UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

ROOSVELT VIEIRA DE OLIVEIRA

RECUPERAÇÃO DE INFORMAÇÃO EM UM AMBIENTE DE APRENDIZAGEM: Uma Abordagem Multiagente

São Luís
2005

ROOSEVELT VIEIRA DE OLIVEIRA

RECUPERAÇÃO DE INFORMAÇÃO EM UM AMBIENTE DE APRENDIZAGEM: Uma Abordagem Multiagente

Dissertação submetida à Coordenação do programa de Pós-Graduação em Engenharia de Eletricidade da UFMA, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Sofiane Labidi

São Luís
2005

Oliveira, Roosevelt Vieira
Recuperação de Informação em um ambiente de Aprendizagem: Uma Abordagem Multiagente / Roosevelt Vieira de Oliveira. - São Luís, 2005.

91f. il.

Mestrado (Mestrado em Ciência da Computação) – Universidade Federal DO MARANHÃO, 2005.

1. Recuperação de Informação Em Um Ambiente De Aprendizagem: Uma Abordagem Multiagente

CDU: 004.89:37

ROOSEVELT VIEIRA DE OLIVEIRA

RECUPERAÇÃO DE INFORMAÇÃO EM UM AMBIENTE DE APRENDIZAGEM: **UMA ABORDAGEM MULTIAGENTE**

Dissertação submetida à Coordenação do programa de Pós-Graduação em Engenharia de Eletricidade da UFMA, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Aprovada em 12 / 04 /2006

BANCA EXAMINADORA

Prof. Dr. Sofiane Labidi (Orientador)
Universidade Federal do Maranhão (UFMA)

Prof. Dr. Rubens Nascimento Melo (Examinador)
Pontifica Universidade Católica do Rio de Janeiro (PUC-Rio)

Prof. Dr. Renato da Veiga Guadagnin (Examinador)
Universidade Católica de Brasília (UCB – Brasília)

Prof. PhD Zair Adbdelouahab (Examinador)
Universidade Federal do Maranhão (UFMA)

Aos meus filhos Daniel, Davi e Roosevelt Jr;

Aos meus pais Francisco e Eronides;

À minha família;

AGRADECIMENTOS

- **A DEUS,** por tudo de bom que ele tem proporcionado à minha vida;
- Ao Prof. Dr. Sofiane Labidi, pelo apoio, incentivo, orientação firme e confiança sempre manifestada em nosso relacionamento;
- Ao Prof. Dr.Edson Nascimento, pela firmeza, sugestões e confiança;
- Aos meus filhos Daniel, Davi e Roosevelt Junior pela força que nos une;
- Aos meus pais Francisco e Eronides Oliveira pela pessoa que sou;
- Aos professores e funcionários do CPGEE.
- Aos colegas que fazem parte do meu contexto social.
- A todos que contribuíram de forma direta e indireta, Gentil, Walker, dentre outros.

"(...) pelo fato de que nada é por natureza um nome, mas somente quando ele se torna símbolo, pois nem mesmo quando sons articulados como o dos animais significam alguma coisa qualquer um deles constitui um nome."

Aristóteles

RESUMO

Neste trabalho, propõe-se um sistema de múltiplos agentes para recuperação de informações em base de dados não semânticos, como por exemplo, a Web. Esses agentes de software têm como objetivo auxiliar o processo de ensino-aprendizagem em ambientes de aprendizagem computadorizada, no que diz respeito à recuperação de informações. Para isso, utilizam-se ontologias para facilitar a interoperabilidade entre os agentes no desempenho de suas tarefas e na contextualização do assunto a ser pesquisado na Web. Faz-se a especificação dos agentes de busca e apresenta-se a implementação de alguns destes, exemplificados através de seu uso no Ambiente de Aprendizagem Netclass.

Palavras-Chave: *Recuperação de Informação, Multiagentes, Ontologia, Educação a Distância.*

ABSTRACT

This work proposes a system of multiple agents for the recovery of information in base of non semantic data, as for example, the Web. Those software agents have as auxiliary objective the teaching-learning process in computer-based learning environment, in what it refers to the retrieval information. For that, ontologies are used to facilitate the intercommunication among the agents while acting their tasks and during the specification of the domain to be researched in the Web. The agents specification is made and the implementation of some of these agents is presented, been exemplified through its use in the Computer-based Learning Environment Netclass.

Keywords: Information Retrieval, Multiagents, Ontology, Distance Learning.

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

Uma ontologia visa desenvolver um conjunto de regras que possibilitam a inferência de forma que a máquina possa, através do acesso a essas regras e a uma coleção de dados e metadados, abstrair um significado semântico das informações disponibilizadas. A vantagem da utilização de uma ontologia é de se lidar com conceitos, representando-os formalmente, e de se livrar de problemas inerentes ao vocabulário da linguagem natural (Bézivin, 1998). Ontologias que facilitam o compartilhamento e reuso de conhecimentos, podem se tornar uma solução para minimizar as dificuldades com o processo de recuperação de informação (Harmelen et al., 1999). Podem agir como um *link* entre os usuários e informação tanto quanto fornecer os conceitos e relações (representação semântica explícita de conhecimento) para usuários a fim de formar e refinar as consultas consistentes. Diante desse contexto, o uso de agentes artificiais tornar-se um indispensável componente de software na recuperação inteligente da informação através do uso de ontologias.

Utiliza-se o projeto NetClass para exemplificar o uso de ontologias em ambientes de aprendizagem na recuperação de informação em bases de dados não semânticos, como por exemplo, a Web. O objetivo do NetClass é definir, projetar e implementar um ambiente computadorizado para um eficiente processo de aprendizagem cooperativa (Labidi et al., 2003a). O Ambiente NetClass une as idéias que fundamentam os Sistemas Tutores Inteligentes aos recursos de uma rede e ao paradigma de aprendizagem cooperativa, definindo um Ambiente de Ensino Inteligente Cooperativo Computadorizado.

1.1 Contexto

Este trabalho situa-se na área de Sistemas Tutores Inteligentes Cooperativos e Recuperação de Informação, e faz parte do Projeto NetClass (Universidade Federal do Maranhão – UFMA). Este projeto tem por objetivo utilizar conceitos de Inteligência Artificial na Educação a fim de desenvolver um STI Cooperativo Computadorizado, onde o processo de ensino-aprendizagem é realizado através da interação entre grupos de alunos, professor e sistema.

Este trabalho trata especificamente do uso de ontologias e agentes artificiais na recuperação de informações na Web. Onde os dados recuperados apresentam-se ao aprendiz na forma de uma lista de URLs pertinentes ao assunto estudado.

1.2 Objetivos

O objetivo deste trabalho é especificar e implementar Agentes de Busca que podem ser utilizados no NetClass, no intuito de recuperar informações na Web (base de dados não semânticos).

Especificamente, pretende-se: 1) descrever os conceitos inerentes à semântica de dados; 2) estabelecer direções gerais para a recuperação de informações; 3) especificar a estrutura dos agentes responsáveis, no NetClass, pela busca de informações na Web; e 4) implementar um protótipo de Agentes de Busca que podem ser utilizados no NetClass.

1.3 Justificativa e Relevância

Representar conhecimentos é o cerne de um Ambiente de Ensino Aprendizagem (Silva, 1999). Mas, nem sempre o conteúdo da base de conhecimentos do sistema contempla todas as necessidades de aprendizagem do aprendiz. Considerando que a Internet possui um número extremamente grande de informações que podem ser utilizadas em um ambiente de aprendizagem, a recuperação de informações da Web torna-se essencial para o bom andamento do processo de ensino-aprendizagem em ambientes computadorizados.

Apesar da Web possuir uma enorme quantidade de documentos didáticos, a menos que saibamos exatamente "*o que*" se quer recuperar, de "*onde*" e "*como*", o crescimento da Web só tende fazer aumentar a dificuldade para se recuperar informação relevante em um tempo razoável. Pois, buscar informação na web torna-se uma tarefa difícil pelos seguintes fatores: 1) falta de estrutura e organização das informações; 2) as informações na Web têm um caráter altamente volátil; 3) o domínio de pesquisa é muito amplo; 4) é muito alta a incidência de informações redundantes; 5) natureza distribuída da Web; 6) é muito difícil garantir a integridade/veracidade das informações publicadas na web; e 7) natureza heterogênea das informações, como por exemplo as informações podem estar disponíveis em diferentes tipos e

formatos e também diferentes línguas ou alfabetos. Portanto, o grande desafio das pesquisas em RI na Web, consiste em se tentar obter eficácia de recuperação (Kobayashi et al., 1999).

Pelo fato de que as informações representadas na Web, em quase totalidade, não possuem caráter semântico, o uso destas em um ambiente de aprendizagem torna-se objeto de estudo, com a finalidade de se identificar tecnologias que possam ser utilizadas no tratamento dessas informações. A intenção é manter-se regras que possibilitem a inferência de forma que o software possa abstrair um significado semântico das informações recuperadas.

1.4 Organização da dissertação

Esta dissertação está organizada em oito capítulos. No próximo capítulo, faz-se uma revisão bibliográfica sobre ontologias, apresentando-se ferramentas que foram utilizadas para a criação das ontologias.

No capítulo 3, descrevem-se os conceitos envolvidos no processo de recuperação de informações. Apresentam-se quais são as principais abordagens descritas na literatura, dando-se enfoque à recuperação de informações nos Modelos da Web.

No capítulo 4, apresentam-se conceitos relacionados aos agentes artificiais, enfatizando-se a possibilidade de se ter arquiteturas com múltiplos agentes inteligentes, ou seja, sociedades de agentes.

No capítulo 5, apresenta-se o Ambiente NetClass de Ensino Inteligente Cooperativo Computadorizado, destaca-se sua arquitetura multiagentes, as interações possíveis entre seus agentes, a organização de seu modelo do domínio, os tipos de atividades pedagógicas possíveis de serem realizadas, as estratégias pedagógicas que podem ser adotadas no processo de ensino-aprendizagem e ainda, o processo de avaliação dos aprendizes nesse ambiente de aprendizagem.

A especificação dos Agentes de Busca NetClass encontra-se no Capítulo 6. Nesse capítulo, mostra-se quais classes compõem esses agentes e apresentam-se as responsabilidades de cada uma delas, face ao processo de recuperação de informações. A partir dessa especificação, no capítulo seguinte (Capítulo 7), apresenta-se um protótipo de

implementação de dois Agentes de Busca, com finalidade de mostrar a viabilidade do uso de agentes na recuperação de informações, conforme descrito no decorrer deste trabalho.

No Capítulo 8, apresentam-se as conclusões deste trabalho e perspectivas futuras.

2 ONTOLOGIAS

Este capítulo visa, apresentar princípios, conceitos fundamentais, componentes e técnicas de construção e manutenção de ontologias.

2.1 Níveis de Especificação de Sistemas Baseados em Conhecimento

A tentativa de criar sistemas diretamente a partir do conhecimento acerca dos processos, ou seja, declarativamente, tornando possível aos computadores realizar dedução automática deu origem à área chamada hoje de Inteligência Artificial Simbólica. Fundamenta-se basicamente na separação entre o conhecimento e o processo dedutivo ou inferência. A concepção de sistema dessa natureza passa por três especificações consecutivas (Newell, 1982):

i) O nível de conhecimento ou epistemológico, que consiste numa especificação abstrata do conhecimento do domínio ou problema que se deseja modelar;

ii) O nível lógico, que converte as especificações de conhecimento em sentenças de lógica formal;

iii) O nível de implementação, que codifica estas sentenças em linguagem computacional, seja numa linguagem ou num banco de dados dedutivo, dito base de conhecimento.

Marcadamente, a existência do nível de conhecimento é de suma importância para o desenvolvimento de sistemas simbólicos, principalmente por uma razão, que se confunde com a própria história da Inteligência Artificial Simbólica. A motivação principal do paradigma declarativo era modelar sistemas num nível mais elevado. O nível de conhecimento guarda uma relação mais próxima e direta com o conhecimento sobre o domínio a ser modelado. O modelo declarativo como um todo consiste numa forma mais flexível de se descrever um domínio, uma vez que, no nível de conhecimento ainda não há compromisso com a implementação. A motivação de evitar escrever código e tentar abstraí-lo colocando o conhecimento do lado de fora dos sistemas é extremamente legítima: a maior parte das vantagens do paradigma declarativo – como flexibilidade, legibilidade, fidelidade semântica,

engajamento ontológico, extensibilidade, reuso, abstração das compilações, portabilidade e capacidade de inferência - já era visionada desde essa época, e corroboraram essa idéia. Ainda não se enxergava, contudo, o potencial do uso do conhecimento por entidades de software, que as ontologias viriam a proporcionar, organizando o conhecimento em conceitos e domínios, e nem que elas implementariam o nível de conhecimento apropriadamente, em tese, sem refletir algum formalismo de representação de conhecimento específico. Saliente-se que esta prática traz em si a possibilidade do conhecimento ser portável, enquanto conhecimento e não enquanto código, e, portanto, independente de máquina.

Cabe lembrar um último argumento favorável às soluções declarativas, que só pôde ser percebido em sua plenitude recentemente, à luz das grandes bases de conhecimento estruturadas, ou ontologias: o engajamento ontológico ou de conhecimento, ou seja, a semântica das sentenças codificadas em lógica guarda uma relação mais direta com o domínio modelado e permite o reuso de um grande volume de informação que desempenha o papel de conhecimento estruturado, disponível para reuso em larga escala por sistemas e programas.

2.2 Formalismos de Representação de Conhecimento Declarativo

Essa idéia de manter fatos e conhecimento do lado de fora dos programas começou a ser fomentada por um sistema em 1958, o Advice Taker (McCarthy 1958), e sedimentou-se com o uso massivo da linguagem Lisp na representação de fatos e regras em sistemas especialistas. Em vez de colocarem-se os fatos dentro das regras, percebeu-se que as regras poderiam ser mais genéricas se os fatos estivessem separados. Com isso, o conceito de processo, ao invés de significar uma ação, passou a denotar muito mais uma decisão de ação a ser tomada – no caso de Lisp, através de casamento de padrões, um primeiro esboço de método de dedução, só que ainda sem a presença de lógica formal.

LISP (List Processing, em português Processamento de Listas), uma linguagem funcional voltada para o manejo de listas, foi criada justamente com o propósito de representar conhecimento de forma mais abstrata.

Em meados dos anos 70, começaram a aparecer os sistemas de inferência orientados a padrões (Hayes-Roth et al 1978), que incorporam a programação relacional ou lógica. Estes sistemas são compostos basicamente de três entidades:

i) Um formalismo fundamentado em lógica matemática, no qual o código do sistema baseado em conhecimento será escrito;

ii) Um método ou estratégia de resolução para o formalismo escolhido, dito mecanismo de inferência;

iii) Estratégias de controle e escalonamento da inferência ou métodos de resolução de conflitos, implementados dentro dos mecanismos de inferência por módulos chamados executivos (Hayes-Roth et al., 1978), com a função de guiar e otimizar a inferência, que normalmente não é consistente ou completa.

2.2.1 Regras de Produção

Particularmente, o emprego de regras enquanto formalismo de representação de conhecimento tem angariado notório sucesso junto aos usuários, devido à sua simplicidade e facilidade de uso. A maior parte dos sistemas especialistas emprega este formalismo. Eis um exemplo de uma sentença lógica transformada em regra orientada ao conseqüente:

```
peixe(x) ^ mamifero(x) ^ grande(x) => baleia(x)
```

Esta regra pode ser entendida como: "Para todo x (quantificador universal implícito), se existe um fato afirmando que x é peixe, outro que é mamífero e um outro que é grande, isto implica a validade de um novo fato, o de que x é uma baleia".

Um dos motivos práticos que fez com que as técnicas de Inteligência Artificial não lograssem a popularidade almejada como ferramentas disponíveis na elaboração de sistemas em geral, deveu-se à postura isolacionista assumida durante a elaboração de sistemas e protótipos, sem as preocupações, comuns às outras áreas de computação, com interoperabilidade, portabilidade e reuso (Pachet, 1995). À parte da demonstração do potencial dos sistemas especialistas, poucos aplicativos se transformaram em produtos, principalmente à dificuldade de integrar as primeiras linguagens padrão – LISP e Prolog – com outras linguagens (Riley, 1999).

Com a percepção de que a integração entre as linguagens convencionais - tais como "C" e Java - e motores de inferência poderia resultar benefícios, foram criados os sistemas de produção. Escreveram-se motores de inferência integráveis a códigos imperativos, com a

vantagem de manter o conhecimento fora dos programas; assim, as regras de produção incorporaram uma separação ainda mais radical e eficiente entre dados e processo pelo uso de estruturas de dados para armazenar as regras, ditas bases de conhecimento, que são "compiladas" em tempo de execução, com indexação eficiente baseada em redes (Forgy, 1982) para o problema da resolução de conflitos, e, por isso, foi empregada maciçamente em sistemas especialistas e agentes. Com isso resolveu-se o problema da compilação, deixando o conhecimento fora dos programas, e, portanto, mais extensível e alterável.

Os sistemas de produção possuem facilidades de troca de variáveis e objetos com as linguagens hospedeiras, o que provê uma integração completa com o código imperativo, podendo inclusive executar métodos de objetos recebidos pelo motor dentro de regras.

2.2.2 Quadros (Frames) e Redes Semânticas

Outro formalismo bastante utilizado são os sistemas baseados em *frames* (Minsky, 1975). Eles foram inspirados na forma como as pessoas resolvem problemas, trazendo da memória estruturas de padrões de situações, que tentam instanciar, e são considerados um dos precursores dos atuais objetos. Os *frames* ou quadros guardam íntima semelhança com os objetos, pois possuem:

i) Atributos para os quadros - preenchidos com valores, ou com facetas, procedimentos ou funções para atribuir valores aos atributos, que não são implementadas por objetos, e

ii) Herança, que são relações entre os quadros do tipo é-um e é-um-subconjunto-de, bem semelhantes aos conceitos de instância e subclasse dos objetos.

Pode-se considerar as Redes Semânticas (Woods, 1975) como um sistema baseado em *frames*, se os arcos da rede forem atributos do *frame* correspondente. Ressalve-se, porém, o fato de que as relações de herança e os conceitos de classe constituem o cerne de um *frame*, e se a hierarquia de conceitos for mais importante na representação do domínio tratado que os atributos, a representação em *frames* ganha em organização. Quando os atributos são mais importantes, as Redes Semânticas são mais adequadas. Por sua expressividade nesses casos, elas têm sido largamente aplicadas em Processamento de Linguagem Natural.

Até o surgimento do projeto KSE (Knowledge Sharing Effort – esforço de compartilhamento de conhecimento) (Farquhar, 1996), os sistemas baseados em frames eram aplicados quase que exclusivamente na especificação do nível lógico de agentes e sistemas especialistas, devido à inexistência de boas ferramentas de apoio para o nível de implementação, tanto de ambientes de programação quanto de edição do conhecimento. O KSE modificou esse quadro substancialmente, provendo uma excelente ferramenta de edição de ontologias, o Editor da Ontolingua, e também tradução para formalismos diversos de representação de conhecimento. A motivação era, principalmente, aproveitar conhecimento existente codificado em grandes bases de conhecimento, como o Cyc (Lenat, 1990), o WordNet (Miller, 1995) e outras. Os formalismos de representação destas bases coincidiam apenas na existência de *frames*, em sua forma original, como em CLIPS, de forma mais expressiva, como em lógica de descrição ou redes semânticas, ou de forma indireta, com linguagens que pudessem facilmente representar frames, como Lisp e Prolog.

Vários benefícios imprevistos foram gerados a partir do advento do KSE. Eles ajudaram a inspirar e impulsionar:

i) A "indústria de ontologias", ou seja, despertou-se o interesse pela codificação de conhecimento reusável e comunicável, em oposição a código imperativo,

ii) A capacidade de comunicação em nível de conhecimento, já que o conhecimento se transformou em dados utilizáveis por um programa com motor de inferência,

iii) E, indiretamente, a capacidade de tradução, por juntar os fatores de conhecimento fora do código, organizado num formalismo lógico, com semântica claramente definida e independente de linguagem.

2.3 Ontologias Reusáveis e Compartilháveis

Os formalismos de representação de conhecimento proporcionaram a preciosa oportunidade de se estruturar conhecimento em ontologias, com conjuntos de conceitos sobre determinados domínios, definindo as propriedades e relações destes conceitos como axiomas. Apesar da palavra "ontologia" denotar uma teoria sobre a natureza do ser ou existência, em

Inteligência Artificial ela pode ser interpretada como o conjunto de entidades com suas relações, definições, axiomas e vocabulário. Uma ontologia é capaz de definir um domínio, ou, mais formalmente, especificar uma conceitualização acerca dele (Gruber, 1995). Normalmente, uma ontologia é organizada em hierarquias de conceitos ou taxonomias. Pelo fato de, idealmente, não refletirem nenhum formalismo específico, e de representarem com freqüência uma interface de vocabulário comum entre usuários e sistemas, pode-se considerar as ontologias como a materialização do nível de conhecimento (Newell, 1982).

