentes e depois a relação entre objetos distribuídos e IAD. Concluiu-se que muitas das propriedades dos problemas que a IAD procura abordar são comuns com os SODs. Com a utilização dos resultados alcançados pelos trabalhos realizados na área de SOD, o projetista do Sistema Multi-Agente (SMA) teria preocupação apenas com os aspectos que realmente envolvem a questão da emergência da inteligência, ignorando os aspectos de distribuição. Portanto, o nível de abstração atendido pelo SODs pode muito bem servir de suporte para o nível SMAs.

PALAVRAS CHAVE

Sistemas Multi-Agente, Objetos Distribuídos

1 Introdução

A Inteligência Artificial Distribuída (IAD) é uma área da Ciência da Computação que recentemente vem ganhando atenção de pesquisadores em Inteligência Artificial (IA) pois, ao contrário dos paradigmas tradicionais da IA, considera como modelo a coletividade e não um único indivíduo [13, 1, 3]. Desta forma, deixam de ter atenção as iniciativas de simular o comportamento humano, seja mental (IA simbolista) ou neural (IA conexionista), passando o foco da atenção para a forma de interação entre as entidades (chamadas

*Financiado pela Universidade Regional de Blumenau (FURB) e CAPES

†Parcialmente financiado pelo CNPq, bolsa 98/03489-9

de agentes) e sua organização social. Para a IAD, a relação do agente com o meio, seja por interação ou organização, é determinante na emergência do comportamento inteligente. Este paradigma é motivado pela observação de vários sistemas naturais, nos quais pode-se perceber o surgimento de um comportamento inteligente a partir da interação de seus elementos. Por exemplo, apesar de uma colônia de formigas ser formada por seres simples, pode-se dizer que o formigueiro como um todo tem “inteligência”; os neurônios são simples células, mas de sua interação e organização emerge um comportamento complexo e inteligente.

Entretanto, já em 1988, Gasser apontava para o problema da falta de um ambiente adequado para o desenvolvimento de agentes [4]. Depois de 12 anos, muitos ambientes de desenvolvimento surgiram provendo algumas facilidades para o desenvolvimento de sistemas de IA distribuída. Contudo, estes ambientes ainda não possuem todas as características desejáveis num ambiente para IAD. Paralelamente, a engenharia de software tem avançado bastante na definição de metodologias e técnicas (Open Distributed Processing (ODP) e Common Object Request Broker Architecture (CORBA), por exemplo) para o desenvolvimento de sistemas em ambientes distribuídos. Um dos resultados é um alto nível de transparência dos aspectos de distribuição, alcançado pela elaboração do conceito de objetos distribuídos. Naturalmente, surge a pergunta: que contribuições as técnicas de objetos distribuídos podem oferecer no desenvolvimento de ambientes para IAD?

Este trabalho tem por objetivo principal avaliar a aplicação da arquitetura de objetos distribuídos no desenvolvimento de ambientes para a IAD. Para isso, alguns objetivos parciais serão buscados: identificar a relação entre objeto e agente; identificar a relação entre objetos distribuídos e IAD; e investigar quais os requisitos de implementação de um ambiente para IAD que a arquitetura de objetos distribuídos pode atender.

2 Objetos Distribuídos

Segundo as recomendações da International Standard Organization (ISO) para o desenvolvimento de sistemas abertos de processamento distribuído (ODP) [7], um **sistema distribuído** é aquele que apresenta as seguintes características: *Distribuição geográfica* (os elementos do sistema estão dispersos geograficamente, podendo ter interações locais ou remotas), *concorrência* (cada componente do sistema pode executar paralelamente aos demais), *inexistência de estado global* (o estado global do sistema não pode ser determinado), *possibilidade de falhas parciais* (qualquer componente do sistema pode falhar sem comprometer os demais), *comunicação assíncrona* (não há um sistema global de sincronização), *heterogeneidade* (não há garantias de que os componentes utilizam a mesma tecnologia — hardware, sistema operacional, protocolos de comunicação, linguagens

de programação, etc.), *autonomia* (cada componente do sistema é independente dos demais), *evolução* (durante seu funcionamento o sistema pode incorporar novas tecnologias) e *mobilidade* (as fontes de informação, nós de processamento e usuários podem se mover fisicamente).