Até meados dos anos 90, o conhecimento de um sistema especialista não podia ser reusado ou compartilhado, organizando-se em bases de conhecimento monolíticas e isoladas, sem interfaces de acoplamento, e, portanto, sem interoperabilidade. A construção de ontologias pode ser vista como um passo importante de evolução na especificação de conhecimento.

A decomposição desse conhecimento em módulos de construção com forte engajamento ontológico – ou seja, similaridade com o conhecimento da forma como ele é organizado, e, especialmente com a terminologia empregada numa área específica – propiciou a criação de ontologias reusáveis, com conceitos instanciáveis e especializáveis.

Ontologias pré-construídas sobre domínios restritos têm sido bastante reutilizadas e podem vir a representar um papel fundamental como fornecedoras de conhecimento para a inferência dinâmica realizada por agentes inteligentes. Fora isso, as ontologias garantem, através de um vocabulário comum de termos e seus respectivos conceitos, definições, relações, axiomas e restrições, a comunicação em nível de conhecimento entre agentes.

Esse compromisso com o vocabulário do domínio desempenha, ainda, um importante papel na identificação de significadosdistintos de uma mesma palavra ou termo, o que pode acarretar uma significativa melhoria de precisão em sistemas de manipulação de informação na Web. Quanto mais específica é uma ontologia, menos ambigüidades e interpretações tornam-se possíveis (Uschold et al., 1999). Para o usuário que está à busca de dados específicos, isso pode significar a economia de horas de trabalho.

Desde que foi criado o projeto KSE (Knowledge Sharing Effort), estão sendo criadas e mantidas ontologias extensíveis, abrangentes, gerais e muito detalhadas, por grupos de pesquisa, abarcando toda a pesquisa da área cujo conhecimento se deseja representar. Esta

orientação ontológica trouxe muitos benefícios, alguns dos quais não previstos, e que só vieram frutificar a época de sua implementação. De início, o KSE e suas ontologias contribuíram para uma maior cooperação entre os grupos de pesquisa responsáveis por manter as ontologias, da mesma forma como mantêm conhecimento, o que, tornando-se uma tendência, pode vir a provocar uma mudança cultural. Outros benefícios são:

i) A oportunidade para os desenvolvedores de reusar ontologias e bases de conhecimento, mesmo com adaptações e extensões. Sem dúvida, o impacto sobre sistemas baseados em conhecimento é enorme: a construção de bases de conhecimento redunda na tarefa mais cara e demorada de um projeto de sistemas especialistas e/ou agentes;

ii) A disponibilização de uma vasta gama de "ontologias de prateleira", prontas para uso, reuso e comunicação por pessoas e agentes. Hoje as ontologias mais maduras, algumas com mais de 2.000 definições, incluem metadados de imagens de satélites e para integração de bases de dados de genoma, catálogos de produtos, osciloscópios, robótica, semicondutores, terminologia médica, o padrão IEEE para interconexões entre ferramentas, e muitas outras (Farquhar, 1996).

2.4 Ontologias e a Internet

A necessidade de melhores ferramentas de busca para a Web terminou por disseminar o termo "agente" para praticamente qualquer aplicação que se propusesse a manipular ou procurar informação na rede. Nesta trilha, o novo termo, que vem herdando essa popularidade, face às promessas de melhoria destas atividades, é o termo ontologia, uma vez que elas estão presentes em muitos sistemas, ferramentas e produtos de manipulação de informação e comércio eletrônico, representadas como hierarquias de palavras-chave, conceitos, e muitas outras formas.

2.4.1 Ferramentas de Recuperação de Informação para a Web Envolvendo Ontologias

Conforme dito anteriormente, os sistemas abertos, as WANs (Wide Area Networks, ou redes de vastas áreas), a Internet, o crescimento da capacidade de conectividade, e a impossibilidade de estruturação de redes cada vez maiores e seus respectivos documentos propiciaram uma re-visita às técnicas de Inteligência Artificial, como a alternativa mais factível para um melhor tratamento aos problemas desses ambientes. Basicamente, dois tipos de solução foram propostos, que não são mutuamente exclusivas (Freitas & Bittencourt, 2000):

i) Dotar os sistemas de inteligência e autonomia para percorrer e selecionar informação relevante na imensidão da rede, o que veio a originar os chamados agentes inteligentes;

ii) Dotar as redes e a própria Internet de inteligência, fazendo com que as páginas possuam uma semântica mais clara e definida, já que as ferramentas que lhe dão suporte, a linguagem HTML (HyperText Markup Language) e o protocolo HTTP (HyperText Transfer Protocol) preocupam-se, em seu estágio atual, com apresentação e navegação, e por isso só podem ser aproveitados em buscas léxicas e mensuração de proximidade de texto, e não em buscas semânticas com contexto delimitado.

Vale a pena salientar que as ontologias representam um papel fundamental em ambos os tipos: no primeiro como elemento de comunicação de agentes, organização, reuso e disseminação de conhecimento, e no segundo como parte intrínseca das linguagens propostas para a definição de páginas com semântica. A Sociedade de Agentes Artificiais a ser apresentada neste trabalho enquadra-se no primeiro tipo.

2.4.2 A Web Semântica

Tim Bemers-Lee vislumbrou o desenvolvimento da Web Semântica como a segunda geração da Web (Berners-Lee et al., 2001), que a partir do uso intensivo de metadados, dados que descrevem dados, visa prover acesso automatizado à informação com base no processamento semântico de dados e heurísticas feito por máquinas.

A Web Semântica representa a evolução e a extensão da Web atual, porém apresentará uma estrutura que possibilitará a compreensão e o gerenciamento dos conteúdos armazenados na Web independente da forma em que estes se apresentem, seja texto, som, imagem ou gráficos, a partir da valoração semântica desses conteúdos e através de agentes que serão programas coletores de conteúdo advindos de fontes diversas capazes de processar as informações e permutar resultados com outros programas (Berners-Lee et al., 2001). Assim, ela visa fornecer estruturas e semântica ao conteúdo das páginas Web, criando um ambiente onde agentes de software e usuários possam trabalhar cooperativamente.

A Web Semântica é um esforço colaborativo conduzido pela W3C com participação de um grande número de pesquisadores. É baseada no RDF (Resource Description Framework), integrada a uma variedade de aplicações usando XML (eXtensib/e Markup Language) para a sintaxe. A Web Semântica visa o discemimento pela própria máquina das informações contidas nos documentos Web. Atualmente, essa compreensão é realizada pelo usuário, onde ele escolhe e define o que verdadeiramente lhe interessa.

Com a utilização de metadados é possível a descrição e uma melhor definição semântica das informações, gerando uma recuperação com maior qualidade e precisão. Assim, diversas tecnologias, como a XML e a RDF, vêm sendo trabalhadas com o objetivo de promover o intercâmbio de metadados desenvolvidos de forma independente, de modo que eles possam ser compreendidos pelo computador, sem o auxílio do discernimento humano. A arquitetura RDF consiste em uma estrutura básica para o processamento de metadados, provendo interoperabilidade entre aplicações que trocam informações, compreendidas pela máquina e entre si na Web e descrevendo formalmente um determinado recurso da Web através de seus metadados.

2.5 Ferramentas para Manuseio de Ontologias

A construção de ontologias para a comunicação em nível de conhecimento entre agentes tem se tornado alvo de estudos e propiciou a elaboração de ferramentas com o objetivo de dar suporte a esta atividade. Assim, vários editores de ontologias estão surgindo, com o objetivo não só de facilitar a sua construção, como também de disponibilizar ontologias públicas para reuso e extensão, integrando diferentes grupos de pesquisa, que pesquisam sobre as mesmas áreas ou áreas afins.

Alguns requisitos tornam uma ferramenta popular: facilidades de acesso e conexão a repositórios, portabilidade e ferramentas de apoio que facilitem o desenvolvimento.

2.5.1 O Editor da Ontolingua

A Ontolingua converteu-se no primeiro editor de ontologias a angariar popularidade, por dispor de um leque de funcionalidades amplo (Farquhar, 1996). A ferramenta disponibiliza uma interface de acesso através de navegadores, possibilitando a qualquer usuário da Web a visualização das ontologias criadas e disponíveis, a criação de novas e o trabalho colaborativo. A arquitetura do servidor Ontolingua está representada na Figura 2-1 . Os colaboradores remotos lêem e contribuem com novos conhecimentos para a biblioteca digital de ontologias em texto, mantida em HTML e acessível através de navegadores da Internet.

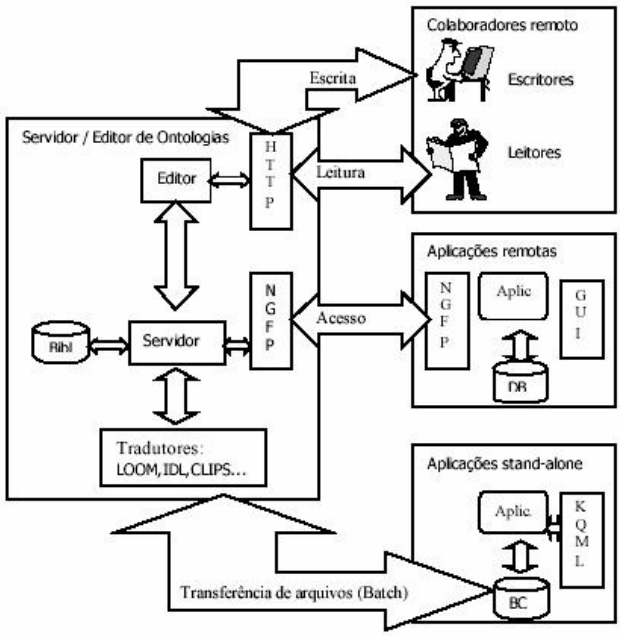

Figura 2-1 - Arquitetura do Servidor Ontolingua

Além das funcionalidades - como um gerador gráfico de ontologias -, a Ontolingua destaca-se pelos critérios cuidadosos com que foi projetada. Por exemplo, a Ontolingua distingue com bastante propriedade os níveis de conhecimento (ou epistemológico), o nível lógico e o de implementação, criando ainda o nível de apresentação. No nível de conhecimento está o conhecimento propriamente dito em formato texto; o nível lógico pode

ser visto como o KIF (Knowledge Interchange Format), pois representa o conhecimento num formalismo, sem preocupação como ele será implementado computacionalmente para os processos de inferência; no nível de implementação, a ontologia é representada num determinado formalismo, como CLIPS ou Prolog; e o nível de apresentação resolve conflitos de conceitos homônimos de áreas distintas, que podem causar problemas quando há interconexões entre estas áreas. O nível de apresentação mantém isto transparente ao usuário, dando a real impressão que o sistema "entende" a diferença conceitual entre eles.

Mesmo fornecendo boas interfaces gráficas para navegadores, falta à Ontolingua uma boa interface para estações de trabalho e computadores, que permitam a usuários, entender, reusar, manusear e verificar ontologias, sem estar necessariamente acessando a Internet.

2.5.2 O Ambiente Protégé

O ambiente Protégé é uma poderosa ferramenta para manipulação de bases de conhecimento. Apesar de, no seu início, ter sido desenvolvido para manipulação de bases dentro do domínio da medicina, acabou evoluindo de tal forma que, atualmente, permite a criação de qualquer base nos mais diversos formalismos para representação de conhecimentos disponíveis (Protégé, 2005).

A principal razão para o desenvolvimento desta ferramenta está no alto custo de desenvolvimento e manutenção de sistemas baseado em conhecimento. Com esta consideração a ferramenta foi desenvolvida e evoluída através dos anos para guiar desenvolvedores e especialistas de domínio durante o processo de desenvolvimento destes sistemas.

O modelo de conhecimento do Protégé é baseado em *frames*. As ontologias no Protégé são formadas por classes, slots, facets e axiomas, onde:

i) Classes são conceitos dentro de um domínio;

ii) Slots descrevem propriedades ou atributos de classes;

iii) Facets descrevem propriedades de Slots;

iv) Axiomas especificam características adicionais.

As classes no Protégé formam uma hierarquia taxonômica. Se uma classe A é subclasse de uma classe B então todas as instâncias de A são também instâncias de B. Inclusive, o Protégé suporta herança múltipla.

A Figura 2-2 mostra a representação de uma ontologia no ambiente Protégé 3.0. No painel a esquerda, observa-se um exemplo de hierarquia de classes.

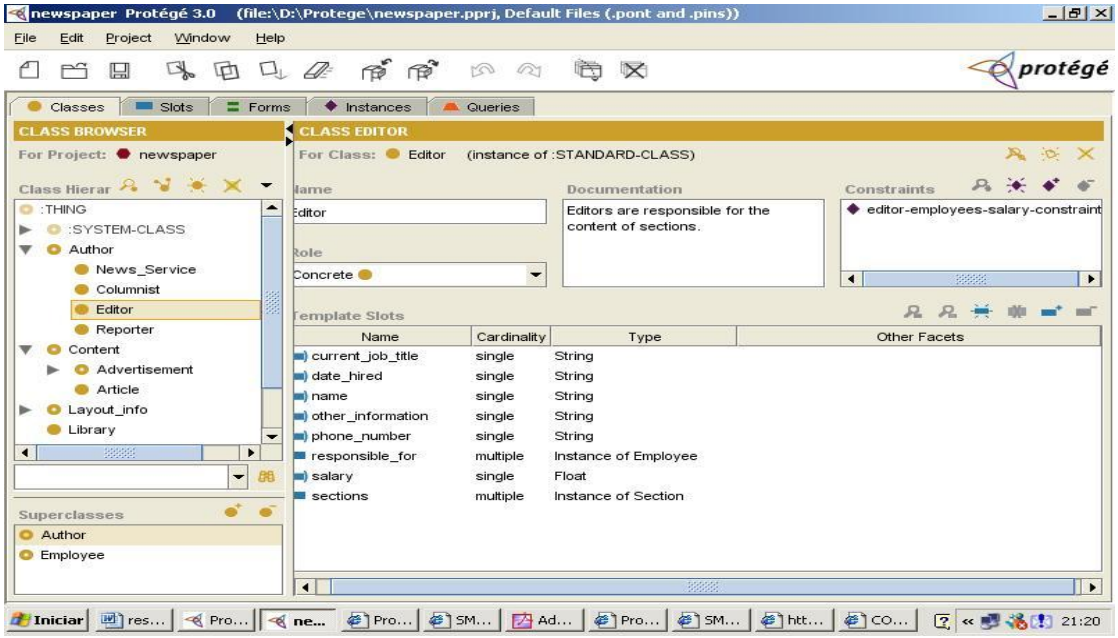

Figura 2-2 - Representação de uma ontologia no ambiente Protégé 3.0

No ambiente Protégé, a partir dos slots definidos para uma determinada classe, são construídos os formulários para adição de instâncias para esta classe. Além disso, possui um vasto conjunto de extensões que o habilitam a converter bases de conhecimento em vários formalismos, dentre eles o Jess, RDF, Prolog e OWL.

O plugin OWL é uma extensão do Protégé que pode ser usado para editar arquivos e banco de dados OWL. O plugin OWL estende o modelo Protégé para representação do conhecimento através de classes, conforme Figura 2-3.

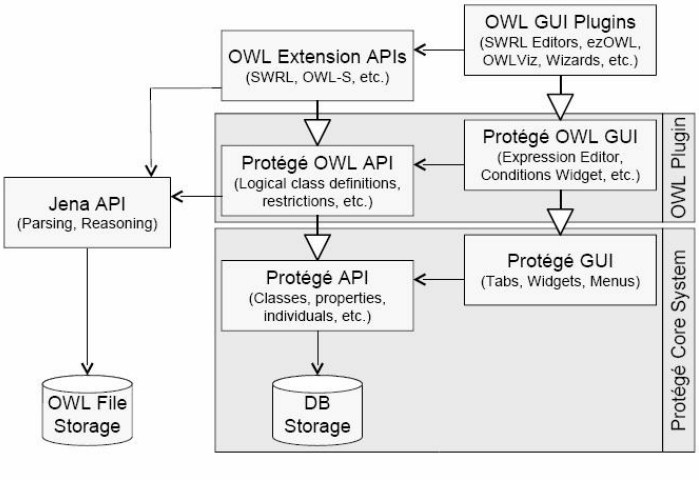

Figura 2-3 - O Plugin OWL como extensão do Protégé

O ambiente Protégé é uma sólida ferramenta para desenvolvimento e manutenção de ontologias. Com uma interface bem enriquecida, possibilita aos desenvolvedores/engenheiros de conhecimento a construção de uma interação com o usuário a partir da definição da própria base.

Com a sua capacidade de extensão, através do desenvolvimento de plugins, permite a construção de verdadeiras ferramentas adicionais para manipulação da base, onde temos de simples componentes de formulários (*widgets*) até poderosos *tabs* com a capacidade de conversão e representação do conhecimento em outros formalismos.

É importante ainda ressaltar que, através da API-Protégé, pode-se 'importar' as funcionalidades desta ferramenta para diversos ambientes, que se julguem ainda mais amigáveis.

3 RECUPERAÇÃO DE INFORMAÇÃO NA WEB

Recuperação de Informação (RI) é a área da Ciência da Computação que estuda formas de representar, armazenar, organizar e acessar itens de informação (Baeza-Yates et al., 1999). O objetivo primordial de um sistema de RI é satisfazer as necessidades de informação do usuário, usualmente expressas em linguagem natural.

3.1 A evolução dos sistemas de RI

Os primeiros sistemas de RI surgiram para dar suporte à pesquisa científica e tecnológica e eram baseados em modelos totalmente voltados para recuperação de texto. Com o desenvolvimento da tecnologia de bancos de dados, surgiram mais tarde as Bibliotecas Digitais (Schatz, 1997), os sistemas de recuperação e ordenação automática de elementos e os sistemas de geração automática de índices.

Os modelos clássicos de RI podem ser classificados em três categorias principais: os que são baseados na Teoria dos Conjuntos (*Booleano*); os baseados no Modelo Algébrico (*Vetor*) e aqueles baseados no Modelo Probabilístico. Uma maior cobertura desses modelos pode ser encontrada em (Baeza-Yates et al., 1999).

As pesquisas em Recuperação de Informação na Internet surgiram com o WAIS (*Wide Area Information Server*), que foi a primeira ferramenta de software a permitir que o usuário pudesse efetuar buscas pela Internet. Aos WAIS, seguiram-se serviços tais como *Gopher*, *Mosaic*, *Netscape*, *Lycos*, *AltaVista,* etc.

Com o surgimento da *World Wide Web,* ou simplesmente Web, grande quantidade de informação poderia agora estar acessível a um grande número de pessoas, simultaneamente, e a um baixo custo. As Bibliotecas Digitais foram estendidas à Internet e a demanda por serviços de RI na Web, cada vez mais eficientes, tem aumentado desde então. A Web pode ser vista como um gigantesco banco de dados distribuído, cujas informações não se encontram organizadas e que pode ser acessado de qualquer ponto da rede (Baeza-Yates et al., 1999). Dessa maneira, os algoritmos que antes eram empregados em domínios estáticos, não podem atender aos requisitos da Web, que se apresenta como um ambiente dinâmico, com uma magnitude, muito maior do que qualquer banco de dados estático e que não segue uma

estrutura padrão. Nesse contexto, têm surgindo muitas técnicas para a recuperação de informações que utilizam conceitos de Inteligência Artificial para a obtenção de um melhor desempenho desta tarefa. Este trabalho mostra como se podem utilizar múltiplos agentes artificiais para recuperação de informações a serem utilizadas em um ambiente de aprendizagem.

3.2 Técnicas de Recuperação de Informação

Algumas técnicas bastante conhecidas, em RI, para criar índices são (Baeza-Yates et al., 1999): *i)* arquivo invertido, técnica muito usada em sistemas que tratam texto como uma seqüência de "palavras", cuja estrutura de dados é composta por dois elementos básicos: *vocabulário* e *ocorrências*; *ii)* árvores e vetores de sufixos, técnica empregada em sistemas que não implementam o conceito de "palavra", como bancos de dados genéticos, por exemplo (Baeza-Yates et al., 1999); e *iii)* a técnica conhecida como "signature files", baseada no armazenamento de códigos para cada bloco de texto (Baeza-Yates et al., 1999). Em razão de sua simplicidade, utiliza-se, na implementação dos Agentes de Busca a primeira técnica. Entretanto, qualquer outra técnica poderia ser utilizada.

Em modelos clássicos de RI, tanto o documento quanto a necessidade de informação do usuário são representados para o sistema por meio de listas (vetores) de termos. Essa forma de representação (visão lógica) é obtida através da aplicação de operações de texto (Baeza-Yates et al., 1999). Uma consulta é a expressão da necessidade de informação do usuário, na forma de uma linguagem de consulta provida pelo ambiente. A forma mais simples de linguagem de consulta é baseada na especificação de palavras-chaves combinadas com operadores booleanos.

Para determinar a relevância de um documento, os sistemas de RI precisam interpretar seu conteúdo e classificá-lo de acordo com sua possível importância para a consulta que está sendo processada. A interpretação do conteúdo de um documento envolve a extração de informações sobre o próprio documento (metadados), tais como: *sintaxe*, *semântica*, *estrutura* e *estilo de apresentação*, por exemplo. Mas, na maioria dos modelos clássicos de RI, a classificação dos documentos baseia-se somente na medida da freqüência com que os termos ocorrem em um dado documento. Uma variação é a contagem do número de documentos em que os termos da consulta aparecem. Essa abordagem simplifica bastante o problema, mas

apresenta falhas, pois sabemos que há perda semântica na representação de um documento ou necessidade de informação, na forma de listas de termos. Havendo perda no armazenamento das informações, logicamente a tarefa de recuperação também estará comprometida. Portanto, em se tratando de semântica, conforme visto na Seção 2.4, pode-se ter como solução i) dotar os sistemas de inteligência (agentes inteligentes); ou ii) dotar a Internet de inteligência (páginas com semântica). A primeira é objeto de estudo deste trabalho e para este caso as ontologias servirão como elemento de comunicação de agentes, organização, reuso e disseminação de conhecimento. No capítulo 6, faz-se a especificação de uma Sociedade Agentes de Busca que utilizam ontologias para recuperar informações da Web.

3.3 Mecanismos de Busca

Até o momento, existem basicamente três formas conhecidas para realizar buscas na web: 1) Máquinas de Busca: possuem parte da web indexada; 2) Diretórios: classificam os documentos em categorias de assuntos; ou 3) explorando a estrutura hipertexto da web. A seguir, apresenta-se o funcionamento de cada um desses mecanismos de busca.

3.3.1 Máquinas de Busca

Normalmente, as máquinas de busca possuem parte da web indexada e armazenada em bases de índices, onde as buscas são efetivamente realizadas. A maioria das máquinas de busca usa uma arquitetura centralizada do tipo "Crawler-Indexador" (cf. Figura 3-1).

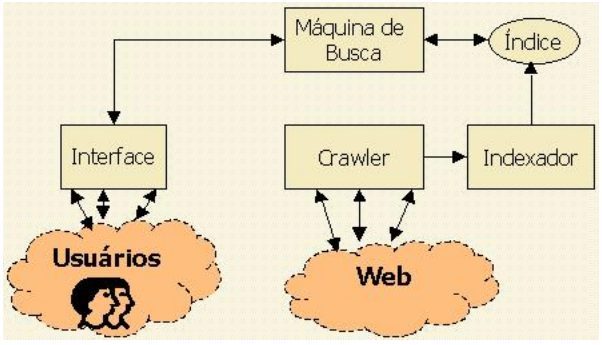

Figura 3-1 - Arquitetura Crawler-indexador (adaptado de (Baeza-yates et al., 1999)).

Crawlers são programas (alguns são baseados em agentes de software) que percorrem a web buscando e enviando novas páginas ou atualizando páginas para os servidores onde serão indexadas. Também são chamados de *robots*, *spiders*, *wanderers*, *walkers* e *knowbots*.

Os Crawlers não migram para as máquinas remotas, eles são baseados na arquitetura cliente-servidor, ou seja, executam localmente no sistema enviado requisições a servidores web remotos. O índice é usado de maneira centralizada para responder consultas submetidas a partir de diferentes locais na web (Baeza-Yates et al., 1999). Alguns serviços de busca como AltaVista, HotBot, NorthernLight e Excite são baseados nesta arquitetura.

Existem variações da arquitetura "Crawler-Indexador", como, por exemplo, a arquitetura *Harvest* (Bowman et al., 1995), a arquitetura da máquina de busca Google (Brin et al., 1998) e a arquitetura do serviço CiteSeer (Bollacker et al., 1998), voltado para recuperação de publicações científicas.

3.3.2 Diretórios

Outra forma de se pesquisar na web é através da utilização de Diretórios, que constituem uma categorização hierárquica de um domínio de conhecimento, mantida por especialistas. Embora a cobertura provida pelos diretórios seja muito baixa, as respostas retornadas ao usuário geralmente apresentam alto grau de relevância. No entanto, por não serem capazes de acompanhar o crescimento da web, estão caindo rapidamente em desuso. Alguns exemplos são Yahoo, LookSmart, Lycos e Netscape. Os Diretórios também são conhecidos como Catálogos ou Páginas Amarelas.

3.3.3 Meta-Buscadores

São servidores web que enviam uma dada consulta para várias máquinas de busca, diretórios e outros bancos de dados, coletam os resultados e os unificam. Alguns exemplos são MetaCrawler e MetaMiner. As principais vantagens de um meta-buscador é a habilidade de combinar os resultados obtidos de muitas fontes e o fato de que o usuário pode submeter a mesma consulta a várias fontes distintas através de uma única interface. Os meta-buscadores diferem uns dos outros na forma de classificar os resultados obtidos das diversas fontes. Em alguns casos, nenhuma classificação é executada. Também diferem na maneira como traduzem a consulta do usuário para a linguagem de consulta específica de cada máquina de busca ou diretório. Por outro lado, o número de páginas que pode ser recuperada de cada fonte pelo meta-buscador é limitado.

3.4 Modelagem da Web

Por sua riqueza, variedade e falta de estrutura de seu conteúdo, um dos problemas que precedem qualquer solução relativa a Web é a forma de abordá-la ou modelá-la. O tema "modelagens da Web" costuma estar presente em muitos congressos sobre a Internet, e constitui tema de intensa pesquisa. Basicamente, existem dois tipos de modelos da Web, baseados em abordagens criadas sob o ponto de vista de Bancos de Dados (Florescu et al., 1998):

i) Modelos baseados em grafos, onde os ponteiros representam os arcos do grafo;

ii) Modelos baseados em dados semi-estruturados, em que partes da Web possuem entidades e atributos, cujo esquema não é completamente conhecido ou obedecido, ou seja, são toleradas páginas que não se adaptam completamente ao esquema.

O primeiro modelo adequa-se melhor a problemas de busca, uma vez que buscas em grafos são problemas relativamente formalizados. O segundo modelo oferece ferramentas mais acuradas de acesso aos atributos, e costuma ser empregado em tarefas como sumarização e extração de dados semi-estruturados. Por estes motivos, a modelagem da Web introduzida neste trabalho enquadra-se como um modelo baseado em dados semi-estruturados.

Os modelos baseados em dados semi-estruturados não tratam toda a Web, mas partes definidas dela, que devem obedecer minimamente a esquemas pré-definidos. A tarefa conhecida como categorização preocupa-se com a identificação de partes ou categorias da Web, e é empregada com esse propósito. A categorização pode ser feita de duas formas:

i) Sobre categorias previamente definidas;

ii) Criando-se categorias de acordo com a semelhança entre as páginas, problema este conhecido como agrupamento (clustering) [Sahami et al 97].

Para extração, a categorização é usualmente efetuada sobre categorias pré-definidas.

3.5 Visão por conteúdo

A visão por conteúdo tenta caracterizar partes da Web pela semântica dos dados apresentados nas páginas, identificando-os através da presença de elementos pertinentes ao contexto em que se espera que o dado se insira. Dois conjuntos de páginas, que serão descritos nas próximas subseções, relacionam-se à visão por conteúdo: as categorias e as classes de páginas.

3.5.1 Categorias

Muitos mecanismos de busca adicionalmente oferecem serviços de diretório ou divisões de categorias, nos quais páginas estão separadas manualmente de acordo com o seu conteúdo. A construção e/ou manutenção destes diretórios constitui uma das principais motivações de pesquisa em classificação automática através de técnicas de aprendizado (Cohen & Singer, 1996), pois, ocasionalmente, o conteúdo deles torna-se obsoleto e desatualizado, faltando páginas novas e relevantes. Estas categorias freqüentemente estão organizadas em hierarquias, e possuem características bastante peculiares:

i) **Similaridade Estrutural**: Curiosamente, o primeiro fato observável nestas categorias e ignorado pelos mecanismos de busca é o estilo de composição das páginas, que denota sua estrutura em termos de aparência. (Cruz et al., 1997) evidenciam a existência de padrões particulares de editoração das páginas pertencentes às categorias, e que podem ajudar a identificá-las, caso sejam analisados as tags de HTML que as constituem. Por exemplo, páginas de educação contêm mais ponteiros que as de outras categorias, páginas de artistas usam tabelas apenas para embelezar o lay-out, e possuem o dobro de imagens de páginas de advogados, que, por sua vez, usam tabelas apenas para apresentar dados, e apresentam uma freqüência significativamente superior de separadores de parágrafos em relação às outras categorias.

ii) **Conteúdo**: Naturalmente, na medida em que se desce na hierarquia, as categorias guardam maior especificidade, e muitas vezes apresentam itens de um mesmo tipo, como, por exemplo, dados sobre tenistas. As páginas classificadas em folhas das árvores de categorias, ou seja, páginas das

categorias mais inferiores, trazem dados específicos que podem ser mapeados, como, por exemplo, dados biográficos dos tenistas, resultados de partidas de torneios, entre outras.

iii) **Entidades**: Algumas destas folhas trazem páginas exclusivamente sobre apenas uma entidade ou instância de uma classe - isto é, sobre apenas uma pessoa, organização, empresa, artigo ou outro assunto específico – bem como seus respectivos dados ou atributos. Há ainda categorias dentro dos serviços de diretório que poderiam ser divididas com uma granularidade mais fina, de forma a comportar uma única entidade. Partes da Web podem ser vistas desta maneira, caracterizando a visão por conteúdo.

3.5.2 As Classes de Páginas

As páginas que apresentam entidades compartilham muitas características comuns entre si, tais como estilo de editoração, padrões de conexão a outras páginas, terminologia e, principalmente, o conjunto de atributos. A criação de extratores baseia-se neste fato, na existência de classes de páginas, que definem uma visão da Web por conteúdo. Em páginas da classe de turismo, por exemplo, com certeza encontram-se dados relacionados a preços, pacotes, reservas e muitos outros itens. Cada classe de página deve corresponder a um conceito na base de conhecimento estruturada (ontologia) do domínio.

A obrigatoriedade de existência de determinados atributos de uma entidade, assim como a obediência deles a determinadas regras de consistência, confere às entidades procuradas a propriedade de discriminação: a existência e consistência de seus atributos em páginas são um indicativo fundamental no reconhecimento de páginas consideradas como membros de uma classe. Por exemplo, páginas consideradas como de chamadas de trabalhos para eventos científicos devem conter pelo menos uma data ou um ponteiro para datas, e, em havendo mais de uma, a distância entre elas deve ser menor que um ano. Para um sub-domínio de Biologia, por exemplo, vertebrados, as páginas deverão conter texto e não poderão conter preços de produtos.

Assim, o emprego de regras para atributos propicia um inolvidável benefício, o uso da extração como refinador à tarefa de categorização. Na realidade, as duas tarefas podem ser consideradas como complementares na manipulação de informação da Web, uma vez que a

extração depende de uma prévia categorização, ou seja, de um reconhecimento e filtragem de páginas candidatas a membros da classe de páginas que está sendo processada.

3.5.3 Grupos de Classes (Clusters)

Uma das hipóteses deste trabalho é a de que muitos ponteiros das páginas que pertencem às classes, conforme definidas acima, apontam para outras páginas contendo entidades pertencentes a um número reduzido de outras classes ou contendo informação referente a páginas com essas características. Por exemplo, em páginas de Ciências, poderão ser encontrados ponteiros para páginas de instituições ou professores, por exemplo, podendo-se localizar ponteiros para assuntos relacionados e/ou páginas de outras classes.

O conceito de "comunidades da Web" (Gibson et al., 1998) reforça essa hipótese. Comunidades são regiões da Web que agrupam um conjunto de páginas autoritativas – as primeiras páginas ou as páginas principais de um site, por exemplo, www.puc-campinas.edu.br -, sendo este conjunto topologicamente fechado, no sentido em que as páginas autoritativas são o prefixo inicial do endereço de páginas apontadas por muitas outras páginas referentes a determinados assuntos ou tópicos. Isso demonstra que determinados tópicos localizam-se dentro de partes definidas da Web. Analogamente, este trabalho supõe que um conjunto fechado de classes e seus relacionamentos reúnem conhecimento sobre entidades dentro de áreas específicas, por exemplo, o meio científico, turismo, e outras áreas.

As entidades, atributos e relacionamentos de um grupo de classes devem coincidir com o conhecimento usual acerca da área específica a que pertencem e, especialmente, os relacionamentos entre as entidades estão refletidos como ponteiros entre suas páginas na Web. Assim, conhecimento a priori sobre a área específica pode ser aplicado para encontrar, extrair e validar dados, disponibilizando as instâncias das entidades extraídas para acesso a usuários ou agentes de software.

O trabalho efetuado pelos extratores atuais resume-se à captura das entidades, relevando o significativo e indicador conteúdo dos *hyperlinks*, que podem ser analisados dentro do contexto onde se localizam. De fato, não só a extração e recuperação, mas qualquer tipo de manipulação de informação sobre a Internet não deve deixar de levar em conta as ligações contextuais entre as informações presentes nas páginas, apresentadas sob a forma de âncoras.

Extratores podem fornecer com segurança endereços de páginas encontrados entre os seus ponteiros a outros extratores, se as entidades extraídas por este último tiverem relacionamento com a entidade extraída pelo primeiro. Afinal, estes endereços são obtidos dentro de um contexto confiável e bem mais seguro e com maior probabilidade de ser relevante do que as listas de resultados dos mecanismos de buscas. As listas de resultados devem, portanto, ter menor precedência de processamento de que estas "dicas" sobre endereços de páginas trocadas entre os extratores. O papel dos mecanismos de busca num acoplamento com extratores para a Web pode, dependendo da dificuldade de serem encontrados termos que prometam uma boa cobertura da classe de páginas procurada, apenas resumir-se a disparar a cooperação entre os extratores, fornecendo novas tentativas quando as sugestões de páginas enviadas pelos outros agentes tivessem se esgotado.

Dessa forma, a complementaridade entre a extração e a recuperação de informações está sedimentada; a extração pode otimizar a recuperação, que por outro lado inicia o processo de extração.

3.6 Visão por funcionalidade

Uma visão alternativa da Web diz respeito à funcionalidade das páginas, dividindo-as de acordo com o seu papel na ligação entre páginas e na apresentação e armazenamento de dados relevantes. Inspirada no trabalho de (Pirolli et al., 1995), esta visão baseia-se no exame de listas de resultados retornados pelos mecanismos de busca para o processamento de uma classe. Assim, dado este propósito e visando a extração integrada, as categorias funcionais estão assim divididas:

i) Páginas-conteúdo, que são páginas que pertencem à classe que está sendo processada, e de onde serão extraídas a(s) entidade(s) em questão (Páginas CLASSIFICADAS);

ii) Listas de páginas-conteúdo, também chamadas de diretórios ou índices, de grande utilidade na localização segura e contextualizada de páginas-conteúdo, por ser composta basicamente de âncoras para páginas-conteúdo (Páginas que têm uma lista de URLs a serem REAVALIADAS);

iii) Mensagens ou Listas de Mensagens, que contêm correspondências ou listas delas sobre assuntos correlatos à entidade que está sendo extraída, que usualmente não possuem utilidade para extração integrada, por serem páginas muito longas, por apontar para páginas trazendo informações muito disparatadas e com relacionamentos difíceis de serem identificados (Páginas que devem ser REJEITADAS);

iv) Recomendações, que são páginas-conteúdo que pertencem à outra classe, podendo ser aproveitadas quando do processamento dessa outra classe, acelerando o processo de busca da mesma (Páginas a serem RECOMENDADAS para outro Agente de Busca);

v) Simplesmente lixo, ou seja, páginas sem qualquer utilidade para a extração, por não pertencerem a nenhum dos itens anteriores (Páginas que devem ser REJEITADAS).

Convém salientar que é possível a existência de listas em páginas-conteúdo. Por exemplo, a bibliografia de um assunto em Biologia pode constituir uma lista de *links* para outras páginas da mesma classe.

Ressalve-se ainda que o comportamento e utilidade das categorias funcionais, e também o relacionamento entre elas para a extração integrada permanece fixo, conforme definido acima. Assim, durante a extração, a classificação de páginas resultantes de consultas a mecanismos de busca com relação a estas categorias, assim como a identificação de páginas autoritativas com relação à entidade processada, proporciona um refinamento fundamental no reconhecimento de páginas contendo dados realmente pertinentes.

3.7 Cruzamento entre Conteúdo e Funcionalidade

Tanto para identificar precisamente as páginas que abrigam as entidades das classes processadas, como para localizá-las rapidamente, beneficiando-se dos relacionamentos entre elas refletidos nas âncoras, as duas visões devem ser usadas simultaneamente. A visão por conteúdo responsabiliza-se por trazer da Web páginas potencialmente pertencentes às classes processadas, garantindo cobertura sobre a Web, enquanto a visão por funcionalidade

encarrega-se de selecionar com rigor as páginas que contêm as entidades (páginas-conteúdo), preocupando-se também em extrair o máximo de informações contextuais relevantes que ajudem a otimizar a busca de mais entidades não só da classe processada como de outras classes do grupo – os relacionamentos entre as páginas -, o que garante uma alta precisão.

4 SISTEMAS MULTIAGENTES

O objetivo deste capítulo é apresentar o conceito de sistemas multiagentes, conforme é utilizado neste trabalho, e apresentar ferramentas que podem ser utilizadas para o desenvolvimento desses agentes, mas antes é necessário fazer uma breve conceituação de agentes artificiais inteligentes.

4.1 Conceito de Agente

Um agente pode ser definido como uma entidade que executa uma ação sobre algo, produzindo um efeito. Por se tratar de uma definição genérica, é necessário definir o contexto particular de aplicação da definição, para que seja possível especificar o conceito de agente adotado neste trabalho. É importante, então, deixar claro que o conceito adotado refere-se ao universo da ciência da computação.

No âmbito da ciência da computação, pode-se falar de agentes artificiais como sendo sistemas de computação que executam determinadas tarefas com características que os enquadram como agentes. Agentes humanos são usuários humanos que projetam ou manipulam computadores (hardware ou software).

Russell e Norvig (Russell et al., 1995) afirmam que o trabalho de concepção de um agente inteligente consiste em definir o que será percebido pelo agente no ambiente, suas possíveis ações e, principalmente, os mecanismos através dos quais ele avaliará suas percepções para "escolher" a ação a ser executada, alcançando diferentes graus de autonomia.

Em (Wooldridge et al., 1995) os autores acrescentaram ao conceito de agentes inteligentes a noção de sociedade, ao afirmar que os agentes inteligentes devem ser capazes de se comunicar, interagindo uns com os outros através de algum tipo de linguagem comum.

Assim, avaliando as definições acima, pode-se dizer que agentes artificiais inteligentes constituem um novo paradigma de concepção de sistemas computacionais, no qual cada sistema deve ser visto como capaz de perceber um ambiente, através de sensores, e agir de forma racional e autônoma sobre esse ambiente. Estes sistemas podem conter mais de um agente, comunicando-se entre si, a fim de alcançarem os objetivos para os quais foram projetados.

4.2 Tipos de Agentes Artificiais

Para uma melhor compreensão da aplicabilidade da tecnologia de agentes, é interessante falar sobre os diversos tipos de agentes e suas mais variadas diferenças para que tenhamos uma melhor noção de utilidade no emprego de agentes. É possível fazer uma classificação de agentes de acordo com vários aspectos como quanto à mobilidade, quanto ao relacionamento inter agentes e quanto à capacidade de raciocínio.

Agentes Móveis: são agentes que tem a mobilidade como característica principal. Isto é, uma capacidade de mover-se seja por uma rede interna local (intranet) ou até mesmo pela Web, transportando-se pelas plataformas levando dados e códigos. Seu uso tem crescido devido alguns fatos como uma heterogeneidade cada vez maior das redes e seu grande auxílio em tomadas de decisões baseadas em grandes quantidades de informação.

Agentes situados ou estacionários: são aqueles opostos aos móveis. Isto é, são fixos em um mesmo ambiente e ou plataforma. Não se movimentam em uma rede e muito menos na Web.

Agentes Competitivos: são agentes que "competem" entre si para a realização de seus objetivos ou tarefas, ou seja, não há colaboração entre os agentes.

Agentes Coordenados ou Colaborativos: agentes com a finalidade de alcançar um objetivo maior realizam tarefas especificas, porém coordenando-as entre si de forma que suas atividades se completem.