Certamente a especificação de um sistema distribuído não é trivial e envolve muita informação. Para lidar com essa grande quantidade de informação, ODP divide a fase de especificação do sistema na construção de cinco modelos definidos a partir de **pontos de vista**¹: *Empresa*

(especifica o sistema em relação aos negócios da empresa, tendo como foco o propósito, escopo e políticas do sistema), *informação* (especifica o significado das informações sendo processadas), *computação* (especifica a distribuição funcional dos objetos e suas interfaces), *engenharia* (especifica os mecanismos e funções necessários para a interação dos objetos distribuídos) e *tecnologia* (especifica as tecnologias utilizadas no sistema).

Na visão de engenharia, o projetista especifica quais os mecanismos que irão implementar as transparências *de acesso* (mascara diferentes tipos de dados e formas de invocação de método entre objetos distribuídos), *de falha* (mascara a eventual falha e recuperação de um objeto), *de localização* (mascara a localização física de uma interface), *de migração* (mascara a troca de localização física de um objeto), *de realocação* (mascara a troca de uma interface por outra compatível), *de replicação* (mascara a troca de um objeto por outro que implementa a mesma interface) e *de persistência* (mascara a eventual desalocação e realocação de um objeto). Para o desenvolvimento de um Sistema Multi-Agente (SMA) justamente estes mecanismos são fundamentais e, por este motivo, somente os conceitos da visão de engenharia serão considerados neste trabalho.

2.1 Visão de Engenharia

No nível mais geral, os objetos estão fisicamente associados a um recurso de processamento, o *nó* (um computador, por exemplo). Cada nó é controlado por um *núcleo* (um sistema operacional, por exemplo) que é responsável pela inicialização do nó, pela criação de grupos de objetos, pelo suporte a comunicação, etc. Ainda no nó, há *cápsulas* que possuem memória e compartilham o processador do nó (um processo, por exemplo). Na cápsula há um objeto especial, chamado gerenciador, que a controla (cf. Fig. 1) Agrupam-

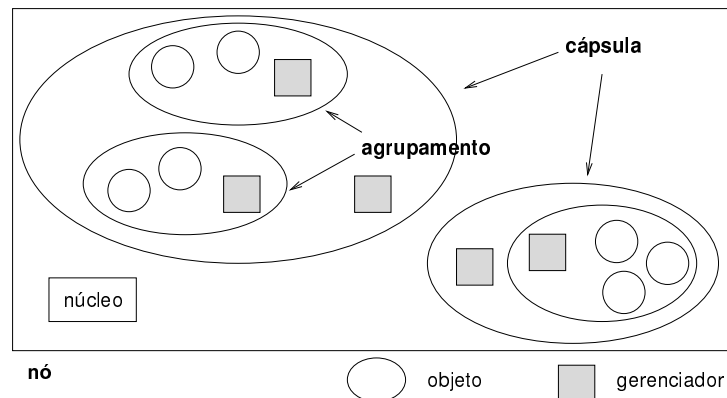


Figura 1: Organização de um nó ODP

¹ Assim como outras técnicas que envolvem vários modelos, Reference Model of Open Distributed Processing (RM ODP) também propõe um conjunto de regras que mantem a consistência entre os me-

se objetos em cápsulas para facilitar a comunicação entre eles, dispensando a utilização da comunicação entre processos oferecida pelo nó.

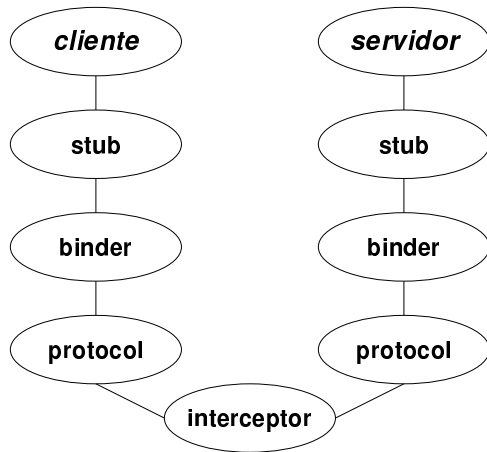


Figura 2: Canais de comunicação

Cada cápsula pode conter vários objetos formando *agrupamentos*. Os objetos de um agrupamento formam uma unidade que pode ser transferida, desativada, reativada e movida para outro nó. A comunicação dos objetos dentro do agrupamento é muito eficiente, já que os objetos são criados juntos e desenvolvidos na mesma linguagem, i.e., essa comunicação normalmente é uma invocação de método. Assim como o nó e a cápsula, cada agrupamento é controlado por um gerenciador.