Agentes Reativos: é um agente que reage a estímulos sem ter memória do que já foi realizado no passado e nem previsão da ação a ser tomada no futuro. Não tem representação do seu ambiente ou de outros agentes e são incapazes de prever e antecipar ações. Geralmente atuam em sociedades como uma colônia de formiga, por exemplo. Baseiam-se muito também na "teoria do caos" no qual afirma que até mesmo no caos existe uma "certa organização". No caso da formiga, por exemplo, uma única delas não apresenta muita inteligência, mas quando age no grupo comporta-se o todo como uma entidade com uma certa inteligência, ou seja, a força de um agente reativo vem da capacidade de formar um grupo e construir colônias capazes de adaptar-se a um ambiente.

Agentes Cognitivos: esses, ao contrário dos agentes reativos, podem raciocinar sobre as ações tomadas no passado e planejar ações a serem tomadas no futuro. Ou seja, um agente cognitivo é capaz de "resolver" problemas por ele mesmo. Ele tem objetivos e planos explícitos os quais permitem atingir seu objetivo final. Para que isso se concretize, cada agente deve ter uma base de conhecimento disponível, que compreende todo os dados e todo o "know-how" para realizar suas tarefas e interagir com outros agentes e com o próprio ambiente. Sua representação interna e seus mecanismos de inferência o permitem atuar independentemente dos outros agentes e lhe dão uma grande flexibilidade na forma de expressão de seu comportamento. Além disso, devido a sua capacidade de raciocínio baseado nas representações do mundo, são capazes de ao mesmo tempo memorizar situações, analisá-las e prever possíveis reações para suas ações.

4.3 Multiagentes

Sistemas Multiagentes são sistemas constituídos de múltiplos agentes que interagem ou trabalham em conjunto de forma a realizar um determinado conjunto de tarefas ou objetivos. Esses objetivos podem ser comuns a todos os agentes ou não. Os agentes dentro de um sistema multiagente podem ser heterogêneos ou homogêneos, colaborativos ou competitivos, dependendo da finalidade da aplicação que o sistema multiagente está inserido.

Arquiteturas com múltiplos agentes reativos são constituídos por um grande número de agentes. Estes são bastante simples, não possuem inteligência ou representação de seu ambiente e interagem utilizando um comportamento de ação/reação. A inteligência surge conforme os agentes trocam de informações entre si e com o ambiente. Esses agentes não são inteligentes individualmente, mas o comportamento global é.

Já os sistemas multiagentes constituídos por agentes cognitivos são geralmente compostos por uma quantidade bem menor de agentes se comparado aos sistemas multiagentes reativos. Aqueles, conforme a definição de agentes cognitivos, são inteligentes e contêm uma representação parcial de seu ambiente e dos outros agentes. Podem, portanto, comunicar-se entre si, negociar uma informação ou um serviço e planejar uma ação futura.

Esse planejamento de ações é possível, pois em geral os agentes cognitivos são dotados de conhecimentos, competências, intenções e crenças, o que lhes permite coordenar suas ações visando à resolução de um problema ou à execução de um objetivo.

4.4 Ferramentas para desenvolvimento de sistemas multiagentes

Existem muitas ferramentas que facilitam o desenvolvimento de sistemas multiagentes distribuídos, como por exemplo: ZEUS, JADE e Microsoft Agent. O Microsoft Agent (MsAgent, 2003) dependente de produtos Microsoft. O ZEUS e o JADE, os quais são descritos nas próximas subseções, apresentam-se como ferramentas de fácil uso, portáveis e que utilizam o padrão FIPA. No ambiente NetClass, utiliza-se o Framework JADE, por este ser suportado pelo Protégé. Mas isso não impede de se utilizar o ZEUS em um outro momento.

4.4.1 ZEUS

A ferramenta ZEUS (Nwana et al., 1999) se caracteriza por ser um ambiente gráfico (cf. Figura 4-1) que fornece uma biblioteca de componentes de software e ferramentas que facilitam uma rápida construção e desenvolvimento de sistemas multiagentes. Ele é gratuito, de código aberto e constitui-se por um conjunto de componentes, escritos em Java, que podem ser descritos em três grupos funcionais (ou bibliotecas): biblioteca de componentes para agentes, uma ferramenta construtora de agentes e um conjunto de utilitários, incluindo um servidor de nomes, um facilitador e um visualizador de agentes.

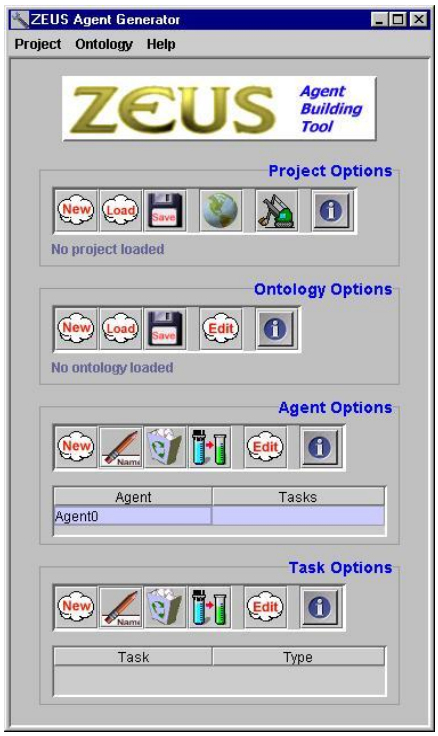

Figura 4-1 - Interface principal do ZEUS Agent Toolkit

A biblioteca de componentes fornece, dentre outras coisas, uma linguagem de comunicação entre agentes baseada em performativas, nesse caso FIPA-ACL. Além disso, ZEUS opera em redes TCP/IP, o que facilita a intercomunicação entre agentes executados em sistemas operacionais diferentes.

4.4.2 Framework JADE

O framework JADE (Java Agent DEvelopment framework) constitui-se de um conjunto de recursos para o desenvolvimento de aplicações *peer-to-peer* baseadas no paradigma de agentes inteligentes com a capacidade de interoperabilidade em diversos ambientes. É totalmente desenvolvido em linguagem Java, dependendo apenas da JVM (Java Virtual Machine) instalada em cada estação que deseja executar aplicações baseadas neste framework. Ele é absolutamente compatível com a arquitetura FIPA (Foudation for Inteligent Physical Agents) para desenvolvimento de agentes, tornando-se cada vez mais aceito pela comunidade científica como uma plataforma confiável para o que se propõe (Silva, 2003).

JADE é um projeto baseado na licença LGPL (Lesser General Public License), disponível então para uso e distribuição gratuita, com permissão para alterações seguindo-se as restrições da licença.

Seguem abaixo algumas características do JADE para o desenvolvimento de sistemas multiagentes:

i) **Plataforma distribuída de agentes:** JADE pode ser dividido em hosts, onde apenas uma aplicação Java e uma JVM é executada em cada host. Cada agente é implementado como uma thread Java e colocado dentro de containers (Agent Containers);

ii) **Graphical User Interface:** Uma interface que gerencia agentes e containers;

iii) **Ferramentas de Debugging**: Ferramentas que auxiliam o desenvolvimento e depuração de sistemas multiagentes baseados em JADE;

iv) **Capacidade Multithreading**: Possibilitada através dos modelos de comportamentos (behaviors);

v) **Compatibilidade com a FIPA**: Composto pelo Sistema Gerenciador de Agentes (SGA), o Diretório Facilitador (DF), e o Canal de Comunicação dos Agentes (CCA), carregados quando o ambiente é inicializado.

vi) **Transporte de mensagens**: Transporte de mensagens no formato FIPA-ACL;

vii) **Biblioteca de protocolos FIPA**: Protocolos prontos para uso;

viii) **Automação de registros**: Abstraindo para o desenvolvedor todo o serviço de registro e cancelamento de agentes;

ix) **Serviço de nomes em conformidade com os padrões da FIPA**: Todos os agentes, ao serem inicializados, são identificados por um identificador único (GUID – Global Unique Identifier);

x) **Integração**: Permite a aplicações externas carregarem agentes autônomos em JADE.

O Framework JADE possui ainda um conjunto bastante útil de ferramentas que possibilitam o gerenciamento de sistemas multiagentes.

De acordo com a definição do padrão FIPA, a Plataforma de Agentes do JADE possui (cf. Figura 4-2):

i) **Sistema Gerenciador de Agentes (SGA)**: é o agente que gerencia o acesso e o uso da plataforma de agentes. Só existe um agente deste tipo por plataforma, responsável pelo provimento do guia de endereços e controle de ciclo de vida, mantendo um diretório de identificadores e estados de agentes;

ii) **Diretório Facilitador (DF)**: é o agente que provê o serviço de páginas amarelas na plataforma;

iii) **Sistema de Transporte de Mensagens (STM)**: é o canal de comunicação entre os agentes. Ele é o agente responsável pela comunicação que ocorre entre todos os agentes, dentro e fora da plataforma. O Gerenciador de Agentes e o Diretório Facilitador também se comunicam através dele.

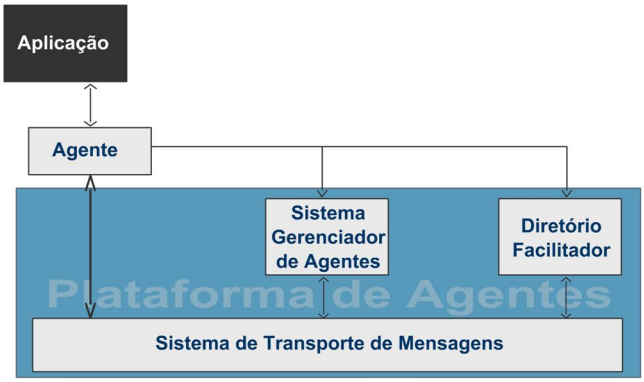

Figura 4-2- Plataforma de Agentes definido pela FIPA

Portanto, o JADE é compatível com a arquitetura definida pela FIPA. O Gerenciador de Agentes e o Diretório Facilitador são criados automaticamente e o Sistema de transporte configurado para atender aos requisitos de comunicação descritos.

O JADE permite a utilização de várias máquinas virtuais Java distribuídas por diversos *hosts*. Como este framework necessita apenas da máquina virtual instalada no *host*, ele independe do Sistema Operacional. Como pode ser visto na Figura 4-3, o JADE ainda contempla o uso de diversos outros dispositivos capazes de rodar uma JVM ou KVM. KVM é um conjunto menor de bibliotecas para dispositivos portáteis. A portabilidade da API possibilita o uso do JADE em dispositivos móveis (celulares, pagers, PDAs, etc.), inclusive integrando-os através de redes *wireless*.

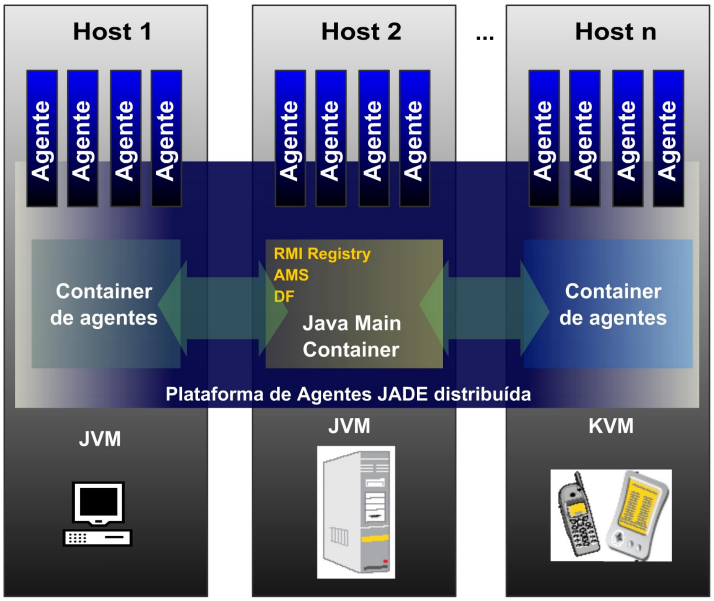

Figura 4-3 - Plataforma de Agentes JADE distribuída em diversos containers

Em cada JVM (ou KVM) temos um *container* de agentes que deve fornecer um ambiente para execução desses agentes, além de permitir que vários agentes compartilhem o mesmo processador em um *host*. Dessa forma, deve existir apenas uma JVM por *host*, sendo possível vários agentes para cada JVM.

O *Java Main Container* é o container onde se encontra o SGA, o DF e o Registro RMI, para invocação remota de métodos. Tal registro não passa de um repositório de nomes que o Java utiliza para registrar e recuperar referências a objetos/containers de agentes que se conectam à plataforma. Os outros *containers* de agentes se conectam ao *Java Main Container* de forma abstrata para o desenvolvedor.

O JADE possui comportamentos comuns para a maioria das tarefas necessárias para a construção de agentes. Entre elas, pode-se destacar o envio e o recebimento de mensagens e a construção de tarefas mais complexas. Os comportamentos mais complexos, normalmente, são divididos em comportamentos mais simples, conforme pode ser observado na Figura 4-4.

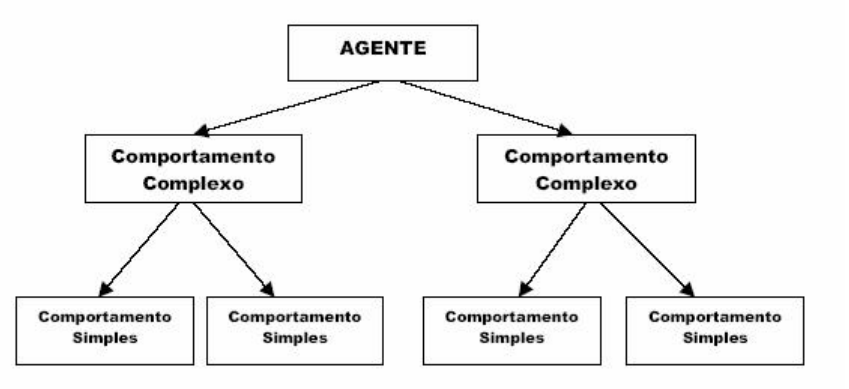

Figura 4-4 - Comportamentos de um agente em JADE

4.4.3 Criação de Ontologias em JADE utilizando o PROTÉGÉ

O ambiente para criação de ontologias Protégé 3.0, através do plug-in BeanGenerator, possibilita a conversão das ontologias criadas em JavaBeans de acordo com a API JADE. Com esta possibilidade torna-se relativamente fácil a criação de modelos de domínio que possam ser acessados por agentes construídos na plataforma JADE.

O plugin BeanGenerator é um Tab Plug-in que pode ser executado no ambiente Protégé. Através deste plugin, pode-se criar ontologias em diversas linguagens (como OWL) e exportá-las para a ferramenta JADE (cf. Figura 4-5).

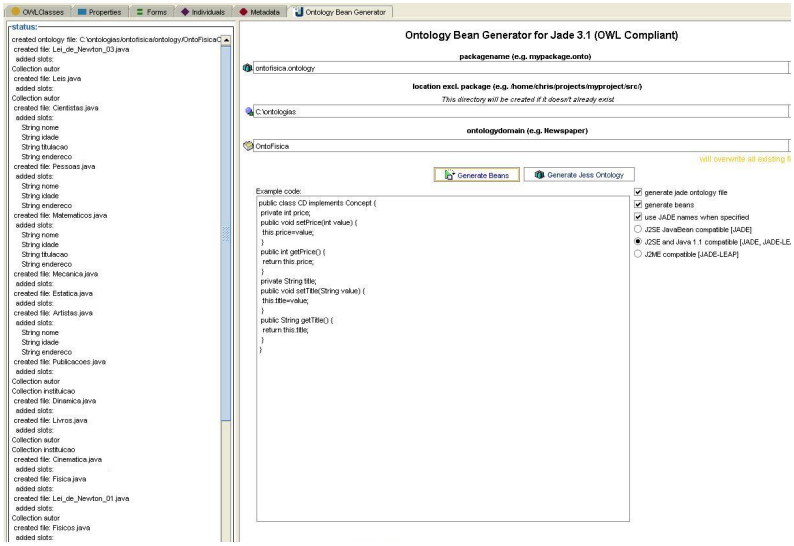

Figura 4-5 - O Tab Plug-In Beangenerator

Cada classe criada no Protégé é mapeada para uma correspondente em Java. A estrutura taxonômica é mapeada em relacionamentos de herança em Java. Os Slots são mapeados em propriedades em Java, associados com seus devidos tipos. Caso seja uma instância, ela será associada a uma instância da classe gerada em Java.

5 AMBIENTE NETCLASS

O Ambiente NETCLASS une as idéias que fundamentam os STI ao paradigma de Aprendizagem Cooperativa, definindo um Ambiente de Ensino Inteligente Cooperativo Computadorizado (Labidi et al., 2003a), conforme ilustrado na Figura 5-1.

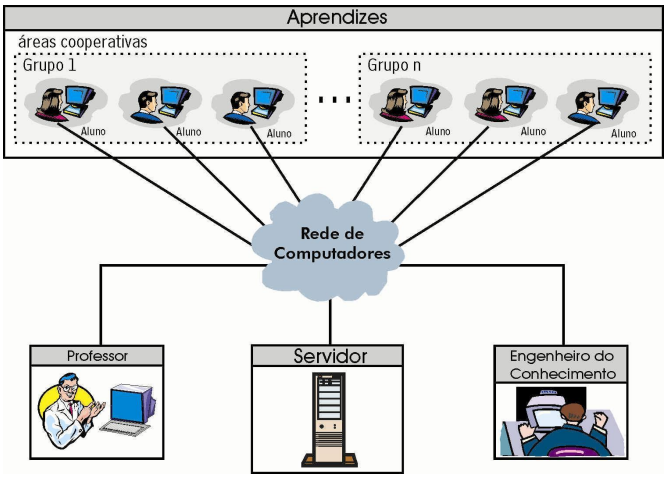

Figura 5-1 Ambiente NetClass

O núcleo do Ambiente NETCLASS encontra-se instalado em um servidor que, operacionalmente, poderá estar distribuído pela rede. Aos serviços disponíveis no servidor, chamamos de Sistema NETCLASS[1].

No Ambiente NETCLASS, os alunos são divididos em grupos distintos, chamados de áreas cooperativas. Eles cooperam e aprendem a partir da interação dentro de seus próprios grupos (interação intragrupo) com o sistema, com o professor, e com os outros grupos (interação intergrupo), através da utilização de recursos multimídia e da tecnologia de redes.

Um grupo ou área cooperativa é formado por três alunos que cooperam e interagem entre si e com o sistema através de terminais. Esses alunos podem estar separados fisicamente e, sendo assim, poderão realizar as interações através de recursos de comunicação de rede (salas de conversação, vídeo-conferência, correio eletrônico, banco de dúvidas, etc.). Dessa

[1] O termo NetClass pode ser entendido como sendo o ambiente de aprendizagem ou o sistema computacional, propriamente dito. No entanto, o contexto do uso de tal termo deixará claro ao que se refere.

maneira, os aprendizes podem realizar atividades sugeridas pelo sistema ou professor, resolver problemas, solicitar informações ou tirar dúvidas, entre outras atividades.

O professor interagindo com um terminal é considerado uma área cooperativa específica. Com o auxílio do sistema, o professor desempenha várias tarefas, visando eficiência no ensino. Por exemplo, o professor utiliza o NETCLASS para formar os grupos de alunos com base em critérios pedagógicos e em informações sobre os aprendizes, determinar alguns parâmetros básicos para o ensino do conteúdo (seqüência de atividades, tempo alocado para cada atividade) e outros parâmetros que auxiliarão o sistema no processo de avaliação dos aprendizes.

Em uma Sessão de Aprendizagem, ocorre a interação entre várias áreas cooperativas interligadas através da rede, objetivando o aprendizado por parte dos alunos. Nas etapas da aprendizagem, têm-se vários tipos de atividades sendo desempenhadas pelos aprendizes. Na Seção 5.3, serão detalhados esses tipos de atividades pedagógicas.

5.1 Arquitetura Multiagentes NETCLASS

O NETCLASS é baseado em uma arquitetura composta por agentes artificiais inteligentes (Agente Tutor, Agente Estrategista, Agente de Modelagem do Aprendiz, Agentes de domínio, Agentes de Busca, etc.) e humanos (alunos/grupos, engenheiro do conhecimento e professor) (Labidi et al., 2000). Antes do detalhamento desta arquitetura, será apresentado o conceito de sistemas multiagentes conforme utilizado neste trabalho.

5.1.1 Agentes do Ambiente NETCLASS

A arquitetura NETCLASS é composta por vários agentes humanos e artificiais. Na Figura 5-2 são mostrados os principais agentes que compõem essa arquitetura. Além destes, outros agentes secundários fazem parte do sistema.