Para que objetos de diferentes agrupamentos possam interagir com as transparências mencionadas, a recomendação sugere uma estrutura com as seguintes camadas (cf. Fig. 2):² **stubs** (responsável pela informação transmitida na interação); **binder** (responsável por manter a associação entre os objetos de um canal); e **protocolo** (responsável pela manutenção da comunicação e, caso seja necessário, pela resolução de problemas de nomes — serviço de diretório). Caso a comunicação seja com objetos de outra organização e/ou tecnologia (outro tipo de rede, por exemplo), há necessidade de adaptações adicionais, feitas por um objeto interceptador.

3 Inteligência Artificial Distribuída

Um **sistema em IAD** é formado pelos seguintes elementos [1]: *Agentes* (entidades ativas do sistema), *Sociedade* (o conjunto formado pelos agentes), *Ambiente* (o conjunto formado pelas entidades passivas (objetos) do sistema), *Interação* (trocas de informação entre os agentes da sociedade, podem ser tanto trocas diretas – por comunicação explícita – quanto indiretas – por meio do ambiente), e *Organização* (restrições sociais aplicadas aos agentes, é o meio que o projetista utiliza para garantir que os agentes autônomos farão aquilo para o qual foram construídos).

3.1 Tipos de Agentes

Os agentes são classificados em dois grupos: agentes reativos e agentes cognitivos. Os agentes reativos tem um comportamento muito simples: reagem “instintivamente” às percepções que tem do ambiente (cf. Fig. 3). Os agentes reativos apresentam as seguintes propriedades: Somente possuem representação de conhecimento implícita no código, ou

² Mesmo que os objetos estejam na mesma cápsula ou nó, um mecanismo é necessário para lidar com problemas de falha, movimentação, etc.

seja, não tem representação do ambiente (objetos e outros agentes); não tem história (fatos que aconteceram e ações que executou); não tem controle deliberativo (planejado) de suas ações; formam organizações do tipo etológico; e as sociedades são formadas por muitos agentes. Em geral, neste tipo de sociedade, o interesse está voltado para o comportamento global que emerge da interação de um grande grupo de agentes.

Os agentes cognitivos possuem um *estado mental*³ e funcionam racionalmente, i.e., raciocinam para construir um plano de ações que leva ao objetivo pretendido (cf. Fig. 3). Portanto, apresentam características particulares que os diferenciam de programas convencionais e dos agentes reativos, dentre elas, pode-se citar: tem autonomia funcional (podem alterar seu funcionamento a fim de adaptarem-se melhor ao seu ambiente [11]); estão continuamente em funcionamento; são sociáveis (possuem a capacidade de comunicação e de modelagem dos outros); possuem representação de conhecimento explícita no código (conhecimento introspectivo); o mecanismo de controle é deliberativo, ou seja, o agente raciocina sobre que ações realizar; tem história; e normalmente, as sociedades são formadas por poucos agentes.

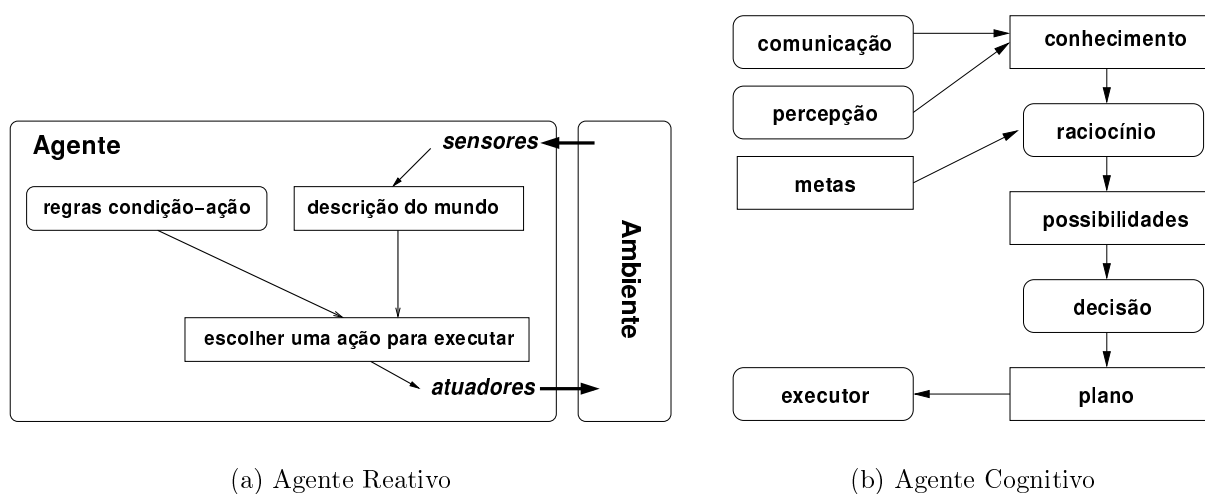


Figura 3: Arquiteturas de funcionamento para agentes. Em (a) tem-se a arquitetura proposta por Russel [11]: O comportamento do agente é determinado unicamente pela percepção do ambiente e por um conjunto de regras. Em (b) tem-se a arquitetura proposta por Demazeau [3]: O conhecimento que o agente possui é formado a partir da sua percepção do ambiente e da comunicação com outros agentes. Dado este conhecimento e uma meta, o agente gera um conjunto de possíveis planos que atingem esta meta. Dadas estas possibilidades, o agente escolhe o melhor plano a ser executado.

³ O estado mental de um agente é definido em vários trabalhos com base nos conceitos lógicos de crenças, desejos e intenções [2].

3.2 Comunicação entre Agentes

Assim como na história da espécie humana a evolução da linguagem teve um papel essencial em SMA. A comunicação entre agentes é um dos seus temas, pois viabiliza a troca de informações e a coordenação de tarefas entre agentes

A maioria dos trabalhos na área de interação entre agentes cognitivos se baseiam na *teoria dos atos de fala* [12]. Essa teoria, inicialmente desenvolvida para modelar a comunicação entre humanos, é utilizada em SMA por representar a interação como troca de conhecimento entre os agentes⁴. Nas aplicações em SMA, as mensagens possuem uma intenção (força ilocucionária) e um conteúdo (força locucionária). Por exemplo a mensagem *tell(A, B, p)* significa que o agente *A* está informando ao agente *B* que ele (*A*) acredita em *p*, a intenção da mensagem é informar e o seu conteúdo é *p*.

Para a implementação de um mecanismo de comunicação entre agentes Mayfield et al. [10] propõem algumas exigências básicas para a linguagem:

1. Quanto a sua forma, deve ser declarativa, sintaticamente simples e legível por pessoas.
2. Quanto ao conteúdo das mensagens, a implementação deve fazer distinção entre conceitos da linguagem de comunicação que expressam os atos de fala (*linguagem externa*) e a linguagem que é usada para expressar os fatos sobre o domínio da aplicação (*linguagem interna*). O objetivo é que a linguagem externa possa encapsular qualquer sentença da linguagem interna.
3. Linguagens de comunicação entre agentes devem ser eficientes.
4. Uma linguagem de comunicação deve prover meios para que os agentes se comuniquem de maneira segura, deve garantir o isolamento e a integridade de dados, deve permitir a autenticação de outros agentes e deve prover mecanismos de troca de dados confidenciais.
5. Uma linguagem de comunicação entre agentes deve-se ajustar bem com a arquitetura da rede. A linguagem deve aceitar tipos diferentes de conexões.

Uma proposta de atingir as características enumeradas acima é a Knowledge Query and Manipulation Language (KQML). KQML é uma linguagem e um protocolo para a comunicação entre agentes, visando suportar a troca de informação e conhecimento

⁴Um ato de fala possui três componentes: força locucionária (as palavras e sentenças e seu significado); força ilocucionária (a intenção da mensagem, por exemplo: *inform*, *ask-to-do*, *answer*, *promise*, etc.); e uma força perlocucionária (o resultado esperado no receptor pela emissão da mensagem, por exemplo: *convince*, *insult*, etc.)