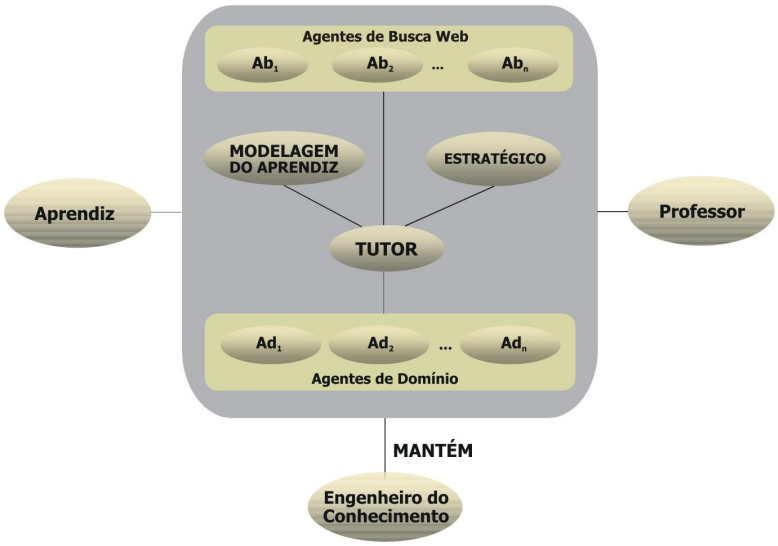

Figura 5-2 Arquitetura multiagentes NETCLASS

a) Agentes Artificiais

- **Agente Tutor**: participa de todas as atividades de ensino. Ele interage com os principais agentes da arquitetura. a fim de apresentar aos aprendizes o conteúdo e tarefas adequados para cada um deles. É responsável pelo controle das interações no sistema durante o processo de ensino-aprendizagem. Além disso, fornece dicas e ajuda o aprendiz no momento da aplicação do assunto.

- **Agentes de Domínio:** um Agente de Domínio é responsável pela representação de um conhecimento específico, ou melhor, representação de um subdomínio do domínio em estudo. Na realidade, existem vários Agentes de Domínio, todos possuindo a mesma estrutura, diferenciando-se uns dos outros apenas pelo seu conhecimento específico. Eles mantêm *links* para recursos armazenados no servidor. Esses recursos podem disponibilizar vários tipos de mídia.

- **Agentes de Busca Web:** um Agente de Busca Web, também chamado simplesmente de Agente de Busca, é um correspondente de um Agente de Domínio, representando Índices Web do seu subdomínio. Os Agentes de Busca mantêm *links* para recursos localizados na Web.

- **Agente de Modelagem do Aprendiz**: é responsável pelo processo de aquisição, representação e manutenção de informações sobre aluno e grupos durante o processo de ensino-aprendizagem.

- **Agente Estrategista**: interage com o Agente Tutor a fim de definir as estratégias pedagógicas mais adequadas a serem adotadas em suas atividades. Por exemplo, o Agente Estrategista decide quais unidades de conhecimento e em que formato o conteúdo será apresentado aos aprendizes.

b) Agentes Humanos

- **Professor**: agente humano que pode assumir diferentes papéis no sistema, dentre os quais, destacam-se os de especialista, orientador e avaliador. Entre outras funcionalidades, o professor poderá: i) Fornecer ao engenheiro do conhecimento o conteúdo a ser ensinado aos aprendizes; ii) Com o auxílio do sistema, formar grupos e reorganizá-los, quando for necessário; iii) Modificar a estratégia pedagógica adotada; iv) Supervisionar e interagir com as áreas cooperativas, monitorando a apresentação dos conteúdos, discutindo e esclarecendo dúvidas dos aprendizes; v) Avaliar os aprendizes, tendo como base as informações do modelo do aprendiz.

- **Aprendizes**: são os alunos, os quais podem ser organizados em grupos. Cada aluno necessita está inserido em um grupo para que possa participar das sessões de aprendizagem. Ele está ligado ao sistema através de um computador e pode estar ou não separado fisicamente dos demais integrantes do seu grupo.

- **Engenheiro do Conhecimento**: é responsável pela manutenção dos agentes da arquitetura. Ele inclui e realiza a edição do conhecimento de cada Agente de Domínio e a organização de seu respectivo subdomínio, bem como dos Agentes de Busca.

Nas próximas seções, faz-se uma breve descrição do Agente Tutor, Agente de Modelagem do Aprendiz, Agente Estrategista e Agentes de Domínio. No próximo capítulo, descreve-se os Agentes de Busca, objetivo principal deste trabalho.

5.2 Agente Tutor

O Tutor Artificial tem o papel de apresentar os conhecimentos ao alunos, dirigir a fase de aplicação do conhecimento, adotar as melhores estratégias pedagógicas e realizar a avaliação dos alunos/grupos. Ele também tem a responsabilidade de controlar as interações dos grupos de estudantes com o sistema durante progresso da aprendizagem. As ações do Tutor dependem do comportamento dos aprendizes. Ele interage com o Agente Estrategista e o Agente de Modelagem do Aprendiz para selecionar as estratégias de ensino mais adequadas. Portanto, este agente determina o que será ensinado, quando e como será feito.

5.3 Agente de Modelagem do Aprendiz

O Agente de Modelagem do Aprendiz é o componente do NetClass responsável por adquirir, manter e representar as informações sobre estudantes individuais e grupos (Serra Jr, 2001).

O Agente de Modelagem é composto por quatro principais classes: `Aquisicao`, `Manutencao`, `Consulta` e `MA` (modelo do aprendiz). As classes `Aquisicao` e `MA` são especializadas em duas outras: uma para informação sobre os aprendizes individuais e outra para grupos de indivíduos. A Figura 5-3 ilustra a arquitetura do Agente de Modelagem do Aprendiz.

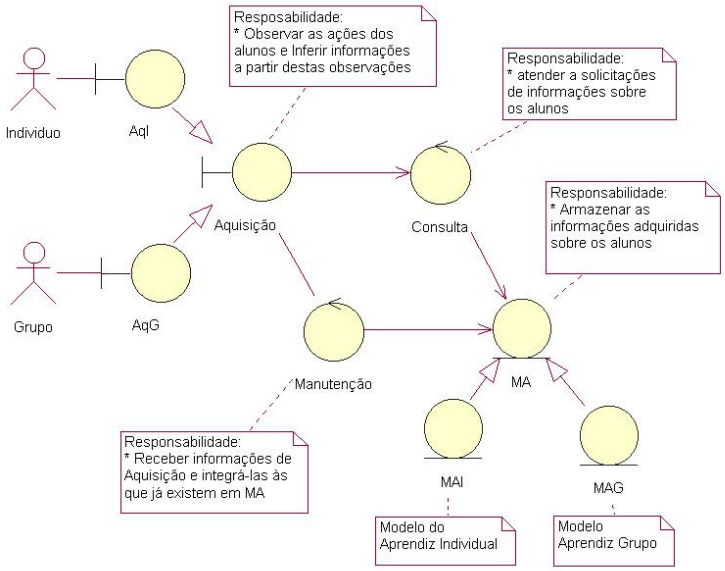

Figura 5-3 - Estrutura do Agente de Modelagem do Aprendiz

De forma geral, as responsabilidades de cada uma destas classes são:

i) `Consulta`: atender às solicitações de informações sobre os aprendizes;

ii) `MA`: armazenar as informações adquiridas sobre os aprendizes. Conforme pode ser visto na Figura 5-3, a classe MAI armazena as informações relativas a um modelo de aprendiz individual e a classe MAG armazena informações relativas a um grupo de alunos;

iii) `Aquisicao`: observar as ações dos alunos e inferir informações a partir destas;

iv) `Manutencao`: receber informações da classe Aquisicao e integrá-las às já existentes no modelo do aprendiz.

Quando um agente necessita de informações sobre um aprendiz, sejam essas informações de um aprendiz individual ou de grupo, ele as solicita ao Agente de Modelagem do Aprendiz e essa interação se dá por meio da classe `Consulta`. A classe `Consulta` é

responsável por atender a todas as solicitações de consultas feitas por outros agentes do sistema.

A classe `Aquisicao` é responsável pelo processo de aquisição das informações sobre os aprendizes a serem representadas no modelo do aprendiz. O processo de aquisição em modelagem do aprendiz pode se dar de forma direta ou indireta. Portanto, a classe `Aquisicao` é especializada nas classes `AquisicaooDireta` e `AquisicaoIndireta`. A classe `AquisicaoDireta` encapsula os métodos e técnicas específicos para a realização da aquisição direta de informações sobre o aprendiz. No caso da classe `AquisicaoIndireta`, os métodos e técnicas específicos servem para realizar a aquisição indireta de informações sobre um aprendiz. De forma similar, a classe `MA` encontra-se especializada em classes `MAI` e `MAG`.

Na aquisição direta, o Agente de Modelagem do Aprendiz interage diretamente com o aprendiz através da classe de interface `AquisicaoDireta`. Quando uma informação específica sobre o aprendiz é necessária, o Agente de Modelagem pode apresenta-se solicitando esta informação diretamente por meio desta classe. Essa classe também pode ser utilizada no momento em que o aprendiz deseja alterar alguns atributos pessoais contidos no modelo do aprendiz. O objeto da classe `Aquisicao` repassa as informações adquiridas para o objeto da classe `Manutencao` para que este atualize as informações presentes no modelo do aprendiz.

A classe `Manutenção`, de posse das informações coletadas pela classe `AquisicaoDireta`, atualiza as informações no modelo do aprendiz em três passos: 1) verifica as informações presentes no modelo; 2) integra as informações atuais às novas informações; e 3) armazena o resultado da integração.

Na aquisição indireta, o Agente de Modelagem do Aprendiz irá inferir informações sobre o aprendiz de forma indireta, a partir de suas ações nas interfaces do sistema. Isto irá ocorrer principalmente durante a atividade de Aplicação do Conhecimento, quando o aprendiz realizará atividades de resolução de problemas.

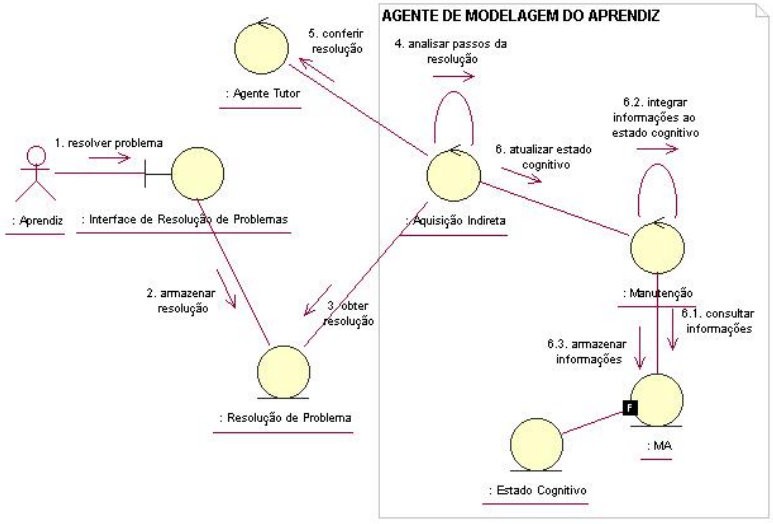

Figura 5-4 Aquisição indireta de informações sobre o aprendiz

Na Figura 5-4 é apresentado um diagrama de colaboração, onde se pode observar as principais interações com a classe `AquisicaoIndireta`.

A exemplo do que ocorre com a aquisição direta, na aquisição indireta, a classe `Manutenção`, em três passos, irá atualizar as informações no modelo do aprendiz: 1) verifica as informações presentes no modelo sobre o estado cognitivo do aprendiz; 2) integra as informações atuais às novas informações; e 3) armazena o resultado da integração.

Conforme pôde se obervar, a classe `Manutencao` trabalha constantemente com a classe `Aquisicao`. Sua responsabilidade é consultar, integrar e armazenar novas informações ao modelo do aprendiz. A manutenção do modelo do aprendiz é feita com o uso da técnica de sobreposição difusa.

5.4 Agente Estrategista

O Agente Estrategista mantém as estratégias a serem adotadas pelo Agente Tutor para cada aprendiz. Por exemplo, o Estrategista define a seqüência de apresentação dos conhecimentos, conforme prévia definição feita pelo professor para cada domínio.

As atividades de ensino-aprendizagem, no sistema NetClass, são classificadas em seis tipos (cf. Figura 5-5). Cada tipo de atividade tem funções específicas que são desempenhadas com o uso de estratégias pedagógicas apropriadas, escolhidas de acordo com o modelo do aprendiz.

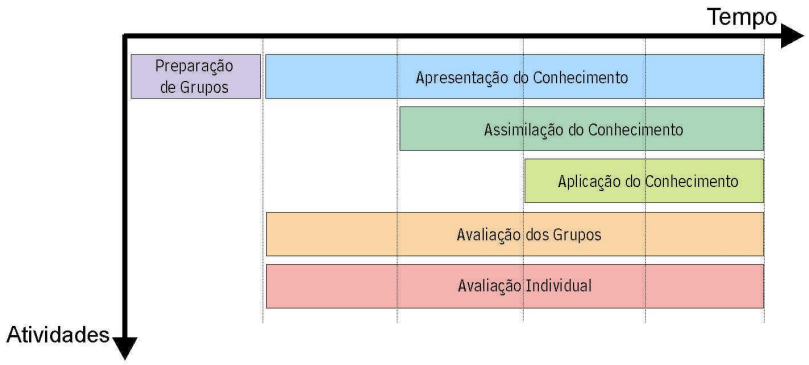

Figura 5-5 Tipos de atividades pedagógicas

Antes de iniciar as sessões de ensino-aprendizagem, o aprendiz preenche alguns formulários, fornecendo ao sistema informações pessoais, preferências e atributos relevantes para a formação dos grupos. Em seguida, é cadastrado no sistema, e finalmente, recebe instruções sobre utilização da interface a ele disponível. Após essas preliminares, o sistema dará início às atividades pedagógicas.

Preparação de Grupos: ocorre sempre no início de uma sessão de aprendizagem, com a finalidade de preparar os grupos para as atividades de ensino-aprendizagem (Serra Jr et al., 2001a). Nesse momento, o professor, com o auxílio do sistema e baseado nas informações dos aprendizes, poderá organizar os alunos em grupos.

Apresentação do Conhecimento: corresponde à apresentação do conteúdo aos alunos através dos terminais. O Agente Estrategista define o conteúdo e a estratégia específica de apresentação mais adequados, com base no perfil de cada aprendiz.

Assimilação do Conhecimento: este é o primeiro tipo de atividade realmente cooperativa, onde, os alunos, organizados em grupos, interagem entre si com o objetivo de assimilar o conhecimento apresentado.

Aplicação do Conhecimento: nesta fase, os alunos trabalham cooperativamente na realização de tarefas, como por exemplo, resolução de problemas (Labidi et al., 2004).

Avaliação do Grupo: a atividade de avaliação é iniciada desde o momento da realização da primeira atividade de ensino-aprendizagem. No entanto, é nas atividades de Avaliação de Grupo e Avaliação Individual que acontece a Avaliação Final de Sessão. Todas as atividades de ensino-aprendizagem (Apresentação, Assimilação e Aplicação do Conhecimento) auxiliam o sistema na composição do processo avaliativo.

Avaliação Individual: o sistema realiza a avaliação individual de forma semelhante à Avaliação do Grupo, com a diferença que o sistema avalia o aluno individualmente e não em grupo.

Mesmo que as pessoas possuam o mesmo nível de conhecimento, normalmente, o estilo de aprendizagem de cada um é diferente. Alguns aprendem mais facilmente participando de atividades de discussão, outros por meio de observações, outros através de exemplos, analogias, etc. Para adaptar o ensino ao estilo de aprendizagem de cada aluno ou grupo são utilizadas as estratégias pedagógicas. Estratégias pedagógicas são métodos e técnicas pedagógicos utilizados pelo professor, objetivando uma maior eficiência no processo de ensino-aprendizagem, frente às diferenças comportamentais dos aprendizes.

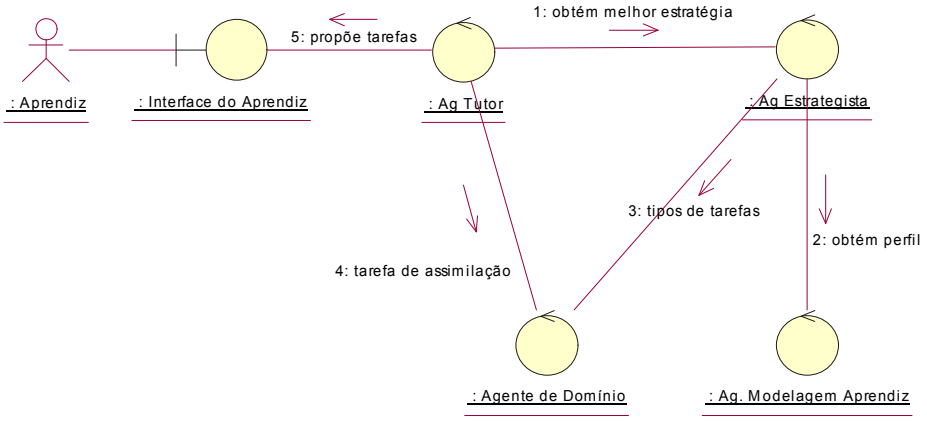

Figura 5-6 - Diagrama de Colaboração na escolha da estratégia

O NetClass possui a possibilidade de múltiplas estratégias pedagógicas, como também introduz o conceito de Agente Estrategista que interage com o Agente de Modelagem do

Aprendiz para selecionar a estratégia mais adequada de ensino com base no modelo do aprendiz.

Existem dois tipos de estratégias utilizadas pelo Agente Estrategista NetClass: **1) Estratégia Global**: estratégia definida ou modificada pelo professor nos intervalos entre as sessões de ensino-aprendizagem; e **2) Estratégias Específicas de Atividades**: estratégias que podem ser modificadas pelo Agente Estrategista ou professor durante as atividades em uma sessão.

A Estratégia Global faz parte do planejamento pedagógico realizado antes do início de uma sessão de ensino-aprendizagem. Essa estratégia diz respeito à seqüência das atividades pedagógicas e ao tempo alocado para a realização de cada uma delas.

Durante as atividades de ensino-aprendizagem e avaliação, muitas estratégias podem ser adotadas. No entanto, o uso destas limita-se aos recursos disponíveis em um ambiente de aprendizagem. As estratégias específicas que podem ser adotadas pelo Agente Estrategista dependem do tipo de atividade.

Cada estratégia pedagógica possui vantagens peculiares no processo de ensino-aprendizagem. Elas devem ser escolhidas adequadamente, de acordo com o estilo de aprendizagem de cada aprendiz, visando a eficiência durante o processo de ensino. Para isso, critérios devem ser estabelecidos com base no tipo de domínio, na forma de ensinar o domínio e nas informações que o sistema tem sobre os aprendizes.

A escolha das Estratégias Específicas é responsabilidade do Agente Estrategista. No entanto, o professor poderá a qualquer momento modificar uma estratégia específica, caso discorde da decisão do Agente Estrategista em razão de perceber que o andamento do curso não está sendo satisfatório. O Agente Estrategista seleciona essas estratégias com base em dois critérios: i) **Domínio a ser ensinado -** a escolha das estratégias está estritamente relacionada ao tipo de conteúdo a ser ensinado. Ao criar o modelo do domínio, o professor define preferências para determinadas estratégias específicas; ii) **Informações sobre o aprendiz -** o Agente Estrategista utiliza informações contidas no modelo do aprendiz.

5.5 Agentes de Domínio

O domínio de conhecimento a ser ensinado no NetClass é organizado em uma base de conhecimentos compartilhada pelos Agentes de Domínio. Cada Agente de Domínio NetClass mantém um domínio específico, chamado de unidade pedagógica.

As unidades pedagógicas são organizadas de acordo com a estrutura pedagógica estabelecida para o domínio a ser ensinado. Estrutura pedagógica é uma organização do conhecimento em critérios de pré-requisitos que visam estabelecer possíveis seqüências nas quais o conteúdo pode ser ensinado.

Uma seqüência de aprendizagem é uma seqüência na qual as unidades pedagógicas podem ser dispostas (para estudo) respeitando a ordem de pré-requisitos imposta pela estrutura pedagógica, conforme Figura 5-7.