entre os agentes [9, 5]⁵. As mensagens KQML podem ser dividida em três camadas: *camada de conteúdo* (possui a mensagem que pode ser em qualquer linguagem de representação), *camada de comunicação* (codifica um conjunto de características de alto nível, tais como a identidade de quem envia e de quem recebe a mensagem), *camada de mensagem* (forma o núcleo da linguagem e determina o tipo de interação entre os agentes). A função primária desta camada é identificar o protocolo usado para transmitir a mensagem e a intenção (performativa) do emissor. A intenção pode ser uma declaração, uma consulta, um comando ou uma outra primitiva dentro de um conjunto de intenções conhecidas). O formato das mensagens KQML é baseada na sintaxe do LISP:

```
(performative
  :language  word
  :ontology  word
  :sender     word
  :receiver  word
  :reply-with word
  :content   expression
  ...)
```

} camada de mensagem
} camada de comunicação
} camada de conteúdo

4 De Objetos para Agentes

Como tanto um objeto como um agente podem apresentar o mesmo comportamento, a comparação entre ambos não será feita sob um enfoque externo, mas irá considerar a estrutura interna de objetos e agentes. Para estes últimos, como há inúmeras arquiteturas de funcionamento, serão consideradas aquelas apresentadas na seção 3. Os componentes internos considerados são: a memória e o funcionamento.

4.1 Comparação entre Agente Reativo e Objeto

Quanto à memória, um agente reativo é um elemento de software mais simples que um objeto pois não possui estado, toda ação é realizada unicamente em função da observação do mundo.

Quanto ao funcionamento, ambos são similares. Assim como o agente reage a uma percepção, um objeto sempre responde a uma mensagem. A resposta do objeto é dada em função dos parâmetros da mensagem e de seu estado enquanto a resposta do agente é função apenas dos parâmetros observados. Entretanto, o agente possui um fluxo de execução próprio (o ciclo: perceber, escolher, agir) enquanto um objeto não tem fluxo de execução pré-definido (num sistema orientado a objetos, o programa tem um fluxo de execução e não os objetos).

⁵KQML é projetado para ser usado com vários mecanismos de transporte (atualmente há implementações que usam TCP/IP, SMTP (email), HTTP e CORBA). O agente de KQML pode falar diretamente a outro agente ou pode enviar mensagens a múltiplos agentes do mesmo grupo.

Resumidamente, um objeto tem controle sobre seu estado mas não tem *autonomia* sobre seu comportamento, se um método for invocado ele é executado [8].

4.2 Comparação entre Agente Cognitivo e Objeto

Quanto à memória, um agente cognitivo tem sua memória representada por estados mentais, enquanto o objeto tem sua memória representada por pares atributo-valor. Os estados mentais têm significado para o agente, já os atributos não têm significado para o objeto (obviamente têm significado para o programador). Essa diferença permite ao agente raciocinar sobre seu estado mental e encontrar soluções (planejar ações) para problemas novos que eventualmente não foram considerados no seu projeto e ter um comportamento pró-ativo a partir do estado mental (*meta*).⁶

Quanto ao funcionamento, além das diferenças já apontadas na seção 4.1, um agente cognitivo tem seus próprios objetivos e seu comportamento é resultado do plano que traçou para alcançá-los. Quanto à parte do funcionamento referente a interação com outros elementos, também há diferenças interessantes:

- Enquanto objetos *invocam* métodos de outros objetos, agentes *requisitam* serviços de outros agentes, estes podem ou não atender a requisição dependendo da organização da sociedade (agentes não são necessariamente benevolentes).
- Os agentes utilizam uma linguagem adequada para comunicação (KQML, por exemplo). Há diferença entre a linguagem na qual o agente foi desenvolvido e a linguagem com que irá se comunicar. Um agente recebe a mensagem e a *interpreta*⁷. Esse fator é determinante para a interoperabilidade dos agentes.
- Considerando que os agentes se comunicam utilizando um protocolo de comunicação (regras para um diálogo), a resposta de uma mensagem é função dos parâmetros da mensagem, do estado mental do agente e do estado do protocolo. Na comunicação entre objetos não existe a noção de protocolo.

⁶ É claro que os estados mentais podem ser construídos a partir de atributos-valor, mas a definição de objeto não restringe a organização de sua memória ao formato de estados mentais. Este mesmo argumento é utilizado pelos usuário de OO quando lhe perguntam: mas isso não pode ser feito com programação estruturada? Ou seja, OO pode ser visto como um conjunto de restrições sobre programas estruturados (lhe impondo uma estrutura de funcionamento, por exemplo: só se acessa os atributos por meio de métodos) e agentes impõem restrições sobre a organização da memória e do comportamento de um objeto.

⁷ Assim como acontece com os estados mentais, as mensagens tem significado para o agente o que lhe permite interpretá-las. Um objeto não interpreta a mensagem, apenas desvia o fluxo de execução para um determinado endereço de memória, justamente porque a linguagem de comunicação é uma linguagem de programação. Portanto, para os objetos as mensagem precisam ter um formato rígido onde se deve especificar exatamente o número e o tipo dos parâmetros, tanto no projeto da interface do objeto quanto na sua utilização (chamada do método).

4.3 Utilização de Objetos na Construção de Agentes

Assim como um programa orientado a objetos pode ser feito numa linguagem não OO (mesmo sendo contra-producente), um agente também pode ser construído a partir de objetos, restringindo-se sua estrutura de memória e funcionamento.

No caso da construção de agentes reativos, o paradigma de Orientação a Objetos (OO) é uma alternativa tão boa quanto a programação estruturada. Entretanto devem-se considerar algumas restrições no desenvolvimento de um agente reativo utilizando programação OO: o objeto deve ter um fluxo de execução próprio (ser uma *thread*); ter um “mailbox” para receber mensagens; e ter uma tabela para as regras condição-ação. O comportamento do agente seria definido pela tabela condição-ação, pois o programa seria o mesmo para qualquer agente (perceber-escolher-agir).

No caso da construção de um agente cognitivo a partir de OO, a quantidade de rotinas que deveriam ser implementadas é muito maior, por exemplo: representação de conhecimento (estados mentais), mecanismo de inferência, aprendizado, etc. Para uma boa parte destes mecanismos, a IA tradicional já oferece soluções.

Entretanto, para tirar vantagem da abordagem de SMA, não é salutar colocar muitas restrições sobre o funcionamento interno do agente. A escolha de uma linguagem de programação e de uma arquitetura deve ser a mais conveniente para a construção do agente. Para o SMA como um todo estas escolhas devem ser irrelevantes.

5 De Objetos Distribuídos para Sistemas Multi-Agentes

A primeira característica em comum entre um Sistema de Objetos Distribuídos (SOD) na visão ODP e um SMA é o contexto onde o sistema funcionará, pois em ambos o sistema é desenvolvido considerando as características de um sistema distribuído. As características e transparências que ODP procura dar ao sistema também são desejáveis em SMA. Logo, certamente ODP tem contribuições a oferecer na definição de ambientes para SMA.

5.1 SMA como um SOD

Uma sociedade de agentes pode muito bem ser vista como um conjunto de cápsulas distribuídas em vários nós e um agente como um agrupamento de objetos (veja, por exemplo, a Fig. 4). Neste contexto, os gerenciadores têm suas funções especializadas: o núcleo do nó tem a função de criar *sociedades de agentes*; o gerenciador de cápsula tem a função de criar, desativar, mover, etc. agentes para a sociedade que representa; e o gerenciador de agrupamento deixa de existir e passa a fazer parte da estrutura interna dos agentes que, para o seu funcionamento, pode criar objetos (cf. Tab. 1).