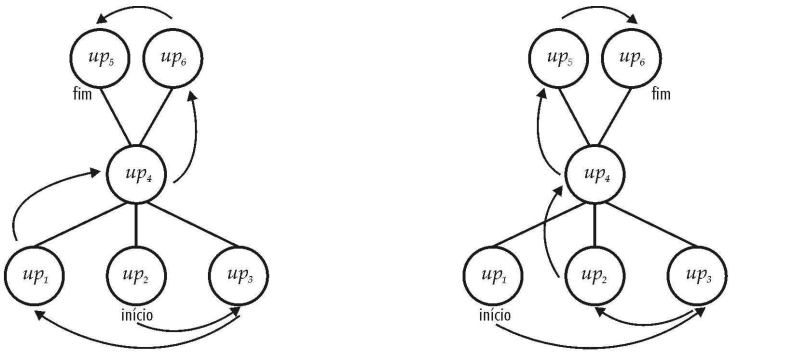

Figura 5-7 - Exemplos de seqüências de aprendizagem

Um Agente de Domínio, a partir das ontologias e instâncias, mantém uma unidade pedagógica como sendo uma visão seqüenciada dos conceitos que devem ser apresentados ao aprendiz.

Cada conceito poderá estar associado a uma instância. Cada instância poderá possuir vários recursos. Recursos são os itens de multimídia (texto, áudio, vídeo) apresentados para introduzir ou reforçar um conhecimento. Os Agentes de Domínio não mantêm as instâncias das ontologias. As instâncias são mantidas na Base de Conhecimentos.

Utiliza-se o Ambiente Protégé para a criação das ontologias. A partir da API JADE-FIPA está sendo implementado um Módulo de Manutenção da Sociedade de Agentes que permitirá, ao Engenheiro do Conhecimento, criar, alterar ou excluir agentes da Sociedade de Agentes de Domínio, conforme Figura 5-8. Na criação de um Agente de Domínio, deve-se definir quais conceitos (unidade pedagógica) do domínio esse irá manter e em qual seqüência.

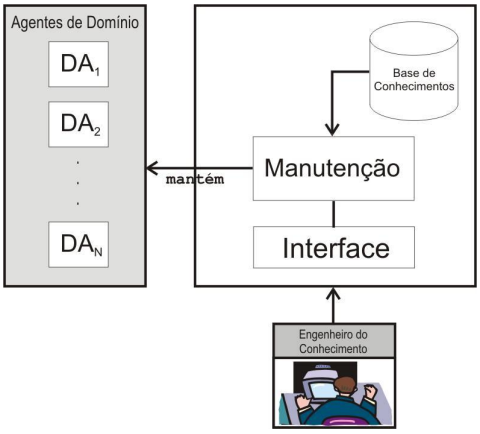

Figura 5-8 - Manutenção dos Agentes de Domínio

Os Agentes de Domínio organizam-se de forma distribuída, ou seja, localizam-se em vários *hosts*, independentemente do Sistema Operacional, possibilitando o uso de dispositivos portáteis nessa arquitetura, inclusive integrando-os através de redes wireless.

6 AGENTES DE BUSCA DO NETCLASS

No NetClass, o Agente Tutor poderá apresentar informações para o aprendiz com base em duas fontes de informações: Agentes de Domínio e Agentes de Busca. Os Agentes de Domínio possuem uma base de conhecimentos baseada em ontologias, conforme visto na Seção 5.5.

Os Agentes de Busca utilizam as mesmas ontologias do Modelo do Domínio, mas recuperam informações da Web. Entretanto, os Agentes de Busca não armazenam recursos Web, mantêm apenas endereços de páginas que podem ser acessados pelo aprendiz.

Em (Nunes, 2001) é definido um Serviço de Busca, utilizando o paradigma de Agentes Móveis no ambiente MATHNET de Ensino-Aprendizagem Cooperativo à Distância. Apesar do ambiente MATHNET possuir arquitetura bastante semelhante ao NetClass, o segundo diferencia-se por utilizar ontologias para representar conhecimentos e recuperar informações da Web. No MATHNET, o aprendiz poderá tirar dúvidas recuperando informações com o auxílio de um serviço de busca baseado em agentes móveis. Mas a busca feita por esse serviço não leva em consideração a contextualização da dúvida do aprendiz no que diz respeito ao uso de uma base ontológica. O Serviço de Busca no MATHNET consulta as informações armazenadas em uma base de recursos que é utilizada para o armazenamento e manutenção dos recursos didáticos (aulas, apontamentos, exercícios, atividades etc.). No NetClass, recupera-se apenas o *link*, os recursos didáticos são mantidos pelos Agentes de Domínio como instâncias das ontologias criadas pelo Engenheiro do Conhecimento para cada subdomínio.

No NetClass, um Agente de Busca, uma vez instanciado, utiliza mecanismos de busca e interage com os outros Agentes de Busca com a finalidade de realizar busca na Web. As informações recuperadas encontrar-se-ão na forma de índices (*links*) e poderão ser apresentadas ao aprendiz, através do Agente Tutor. Observe na Figura 6-1 que o aprendiz interage com o Tutor através da Interface do Aprendiz.

Muitas são as classes envolvidas no processo de ensino-aprendizagem, as principais são (cf. Figura 6-1): `AgenteTutor`, `AgenteBusca`, `AgenteDominio`, `AgenteModelagemAprendiz`, etc. A `AgenteBusca` interage principalmente com a

`AgenteTutor` com a finalidade de atender às demandas de busca na Web feitas pelo andamento do processo de aprendizagem mediado pelo Tutor Artificial.

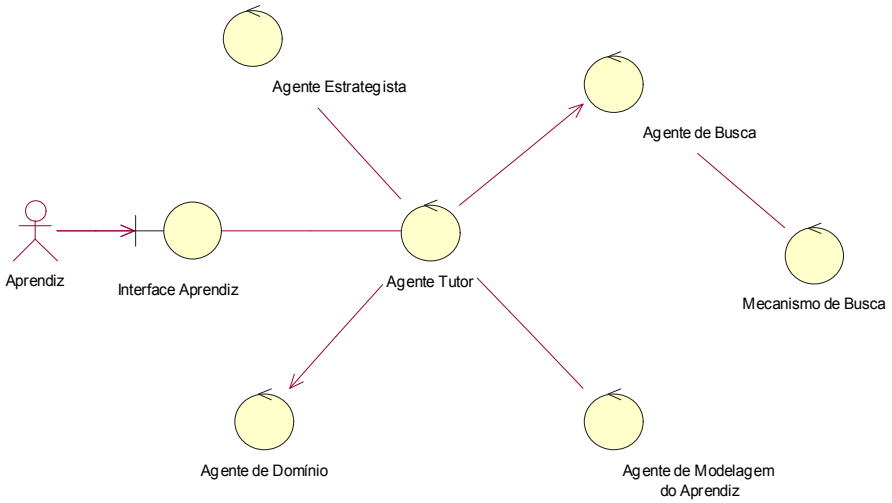

Figura 6-1 – Visão Geral – Classes NetClass

6.1 Arquitetura da Busca

Os Agentes de Busca se conectam a mecanismos de busca, como, por exemplo, Google, Altavista, Excite e outros, aproveitando os seus índices, uma vez que não é necessário sempre indexar ou percorrer toda a Web para recuperar informações. A arquitetura de busca do NetClass é apresentada na Figura 6-2.

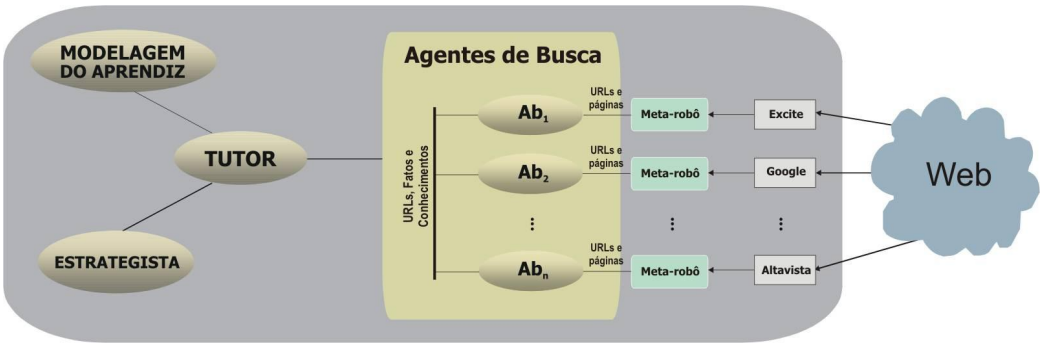

Figura 6-2 – Arquitetura de busca do NetClass

Um Agente de Busca efetua uma consulta ao mecanismo de busca com palavras-chave. O resultado são endereços de páginas Web, algumas com conteúdo relevante e outras irrelevante. A partir da ontologia, regras e fatos mantidos pelo agente, este é capaz de discernir sobre os índices que são úteis para o seu domínio específico e os que não são. Para isso, são utilizados métodos de categorização de recuperação de informação. Uma vez que os agentes possuirão responsabilidades distintas, praticamente sem interseção, cooperando uns com os outros pela troca de sugestões de endereços de páginas candidatas a membros das classes que processam, a arquitetura baseia-se na abordagem de Resolução Distribuída de Problemas (RDP) (Oates et al., 1994). Os Agentes de Busca trocam mensagens contendo regras de reconhecimento e fatos (conhecimento dos agentes), além das URLs (dados). Portanto, os resultados da busca poderão conter páginas de conteúdo desprezível para um agente, mas relevante para um outro, sendo assim, os Agentes de Busca podem comunicar-se com outros agentes a fim de trocar informações com um agente responsável por um outro domínio específico.

Conforme pode ser observado na Figura 6-3, a estrutura estática da classe que implementa um Agente de Busca é composta, principalmente, por: `MetaRobo`, `Validar`, `PreProcessar`, `Classificar`, `Consultar`, `Rejeitadas`, `Classificadas`, `URLsAlta`, `URLsBaixa`.

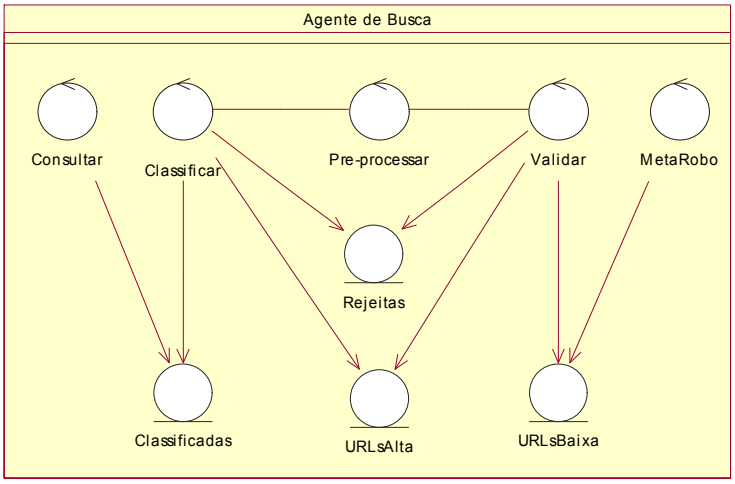

Figura 6-3 - Composição de um Agente de Busca

As ações relacionadas às classes `MetaRobo`, `Validar`, `PreProcessar` e `Classificar` serão detalhas nas próximas seções.

Cada Agente de Busca é um especialista no reconhecimento de páginas que correspondem a instâncias de classes que ele processa (p. ex., páginas sobre assunto de vertebrados) e na extração de atributos dessa entidade (p. ex., fonte da informação, instituição, professor, etc), procurando também identificar páginas e ponteiros úteis a outros agentes. A recomendação que um agente faz a um outro é feita através do armazenamento na fila de URLs de Alta Prioridade.

6.2 Mediação do Agente Tutor

O conhecimento mantido pelo NetClass pode ser visto a partir do conjunto das sociedades de agentes de software. Um agente, responsável por um domínio específico, é parte do todo e deve ser visto como uma entidade social. Portanto, percebe-se a necessidade de um agente mediador, chamado no NetClass de Agente Tutor, com capacidade de:

i) Combinação ou integração de informação;

ii) Discernir o que será apresentado ao aprendiz;

iii) Transformação da informação para um formato compreensível ao usuário;

iv) Notificação, isto é, aviso ao usuário que há informação que lhe pode ser útil, entregando-a sob sua concordância.

A partir das interações com as sociedades (de Domínio e de Busca) e demais agentes (Modelagem do Aprendiz e Estrategista), o Tutor adota as decisões do andamento do curso.

A existência do Agente Tutor adiciona valor aos serviços fornecidos, atuando como uma recepção e interface amigável e interoperável dentro de uma arquitetura que reúne esses serviços, refinando os pedidos para o grau de granularidade adequado, traduzindo-os e otimizando-os (Arens et al 1996). O Tutor poderá ainda executar tarefas ligadas à segurança e permissões, e até planejar a execução otimizada de funções típicas de agentes de informação, como busca, recuperação de informação e outras.

É mantida pelo Agente Tutor uma base de dados ontológica com a relação dos nomes de todos os Agentes de Busca e seus respectivos domínios específicos. Através desta, o Tutor tem condições de identificar o agente responsável pela busca de uma determinada classe de páginas (subdomínio).

O resultado que pode ser obtido do Tutor pelo Agente de Busca, correspondente ao domínio, será apresentado ao aprendiz como uma relação de endereços que podem ser acessados através da Interface do Aprendiz. O usuário poderá, depois de ter acessado o endereço, opinar sobre o quanto este endereço é relevante para o assunto que está sendo estudado. Cada endereço terá um grau de relevância. O agente poderá, a partir de regras, diminuir o grau de relevância da página ou descredenciá-la, considerando-a página rejeitada, principalmente, quando vários usuários a classificam como irrelevante. O Engenheiro do Conhecimento poderá comparar a opinião dos alunos com a dos agentes como uma forma de avaliar as regras definidas. Este poderá reavaliar as regras que estão sendo utilizadas pelo agente.

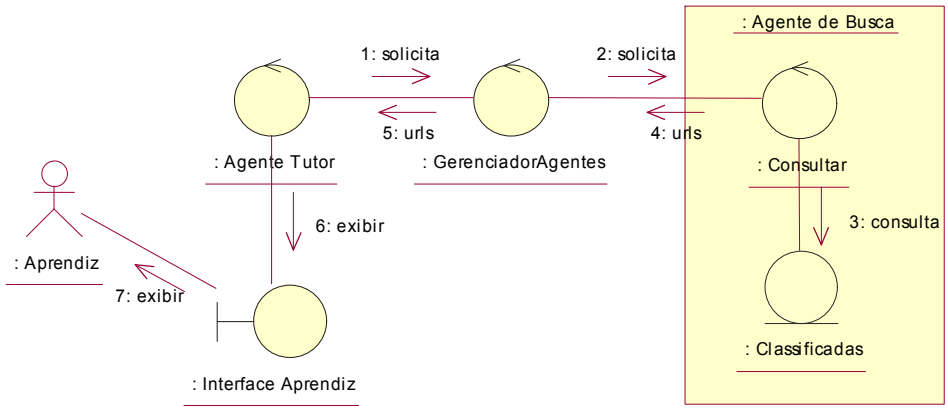

Figura 6-4 - Interações no Cenário Básico Tutor-AgenteBusca

Apresenta-se na Figura 6-4 um cenário básico das interações entre o Tutor e um Agente de Busca responsável por uma classe de páginas (subdomínio). Nesse aspecto dinâmico dos objetos, pode-se perceber que o Tutor toma a iniciativa de exibir ao Aprendiz uma relação de *links* relacionados ao assunto que está sendo estudado.

As interações entre os objetos que provêm o comportamento descrito no cenário básico das interações entre o Tutor e um Agente de Busca são:

1) O Agente Tutor solicita informações de um Agente de Busca relacionado ao assunto que está sendo estudado. O Tutor mantêm informação de qual Agente de Busca relaciona-se ao assunto e comunica-se com o Gerenciador de Agentes;

2) O Gerenciador de Agentes solicita ao agente que mantém a informação demanda. Este gerenciador é responsável pelo provimento do guia de endereços e controle de ciclo de vida, mantendo um diretório de identificadores e estados de agentes;

3) O Agente de Busca consulta a base de páginas classificadas;

4) O Agente de Busca envia para o Gerenciador a relação de URLs classificadas como pertinentes ao subdomínio, a fim de que sejam entregues ao Tutor;

5) O Gerenciador entrega ao Tutor a relação de URLs;

6) O Agente Tutor exibe na Interface do Aprendiz;

7) O Aprendiz poderá ver a relação de URLs relacionadas ao assunto que está sendo estudado.

6.3 Meta-robô

Uma questão relevante no desenvolvimento de sistemas para a Internet consiste na construção de robôs coletores. Todos os mecanismos de busca usam robôs para coletarem os documentos e os representarem com palavras-chave e suas respectivas freqüências em bases de índices, que servem para responder aos requisitantes (agentes artificiais ou humanos) quais documentos são mais relevantes às suas consultas. A proliferação destes robôs ameaçava inviabilizar o tráfego na rede e sobrecarregar os servidores durante a coleta das páginas para os índices. Para sanar o problema, foram criadas convenções para robôs "bem-comportados", que, entre outras características:

i) Aproveitam índices de outros mecanismos de busca e serviços de outros robôs, evitando redundância de esforços;

ii) Alternam vários servidores, evitando sobrecarga;

iii) Processam apenas os dados que interessam (tipos de arquivos, data dos arquivos, etc);

iv) Sabem fugir de loops na procura de ponteiros, e de ponteiros repetidos;

v) Podem ser controlados interativamente.

Um meta-robô é um robô que pode conectar-se a múltiplos mecanismos de busca - como Google, Altavista, Excite e outros – aproveitando os seus índices, uma vez que não é necessário, para a recuperação de página de uma classe, indexar ou percorrer toda a Web. O meta-robô proposto segue todas as diretrizes enumeradas acima. Ele funciona da seguinte forma: efetuam-se consultas aos mecanismos de busca com palavras-chave que garantam cobertura das páginas retornadas em relação à classe de páginas processada pelo agente. Por exemplo, o termo 'vertebrados' assegura a cobertura das páginas que tratam dos animais vertebrados. Devido à falta de precisão, o conjunto de páginas resultante das consultas recai em vários grupos funcionais além do grupo de páginas-conteúdo associado ao meta-robô, apresentando muitas listas, mensagens, páginas conteúdo de outras classes, e lixo.

Entende-se que os links sugeridos pelos outros agentes da Sociedade devem ser vistos com uma prioridade maior em relação aos recuperados pelos robôs. Portanto, cada agente mantém os links sugeridos em uma fila de alta prioridade e os links recuperados pelo robô em uma fila de baixa prioridade.

O uso de meta-robô é feito quando o agente não possui, em sua lista de links sugeridos por outros agentes, páginas a serem processadas, ou seja, ele só deve ser posto em ação caso não haja URLs na fila de alta prioridade.

6.4 Modelo de Interação

Os Agentes de Busca devem modificar o seu comportamento quando um novo agente é introduzido na sociedade. Este novo agente deve assemelhar-se aos da sociedade em funcionalidade e estrutura, mas não em conhecimento, exceto da ontologia usada na comunicação entre eles.

O modelo de interação dos agentes da arquitetura assemelha-se ao implementado em vários sistemas multiagentes, como o InfoSleuth (Bayardo Jr et al., 1996). Ao entrar no sistema, os agentes registram-se e anunciam-se aos outros agentes, mandando as regras gerais e específicas (para um agente específico) de reconhecimento de páginas e ponteiros úteis a si próprio, que serão empregadas pelos outros para lhe indicarem sugestões de páginas a processar. Além de recebê-las dos outros agentes, o novo agente receberá também, reciprocamente, as regras para identificar páginas e ponteiros úteis a eles. Assim, quando um agente acha informação que dispara alguma dessas regras referentes aos outros, este agente repassa a informação (ponteiro ou página e o conjunto de regras e conceitos usados para encontrá-la) ao agente que lhe enviou a(s) regra(s) disparada(s).

Este modelo de interação obedece ao teste de sociabilidade em seus dois critérios. No momento em que um novo agente entra no sistema, os outros agentes modificarão seu comportamento, tentando identificar informação útil para ser entregue a ele. Os agentes possuem ainda a mesma estrutura e código, o que acarreta várias vantagens de reuso e flexibilidade. Torna-se, inclusive, fácil modificar o comportamento de um agente ou de todo o sistema; pode-se incluir no ambiente um agente que anuncie novas regras para que os outros identifiquem páginas de suas próprias classes, para melhorias de performance ou inclusão de novos atributos a serem extraídos.

6.5 Conhecimento dos Agentes

O conhecimento dos agentes estará codificado em três ontologias e mais as bases de regras e funções. As ontologias encontram-se descritas logo abaixo:

1) Ontologia do domínio tratado, nos estudos de caso deste trabalho, a ontologia na área de Biologia;

2) Ontologia da Web, contendo, no mínimo, definições de hyperlink, termo e freqüência, e de página da Web em suas várias representações e atributos - como listas de palavras-chave e suas freqüências, ponteiros, e-mails e outros. Esta ontologia pode adicionalmente conter definições e instâncias de protocolos, tipos de arquivos e outros conceitos relativos à Internet.