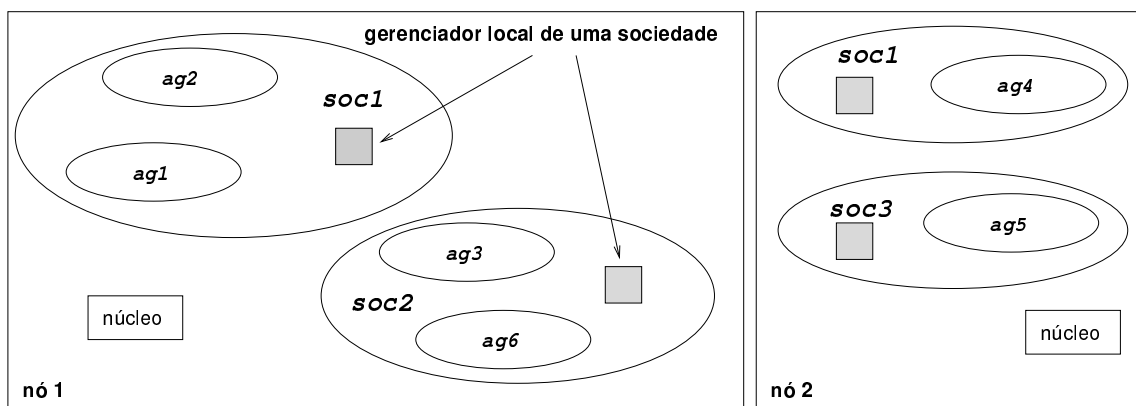


Figura 4: Ambiente Multi-Agentes na visão engenharia do ODP. Três sociedades (*soc1*, *soc2* e *soc3*) distribuídas em dois nós. A sociedade *soc1* é formada por três agentes (*ag1*, *ag2* e *ag4*), dois no nó 1 e um no nó 2. A *soc2* é formada por dois agentes (*ag3* e *ag6*) alocados no nó 1. A *soc3* é formada por um único agente (*ag5*) alocado no nó 2.

5.2 Comunicação entre Agentes e Canais entre Objetos Distribuídos

A transparência dada pela recomendação ODP é, em grande parte, gerada pela noção de canais de comunicação. É interessante observar a correspondência entre que as especificações da comunicação entre agentes (KQML, por exemplo) e a arquitetura em camadas do ODP: o *stub* corresponde a camada de mensagem da KQML⁸; o *binder* corresponde a camada de comunicação; e o protocolo corresponde a camada de mensagem, com muitas das duas funções sendo desempenhadas pelo roteador e facilitador KQML.

A figura do interceptador também existe. Na comunicação entre agentes três situações distintas podem ocorrer: (i) ambos serem da mesma sociedade e terem a mesma linguagem de comunicação (Agent Communication Language (ACL)); (ii) ambos serem de sociedades diferentes e terem a mesma ACL; e (iii) serem de sociedades diferentes e terem ACL diferentes. Nos casos (i) e (ii), as camadas stub-binder-protocolo são suficientes para concluir a comunicação. Mas no caso (iii), o interceptador precisa fazer as traduções necessárias entre as ACLs das duas sociedades.

5.3 Resultados

O primeiro benefício de adotar um sistema que segue o RM-ODP como suporte na construção de um ambiente para SMA é que o SMA terá as propriedades dos SODs: abertura⁹, integração, flexibilidade, modularidade, federação, gerenciabilidade, qualidade, segurança

⁸A transparência de acesso em KQML é praticamente desnecessária, já que KQML independe de linguagem de programação. Apenas o conteúdo das mensagens eventualmente devem ter uma adaptação (no envio de uma imagem, por exemplo).

⁹O tipo de abertura que o RM-ODP garante é baseada na utilização de padrões. Entretanto, a padronização é uma restrição muito forte para os agentes. Neste caso, a abertura é garantida pela capacidade de adaptação e aprendizado dos agentes.

e transparência. O segundo benefício é que os agentes terão as propriedades dos agrupamentos e poderão ser transferidos, desativados, reativados e movidos para outro nó.

Se um SMA realiza a comunicação entre os agentes baseado, por exemplo, em implementações da especificação CORBA (que segue, em parte, as recomendação ODP) garante-se a transparência (de falha, localização, migração, realocação e replicação) na comunicação entre agentes e atende-se os requisitos 4 e 5

Conceitos ODP	SMA como SOD
Componentes	
nó	nó
cápsulas	grupo de agentes de uma mesma sociedade
agrupamento	agente
objeto	deixa de ser considerado
Gerenciadores	
de nó (núcleo)	inicia gerenciadores de sociedade
de cápsula	realiza as funções locais de uma sociedade
de agrupamento	passa a fazer parte da arquitetura interna do agente

Tabela 1: Concepção de um SMA como umSOD

de uma ACL (cf. visto na Sec. 3.2). A concepção de um SMA seguindo as recomendações ODP foi utilizada no desenvolvimento da ferramenta SACI, apresentando bons resultados [6].

O mesmo resultado alcançado na comparação entre objetos e agentes vale para o nível sistêmico: um SOD pode ser usado como base para a construção de um SMA a partir dele, porém não é suficiente. Certos aspectos próprios dos SMA, que não existem em SOD, devem ser adicionados, por exemplo: o conceito de organização, papéis sociais dos agentes, definição de uma ACL, etc.

6 Conclusões e Trabalhos Futuros

A princípio, poderia-se inferir que ambas as áreas (ODP e SMA) tinham o mesmo objetivo. Entretanto, constatou-se que atuam em níveis de abstração diferentes: objetos/agentes, dados/conhecimento, interação por invocação/interação por meio de uma linguagem de alto nível.

Contudo não há somente diferenças. Como principal resultado deste trabalho, tem-se a constatação de que muitas das propriedades dos sistemas que a IAD procura abordar são comuns aos SOD. Portanto, muitos dos trabalhos realizados na área de SOD podem (e devem) ser utilizados na construção de SMA. O projetista do SMA teria preocupação com os aspectos que realmente envolvem a questão da emergência da inteligência, ignorando os aspectos de distribuição. Portanto, o nível de abstração atendido pelo ODP pode muito bem servir de suporte para o nível dos SMA. Aliás, tal sinergia é recomendável para que pesquisadores de duas área não dispensem esforços na solução de um mesmo problema.

Assim como a arquitetura de objetos distribuídos pôde contribuir na implementação de SMA, sugere-se como tema de novos trabalhos a verificação das contribuições que

as metodologias de desenvolvimento OO e SOD podem fornecer na definição de uma metodologia para desenvolvimento de sistemas baseados em agentes, que no momento inexistente [14].

Referências

- [1] ALVARES, L. O., SICHMAN, J. S. Introdução aos sistemas multiagentes. In: MEDEIROS, C. M. B. (Ed.) *Jornada de Atualização em Informática*. Brasília: SBC, agosto 1997. v. 16, Cap. 1, p. 1ss.
- [2] COHEN, P. R., LEVESQUE, H. J. Intention = choice + commitment. In: National Conference on Artificial Intelligence, 6., 1987, Seattle, WA. *Proceedings...* Morgan Kaufmann, c1987.
- [3] DEMAIZEAU, Y., MÜLLER, J.-P. (Eds.). *Decentralized artificial intelligence*. Amsterdam: Elsevier, 1990.
- [4] GASSER, L. Social conceptions of knowledge and action: DAI foundations and open systems semantics. *Artificial Intelligence*, v. 47, n. 1-3, p. 107-138, 1991.
- [5] HÜBNER, J. F., COELHO, A. R. Avaliação da comunicação entre agentes utilizando KQML. In: Seminário de Computação, 7., 1998, Blumenau. *Anais...* Blumenau: FURB/DSC, c1998. p. 11-22.
- [6] HÜBNER, J. F., SICHMAN, J. S. SACI: Uma ferramenta para implementação e monitoração da comunicação entre agentes. Submitted to SBIA'2000 (<http://www.lti.pcs.usp.br/saci>).
- [7] ISO. *Reference model of open distributed processing*, 1995.
- [8] JENNINGS, N. R., WOOLDRIDGE, M. J. *Agent technology: foundations, applications, and markets*. London: Springer Verlag, 1998.
- [9] LABROU, Y., FININ, T. *A proposal for a new kqml specification*. UMBC, Baltimore, 1997.
- [10] MAYFIELD, J., LABROU, Y., FININ, T. Desiderata for agent communication languages. In: AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments, 1995, Stanford, CA. *Proceedings...* c1995.
- [11] RUSSEL, S., NORVIG, P. *Artificial intelligence: a modern approach*. New Jersey: Prentice-Hall, 1995.
- [12] SEARLE, J. R., VANDERVEKEN, D. *Foundations of illocutionary logic*. Cambridge: Cambridge University Press, 1985.
- [13] WEISS, G. (Ed.). *Multiagent systems: A modern approach to distributed artificial intelligence*. London: MIT Press, 1999.
- [14] WOOLDRIDGE, M., JENNINGS, N. R. Pitfalls of agent-oriented development. In: International Conference on Autonomous Agents, 2., 1998, Minneapolis, St. Paul. *Proceedings...* Editors SYCARA, K. P., WOOLDRIDGE, M. ACM, c1998.