3) Ontologias particulares de cada agente, que podem subdividir-se em:

i) Ontologias de outras áreas de conhecimento, porém de interesse específico do agente. Nem todo agente possui ontologias deste tipo;

ii) Ontologia principal do agente, que inclui todas as outras ontologias, usando-as tanto para como vocabulário de comunicação, como também para o desempenho de suas funções. Esta ontologia não possui classes, mas apenas instâncias de conceitos, casos, extratores e reconhecedores de classes e categorias funcionais.

O conjunto de regras pode ser dividido em dois grupos:

1) Regras de reconhecimento, sendo que um subconjunto mais simples das de reconhecimento será enviado aos outros agentes;

2) Regras de recomendações, que identificam páginas e ponteiros a serem sugeridos aos outros agentes, e que, em princípio, foram recebidas deles.

6.6 Tarefas dos Agentes

Como pode ser observado na Figura 6-5, cada agente desempenha três tarefas consecutivas no processamento de cada página: validação, pré-processamento e classificação. A tarefa de pré-processamento depende dos resultados da tarefa de validação, páginas inválidas (que não foram aprovadas na validação) não são processadas.

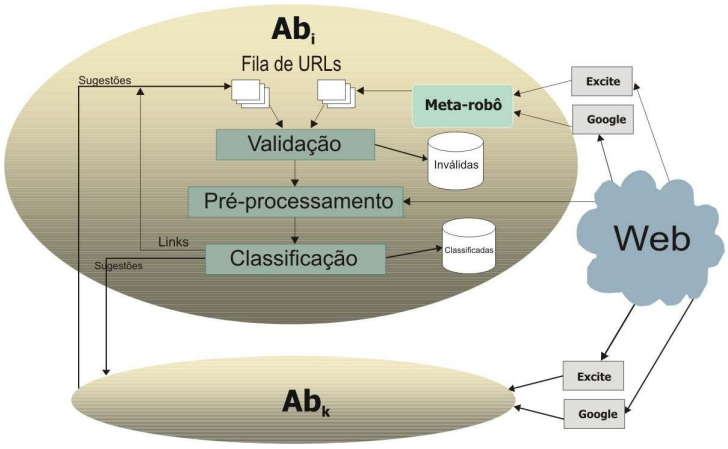

Figura 6-5 - Tarefas de um Agente de Busca

Cada agente manipula três filas de URLs durante sua execução: duas são processadas pelo próprio agente, uma de baixa e outra de alta prioridade, e a terceira contém sugestões de *links* para serem enviadas aos outros agentes.

O controle de chaveamento entre as fases de processamento é efetuado por uma classe da ontologia de manipulação integrada, o monitor de processamento. Esta classe contém o atributo Page-Status, que vai sendo modificado à medida que transcorre o processamento da página, assumindo sucessivamente os estados de ENFILEIRADA, VÁLIDA ou CLASSIFICADA. Ao final das inferências, caso a página não consiga ser reconhecida e classificada, ela estará categorizada funcionalmente como INVÁLIDA, LISTA (Lista Re-avaliável, ou seja, tratada como uma lista, onde cada frame é uma nova URL), RECOMENDAÇÃO (página reconhecida como pertencente à outra classe de páginas), INACESSÍVEL (é válida não pode ser recuperada para ser avaliada) ou REJEITADA (Mensagem ou Lixo). As próximas subseções apresentam descrições detalhadas dessas fases de processamento.

6.6.1 Validação

Nesta fase, concretiza-se a recomendação para os robôs de só recuperarem páginas escritas em formatos que os agentes possam processar. Com efeito, a validação elimina trabalho redundante, verificando se o *link* da página já está presente na base de *links*, seja ela válida ou não, indicando se já foram processadas, se está acessível, se o seu formato e protocolo são compatíveis, etc. Mesmo as páginas sendo inválidas ficam armazenadas na base de dados (endereço e data de acesso), porque o robô freqüentemente depara-se com ponteiros repetidos, só as recuperando novamente se sua data de última modificação foi alterada. A validação acelera todo o processamento, além de evitar trabalho redundante, economiza banda passante e tempo de processamento, gastos com a recuperação de páginas inúteis.

A maior parte das regras de validação é reusável por todos os agentes, mas regras específicas podem ser implementadas, referenciando outros campos, como data e tamanho, ou mesmo tratando outros protocolos.

6.6.2 Pré-processamento

A fase de pré-processamento tem por meta representar as páginas de diversas maneiras, com dados extraídos delas, aplicando, se necessário, técnicas de recuperação de informação e processamento de linguagem natural, como stop-lists, stemming e etiquetagem (*tagging*) (Appelt et al., 1999). Estes dados são repassados para o motor de inferência para que possam auxiliar a fase posterior (de reconhecimento e extração), e podem vir a ser enriquecidos com outros que venham a provarem-se úteis.

Os primeiros quatro itens a serem obtidos, através da conexão à URL, são Protocol, Content-Type, Page-Date e Length. Os outros itens são preenchidos durante o pré-processamento, depois da validação.

6.6.3 Classificação

Durante esta fase, o agente classifica as páginas em grupos funcionais, como LISTA, RECOMENDAÇÃO (página reconhecida como pertencente à outra classe de páginas), REJEITADA ou CLASSIFICADA (Reconhecida como membro da classe processada pelo agente ou da classe de outro agente do sistema). Esta tarefa precisa ser realizada com precisão, sob pena de perda de informação relevante (se as páginas membros não são reconhecidas), perda de tempo (se tentar extrair atributos a partir de páginas inúteis) e até mesmo de perda de consistência dos dados extraídos (se os atributos forem extraídos destas páginas irrelevantes). Particularmente, o reconhecimento de listas deve apresentar boa precisão, caso contrário, uma infinidade de endereços de importância duvidosa, advindos de falsas listas, serão inseridos na fila de processamento de alta prioridade.

As páginas reconhecidas nessa fase como sendo do sub-domínio do agente, têm seus atributos armazenados em uma base de índices (páginas classificadas), com um nível de relevância determinado pelo agente. Quando consultado, o agente irá responder com essas informações ao Tutor Artificial. Pode-se ainda considerar a possibilidade de armazenar em um *cache* cópias das páginas consideradas classificadas. Isso possibilitaria o acesso ao conteúdo do recurso, no caso do seu servidor está inoperante no momento da tentativa de acesso pelo aprendiz. Para este tipo de armazenamento, pode-se definir um tempo máximo (*timeout*) de armazenamento do recurso para evitar-se guardar conteúdos ultrapassados.

As páginas categorizadas como lixo, são armazenadas na base de páginas rejeitadas para evitar-se analisá-las em uma outra oportunidade, evitando-se desperdício de recursos computacionais. As páginas categorizadas como listas têm as URLs sendo enviadas para a lista de URls de Alta Prioridade. As páginas recomendações têm seus endereços enviados como sugestões para o agente correspondente.

Para realizar a categorização, palavras-chave da página ou de uma determinada região dela, são conferidas uma lista de termos dos dicionários, associados à respectiva tabela de categorias. Se uma das palavras-chave está contida num dos termos que designa a categoria, testa-se se o termo inteiro está na página ou região. Se isto ocorrer, isto indica que o atributo pertence àquela categoria, e, se for possível, múltiplas categorias, o processo continua até a última palavra da região ou página, caso contrário considera-se findo o processamento do atributo para a página.

Para detalhar melhor o processo de classificação, apresenta-se na Figura 6-6 o cenário básico das interações para o reconhecimento de uma página como membro do domínio (classificada) de um Agente de Busca.

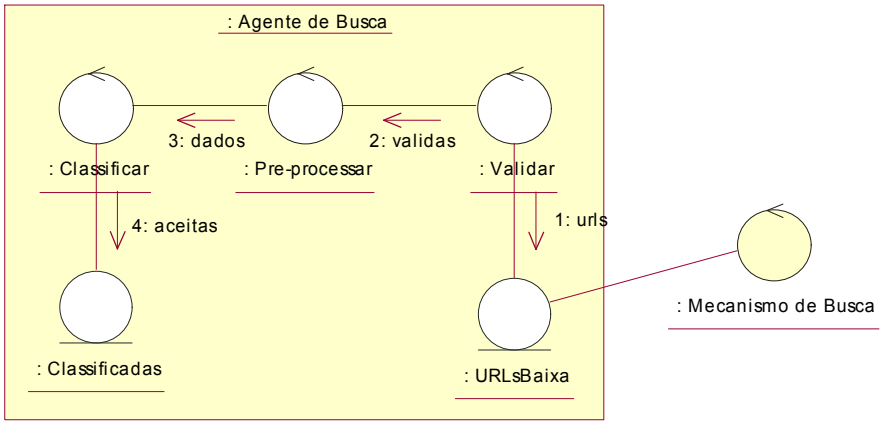

Figura 6-6 - Interações na classificação - Cenário básico

Os passos que compõem o cenário apresentado na Figura 6-6 são os seguintes: 1) o objeto da classe `Validar` obtém a relação de índices (URLs) de Baixa Prioridade; 2) esse objeto envia as URLs válidas para o pré-processamento; 3) o objeto da classe `PreProcessamento` extrai os atributos pertinentes e os envia para o processo de

classificação; 4) as páginas reconhecidas como pertencentes ao domínio do agente são armazenadas.

6.7 Reuso do Conhecimento

Pode-se afirmar que foi a busca por reuso do conhecimento que reavivou as abordagens baseadas em conhecimento, a partir da implementação de ontologias. Na arquitetura apresentada, três tipos de reuso do conhecimento dos agentes são possíveis:

i) Reuso das ontologias "de prateleira", já publicamente disponíveis;

ii) Reuso das ontologias usadas na comunicação entre os agentes, ou seja, o vocabulário a respeito de páginas da Web enquanto documentos, acrescido de conhecimento considerado necessário a todos os agentes, por exemplo, sobre as categorias.

iii) Reuso da ontologia sobre os grupos funcionais, incluindo a maior parte das regras de reconhecimento. Para alguns destes grupos, as regras são gerais (mensagens é o melhor exemplo), contudo as regras instanciáveis são em maior número; pela definição do relacionamento entre o agente e o grupo funcional, é que a regra poderá opcionalmente ser instanciada com casos e conceitos de dicionários.

Como é perceptível, a definição de novos conceitos e ontologias será mínima; ao invés disso, o conhecimento dos agentes instanciará conceitos pré-definidos. Portanto, o desenvolvimento de um novo agente torna-se rápido, exceto pela tarefa de aquisição de conhecimento necessário à categorização (ou reconhecimento) das páginas em grupos funcionais, realizável de duas maneiras: através da engenharia de conhecimento, examinando exemplos de páginas para entender seus padrões, ou através de técnicas de aprendizado automático. A escolha vai depender de fatores como a complexidade das regras, número de páginas a anotar e heterogeneidade das páginas.

7 IMPLEMENTAÇÃO DOS AGENTES DE BUSCA

Neste trabalho, apresenta-se um protótipo de implementação de dois Agentes de Busca, com a finalidade de recuperar da Web e manter URLs pertinentes ao domínio de Biologia, especificamente, o assunto de vertebrados e invertebrados.

Para realizar a implementação, escolheram-se as ferramentas e linguagens mais adequadas para fazê-lo. Foram escolhidas a linguagem Java (Sun, 2005), o motor de inferência Jess (Friedman-Hill, 2000), o editor de ontologias Protégé (Seção 2.5), e a ferramenta de desenvolvimento de sistemas multiagentes JADE (Seção 4.4.2) para a criação dos múltiplos agentes de software.

No JADE temos um `GerenciadorAgentes` responsável pelo provimento do guia de endereços e controle de ciclo de vida, mantendo um diretório de identificadores e estados de agentes. Cada Agente de Busca mantém quatro tipos dados: URLs com Alta Prioridade, URLs com Baixa Prioridade, Páginas Classificadas e Páginas Rejeitadas. O JADE, utilizando FIPA-ACL, possui um canal de comunicação entre os agentes, conforme está detalhado na Seção 4.4.2.

7.1 Motor de Inferência

Para a escolha de um motor de inferência a ser utilizado pelos Agentes de Busca, considera-se que para representar ontologias, faz-se necessário o uso de um formalismo de representação de conhecimento capaz de raciocinar sobre o formalismo frames, além disso, seria necessário que a linguagem empregada pelo motor se incluísse entre as traduzíveis pelos servidores de ontologias, como por exemplo, o Protégé

O motor de inferência com encadeamento para frente Jess (Java Embedded Expert System Shell) (Friedmann-Hill, 1997) uma das mais populares ferramentas de desenvolvimento de agentes e sistemas especialistas da história, possuindo alguns milhares de usuários, principalmente devido à boa integração entre objetos Java e os formalismos regras de produção e frames, tornou-se um candidato natural. O Jess implementa a maior parte das funcionalidades do sistema de produção CLIPS (“C” Language Integrated Production System) (Riley, 1999) para Java, o CLIPS integrava-se à linguagem “C”. Contudo, a sutil e

fundamental diferença de representação entre Jess e CLIPS reside na forma em que as classes são representadas nos dois motores. CLIPS inclui uma linguagem interna, COOL ("C" Object Oriented Language) (Giarratano & Riley, 1998), que representa classes como frames. Já Jess usa Java beans - componentes em Java que provêem reflexão – mas, como foi projetado para servir à comunidade de orientação a objetos, são incapazes de representar declarativamente listas estruturadas, e muito menos frames. Em Jess, um atributo não pode ser instância de uma classe declarativa, característica imperiosa em frames e ontologias. Felizmente, a popularidade de Jess e Java terminou por gerar uma solução para o problema de representação de frames. Um componente integrando as ferramentas Jess e Protégé chamado JessTab (Eriksson, 2000) foi desenvolvido a partir de iniciativa da equipe do Protégé, habilitando o Jess a reusar, definir e refinar ontologias através do Protégé. Este componente substitui as definições de classes declarativas do Jess de forma a implementar a mesma expressividade para frames do CLIPS, incluindo muitas funções relativas às facetas dos slots, que não teriam sentido em Jess. Uma lição aprendida deste caso é que a capacidade expressiva de representar frames constitui um requisito mínimo para motores de inferência quando se almeja reusar ontologias.

O Jess conta ainda com um ambiente de desenvolvimento com interface gráfica, que possibilita ao usuário desenvolver sistemas, applets e verificar a eficiência da rede de regras compilada. Possui ainda capacidade de encadeamento para trás, além de muitas outras facilidades implementadas pelos usuários e adicionadas ao pacote, como a manipulação de bancos de dados como bases de fatos, lógica difusa (Bittencourt, 1998), entre outras.

Entre outras qualidades do Jess, podem ser citadas a capacidade de raciocinar e manipular diretamente sobre objetos, métodos e variáveis Java dentro de regras Jess, a possibilidade de criação de regras e fatos Jess dentro de código Java, a passagem de objetos nos dois sentidos entre Java e Jess. Portanto, de acordo com o que está exposto, optou-se por utilizar o Jess como motor de inferência.

7.2 Agentes de Busca

Com o uso das ferramentas descritas na seção anterior, foram criados dois agentes para fazer a simulação de uma Sociedade de Agentes de Busca. Um agente, chamado de VERTEB, versará sobre o assunto Vertebrados no contexto da Biologia e um outro, chamado de

INVERTEB, tratará do assunto Invertebrados no mesmo contexto. Para isso, utiliza-se a ontologia de Biologia encontrada em (DAML, 2005a).

Os dois Agentes de Busca foram criados por meio do uso da ferramenta JADE e, para simplificar, associou-se a ambos o mesmo mecanismo de busca: o Google. Entretanto, poder-se-ia utilizar qualquer outro. Usou-se o Google através da importação da API-google fornecida em (Google, 2005).

Os agentes são executados em ambiente Windows, mas, como foram escritos em Java, podem ser portados para qualquer outro ambiente que suporte a JVM. As ontologias do sistema foram escritas no editor de ontologias Protégé e convertidas dinamicamente para Jess. Os dados mantidos pelos agentes são armazenados em XML.

Considera-se que esses agentes irão realizar busca com a finalidade de dar suporte ao aprendizado de animais Vertebrados e Invertebrados pelos alunos, usuários do sistema. Considera-se a existência de Agentes de Domínio correspondente para cada um desses agentes.

Com o uso de ontologias, está sendo implementado, no sistema NetClass, um Agente de Domínio para o assunto Vertebrados e um outro para o assunto Invertebrados. As instâncias das ontologias são armazenados em vários formatos de mídia (texto, vídeo, som etc).

No decorrer da aprendizagem, o Agente Tutor pode a qualquer momento solicitar da Sociedade de Busca uma relação de URLs sobre o assunto que está sendo estudado pelo aprendiz. Por exemplo, para o assunto Vertebrados, será acionado o agente VERTEB. O Agente VERTEB informa ao Tutor quais são os resultados alcançados através de pesquisas na Web. Esse resultado é informado na interface do aprendiz como uma relação de *links*, conforme pode ser visto na Figura 7-1. A interface do aprendiz permite que o aprendiz possa navegar pela relação de *links* exibidos como resultados.

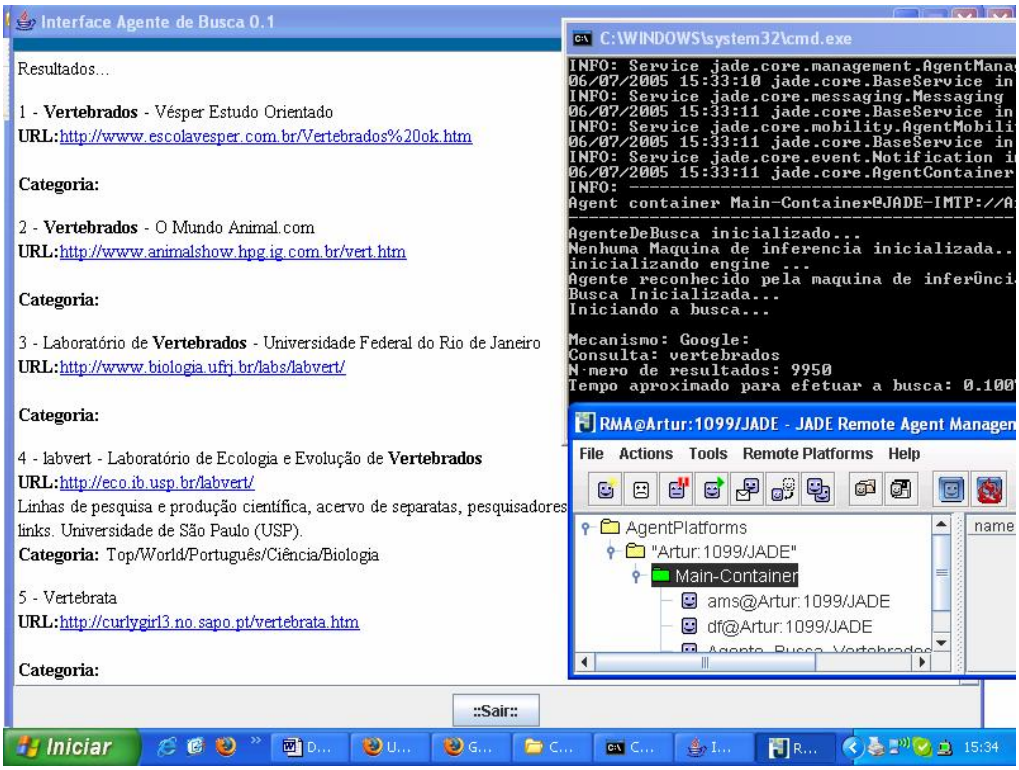

Figura 7-1 - Interface - Protótipo Agente VERTEB

Para o protótipo desenvolvido, considera-se que, por não contemplar o objetivo deste trabalho, o Agente Tutor no domínio de Vertebrados não foi desenvolvido. Portanto, faz-se uma simulação da existência deste de forma que os agentes desenvolvidos, relacionados à busca, são iniciados por uma interface tipo console e os resultados são exibidos diretamente a uma interface tipo GUI (cf. a Figura 7-1).

7.3 Simplificação da Arquitetura

Os agentes descritos procuram atender às especificações descritas no capítulo anterior. Entretanto, foram feitas algumas simplificações para realizar o experimento a fim de provar o que foi exposto até o momento.

O meta-robô de cada agente não foi implementando e esta função está incorporada às atividades dos agentes. A extração basear-se-á apenas em delimitadores e funções para formatação e classificação do atributo a ser extraído.

Como o objetivo dos experimentos consiste em provar que a arquitetura capacita-se a, cooperativamente, encontrar páginas pertencentes a classes de páginas, as duas classificações – a categorização funcional e a classificação por conteúdo nas subclasses processadas por cada agente – foram consideradas como tarefas primordiais. Como a tarefa de classificação requer certa complexidade, optou-se, neste experimento, pela identificação apenas da existência de atributos. Com isso, as duas classificações não ficam prejudicadas e os agentes continuam a seguir a premissa de reconhecer páginas de classes a partir de seu conteúdo. Também não foi tratado o problema de conteúdos exatamente iguais em páginas diferentes, que consta entre os cuidados associados à recuperação.

Outra simplificação diz respeito à concorrência do processamento de páginas. Embora o sistema pudesse ter sido confeccionado de modo a processar várias páginas ao mesmo tempo, foi percebido que, dada a latência causada pelo pré-processamento na geração das representações de uma página, os resultados demorariam mais, aparecendo apenas ao fim do processamento do conjunto de páginas simultaneamente tratadas. Para apressar os testes, então, foi processada uma página por vez.

Uma última simplificação refere-se aos tipos de arquivos tratados. Por enquanto, os agentes somente estão aptos a processar páginas cujos conteúdos estejam codificados em HTML.

7.4 Conhecimento dos Agentes

Cada agente possui uma ontologia que contém os conceitos a serem usados apenas por ele. Todavia, boa parte das instâncias relativas ao domínio, como a instância Vertebrados, podem ser úteis e reusadas por outros agentes.

O conhecimento específico de um agente restringe-se ao que ele necessita para o cumprimento de suas tarefas particulares, compreendidos por:

i) Ontologias de seu interesse exclusivo que lhe permitam melhores inferências nas tarefas de reconhecimento;

ii) Regras para serem enviadas aos outros agentes, para que eles identifiquem ponteiros e páginas para o agente;

iii) Regras de sugestão recebidas de outros agentes, para o envio a eles de páginas e ponteiros;

iv) Regras específicas de reconhecimento, opcionalmente, de validação.

7.5 Construção dos Agentes

Um modelo de agente foi desenvolvido atendendo os requisitos de reuso. Uma vez pronto, o modelo foi instanciado na criação do agente VERTEB para páginas com conteúdo sobre animais vertebrados. Para a confecção de qualquer agente do sistema, nenhum código precisa ser reescrito; apenas os seguintes itens serão definidos:

i) Os dados relativos à entidade cujos dados serão recuperados;

ii) Termos para as consultas aos mecanismos de busca;

iii) Entradas (palavras-chave) nos dicionários, que estão nos conceitos, usadas na fase de classificação;

iv) Regras de classificação e, opcionalmente, de recomendação para outros agentes;

v) Um número pequeno de regras específicas, não reusáveis, de reconhecimento;

vi) Indicação de quais conhecimentos (regras, e, se necessário, conceitos) serão anunciados aos outros agentes para que eles lhe indiquem um ponteiro ou página. Esse conhecimento pode incluir regras específicas e/ou partes da instância de relacionamento entre o agente e a sua classe de páginas reconhecidas.

7.6 Manutenção da Sociedade de Agentes

Está sendo implementado no NetClass um Módulo de Manutenção de Agentes que permite, ao Engenheiro do Conhecimento, criar, alterar ou excluir agentes da Sociedade de

Agentes de Busca e Sociedade de Agentes de Domínio, conforme pode ser observado na Figura 7-2.

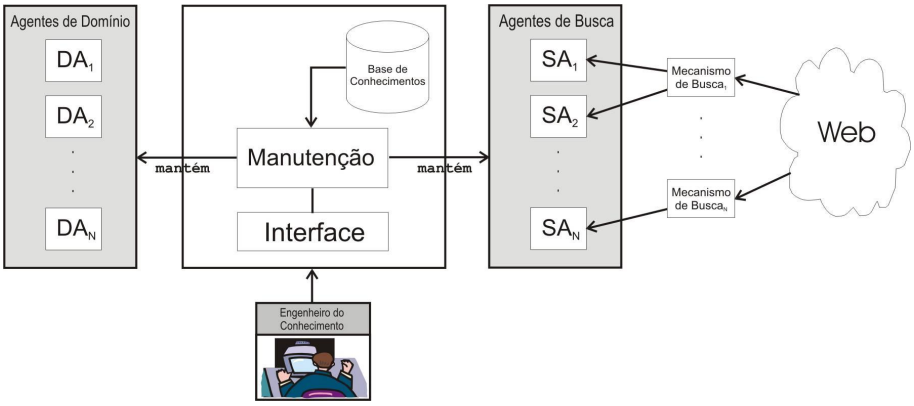

Figura 7-2 - Manutenção dos Agentes de Domínio e Agentes de Busca

O Módulo de Manutenção está sendo implementado com o uso da linguagem Java, além de outras tecnologias relacionadas. Primariamente, ele visa prover ao Engenheiro do Conhecimento uma forma fácil para criação de Agentes de Busca e Agentes de Domínio a partir de um conjunto de comportamentos pré-definidos em uma ontologia. Através da interface, pode-se selecionar um agente e visualizar informações a respeito deste.

Para os Agentes de Domínio, podemos visualizar no Módulo de Manutenção a lista com todas as unidades pedagógicas definidas para cada agente. Para os Agentes de Busca, pode-se editar os tópicos que podem ser incluídos na busca, escolher os mecanismos que este agente deverá utilizar e editar os itens, conforme descrito na seção anterior.

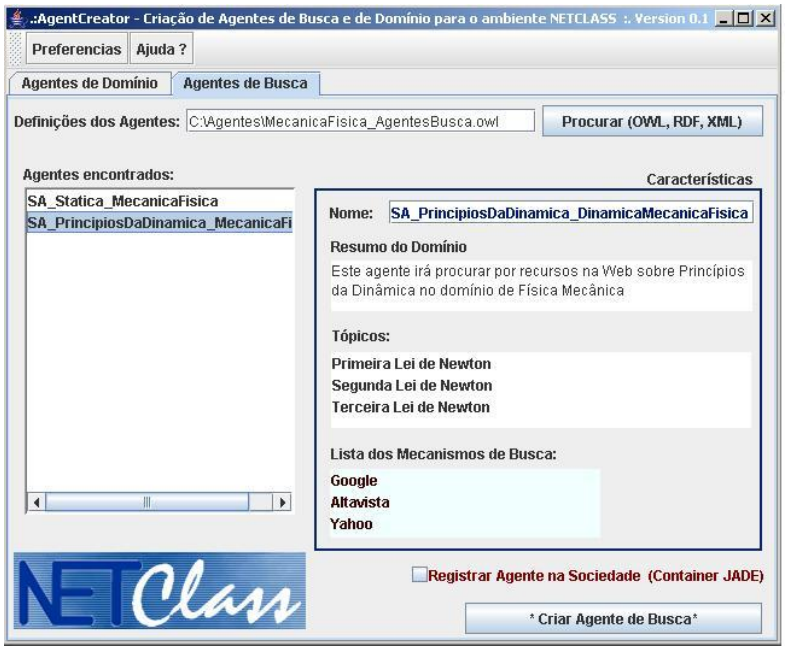

Figura 7-3 - Interface da Manutenção

Ao criar um agente, opcionalmente, pode-se registrá-lo automaticamente em um container JADE, onde se pode optar por um container local ou remoto. Os agentes registrados são criados com o nome carregado da própria ontologia adicionando-se os identificadores padrão da plataforma JADE.

8 CONCLUSÃO

Nesta dissertação, foi descrito o processo de recuperação de informações no sistema NetClass. Foram especificados, projetados e implementados dois Agentes de Busca NetClass que serviram como experimento para mostrar que pode-se utilizar de forma muito eficiente agentes de software para recuperar na Web informações relevantes para determinado domínio.

O uso de ontologias permite a interoperabilidade entre agentes de software. Por isso, o NetClass utiliza ontologias para representar conhecimentos e recuperar informações da Web. Nessa ambiente, a representação de conhecimentos e recuperação de informações são feitas utilizando-se uma arquitetura multiagentes, onde temos os Agentes de Domínio como a base de conhecimentos e os Agentes de Busca mantendo uma base de índices Web.

Por questão de simplificação, associou-se aos dois agentes protótipos o mesmo mecanismo de busca: o Google, através da importação da API-google fornecida em (Google 2005). Entretanto, poder-se-ia utilizar qualquer outro.

Apesar de existirem muitas ontologias já definidas e padronizadas (DAML, 2005b), existe ainda muito trabalho a ser desenvolvido na modelagem do conhecimento. A simples definição de ontologias *ad hoc* para um sistema, sem levar em consideração o que já está padronizado ou por padronizar, apresenta-se como um potencial problema de escalabilidade de conhecimentos.

A ferramenta Protégé é bastante eficiente no que diz respeito à edição de ontologias e, através de plugins, amplia em muito as suas possibilidades de utilização. Além disso, sua interface é bastante amigável, o que facilita o trabalho de edição. Por outro lado, o Jess, apesar de tecnicamente eficiente, exige do usuário (desenvolvedor) tempo para adaptar-se à sua sintaxe.

O NetClass encontra-se ainda em fase de desenvolvimento, o que gerou dificuldades para o entendimento de interações e testes com alguns componentes que sejam os de busca, principalmente, o Agente Tutor.

Sugere-se para trabalhos futuros a implementação dos Agentes de Domínio com o uso de ontologias, conforme descrito neste trabalho. Percebe-se que os Agentes de Busca podem ter seu foco de ação ampliado na medida em que o NetClass for sendo implementando. Por exemplo, podem-se imaginar terminais com navegadores, onde as URLs acessadas pelos aprendizes seriam supervisionadas pelo sistema multiagentes. De forma que os Agentes de Busca poderiam inferir (com um grau de certeza) se, as páginas acessadas estão ou não dentro do domínio estudado. Isso poderia ser interessante para evitar que usuários naveguem por páginas não relevantes, mas seria necessário considerar a possibilidade de erros nas regras dos agentes e, conseqüente, inibição por parte do aprendiz.

REFERÊNCIAS

(Arens et al., 1996) Arens Yigal, Hsu, Chun-Nan, Knoblock, Craig, 1996. *Query Processing in the SIMS Information Mediator*, in Huhns, Michael, Singh, Munindar, Eds. 1997. 'Readings in Agents', Morgan Kaufman Publishers, USA.

(Appelt et al., 1999) Appelt, D. E.; Israel, D. J.: 1999. *Introduction to Information Extraction Technology*. International Joint Conference of Artificial Intelligence.

(Baalen et al., 1993) Van Baalen, J., Fikes, R..; 1993. *The Role of Reversible Grammars in Translating Between Representation Languages*. Disponível em: http://www.ksl.stanford.edu/KSL_Abstracts/KSL-93-67.html - acessado em Março/2005.

(Baeza-Yates et al., 1999) Baeza-Yates, R. and Ribeiro-Neto, B. (eds.) *Modern information retrieval*, ACM Press, New York. 1999.

(Bastos et al., 2004) Bastos, O. Labidi, S. and Nascimento, E. *Increasing Cognitive Earnings using Open Challenge Learning Objects*. In the Proceedings of the International Conference on Computers and Advances Technology in Education (CATE-04). Auguest 16,18, 2004. Kauai, Hawai, USA.

(Bayardo et al., 1996) Bayardo Jr., R. J., Bohrer, W., Brice, R., Cichoki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusiunkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A,Woelk, D. 1996. *InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments*, in Huhns, Michael, Singh, Munindar, Eds. 1998 .Readings in Agents. Morgan Kaufman Publishers, USA.

(Berners-Lee et al., 2001) Berners-Lee, T.; Lassila, O. Hendler, J. – *The Semantic Web* – Scientific American – http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html

(Bowman et al., 1995) Bowman, C. M., Danzig, P. B., Hardy, D. R., Manber, U. and Schwartz, M. F. *The harvest information discovery and access system*. Computer Networks and ISDN Systems 28, pp. 119-125. 1995.

(Cohen & Singer, 1996) Cohen, W. and Singer, Y. 1996. *Learning to Query the Web*. AAAI Workshop on Internet-Based Information Systems.

(Cruz et al., 1997) Cruz, I.; Borisov, S.; Marks, M. A.; and Webb, T.R.; 1997. *Measuring Structural Similarity Among Web Documents: Preliminary Results*. Proceedings of the Electronic Publishing, Artistic Imaging, and Digital Typography 7th International Conference on Electronic Publishing, EP'98. St. Malo, France.

(DAML, 2005a) DAML Ontology Library (2005), http://www.cyc.com/2003/04/01/cyc, Abril.

(DAML, 2005b) DAML Ontology Library (2005), http://www.daml.org/ontologies, Abril.

(Eriksson, 2000) JessTab plugin for Protégé. Disponível em: http://www.ida.liu.se/~her/JessTab - acessado em Março/2005.

(Farquhar, 1996) Farquhar, A., Fikes, R., Rice, J.; 1996. *The Ontolingua Server: a Tool for Collaborative Construction*, Computer Science Department, Stanford University.

(Florescu et al., 1998) Florescu,D., Levy,A., Mendelzon,A. 1998. *Database Techniques for the Worlrd Wide Web*. SIGMOD record, 27(3):59-74.USA.

(Forgy, 1982) Forgy, C. L.; 1982. Rete: *A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem*, Artificial Intelligence 19(1982), 17-37. USA.

(Freitas & Bittencourt, 2000) Freitas, F.; Bittencourt, G. 2000. *Cognitive Multi-Agent Systems for Integrated Information Retrieval and Extraction over the Internet*. Proceedings of the International Joint Conference SBIA'2000 (the Brazilian Artificial Intelligence Symposium) and IBERAMIA'2000 (the Ibero-American Articial Intelligence Conference). Springer-Verlag. Berlim.

(Friedman-Hill, 2000) Friedmann-Hill, E. 2000. Jess, *The Java Expert System Shell*. http://herzberg.ca.sandia.gov/Jess.

(Giarratano & Riley, 1998) Giarratano, J., Riley, G. 1998. *Expert Systems: Principles and Programming*. Brooks/Cole Pub Co. USA.

(Gibson et al., 1998) Gibson,D., Kleinberg,J., Raghavan, P. 1998. *Inferring Web Communities from Link Topology*. Proc. 9th ACM Conference on Hypertext and Hypermedia, USA.

(Google, 2005) Google Web APIs (2005), http://www.google.com/apis, Abril.

(Gruber, 1995) Gruber, Thomas R.; 1995. *Towards Principles for the Design of Ontologies Used for Knowledge Sharing*. International Journal of Human and Computer Studies, 43(5/6): 907-928.

(Harmelen et al., 1999) van Harmelen, F. & Fensel, D. *Practical Knowledge Representation for the Web*. In. Fensel, D. (Ed.). Proceedings of the IJCAI'99 Workshop on Intelligent Information Integration, 1999.

(Hayes-Roth et al., 1978) Hayes-Roth, F., Waterman, D. A. and Lenat, D. B.; 1978. *Principles of Pattern-Directed Inference Systems*. In Pattern-Directed Inference Systems, Academic Press, New York, NY, USA.

(Kobayashi et al., 1999) Kobayashi, M. and Takeda, K. *Information retrieval on the web: selected topics*. IBM Research, Tokyo Research Laboratory, IBM Japan, Ltd. December, 16, 1999.

(Labidi et al., 2000) Labidi, S.; Silva, J.; Coutinho, L.; Costa, E. *Agent-Based Architecture for Cooperative Learning Environment*. Anais do XI Simpósio Brasileiro de Informática na Educação (SBIE'2000). Maceió-AL: 18 a 20 de Novembro, 2000.

(Labidi et al., 2003a) Labidi, S., Souza, C., Nascimento, E. *NetClass: Cooperative Learner Modeling in a Web-Based Environment*. In the 6th Int. Conf. on Computer Based Learning in Science. Proceedings of the 6th Int. Conf. on Computer Based Learning in Science (CBLIS). Nicosia, Cyprus: University of Cyprus, 2003.

(Labidi et al., 2003b) Labidi, S., Costa, N., Ferreira, J. *Modeling of an Authoring Tool for an Inteligent tutoring System*. In the Proceedings of the 6th Int. Conf. on Computer Based Learning in Science (CBLIS), 2003, Nicosia. University of Cyprus, 2003.

(Labidi et al., 2004) Labidi, S., and Borges, H. and Teixeira, C. *Problem Resolution in the NetClass Tutoring System*. In the Proceedings of the International Conference on Computers and Advances Technology in Education (CATE-04). Auguest 16,18, 2004. Kauai, Hawai, USA.

(McCarthy, 1958) McCarthy, John; 1958. *Programs with Common Sense*. In Proceedings of the Symposium on Mechanisation of Thought Processes , 1:77-84, London. Her Majesty's Stationery Office, UK.

(Meghini et al., 1998) Meghini, C., Sebastiani, F. and Straccia, U. *A model of multimedia information retrieval*. Tech. Rep. B4-21-0998, CNR-IEI. Sept. 1998.

(Miller, 1995) Miller, G. 1995. *WordNet: A Lexical Database for English*. Communications of the ACM. 38(11):39-41. EUA.

(Minsky, 1975) Minsky, M.; 1975. *A Framework for Representing Knowledge*. In The Psychology of Computer Vision, p.211-281,McGraw-Hill, New York. USA.

(MsAgent, 2003) MICROSOFT AGENT (2005), http://www.microsoft.com/msagent/, Março.

(Newell, 1982) Newell, A.; 1982. *The Knowledge Level*. Artificial Intelligence 18(1):87-127.

(Nwana et al., 1999) Nwana, H.; Ndumu, D.; Lee, L.; Collins, J. *ZEUS: a toolkit for building distributed multi-agent systems*. In Applied Artificial Intelligence Journal, Vol 13 (1), 1999. pp. 129-186.

(Oates et al., 1994) Oates, T.; Prasad, M.; Lesser, V. 1994. *Cooperative Information Gathering: A Distributed Problem Solving Approach*. Computer Science Technical Report 94-66- version 2. University of Massachusetts, Amherst, USA.

(Pachet, 1995) Pachet, F. 1995. *On the embeddability of Production Rules in Objectoriented Languages*. Journal of Object-Oriented Programming (Joop).Jul-ago 1995.

(Pirolli et al., 1995) Pirolli,P.,Pitkow,J.,Rao,R.:. 1995. Silk from a Sow's Ear: Extracting Usable Structures from Web. http://www.acm.org/sigchi/chi96/proceedings/papers/Pirolli_2/pp2.html

(Protégé, 2005) Protégé-2000 (Release 3.1) (2005), http://protege.stanford.edu, Março.

(Riley, 1999) Riley, G. 1999. *CLIPS: A Tool for Building Expert Systems*. Disponível em: http://www.ghg.net/clips/CLIPS.html - acessado em Março/2005

(Russel et al., 1995) Russell, S. J.; Norving P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

(Schatz, 1997) Schatz, B. *Information retrieval in digital libraries: bringing search to the Net*, Science, 275, 327-334. January 1997.

(Schäuble, 1997) Schäuble, P., *Multimedia information retrieval: content-based information retrieval from large text and audio databases*, Kluwer Academic Publishers, Boston. 1997.

(Serra Jr et al., 2001a) Serra Jr., G.; Coutinho, L.; Labidi, S. *Formation of Groups for Cooperative Learning: a Genetic Algorithm Approach*. In Proceedings of Conference on Computers and Advanced Technology in Education (CATE 2001). Banff, Canada: June 27-29, 2001.

(Serra Jr, 2001b) Serra Jr, G. *Agente de Modelagem do Aprendiz para o Ambiente MATHNET de Ensino Cooperativo Computadorizado*. São Luís-MA, 2001. Dissertação (Mestrado em Ciência da Computação)-Universidade Federal do Maranhão.

(Silva, 1999) Silva, J. *Aquisição de Conhecimentos e Manutenção para uma Sociedade de Agentes Tutores Artificiais*. Campina Grande-PB, 1999. Dissertação (Mestrado em Ciência da Computação)-Universidade Federal da Paraíba.

(Silva, 2003) Silva, Leonardo A. *Estudo e Desenvolvimento de Sistemas Multiagentes usando JADE: Java Agent Development framework.* Universidade de Fortaleza – UNIFOR. Centro de Ciências Tecnológicas – CCT. Curso de Informática. Monografia de Conclusão de Curso. Fortaleza – Ceará – Brasil, 2003.

(Sun, 2005) Sun Microsystems (2005), http://java.sun.com/, Março.

(Uschold et al., 1999) Uschold, M., Gruninger, M. 1999. A Framework for Understanding and Classifyig Ontology Applications. Disponível em: http://sunsite.informatik.rwthaachen.de/Publications/CEUR-WS/Vol-18/11-uschold.pdf - acessado em Fevereiro/2005.

(Wilks, 2000) Wilks, Y. *IR and AI*: traditions of representation and anti-representation in information processing, 2000.

(Woods, 1975) Woods, W. A.; 1975. *What's in a link: Foundations for semantic networks. In Representation and Understanding Studies in Cognitive Science*, Academic Press, New York, NY. USA.

(Wooldridge et al., 1995) Wooldridge, Michael; Nicholas R. Jennings (1995). *Agent Theories, Architectures, and Languages: a Survey*. In Wooldridge and Jennings Eds., Intelligent Agents, Berlin: Springer-Verlag, 1995.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo