UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

JAIR ABÚ BECHIR LÁSCAR ALARCÓN

Airetama

Um Arcabouço Baseado em Sistemas Multiagentes para a Implantação de Comunidades Virtuais de Prática na Web

Goiânia
<2010>

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

AUTORIZAÇÃO PARA PUBLICAÇÃO DE DISSERTAÇÃO EM FORMATO ELETRÔNICO

Na qualidade de titular dos direitos de autor, **AUTORIZO** o Instituto de Informática da Universidade Federal de Goiás – UFG a reproduzir, inclusive em outro formato ou mídia e através de armazenamento permanente ou temporário, bem como a publicar na rede mundial de computadores (*Internet*) e na biblioteca virtual da UFG, entendendo-se os termos "reproduzir" e "publicar" conforme definições dos incisos VI e I, respectivamente, do artigo 5º da Lei nº 9610/98 de 10/02/1998, a obra abaixo especificada, sem que me seja devido pagamento a título de direitos autorais, desde que a reprodução e/ou publicação tenham a finalidade exclusiva de uso por quem a consulta, e a título de divulgação da produção acadêmica gerada pela Universidade, a partir desta data.

Título: Airetama – Um Arcabouço Baseado em Sistemas Multiagentes para a Implantação de Comunidades Virtuais de Prática na Web

Autor(a): Jair Abú Bechir Láscar Alarcón

Goiânia, <04 > de de <2010>.

Jair Abú Bechir Láscar Alarcón – Autor

Cedric Luiz de Carvalho – Orientador

JAIR ABÚ BECHIR LÁSCAR ALARCÓN

Airetama

Um Arcabouço Baseado em Sistemas Multiagentes para a Implantação de Comunidades Virtuais de Prática na Web

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Computação.

Área de concentração: Sistemas de Informação.

Orientador: Prof. Cedric Luiz de Carvalho

Goiânia
<2010>

JAIR ABÚ BECHIR LÁSCAR ALARCÓN

Airetama

Um Arcabouço Baseado em Sistemas Multiagentes para a Implantação de Comunidades Virtuais de Prática na Web

Dissertação defendida no Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Computação, aprovada em <04 > de de <2010>, pela Banca Examinadora constituída pelos professores:

Prof. Cedric Luiz de Carvalho
Instituto de Informática – UFG
Presidente da Banca

Prof. Vinícius Sebba Patto
Instituto de Informática – UFG

Prof. Jaime Simão Sichman
Depto. Eng. Computação e Sistemas Digitais e Laboratório de Técnicas Inteligentes – USP

Jair Abú Bechir Láscar Alarcón

Graduou-se em Ciência da Computação na PUC Goiás - Pontifícia Universidade Católica de Goiás. Durante sua graduação, exerceu a função de monitor, com bolsa, da disciplina Introdução à Ciência da Computação I, Introdução à Ciência da Computação II, Estruturas de Dados I, Estruturas de Dados II, além de pesquisador em um trabalho de iniciação científica do Departamento de Informática. Durante o Mestrado, na UFG - Universidade Federal de Goiás, foi bolsista da CAPES e desenvolveu um trabalho prático no desenvolvimento de um arcabouço baseado em Sistemas Multiagentes para a implantação de Comunidades Virtuais de Prática na Web.

Dedico este trabalho à minha família, pelo apoio emocional, em especial meus pais, que me trouxeram do Chile juntamente com meus quatro irmãos, para garantir a nossa educação. A minha dívida com eles é imensurável.

À minha namorada Bruna, pela compreensão e motivação, além de ser muito linda. Agradeço também ao *TheFuntu* (Roni), maior amigo que tive apesar de nunca tê-lo visto pessoalmente. Foi graças à namorada dele que conheci, sem querer, a minha.

Aos meus pouquíssimos amigos, em especial ao Bruno Bezerra por sempre ter me apoiado no ensino médio, aos amigos da banda Senso Mor (na qual tive a honra de tocar bateria, mas tive que largar devido aos estudos), e aos amigos de Warcraft 3 e WOW Felipe Perotti e Yuri apesar de serem *noobs*.

Aos professores da Universidade Católica de Goiás que me aceitaram como monitor e pesquisador: Alexandre Ribeiro e José Luiz.

Aos programadores que admiro muito: John Romero (a lenda), Shigeru Miyamoto (ele mesmo), Alan Kay (também biólogo e guitarrista de Jazz), Jordan Mechner (também psicólogo) e muitos outros.

Aos pais da computação: Alan Turing, Charles Babbage e Blaise Pascal, que servem como inspiração à toda comunidade científica.

Aos pioneiros da computação: Douglas Engelbart, Steve Jobs, Steve Wozniak, Dennis Ritchie, Ken Thompson, Edsger Dijkstra e Linus Torvalds.

À toda saga da familia Gracie, em especial Hélio, Royce, Rickson, Ryan, Renzo, Ralek e Kyra, pela filosofia que criaram do BJJ, e pelos inesquecíveis *armlocks*.

À todas as outras grandes personalidades que se destacaram por buscarem melhorar o mundo de alguma forma.

À Deus, o maior mistério do universo, eu peço desculpas pelo fato de ser agnóstico.

Agradecimentos

A realização deste trabalho em muito se deve à colaboração e apoio de diversas pessoas, às quais transmito os mais sinceros agradecimentos:

Ao meu orientador Cedric, pelos ensinamentos, sugestões, conselhos, paciência, atenção e inestimáveis críticas construtivas.

Aos colegas do mestrado e professores, que considero, sem exceção, pessoas extraordinárias.

À CAPES, pelo importante incentivo ao ter concedido a bolsa de estudos.

Ao Tim Berners-Lee, que considero ser um dos maiores gênios vivos da humanidade, idealizador da Web, e ainda, a nova geração da Web, denominada Web Semântica.

Ao Andy Seaborne, um dos maiores colaboradores da Web Semântica, pela ajuda dada sobre o Jena, sempre respondendo cordialmente minhas dúvidas nos fóruns de discussão.

Ao Caire Giovanni, pela ajuda sobre o JADE, também respondendo atenciosamente meus questionamentos nos fóruns da comunidade.

Finalmente, ao presente leitor, o real motivo desta dissertação.

Ciência da Computação tem tanto a ver com o computador como a Astronomia com o telescópio, a Biologia com o microscópio, ou a Química com os tubos de ensaio. A Ciência não estuda ferramentas, mas o que fazemos e o que descobrimos com elas.

Edsger Dijkstra,
Recebeu o Prêmio Turing de 1972 por suas grandes contribuições à Ciência da Computação..

Resumo

Alarcón, Jair. **Airetama**. Goiânia, <2010>. 106p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

O objetivo desta dissertação é apresentar o arcabouço Airetama. Este arcabouço é baseado em Sistemas Multiagentes e nos princípios da Web Semântica. Ele fornece uma infra-estrutura semântica, distribuída e *open-source* para a criação de Comunidades Virtuais de Prática na Web. Possibilita, através do uso de agentes, o acoplamento de ferramentas que utilizam recursos e tecnologias semânticas. A inserção de semântica na Web atual tem como principal objetivo permitir que tais agentes de *software* possam utilizar suas páginas de maneira mais inteligente, oferecendo melhores serviços.

Palavras–chave

Comunidades Virtuais de Prática, Sistemas Multiagentes, Web Semântica, Web 3.0.

Abstract

Alarcón, Jair. **Airetama: Um Arcabouço Baseado em Sistemas Multiagentes para a Implantação de Comunidades Virtuais de Prática na Web**. Goiânia, <2010>. 106p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

The objective of this dissertation is to present the framework Airetama. This framework is based on Multiagent Systems and Semantic Web principles. It provides a semantic, distributed and open-source infrastructure for the creation of Virtual Communities of Practice on the Web. It makes possible, through the use of agents, coupling of resources and tools that use semantic technologies. Integration of semantic in the current Web has as main objective to allow such software agents can use their pages more intelligently, thus offering better service.

Keywords

Communities of Practice, Multiagent Systems, Semantic Web, Web 3.0

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Códigos de Programas

Lista de Abreviaturas e Siglas

FOAF - The Friend of a Friend
HTML - HyperText Markup Language
HTTP - Hypertext Transfer Protocol
OWL - Web Ontology Language
RDF - Resource Description Framework
RDFS - RDF Schema
RDQL - RDF Data Query Language
RSS - Really Simple Syndication
SIOC - Semantically-Interlinked Online Communities
SPARQL - SPARQL Protocol and RDF Query Language
UML - A Unified Modeling Language
URI - Uniform Resource Identifier
URL - Uniform Resource Locator
W3C - World Wide Web Consortium
WWW - World Wide Web
XFN - XHTML Friends Network
XHTML - eXtensible HyperText Markup Language
XML - eXtensible Markup Language
XMLS - XML Schema

CAPÍTULO 1

Introdução

O advento da Web[1] mudou a forma como as pessoas se comunicam, como as informações são disseminadas e recuperadas, e também como os negócios são conduzidos. Embora existam diversas ferramentas que utilizam as informações contidas na Web atual (ano de 2010), elas enfrentam os mesmos problemas e desafios [4]:

- Precisão baixa nas buscas;
- Resultados muito dependentes de vocabulário, pois mecanismos de busca são fortemente baseados em palavras-chave;
- A informação não é recuperada, mas sim a sua localização;
- Buscadores Web frequentemente são aplicações isoladas: foram projetados para serem utilizadas por seres humanos, não sendo legíveis para outras ferramentas de *software*.

O maior obstáculo para fornecer um melhor apoio aos usuários da Web é que, atualmente, o significado do conteúdo Web não é acessivel para máquinas (*machine-accessible*). Existem ferramentas que podem recuperar textos, dividí-los em partes, verificar a ortografia, e contar suas palavras. Mas, a capacidade de tais ferramentas para interpretar sentenças e extrair informações úteis é bastante limitada.

Desta forma, a adição de semântica fornece uma maneira formal de máquinas processarem significado de conteúdo Web.

A Web atual é uma rede de documentos. A Web Semântica será uma rede de dados composta por: pessoas, lugares, eventos, músicas, filmes, organizações, ou qualquer outro tipo de conceito. O objetivo principal da Web Semântica não é, pelo menos inicialmente, treinar as máquinas para que se comportem como pessoas, mas sim desenvolver tecnologias e linguagens que permitam que os computadores interpretem e processem estes conceitos [11].

[1] Em 1989 Tim Berners-Lee teve a grande ideia de combinar dois elementos distintos: a Internet e o Hiper-texto, culminando assim na nova invenção, denominada *World Wide Web*, também conhecida como WWW ou simplesmente Web [10].

1.1 Motivação

A Web é uma grande conquista tecnológica com um número crescente de usuários e fontes de informações. Contudo, o aumento da complexidade na Web afeta diretamente os usuários, deixando-os responsáveis por controlar manualmente o acesso, extração, interpretação e manutenção da informação.

Com o aumento exponencial da quantidade de informações na Web, surge a necessidade de desenvolver sistemas inteligentes, capazes de auxiliar no processamento e recuperação de informações, diminuindo assim o esforço dos usuários.

A inserção de semântica na Web atual tem como motivação permitir que agentes de *software* possam utilizar suas páginas de forma mais inteligente, oferecendo assim melhores serviços.

A consolidação da Web Semântica é um problema importante, pois a sua realização possibilitará que tais agentes de *software* possam auxiliar os humanos na busca, processamento e organização de informações, de forma que, todas as áreas do conhecimento humano serão beneficiadas.

As tecnologias da Web Semântica ainda estão em desenvolvimento e levará um certo tempo até que possam ser utilizadas em larga escala. São grandes os esforços científicos e de instituições para tentar criar padrões, no entanto alguns cientistas acreditam que a Web Semântica não irá decolar.

Não é possível definir ou prever uma aplicação principal para essas tecnologias. Diversas aplicações podem ser criadas, tais como: agentes de viagens (*e-Tourism*) que efetuam um planejamento conforme as restrições dadas pelo usuário; agentes de recomendações (livros, músicas, filmes) baseado no perfil e atividade do usuário; agentes comerciais que auxiliem a procura de produtos e serviços (*e-Commerce*); agentes que indicam empregos; entre inúmeras outras possibilidades como na área de saúde (*e-Health*), educação (*e-Learning*), redes sociais etc.

O conhecimento ainda não é servido em lojas de lanches rápidos, como no filme “Inteligência Artificial” de Steven Spielberg, entretanto, é fascinante pensar nesta possibilidade. A existência de um “guru” que responda a todas as nossas perguntas, e que tenha informações sobre todas as áreas de conhecimento é uma perspectiva bastante ousada, mas que talvez não esteja tão distante assim de nossos tempos [80].

1.2 Objetivo

O foco deste trabalho é apresentar o desenvolvimento de um arcabouço[2] baseado em Sistemas Multiagentes, que possa permitir a implantação e o trabalho colaborativo de Comunidades Virtuais de Prática. Esta proposta se insere no contexto da nova geração da Web, denominada Web Semântica.

O arcabouço desenvolvido foi denominado Airetama, que na língua Tupi-Guaraní significa "colméia". Este nome reflete um dos principais propósitos do projeto, isto é, o trabalho colaborativo.

O Airetama faz parte do projeto DWeb[3], do Instituto de Informática da Universidade Federal de Goiás. O Projeto DWeb visa contribuir com a implementação, desenvolvimento e consolidação da Web Semântica, em conformidade com os padrões da comunidade científica, em especial, os padrões divulgados pelo W3C (*The World Wide Web Consortium*), propondo um ambiente baseado em Comunidades Virtuais de Prática, onde recursos de informação possam ser tratados semanticamente, agregando inteligência, interoperabilidade e integração ao ambiente da Web existente [36].

A idéia por trás do projeto é construir um ambiente intuitivo e universal para compartilhamento de dados, informação e conhecimento, de uma forma transparente e acessível.

Tim Berners-Lee afirma que: "A Web é mais uma criação social do que uma criação técnica. Eu a idealizei para um efeito social - para ajudar as pessoas a trabalharem juntas - e não como um brinquedo técnico. O objetivo principal é o apoio e melhoria da nossa existência no mundo" [10].

Portanto, este trabalho compartilha a visão de Tim Berners-Lee, de que a Web não deve ser tratada apenas como uma rede de computadores, mas sim como uma rede de pessoas.

Além do objetivo principal apresentado, este trabalho pretende atingir os seguintes objetivos secundários que convergem para o objetivo principal:

- Esboçar uma arquitetura que permita organizar semanticamente a informação de comunidades virtuais com interesses e atividades específicas em um domínio bem definido;
- Contribuir com o avanço do projeto DWeb, facilitando o estabelecimento das Comunidades Virtuais de Prática no ambiente Web;

[2] Também pode ser denominado *framework*, *middleware*, plataforma, API ou ambiente, dependendo do contexto.

[3] Acrônimo de *Dream Web*, ou seja, a Web dos sonhos.

- Fornecer um ambiente de desenvolvimento que auxilie na criação e integração de ferramentas semânticas, que forneçam serviços inteligentes para estas comunidades;
- Facilitar o desenvolvimento integrado destas ferramentas, de forma transparente, distribuída e semântica;
- Disseminar e motivar o uso dos conceitos, tecnologias e linguagens da Web Semântica em ambientes direcionados para o trabalho colaborativo;
- Criar um ambiente que possibilite a coexistência de sistemas heterogêneos;
- Promover o desenvolvimento, integração e reutilização de informações de forma semântica, através de ontologias e metadados;
- Criar um infra-estrutura semântica de agentes inteligentes capazes de interpretar e processar estas informações.

1.3 Metodologia

Como pode ser visto na Figura 1.1, o processo de implementação do projeto Airetama foi decomposto em 6 etapas.

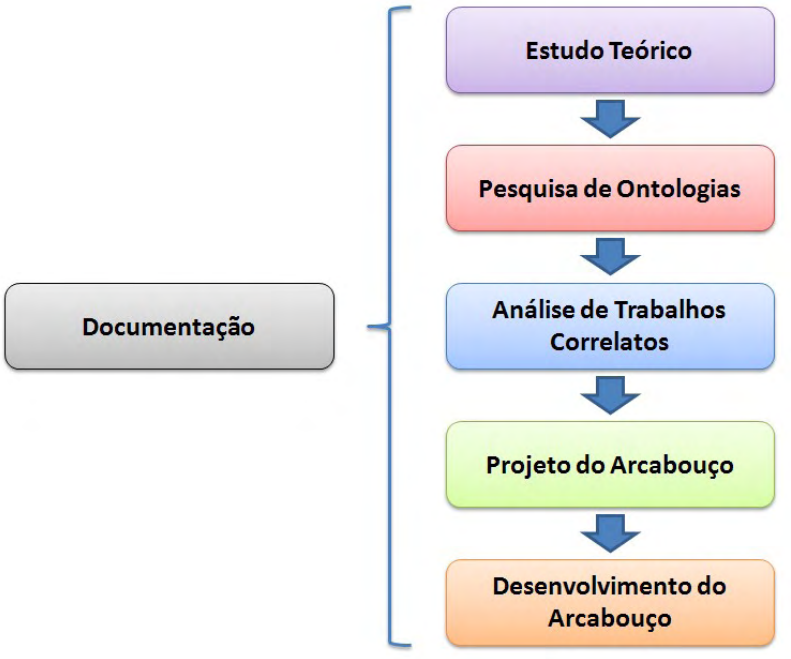


Figura 1.1: *Implementação do Airetama.*

Na etapa Estudo Teórico foi feita toda a pesquisa bibliográfica necessária para o desenvolvimento do arcabouço. A etapa Pesquisa de Ontologias consistiu no levantamento e análise das principais ontologias utilizadas pela comunidade científica, em especial aquelas recomendadas pelo W3C.

Na etapa Análise de Trabalhos Correlatos foram estudados os principais trabalhos relacionados com o tema. Na etapa Projeto do Arcabouço foi efetuado o levantamento de requisitos, e também foi definida uma arquitetura, visando esboçar uma solução adequada às restrições e necessidades identificadas. Na etapa Desenvolvimento do Arcabouço foi feita a implementação do mesmo, utilizando a arquitetura definida. A etapa de Documentação (que consiste principalmente na escrita da dissertação, relatórios e artigos) foi feita paralelamente no decorrer de todas as outras 5 etapas.

1.4 Organização da Dissertação

No Capítulo 2, são introduzidos os fundamentos teóricos utilizados neste trabalho. Inicialmente, são apresentados os principais conceitos das Comunidades Virtuais. Em seguida, são tratados os principais temas da Web Semântica, mostrando os conceitos de Web 1.0, 2.0 e 3.0, XML, RDF, RDFS, Ontologias, OWL, SPARQL e Microformatos. Também são abordados os agentes e sistemas multiagentes, além das arquiteturas de agentes, arquiteturas de sociedades de agentes, e arquiteturas distribuídas.

No Capítulo 3, são apresentadas algumas das principais ontologias que estão sendo utilizada pela comunidade científica: FOAF, SIOC, Relationship, iCalendar, Goodrelations, SKOS, Geo Ontology, DOAC, DOAP, Dublin Core e Music Ontology.

No Capítulo 4, são descritos os principais trabalhos relacionados com esta dissertação: Coopractice, SemantiCore, fGrin, OntoShare, Twine, Semantic MediaWiki, DBpedia, Swicki, WordNet, Wordnet.PT, Cyc e Powerset.

No Capítulo 5, são apresentados os Atores, Requisitos Funcionais e Requisitos Não Funcionais que serviram como base para o projeto do arcabouço Airetama. Em seguida, é descrita a definição de sua arquitetura, formada pelos seguintes componentes: um Portal, uma Sociedade de Membros, uma Sociedade de Controladores, uma Sociedade de Ferramentas, um Repositório Semântico, e o Relacionamento entre os Componentes.

O Capítulo 6 é dedicado ao desenvolvimento do arcabouço Airetama, utilizando as especificações do capítulo anterior. No início do capítulo são elencadas as principais tecnologias empregadas para o desenvolvimento do arcabouço. Em seguida, é descrita a implementação de cada componente do mesmo.

Finalmente, o Capítulo 7 fornece algumas considerações finais, enumerando contribuições e possíveis trabalhos futuros.

CAPÍTULO 2

Fundamentos Teóricos

O ponto de partida para a criação do arcabouço Airetama surgiu com a necessidade de dar suporte às Comunidades Virtuais de Prática na utilização dos recursos e tecnologias relacionadas com a Web Semântica.

Para a elaboração deste arcabouço, foi necessária a análise de três áreas distintas: Comunidades Virtuais de Prática, Sistemas Multiagentes e Web Semântica. A Figura 2.1 ilustra as 3 áreas que se relacionam com o arcabouço.

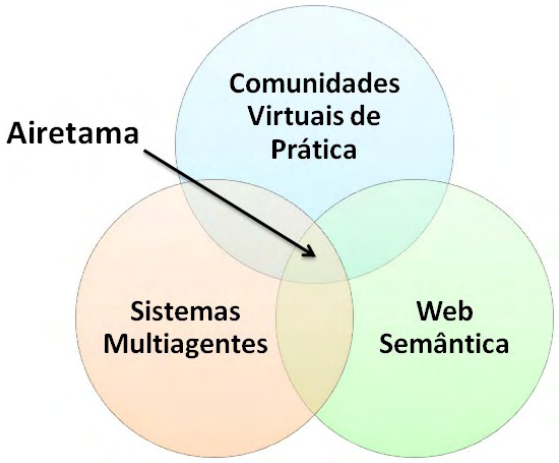


Figura 2.1: *Relacionamento entre o arcabouço Airetama e as áreas envolvidas no seu desenvolvimento.*

Este capítulo fornece uma visão geral sobre os principais conceitos relacionados com estas 3 áreas. Inicialmente, são descritos os tipos de comunidades virtuais, com foco nas Comunidades Virtuais de Prática, que é o tipo utilizado no arcabouço.

Em seguida, são identificadas as principais tecnologias relacionadas com a Web Semântica, juntamente com alguns exemplos e descrições breves. Também são descritos conceitos relacionados com agentes e sistemas multiagentes.

Ao final do capítulo são apresentadas as principais arquiteturas de *software* correspondentes a agentes, sistemas multiagentes e sistemas distribuídos.

2.1 Comunidades Virtuais de Prática

O termo **comunidade** é definido como um grupo de pessoas que compartilham o mesmo interesse ou estão inseridas em um mesmo contexto [55].

Já o termo **comunidade virtual**[1] é definido como um agrupamento de indivíduos alinhados em torno de um interesse comum, conferindo-lhe a característica de possuir comunicação assíncrona [6].

A Tabela 2.1 apresenta as 25 comunidades virtuais mais populares[2], juntamente com seu *ranking* e visitas mensais [42].

Tabela 2.1: *Top 25 de Redes Sociais*

Rank	Site	Visitas Mensais
1	facebook.com	1.191.373.339
2	myspace.com	810.153.536
3	twitter.com	54.218.731
4	flixter.com	53.389.974
5	linkedin.com	42.744.438
6	tagged.com	39.630.927
7	classmates.com	35.219.210
8	myyearbook.com	33.121.821
9	livejournal.com	25.221.354
10	imeem.com	22.993.608
11	reunion.com	20.278.100
12	ning.com	19.511.682
13	blackplanet.com	10.173.342
14	bebo.com	9.849.137
15	hi5.com	9.416.265
16	yuku.com	9.358.966
17	cafemom.com	8.586.261
18	friendster.com	7.279.050
19	xanga.com	7.009.577
20	360.yahoo.com	5.199.702
21	orkut.com	5.081.235
22	urbanchat.com	2.961.250
23	fubar.com	2.170.315
24	asiantown.net	1.118.245
25	tickle.com	109.492

Existem diversas classificações possíveis para as comunidades virtuais, sendo que as mais comuns são [70]:

[1] Também chamadas de Redes Sociais.
[2] Em termos de acesso aos seus portais na Web.

- **Comunidades Virtuais de Relacionamentos**: são construídas baseadas em relacionamentos especiais entre pessoas, como por exemplo relações de família;
- **Comunidades Virtuais de Lugar**: baseadas em indivíduos que compartilham o mesmo habitat ou local;
- **Comunidades Virtuais de Conhecimentos**: ajudam a encontrar pessoas com os mesmos objetivos, valores e concepção sobre determinado assunto;
- **Comunidades Virtuais de Memória**: baseadas em algum fato ou acontecimento passado compartilhado.

Este trabalho está direcionado com um outro tipo de comunidade virtual, chamada de **Comunidade Virtual de Prática**. Etienne Wenger [91] define este termo da seguinte forma:

"As comunidades de prática são grupos de pessoas, com distintos níveis de conhecimentos, habilidades e experiência, que se implicam de um modo ativo em processos de colaboração em uma atividade comum e, nestes processos, experimentam e recriam continuamente sua identidade compartilhada e constroem conhecimento, tanto pessoal quanto coletivo, melhorando assim as práticas da comunidade às quais pertencem."

As Comunidades Virtuais de Prática são baseadas em 3 dimensões [91]:

- **Domínio**[3]: domínio de interesse compartilhado, no qual os membros possuem um nível mínimo de conhecimento, e ocorre acumulo de experiência dentro deste domínio;
- **Comunidade**[4]: membros engajados em atividades conjuntas, discussões, ajuda mútua e compartilhamento de informações. Interação é requisito básico para membros pertencerem a uma comunidade;
- **Prática**[5]: uma Comunidade Virtual de Prática não é meramente uma Comunidade de Interesses, pois tem como requisito a prática compartilhada, onde ocorre interação através de problemas, soluções, apresentações, e da construção de um ambiente comum de conhecimento.

Etienne Wenger ainda realizou um levantamento de características que poderiam dar suporte às Comunidades Virtuais de Prática [92]: integração entre trabalho e conhecimento (espaço para gerenciar participação em múltiplos grupos); trabalho (espaços para projetos); estrutura Social (portal da comunidade); conversação (grupos de discussão); interações (locais de encontro síncronas); instrução (espaços de aprendizagem virtual); compartilhamento de conhecimento (acesso à experiência dos membros); documentos (bases de conhecimento).

[3] Sobre o que?
[4] Como funciona?
[5] O que produz?

2.2 Web Semântica

A Web Semântica está sendo criada com o intuito de eliminar (ou minimizar) as limitações da Web atual, também chamada de *World Wide Web* ou WWW. O objetivo da introdução de semântica na Web é tornar a informação "compreensível" para o computador.

Tim Berners-Lee, idealizador da WWW, HTML, URIs e HTTP, fundador e atual diretor do W3C [84] afirma: "A Web Semântica não é uma Web separada, mas uma extensão da atual. Nela, a informação é dada com um significado bem definido, permitindo melhor interação entre os computadores e as pessoas" [11].

Ele também afirma que a Web atual é um banco de documentos mundial, e com sua evolução para Web Semântica passará a ser um banco de dados mundial. Desta forma, os computadores irão ter a capacidade de processar os dados de forma semântica e não apenas sintática como vem sendo feito atualmente.

Diversos padrões vem sendo criados para possibilitar a consolidação da Web Semântica. O W3C é a principal instituição responsável pela padronização de tecnologias associadas com a Web Semântica, juntamente com a comunidade acadêmica.

A seguir, é feita uma apresentação dos principais conceitos relacionados com a Web Semântica, iniciando com um breve histórico, e em seguida são descritas as suas principais tecnologias.

2.2.1 Web 1.0, 2.0 e 3.0

A primeira geração da Web, chamada Web 1.0, foi a implantação e popularização da rede em si. A segunda geração da Web, denominada Web 2.0, é a utilizada atualmente, centrada nos mecanismos de busca como Google e nos *sites* de colaboração, como Wikipedia e YouTube, e os *sites* de relacionamento social, como o Facebook. A terceira geração da Web, conhecida como Web 3.0, pretende ser a organização e o uso de maneira mais inteligente de todo o conhecimento já disponível na Web atual.

Na Web 1.0, surgiu uma grande quantidade de informações, no entanto, o conteúdo era pouco interativo. O usuário apenas tinha um papel de espectador, não alterando o conteúdo dos *sites*. Nela, surgiram grandes *sites* como Altavista, Hotmail, UOL, Terra, Yahoo! e Cadê.

Já na Web 2.0, ocorreu uma evolução que consistiu na construção coletiva do conhecimento. Os internautas passaram a modificar o conteúdo dos *sites*, no entanto surgiu a dificuldade em lidar com o excesso de informação. O termo Web 2.0 foi criado em 2004 pela empresa O'Reilly Media [66]. Exemplos de *sites* que representam a Web 2.0 com seu uso coletivo de conhecimento são Del.icio.us, Flickr, Google Maps, YouTube, Amazon, Blogz e Wikipédia.

O termo Web 3.0 foi empregado pela primeira vez pelo jornalista John Markoff, num artigo do New York Times [62]. Ainda não é clara a fronteira para a Web 3.0, e embora os termos Web 1.0, 2.0 e 3.0 sejam controversos, a Web 3.0 pode ser entendida como um conjunto de tecnologias mais eficientes para ajudar os computadores a organizar e analisar a informação disponível na Web.

Um dos objetivos destas tecnologias é fornecer a fundação para sistemas que possam raciocinar da forma mais próxima possível de seres humanos, com o objetivo de ajudá-los de forma mais precisa e eficiente.

Desta forma, a Web 3.0 é a tentativa de tornar a *World Wide Web* (rede mundial) em uma *World Wide Database* (base de dados mundial). Exemplos de alguns *sites* que estão se esforçando para evoluir para Web 3.0 são: Facebook, Linkedin, Amazon e Lastfm.

2.2.2 XML

XML (*Extensible Markup Language*) é uma linguagem que permite a construção de documentos legíveis para seres humanos e que podem ser facilmente tratados por máquinas [83].

A linguagem XML possui um conjunto de marcadores que é mais expressivo e flexível do que a linguagem HTML, uma vez que estes marcadores podem ser definidos de acordo com as necessidades do usuário. A Figura 2.2 ilustra um exemplo de um documento XML que representa um currículo.

```
<?xml version="1.0" encoding="UTF-8"?>
<curriculo>
  <InformacaoPessoal>
    <DataNascimento>05-20-78</DataNascimento>
    <Nomecompleto>Manuel da Silva</Nomecompleto>
    <Contatos>
      <Telefone>9999-9999</Telefone>
      <CorreioEletronico>email@email.com</CorreioEletronico>
    </Contatos>
    <Nacionalidade>Portuguesa</Nacionalidade>
    <Sexo>M</Sexo>
  </InformacaoPessoal>
  <objetivo>Atuar na area de TI</objetivo>
  <Experiencia>
    <Cargo>Suporte tecnico</Cargo>
    <Empregador>Empresa de TI, Porto - Portugal</Empregador>
  </Experiencia>
  <Formacao>Superior Completo</Formacao>
</curriculo>
```

Figura 2.2: *Exemplo de um documento XML.*

2.2.3 RDF

RDF (*Resource Description Framework*) é uma linguagem utilizada para descrever recursos na Web. Esses recursos, a princípio, podem ser qualquer coisa: um título, uma pessoa, um autor, uma data de modificação, um conteúdo, informações de licença, uma página da Web etc [87].

O RDF foi projetado para alcançar os seguintes objetivos [87]:

- Ser um modelo simples de dados;
- Capacitar uma inferência semântica e provável;
- Utilizar um vocabulário baseado em URIs extensível;
- Utilizar uma sintaxe baseada em XML;
- Auxiliar o uso de esquemas XML (*XML Schemas*);
- Permitir que qualquer pessoa crie declarações sobre qualquer recurso.

Cada tripla RDF representa uma declaração (*statement*) de um relacionamento entre conceitos que são representados por nós de um grafo. Como pode ser observado na Figura 2.3, cada declaração é formada por 3 partes:

- Um sujeito;
- Um objeto;
- Um predicado (também chamado de propriedade) que denota um relacionamento.

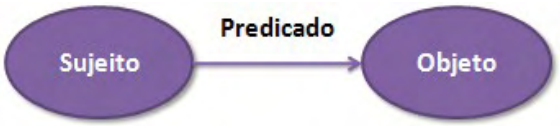


Figura 2.3: *Exemplo de uma declaração RDF.*

A direção do arco é importante e sempre aponta em direção ao objeto. O sujeito é nomeado utilizando um URI ou pode ser "anônimo". O predicado é nomeado utilizando um URI, no entanto, normalmente, é representado através de *qnames*, que são um formato de representação abreviado e conveniente. O objeto pode ser nomeado por um URI, ser anônimo, ou ser um literal.

Um conjunto de triplas é denominado grafo RDF. A Figura 2.4 ilustra um exemplo de um grafo RDF que descreve 3 declarações sobre uma página Web.

Como pode ser visto na Figura 2.4, o grafo indica 3 declarações sobre uma página *index.html*: a primeira indica a data de criação da página; a segunda denota a linguagem utilizada na página; e a terceira aponta o criador da página. As duas primeiras declarações são representadas com retângulos para indicar que são literais e a última utiliza uma elipse para representar um URI.

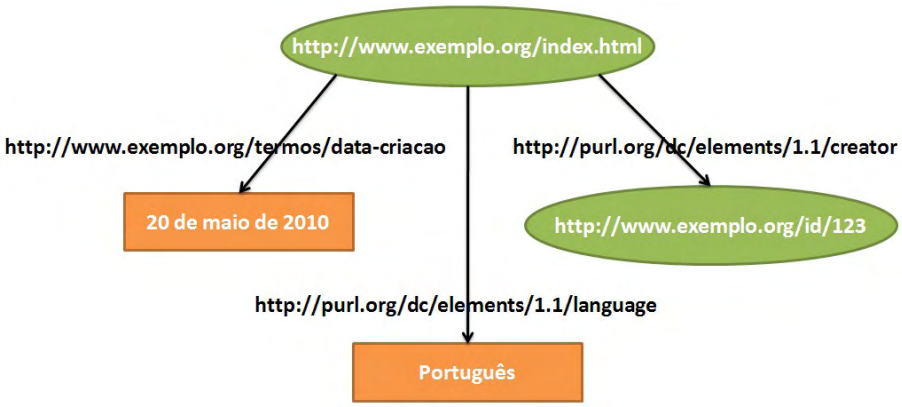


Figura 2.4: *Exemplo de um grafo RDF.*

A Figura 2.5 mostra um documento RDF que representa o grafo da Figura 2.4.

```
<http://www.exemplo.org/index.html>
<http://www.exemplo.org/termos/data-criacao>
"20 de maio de 2010"

<http://www.exemplo.org/index.html>
<http://www.purl.org/dc/elements/1.1/language>
"Português"

<http://www.exemplo.org/index.html>
<http://www.purl.org/dc/elements/1.1/creator>
<http://www.exemplo.org/id/123>
```

Figura 2.5: *Exemplo de um documento RDF no formato "N-TRIPLES".*

A notação de triplas, chamada de N-TRIPLES, pode resultar em várias linhas longas em uma página. Existem outras notações tais como a notação abreviada, que pode ser vista na Figura 2.6, e a notação Turtle (muito parecida com a notação N3) que pode ser vista na Figura 2.7.

```
ex:index.html extermos:data-criacao "20 de maio de 2010"
ex:index.html dc:language "Português"
ex:index.html dc:creator exid:123.
```

Figura 2.6: *Documento RDF com notação abreviada.*

```
<http://www.exemplo.org/index.html>
   extermos:data-criacao "20 de maio de 2010" ;
   dc:language "Português";
   dc:creator <http://www.exemplo.org/id/123>
```

Figura 2.7: *Documento RDF com notação Turtle/N3.*

2.2.4 RDFS

RDF Schema (também abreviado como RDFS, RDF(S), RDF-S, ou RDF/S) é uma linguagem de representação de conhecimento extensível, que fornece elementos básicos para a descrição de ontologias. RDF é utilizada como uma linguagem de propósitos gerais para representação de informações na Web [89].

RDFS é uma extensão do RDF e permite afirmar que uma classe (tipo) é uma subclasse de outra, e que uma propriedade é especificada por um âmbito (sujeito) ou domínio (objeto) [88].

Recursos podem ser divididos em grupos chamados classes. Os membros (também chamados de indivíduos) de uma classe são conhecidos como instâncias da classe. Classes são por si próprias recursos e são frequentemente identificadas por uma referência URI, e são descritas utilizando propriedades RDF.

A propriedade *rdf:type* pode ser utilizada para expressar a instância de uma classe. A Figura 2.8 ilustra um exemplo de uma propriedade *rdf:type* utilizada para declarar que John é uma instância da classe *Person*.

```
John rdf:type foaf:Person
```

Figura 2.8: *Exemplo de uma propriedade rdf:type.*

Se a classe C é uma subclasse da classe C', então todas as instâncias de C também serão instâncias de C'. A propriedade *rdfs:subClassOf* pode ser utilizada para expressar que uma classe é subclasse de outra. O termo superclasse é utilizado como inverso de subclasse. Se a classe C' é uma superclasse de uma classe C, então todas as instâncias de C também são instância de C'.

A propriedade *rdfs:subClassOf* permite declarar que um recurso é uma classe de outros recursos. A Figura 2.9 ilustra um exemplo de uma propriedade *rdfs:subClassOf* utilizada para expressar que toda *Person* é um *Agent*.

```
foaf:Person rdfs:subClassOf foaf:Agent
```

Figura 2.9: *Exemplo de uma propriedade rdfs:subClassOf.*

Além de permitir a utilização de classes e propriedades, RDFS também permite definir restrições (*constraints*). As propriedades *rdfs:range* e *rdfs:domain* estabelecem restrições (limites) sobre as propriedades de um recurso. *rdfs:range* é usada para indicar quais os valores que uma determinada propriedade pode ter. *rdfs:domain* especifica quais classes podem utilizar determinada propriedade.

2.2.5 Formalismos para Representação de Conhecimento

Os formalismos para representação de conhecimento mais utilizados são [82]:

- **Lógica**: a lógica é a base para a maioria dos formalismos de representação de conhecimento, seja de forma explícita, como nos sistemas especialistas baseados na linguagem Prolog, seja implícita na forma de representações específicas que podem facilmente ser interpretadas como proposições ou predicados lógicos;
- **Redes Semânticas**: uma rede semântica consiste em um conjunto de nós conectados por um conjunto de arcos. Os nós, em geral, representam objetos, e os arcos representam relações binárias entre esses objetos (é um, tipo um, tipo de, maior que) ou definem novas entidades (altura, cor);
- **Frames**: os *frames* (ou quadros) foram introduzidos para permitir a expressão das estruturas internas dos objetos, mantendo a possibilidade de representar herança de propriedades como as redes semânticas. O método de frames também está na origem das idéias que levaram às linguagens de programação orientadas a objetos;
- **Taxonomias**: as taxonomias tiveram sua origem nas ciências da vida[6], sendo utilizadas para nomear, descrever e classificar seres vivos. Tem como objetivo organizar todas as entidades de um universo na forma de hierarquia;
- **Tesauros**: um tesauro é um vocabulário controlado, estruturado e organizado em uma ordem conhecida, de modo que os relacionamentos de equivalência, homográficos, hierárquicos e associativos entre termos sejam indicados e identificados claramente por indicadores padronizados dos relacionamentos.
- **Ontologias**: definem conceitos e seus relacionamentos utilizados para descrever e representar um domínio de conhecimento. A idéia básica é criar formas de explicitar conceitos e o relacionamento destes conceitos em vários domínios de conhecimento, permitindo que máquinas possam trabalhar sobre estes conceitos, relacionando-os e inferindo novos conceitos [38] e [90].

O termo ontologia foi originado na filosofia grega (Aristóteles 384-322 a.C). Na filosofia, uma ontologia é uma teoria sobre a existência da natureza, sobre que tipos de coisas existem ou o que se dizer sobre o mundo.

Segundo Gruber [48], uma ontologia é uma especificação explícita e formal de uma conceitualização compartilhada. Ou seja, uma ontologia conceitua um modelo abstrato de algum fenômeno do mundo em um conhecimento consensual, isto é, compartilhado por todos. Além disso, os conceitos, as propriedades, as funções, os axiomas devem ser especificados explicitamente e serem manipuláveis por computador.

[6] Uma taxonomia bastante conhecida é a de Lineu [65].

A Figura 2.10 apresenta um trecho de diagrama simples que representa uma ontologia que descreve o planeta Terra.

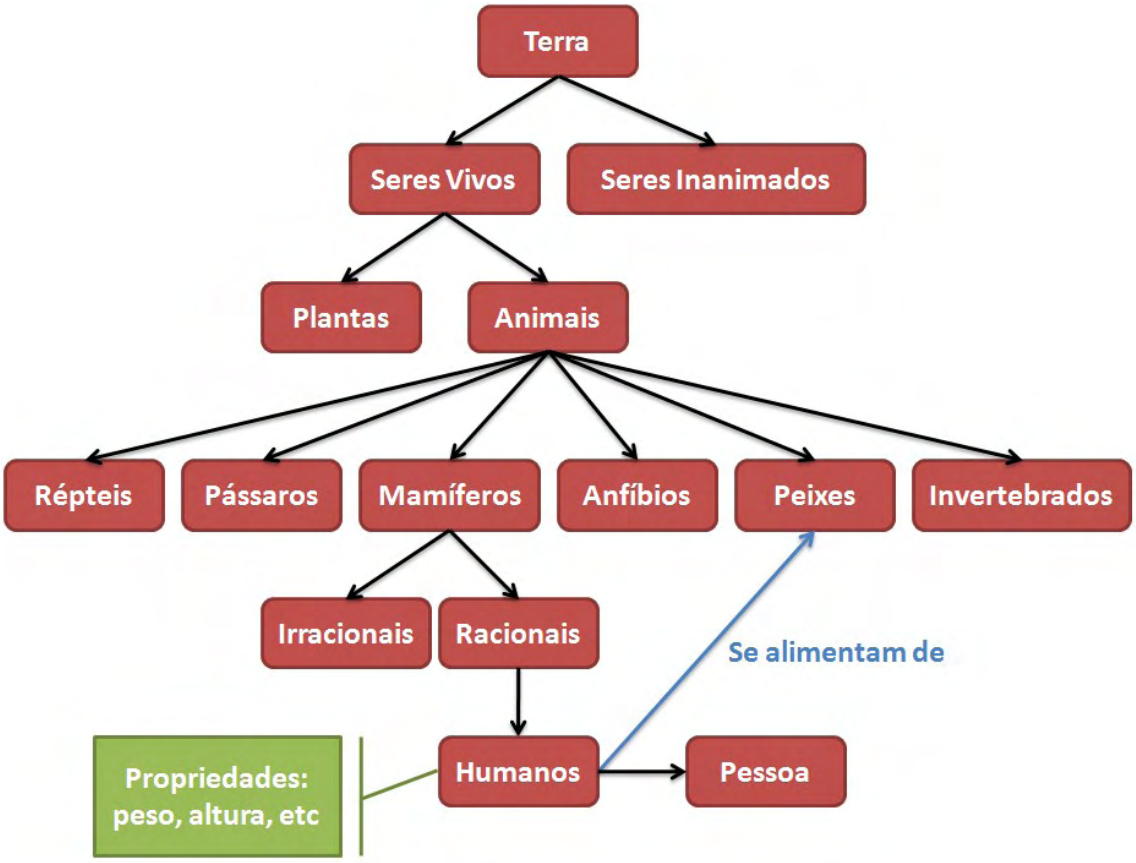


Figura 2.10: *Representação de uma ontologia.*

Os componentes usuais de ontologias incluem [38]:

- **Indivíduos**: instâncias ou objetos de classes;
- **Classes**: conjuntos, coleções, conceitos, tipos de objetos, tipos de coisas;
- **Atributos**: aspectos, propriedades, características, traços, parâmetros que objetos (e classes) podem ter;
- **Relações**: formas em que classes e indivíduos podem ser relacionados com outros;
- **Termos** de Funções: estruturas complexas formadas por certas relações que podem ser utilizadas em lugar de termos individuais de uma declaração;
- **Restrições**: descrições que devem ser verdadeiras durante uma entrada (*input*);
- **Regras**: declarações no formato se-então (*if-then*)[7], para inferências lógicas;
- **Axiomas**: declarações que descrevem de forma lógica um domínio. Consistem apenas em declarações que são aceitas como verdadeiras por si mesmas;
- **Eventos**: a mudança de atributos ou relações.

[7] Antecedente-consequente.

Diferentes linguagens de ontologias provêem diferentes facilidades. Dentre estas linguagens pode-se citar: a SHOE, XOL, OIL e DAML. Estas duas últimas foram combinadas e formaram a DAML+OIL. No entanto, a linguagem para ontologias recomendada pela W3C é a OWL.

A OWL é uma revisão da linguagem DAML+OIL. Embora seja baseada em RDF e RDF Schema e utilize a sintaxe XML, OWL é mais expressiva do que XML, RDF e RDF Schema. A OWL foi projetada para ser usada por aplicações que necessitem processar o conteúdo de informações, em vez de somente apresentar a visualização destas informações.

2.2.6 OWL

A **OWL** (*Web Ontology Language*) é uma linguagem para definir e instanciar ontologias na Web. É uma extensão semântica da linguagem RDF, permitindo especificar dependências lógicas adicionais entre estruturas de informação [86] e [85].

```
<owl:Class rdf:about="#VegetarianPizza">
  <rdfs:label xml:lang="pt">PizzaVegetariana</rdfs:label>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Pizza"/>
        <owl:Class>
          <owl:complementOf>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#hasTopping"/>
              <owl:someValuesFrom rdf:resource="#MeatTopping"/>
            </owl:Restriction>
          </owl:complementOf>
        </owl:Class>
        <owl:Class>
          <owl:complementOf>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#hasTopping"/>
              <owl:someValuesFrom rdf:resource="#FishTopping"/>
            </owl:Restriction>
          </owl:complementOf>
        </owl:Class>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Figura 2.11: *Exemplo de uma classe OWL [40].*

RDF é fácil de utilizar em relação à OWL. No entanto, apesar de ser mais complexa, OWL proporciona muito mais expressividade. A Figura 2.11 mostra um trecho da ontologia *pizza.owl*, onde a classe *VegetarianPizza* descreve uma pizza vegetariana.

2.2.7 SPARQL

O SPARQL (Acrônimo recursivo de SPARQL Protocol and RDF Query Language)[8] é uma linguagem de consulta e manipulação de modelos RDF e é baseado nas antigas linguagens de consulta RDQL, rdfDB e SeRQL.

O SPARQL está atualmente em discussão no *W3C Working Draft* e se tornou uma recomendação oficial do W3C em janeiro de 2008. Ele é implementado em diversas ferramentas, inclusive o Jena (que é descrito no Capítulo 6). O pacote do Jena que implementa o SPARQL é o ARQ [29].

Como foi visto anteriormente, o RDF possibilita descrever dados de forma descentralizada e distribuída. Os modelos RDF podem ser unidos facilmente, e serializados para intercâmbio por meio HTTP. Desta forma, as aplicações podem ser fracamente acopladas em múltiplos bancos de dados sobre a Web.

Existem diversas formas de executar consultas SPARQL. Uma forma é através do comando *sparql –query teste.rq*, onde "teste.rq" representa o arquivo que possui a consulta no formato SPARQL. A Figura 2.12 mostra o resultado da consulta utilizando o Código 2.1.

Código 2.1 – Arquivo teste.rq contendo a consulta SPARQL.

```
1 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2 SELECT ?url
3 FROM <bloggers.rdf>
4 WHERE {
5   ?contributor foaf:name "Jon Foobar" .
6   ?contributor foaf:weblog ?url .
7 }
```

```
---------------------------
| url |
===========================
| <http://foobar.xx/blog> |
---------------------------
```

Figura 2.12: *Resultado de uma consulta SPARQL.*

O SPARQL é uma tecnologia interessante, pois ilustra o uso da Web Semântica como um único e enorme banco de dados.

[8] pronunciado como "sparcol".

2.2.8 Microformatos

Microformatos são conjuntos de padrões para formatação de dados comuns em páginas ou documentos Web, de forma que estes dados possam ser interpretados tanto por humanos quanto por máquinas. São um conjunto de formatos abertos projetados para acrescentar marcações semânticas em qualquer documento XML, especialmente HTML e XHTML [22].

Aplicações, como buscadores, podem extrair informações específicas de páginas que usam microformatos. Existem microformatos para os mais variados tipos de dados, como: localização geográfica (latitude e longitude); categorização de conteúdo (*tag*); autor do texto; contato de redes de mensagens instantâneas; *link* para perfil pessoal em *sites* como Flickr, Blogger.com e Del.icio.us; menus de *sites*; listas retráteis (como FAQs); e currículos profissionais.

hCard é um microformato para publicar informações de contato de pessoas, companhias, empresas, e organizações (como um cartão de visitas) para páginas (X)HTML, Atom, RSS, ou XML. A Figura 2.13 ilustra um exemplo de microformato hCard.

```
<div class="vcard">
  <div class="fn">Joe Doe</div>
  <div class="org">The Example Company</div>
  <div class="tel">604-555-1234</div>
  <a class="url" href="http://example.com/">http://example.com/</a>
</div>
```

Figura 2.13: *Exemplo de microformato hCard.*

hResume é um microformato para publicação de currículos. Ele pode ser embutido em páginas HTML, XHTML, Atom, RSS, ou XML [54]. Um exemplo de *site* que implementa microformatos hResume é a rede social LinkedIn [21], onde são utilizados para publicar as páginas de perfil. Outros microformatos bastante conhecidos são hEvent, hCalendar, XFN, hReview, VoteLinks, rel-license, geo, XOXO, hAtom, hAudio, hMedia, hProduct, e hRecipe [22].

Microformatos são bastante populares para embutir metadados em páginas XHTML. A principal vantagem dos microformatos em relação ao RDF, é a possibilidade de inserir os metadados diretamente no XHTML. No entanto, o RDF possui várias vantagens em relação aos Microformatos: os recursos são representados por URIs, permitindo o acesso remoto de dados; é totalmente extensível com arquitetura aberta; utiliza uma linguagem poderosa (OWL); tem a habilidade de utilizar, compartilhar, e extender qualquer número de vocabulários; não é limitado pela codificação utilizada [75].

2.3 Agentes e Sistemas Multiagentes

Mesmo que sistemas possam ser baseados em um agente isolado, normalmente eles consistem em múltiplos agentes. Estes Sistemas Multiagentes (SMA) podem modelar sistemas complexos e introduzir a possibilidade dos agentes possuirem objetivos comuns ou confitantes [8].

Os agentes computacionais formam uma sub-área da Inteligência Artificial, e os Sistemas Multiagentes formam uma sub-área da Inteligência Artificial Distribuída.

A idéia principal em um sistema multiagente é que um comportamento global inteligente pode ser alcançado a partir do comportamento individual dos agentes. Em um SMA não é necessário que cada agente seja individualmente inteligente para alcançar um comportamento global inteligente.

2.3.1 Agentes

Embora o termo **agente** exista em diversas áreas como sociologia, economia, comportamento animal e robótica [44], o termo neste trabalho se restringe aos agentes inteligentes computacionais que são os agentes estudados na Ciência da Computação.

No entanto, dentro da Ciência da Computação, existem diversas definições a respeito de agentes:

- "Um agente é tudo o que pode ser considerado capaz de **perceber** seu ambiente por meio de sensores e de **agir** sobre esse ambiente por intermédio de atuadores" [71].
- "Um agente é um sistema que tenta preencher um conjunto de objetivos em um ambiente complexo e dinâmico além de possuir as seguintes propriedades: **autonomia** que é a capacidade de tomar decisões por si próprio; **adaptabilidade** que é a capacidade de melhorar a sua experiência com o tempo; e **eficiência** que consiste em ser bem sucedido em atingir objetivos" [60].
- "Um agente é aquele que possui as seguintes propriedades: **autonomia** que é a capacidade do agente operar sem intervenção humana; **habilidade social** que é a capacidade do agente interagir com outros agentes (também humanos) através de alguma espécie de linguagem; **reatividade** que é a capacidade do agente perceber seu ambiente; e **pró-atividade** que é a capacidade do agente apresentar um comportamento orientado a objetivos (tomar a iniciativa)" [94].

Embora não exista uma definição consensual, todas as definições aceitam o fato do agente ser essencialmente um *software* que provê autonomia e comportamentos parecidos com o de um agente humano. Assim, pode-se afirmar que o principal objetivo de um agente é basicamente substituir um humano na execução de determinada tarefa. Outras definições e classificações do termo agente podem ser vistas em [46].

2.3.2 Sistemas Multiagentes

Os **Sistemas Multiagentes** formam uma sub-área da Inteligência Artificial Distribuída e envolve o estudo de agentes autônomos em um universo multiagente.

Sistemas Multiagentes são utilizados para o compartilhamento de recursos além da resolução de problemas em ambientes dinâmicos e heterogêneos.

Sistemas Multiagentes possuem vários agentes que trabalham em conjunto. Podem ser classificados em colaborativos ou concorrentes [33].

O sistema colaborativo é um grupo de agentes que cooperam na resolução de problemas que estão além da capacidade de resolução de um agente isolado.

Já o sistema concorrente é aquele no qual os agentes não colaboram por um objetivo em comum, cada um trabalhando apenas para seu próprio objetivo.

Usualmente, cada agente no Sistemas Multiagentes possui um conjunto de capacidades comportamentais que definem sua competência, um conjunto de objetivos, e a autonomia necessária para utilizar suas capacidades comportamentais a fim de alcançar seus objetivos.

Os Sistemas Multiagentes também podem ser classificados como homogêneos ou heterogêneos. O sistema homogêneo é um grupo de agentes idênticos, já que possuem os mesmos objetivos, ações e domínios de conhecimento.

Ao contrário do sistema homogêneo, o sistema heterogêneo é aquele no qual os agentes são distintos, já que podem ter objetivos, ações ou domínios de conhecimento diferentes.

Outra classificação possível é em relação à comunicação entre os agentes. Se os agentes se comunicam entre si é intitulado um sistema não-comunicativo (*non-communicating*), caso contrário é intitulado um sistema comunicativo (*communicating*) [77].

A área de Sistemas Multiagentes é focada na coordenação e gerenciamento de comportamento de agentes, e está fortemente relacionada com outras áreas [77]:

- **Computação Distribuída**: muitos processadores compartilham dados, mas não controle. Esta é uma área focada em assuntos de paralelização ou sincronização.
- **Inteligência Artificial Distribuída**: tanto os dados quanto a inteligência é compartilhada. Esta é uma área focada na resolução de problemas, comunicação e coordenação.
- **Resolução de Problemas Distribuídos**: decomposição (ou compartilhamento) de tarefas; gerenciamento de informação.

2.4 Arquiteturas de Software

O dicionário Aurélio [45] define **arquitetura** como a disposição das partes ou elementos de um sistema.

A origem da **arquitetura de software** como um conceito foi primeiramente identificado no trabalho de pesquisa de Edsger Dijkstra em 1968 e David Parnas no início de 1970.

A definição de arquitetura de *software* segundo o padrão ISO/IEEE 1471-2000 [51] é:

"Arquitetura é a organização fundamental de um sistema incorporada em seus componentes, seus relacionamentos com o ambiente, e os princípios que conduzem seu design e evolução."

2.4.1 Arquiteturas de Agentes

Existem diversas arquiteturas de agentes, sendo que, de um extremo, há aquelas baseadas em agentes reativos e do outro, aquelas baseadas em agentes pró-ativos. Ambas arquiteturas podem ser classificadas em quatro principais grupos [7]:

- **Simbólicas**: utilizam técnicas tradicionais (baseadas em lógica) de representação e manipulação simbólicas; possuem a vantagem de serem fáceis de programar pelo fato dos humanos entenderem facilmente a lógica; e possuem a desvantagem de serem difíceis para traduzir o mundo real em uma descrição simbólica precisa e adequada; além de demandar um tempo considerável para a execução de seus resultados.
- **Reativas**: baseadas em mecanismos de estímulo e resposta; diferentemente das arquiteturas baseadas em lógica, não possuem nenhum modelo simbólico ou raciocínio simbólico; a vantagem é o processamento rápido deste tipo de arquitetura (embora não seja tão bom quanto o simbólico); e a desvantagem é a dificuldade de implementar aprendizado nos agentes.
- **Baseadas em BDI** (*Belief*, *Desire*, *Intention*): provavelmente é a arquitetura mais popular; baseada em quatro estruturas de dados chave - crenças (*Beliefs*), desejos (*Desires*), intenções (*Intentions*) e um interpretador. O interpretador é responsável por atualizar estes quatro elementos.
- **Baseadas em camadas**: são arquiteturas híbridas, que permitem aos agentes tanto um comportamento reativo quanto deliberativo (pró-ativo); para fornecer tal flexibilidade, utilizam um subsistema hierárquico em camadas.

2.4.2 Arquiteturas de Sociedades de Agentes

As duas principais arquiteturas de sociedades de agentes são [37]:

- **Arquitetura de Quadro-Negro** (*Blackboard*): Os sistemas de quadro-negro fornecem uma estrutura de controle central, denominado quadro-negro a qual é dividida em regiões ou níveis. Nesta arquitetura todas as interações ocorrem através do quadro-negro. Os agentes lêem e escrevem em um ou mais níveis sob a supervisão de um mecanismo global de escalonamento;
- **Arquitetura Baseada em Troca de Mensagens** (*Message Passing*): os agentes comunicam-se entre si através da troca de mensagens e, para tanto, torna-se necessário que os nomes dos agentes sejam conhecidos. A organização das interações é feita, com base em protocolos, que definem as etapas da conversação entre os agentes para cada tipo de interação possível na sociedade. Os protocolos e os formalismos para representação de mensagem podem ser bastante variados. Encontram-se na literatura vários protocolos como por exemplo, protocolos de apresentação, troca de conhecimentos e aprendizagem cooperativa.

As arquiteturas podem ser classificadas segundo a comunicação nas sociedades de agentes [37]:

- **Indireta**: determinando uma arquitetura de Quadro-Negro;
- **Direta**: determinando uma arquitetura baseada em troca de mensagens.

Segundo o tipo de seus agentes podem ser classificadas em [37]:

- **Homogêneas**: quando os agentes possuem a mesma arquitetura;
- **Heterogêneas**: quando os agentes possuem arquiteturas diferentes.

Também podem ser classificadas segundo a mobilidade de seus agentes [37]:

- **Fechadas**: quando os agentes são fixos;
- **Abertas**: quando há possibilidade de agentes entrarem/saírem da sociedade.

Quanto às regras de comportamento as sociedades classificam-se em [37]:

- **Baseadas em leis**: quando existem regras explícitas de comportamento para toda a sociedade;
- **Não Baseada em Leis**: quando não existem regras de comportamento explícitas.

2.4.3 Arquiteturas Distribuídas

Esta seção descreve os principais modelos de arquiteturas de *software* distribuído: Arquitetura Cliente-Servidor, Arquitetura MVC (Modelo, Visão e Controle), Arquitetura em Camadas, Arquitetura Baseada em Objetos, Arquitetura Baseada em Eventos e Arquitetura Centrada em Dados [79].

A Figura 2.14(a) mostra a Arquitetura Cliente-Servidor, que é uma das mais comuns, e consiste em apenas dois tipos de componentes: os clientes, que efetuam requisições de serviços, e servidores que retornam respostas de requisições para os clientes.

A Figura 2.14(b) representa a Arquitetura MVC (Modelo, Visão e Controle), que decompõe ainda mais o *software*, dividindo-o em 3 partes lógicas: o Modelo que consiste no nível de dados, a Visão que consiste no nível de interface de usuário, e o Controle que consiste no nível de processamento.

(a) (b)

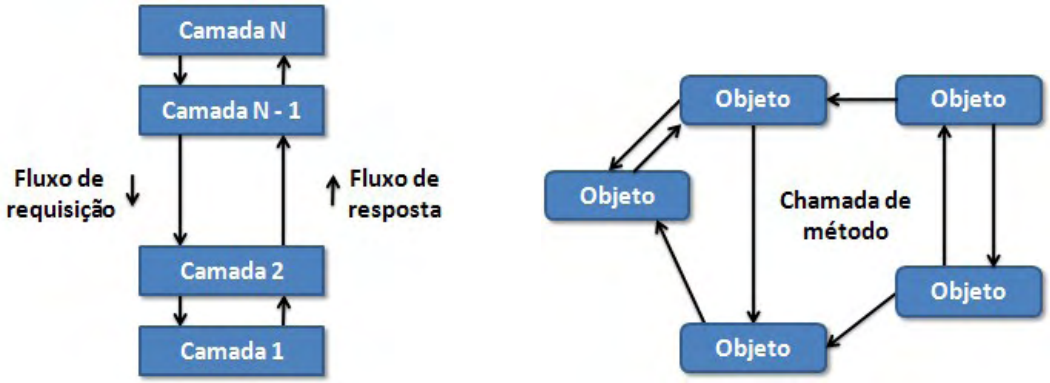
Figura 2.14: *(a) Arquitetura Cliente-Servidor; (b) Arquitetura MVC (Modelo, Visão e Controle).*

A Figura 2.15(a) apresenta a Arquitetura baseada em Camadas, na qual cada componente tem permissão de chamar componentes de um nível abaixo, mas não o contrário. O controle flui de camada para camada; requisições descem pela hierarquia, e resultados fluem para cima.

A Figura 2.15(b) ilustra a Arquitetura Baseada em Objetos, onde cada componente é um objeto, ou seja, possui atributos e métodos. Esses componentes são interligados por meio de chamadas de procedimento remota.

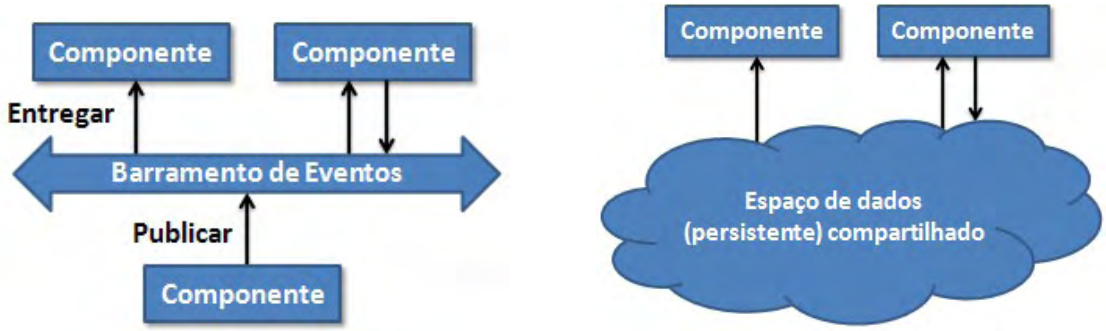
A Figura 2.16(a) apresenta a Arquitetura Baseada em Eventos, na qual processos se comunicam por meio da propagação de eventos que, opcionalmente, também transportam dados.

A Figura 2.16(b) ilustra a Arquitetura Centrada em Dados, que pode ser combinada com a Arquitetura Baseada em Eventos, tornando os processos desacoplados no tempo, ou seja, ambos não precisam estar ativos quando ocorre a comunicação.



(a) (b)

Figura 2.15: *(a) Arquitetura em Camadas; (b) Arquitetura baseada em Objetos.*



(a) (b)

Figura 2.16: *(a) Arquitetura baseada em Eventos; (b) Arquitetura centrada em Dados.*

2.5 Considerações sobre o Capítulo

Ao longo deste capítulo, foram apresentados os conceitos mais importantes para a compreensão e construção do arcabouço Airetama. As áreas relacionadas com o arcabouço Airetama que foram descritas no capítulo foram: Comunidades Virtuais de Prática, Sistemas Multiagentes e Web Semântica.

Existe muita polêmica a respeito destes conceitos, em especial sobre a Web 3.0, pois enquanto alguns estudiosos afirmam que não é possível a criação da Web Semântica (pessimismo provavelmente originado pelo descrédito criado através dos grandes fracassos da Inteligência Artifial), outros pesquisadores, como os da Radar Network, já estão pensando na Web 4.0.

No entanto, este trabalho se baseia nos dois pilares da Web Semântica: ontologias e agentes. A vantagem em utilizar um abordagem ontológica para armazenamento de dados, sem nenhuma utilização de banco de dados tradicionais, é o foco nas características de inteligência, interoperabilidade e integração destes dados.

A utilização de agentes para implementação de ferramentas torna a inclusão, mudança ou atualização destas ferramentas mais fácil, pois agentes são mais modulares e independentes que o software convencional. A utilização de Comunidades Virtuais de Prática serve principalmente para contextualizar os conceitos manipulados pelos usuários em um domínio específico.

No próximo capítulo, são apresentadas diversas ontologias que estão sendo utilizadas na Web Semântica.

CAPÍTULO 3

Ontologias

Neste capítulo são apresentadas diversas ontologias utilizadas na Web Semântica atualmente (ano de 2010). As duas primeiras, FOAF e SIOC, são utilizadas diretamente no projeto Airetama. As outras ontologias descritas neste capítulo são recomendadas pelo W3C e podem servir como expansões futuras para o repositório semântico do Airetama.

Diversas outras ontologias podem ser encontradas no Swoogle [30], que efetua buscas de ontologias entre mais de 10 mil ontologias. Outro *site* bastante útil é o diretório de ontologias SchemaWeb [25] que mostra mais detalhes como: nome da ontologia, descrição, Namespace (URI da ontologia), local onde pode ser baixada, página oficial, contato e versão. Há também a biblioteca de ontologias OntoSelect [24], que fornece detalhes como: nome, domínio, formato, linguagem, número de classes e propriedades.

As bio-ontologias, que são as ontologias relacionadas com biologia, tem crescido bastante na Web Semântica. O *site* SyntheticBiology [13] fornece diversos *links* de ontologias nesta área para vários domínios: genética, processos biológicos, ciências da vida, sequência de proteínas e nucleotídeos, aminoácidos, microarrays etc. Outro *site* bastante difundido em ontologias biológicas e biomédicas é o OBO Foundry [23], que reúne ontologias dos mais diversos domínios nessa área: processos celulares, bioquímica, anatomia, imunologia, neurociência, medicina etc.

3.1 FOAF

FOAF (um acrônimo de *Friend of a Friend*) é uma ontologia que serve para descrever pessoas (suas atividades, interesses, relações com outras pessoas etc.), como estão conectadas e as coisas que elas criam e fazem. Ela é uma tecnologia aberta descentralizada para conectar *sites* sociais, e as pessoas que os descrevem [16].

O projeto FOAF é dedicado a unir pessoas e informações através da Web. Ele integra três tipos de rede: as redes sociais de colaboração humana, amizade e de associação; redes de representação que descrevem uma visão simplificada de um universo; e redes de informação baseadas na Web para compartilhar descrições publicadas de forma independente e inter-relacionada [17].

FOAF não compete com os *sites* Web sociais, em vez disso ele fornece uma abordagem na qual os diferentes *sites* podem compartilhar recursos, e pelo qual os usuários podem manter algum controle sobre suas informações em um formato não-proprietário.

A ontologia FOAF tem evoluído gradualmente desde a sua criação em meados de 2000 por Dan Libby Brickley e Miller. Existe agora um núcleo estável de classes e propriedades que não serão alterados, além de ajustes modestos de sua documentação para acompanhar a execução e *feedback* as melhores práticas emergentes. Novos termos podem ser adicionados a qualquer momento e, consequentemente, esta especificação é um trabalho em evolução.

O projeto FOAF é muito importante para a chamada convergência de redes sociais. Atualmente, se um usuário quiser utilizar 10 redes sociais, ele deve recriar 10 vezes a sua conta, preenchendo formulários e inserindo repetitivamente seus dados nos *sites*. Futuramente, o usuário terá apenas um único perfil na Web que será compartilhado por diversas redes sociais e serviços.

FOAF é uma tecnologia simples que torna mais fácil de compartilhar e utilizar informações sobre pessoas e suas atividades (por exemplo, fotos, calendários, blogs), para transferir informações entre *sites*, e para estendê-las automaticamente, fundí-las e reutilizá-las *online*.

O resultado do projeto FOAF é uma rede de documentos que descrevem uma rede de pessoas e seus dados. Embora esses documentos FOAF nem sempre concordem ou digam a verdade, eles têm a característica útil de poder ser facilmente incorporados, permitindo descrições parciais e descentralizadas para ser combinadas de diversas maneiras.

A Figura 3.1 apresenta um exemplo de um documento FOAF, e a Figura 3.2 mostra o grafo equivalente. Ambas as figuras mostram um exemplo simples no qual duas pessoas são conectadas através da propriedade *foaf:knows*.

```
<rdf:Description rdf:about="airetama/akira@yahoo.com.foaf.rdf">
    <foaf:firstName>Akira</foaf:firstName>
    <foaf:surname>Sato</foaf:surname>
    <foaf:birthday>1967-01-04</foaf:birthday>
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
    <foaf:knows rdf:resource="airetama/juanes@gmail.com.foaf.rdf"/>
</rdf:Description>
<rdf:Description rdf:about="airetama/juanes@hotmail.com.foaf.rdf">
    <foaf:firstName>Juan</foaf:firstName>
    <foaf:surname>Escobar</foaf:surname>
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
</rdf:Description>
```

Figura 3.1: *Exemplo de um documento FOAF.*

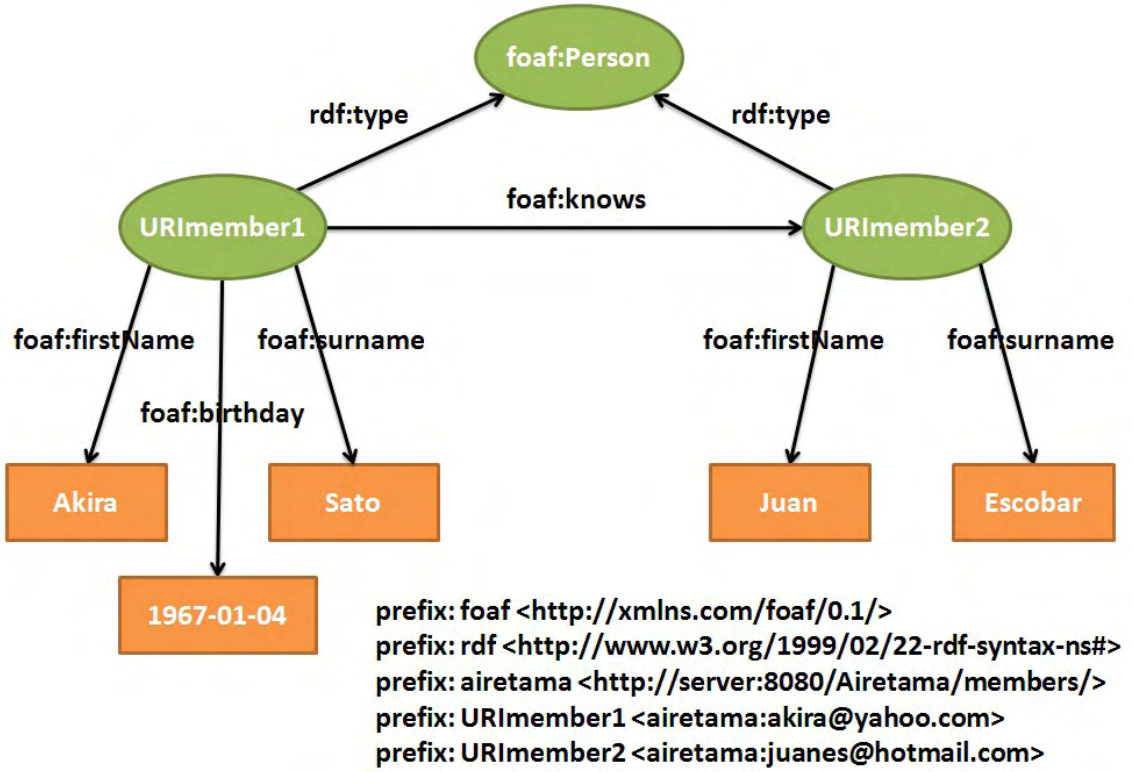


Figura 3.2: *Representação em Grafo do documento FOAF.*

O interessante da ontologia FOAF é que a maioria dos recursos que podem ser acessados na Web possuem algum tipo de relacionamento com pessoas. Assim esta ontologia serve como ponto de partida onde outras ontologias de qualquer domínio podem ser acopladas de forma extensível.

No Capítulo 6 são apresentados os termos (classes e propriedades) da ontologia FOAF que são utilizados pelo Airetama.

3.2 SIOC

SIOC (*Semantically-Interlinked Online Communities* - Comunidades Online Semanticamente Interligadas)[1] é uma ontologia na Web Semântica para a representação de dados da Web Social em RDF, e visa permitir a integração de informações de comunidades *online* [26].

O projeto SIOC foi iniciado em 2004 por John Breslin e Uldis Bojars em DERI NUI Galway. Em 2007, o SIOC foi apresentado por 16 organizações e se tornou um membro da W3C (*W3C Member Submission*) [27].

[1] Pronuncia-se “shock”.

Recentemente, foi adotado em uma variedade de aplicações de *software* comercial e *open-source*, e é comumente usado em conjunto com o vocabulário FOAF para expressar o perfil pessoal e informação de rede social.

Ao se tornar uma forma padrão para expressar o conteúdo gerado pelo usuário a partir de determinados *sites*, SIOC permite novos tipos de cenários inovadores para o uso de dados.

A ontologia SIOC fornece os principais conceitos e propriedades necessárias para descrever as informações de comunidades *online* (por exemplo, fóruns de discussão, *mailing list*, *wikis*, *blogs*, etc) sobre a Web Semântica.

Os *sites* de comunidades *online* têm substituído os meios tradicionais de manter uma comunidade informada, através de bibliotecas e de publicação. Eles são uma valiosa fonte de informação e, muitas vezes, as informações podem ser encontradas somente em um *site* de comunidade *online*.

Mas há um problema, pois as informações normalmente estão fragmentadas, *sites* de comunidades *online* são como "ilhas sem pontes ligando-as". A descentralização da Web exige compromissos, que são ontologias na Web Semântica.

A ontologia SIOC consiste em um padrão aberto legível para máquinas (*machine-readable*) e baseado em RDF e RDFS, que expressa tanto os dados quanto seus respectivos meta-dados. Como foi abordado anteriormente, os documentos SIOC podem ser utilizados em conjunto com outras ontologias existentes, enriquecendo ainda mais a descrição da informação.

A Figura 3.3 representa a especificação SIOC, mostrando seus principais termos (*The SIOC Core Ontology definitions*).

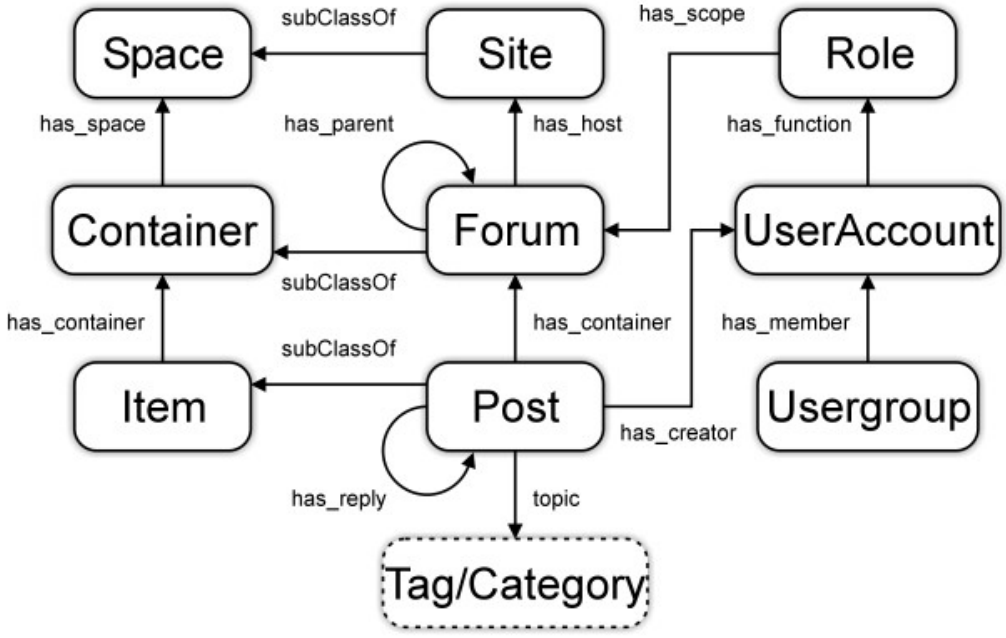


Figura 3.3: *Classes e propriedades SIOC [27].*

A Figura 3.4 mostra um exemplo de um documento que utiliza a ontologia SIOC em conjunto com a FOAF, e a Figura 3.5 ilustra sua representação equivalente em forma de grafo. Ambas as figuras apresentam um exemplo de uma pessoa que é conectada a uma comunidade através da propriedade *sioc:member_of*.

```
<rdf:Description rdf:about="airetama/akira@yahoo.com.foaf.rdf">
    <foaf:firstName>Akira</foaf:firstName>
    <foaf:surname>Sato</foaf:surname>
    <foaf:birthday>1967-01-04</foaf:birthday>
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
    <sioc:member_of rdf:resource="airetama/DNACommunity.sioc.rdf"/>
</rdf:Description>
<rdf:Description rdf:about="airetama/DNACommunity.sioc.rdf">
    <sioc:name>DNA Community</foaf:firstName>
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
</rdf:Description>
```

Figura 3.4: *Exemplo de um documento SIOC.*

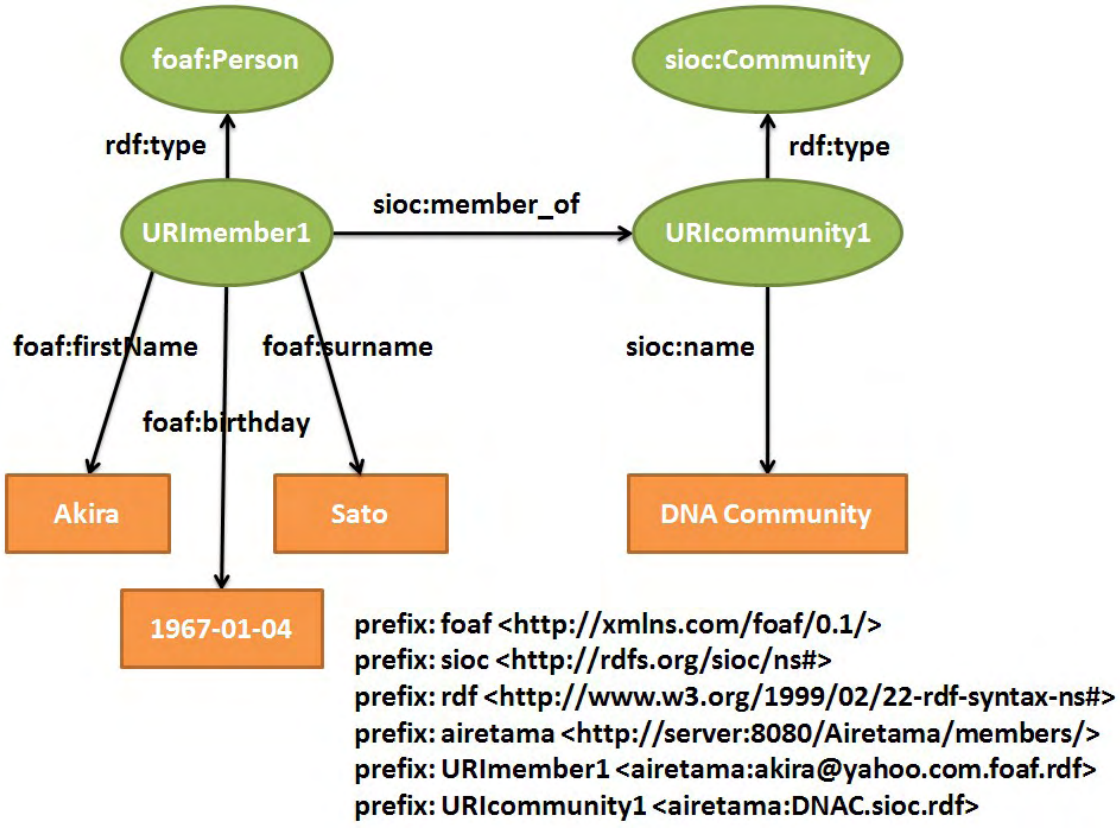


Figura 3.5: *Representação em Grafo de um documento SIOC.*

3.3 Relationship

Relationship é uma ontologia para descrever as relações entre as pessoas [35]. Cada propriedade Relationship é na verdade uma subpropriedade do termo *foaf:knows* da ontologia FOAF. As subpropriedades Relationship são mais expressivas pois, além de indicar o tipo de relacionamento entre duas pessoas, elas indicam se as relações são simétricas, transitivas ou inversas.

Algumas propriedades, utilizadas para conectar indivíduos do tipo *foaf:Person*, são listadas a seguir:

- **apprenticeTo**: representa um aprendiz de uma pessoa que tem a função de orientador ou professor;
- **childOf**: representa uma pessoa que nasceu ou foi criada por outra pessoa;
- **collaboratesWith**: representa uma pessoa que trabalha em um objetivo em comum com outra;
- **employerOf**: representa uma pessoa contratada por outra;
- **friendOf**: uma propriedade que representa uma amizade mútua;
- **neighborOf**: representa uma pessoa que vive na mesma localidade que outra;
- **worksWith**: representa uma pessoa que trabalha com outra que possui o mesmo empregador.

Recentemente, novos termos estão surgindo, como o *wouldLikeToKnow*, que representa a pessoa que deseja conhecer alguém. Muitos outros termos da ontologia Relationship podem ser encontrados em [35].

3.4 iCalendar

A ontologia **iCalendar** serve para representar eventos, com o objetivo de integrar dados de calendário com outros dados da Web Semântica, como dados de redes sociais, conteúdo RSS, e dados multimídia [14].

A Figura 3.6 mostra um exemplo de um documento iCalendar.

```
<Vevent>
        <uid>20020630T230445Z-3895-69-1-7@jammer</uid>
        <dtstart>2002-07-03</dtstart>
        <dtend>2002-07-06</date>
        <summary>Scooby Conference</summary>
        <location>San Francisco</location>
</Vevent>
```

Figura 3.6: *Exemplo de um documento iCalendar.*

3.5 Goodrelations

A ontologia **GoodRelations** fornece o vocabulário para a anotação de comércio eletrônico (*e-Commerce*) na Web e pode ser utilizada para: vender, alugar, reparar, dispor, manter, fornecer serviços de *commodities* etc [18].

GoodRelations permite descrever relacionamentos entre recursos Web, ofertas feitas com esses recursos, pessoas jurídicas, preços, termos e condições, além de produtos e serviços. GoodRelations é oficialmente utilizado pelo Yahoo SearchMonkey e outras aplicações importantes. Ele fornece um vocabulário para expressar coisas como a oferta de venda de um telefone celular de um certo modelo e preço. Outro exemplo seria o aluguel ou manutenção de carros. Diversos outros detalhes podem ser expressos, como por exemplo: opções de entrega e pagamento, desconto sobre quantidades etc.

GoodRelations foi feito para aumentar a visibilidade de produtos e, assim como qualquer outra ontologia, pode ser utilizado em motores de busca de última geração e sistemas de recomendação.

3.6 SKOS

SKOS (*Simple Knowledge Organization System*) é um modelo de dados padrão para compartilhamento e integração de sistemas de organização de conhecimento através da Web. Muitos sistemas de organização de conhecimento, tais como: tesauros, taxonomias, esquemas de classificação, compartilham uma estrutura semelhante, e são usados em aplicações similares. SKOS captura muito dessas semelhanças, tornando-as explícitas, permitindo assim o compartilhamento de dados em diversas aplicações [28].

SKOS classifica recursos em termos de mais amplo ou mais específico, permite a designação de etiquetas preferenciais e alternativas e permite que as pessoas transportem rapidamente dicionários e glossários para a Web. A iniciativa SKOS tem como objetivo resolver o problema no FOAF para descrever temas, categorias (folksonomias), hierarquias e assuntos.

A Figura 3.7 apresenta um exemplo simples de um documento SKOS.

```
ex:renaissance skos:related ex:humanism.
ex:humanism skos:related ex:philosophicalAnthropology.
ex:philosophicalAnthropology skos:related ex:philosophyOfMind.
ex:philosophyOfMind skos:related ex:cognitiveScience.
```

Figura 3.7: *Exemplo de um documento SKOS [28].*

3.7 Geo Ontology

Geo Ontology é um vocabulário RDF que proporciona à comunidade da Web Semântica um espaço para representar latitude (lat), longitude (long), altura (alt) e outras informações sobre localizações [32].

Este vocabulário fornece apenas os termos básicos que podem ser usados em RDF quando houver necessidade de descrever as latitudes e longitudes. Pode-se assim descrever não só os mapas, mas as entidades que estão posicionados no mapa.

Os valores das coordenadas são os mesmos utilizados pelo GPS (*Global Positioning System*), bastante difundido na cartografia, navegação e áreas relacionadas. Esta ontologia é utilizada em diversos *sites* importantes como: *Geocoder.us*, *OpenGuides.org* e *Yahoo Maps*.

A Figura 3.8 apresenta um exemplo *standalone* de um documento que utiliza a Geo Ontology.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
  <geo:Point>
    <geo:lat>55.701</geo:lat>
    <geo:long>12.552</geo:long>
  </geo:Point>
</rdf:RDF>
```

Figura 3.8: *Exemplo de um documento que utiliza a Geo Ontology.*

A Figura 3.9 mostra um exemplo de documento que combina as ontologias Geo, FOAF e o padrão Dublin Core.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns="http://xmlns.com/foaf/0.1/">
<Person>
 <name>Dan Brickley</name>
 <homepage dc:title="Dan's home page" rdf:resource="http://danbri.org/"/>
 <based_near geo:lat="51.47026" geo:long="-2.59466"/>
 <rdfs:seeAlso rdf:resource="http:/danbri.org/foaf.rdf"/>
</Person>
</rdf:RDF>
```

Figura 3.9: *Exemplo de um documento que combina Geo, FOAF e Dublin Core [32].*

3.8 DOAC

DOAC é um vocabulário de metadados RDF para descrever a capacidade profissional de um trabalhador (funcionário) assim como um currículo que as pessoas usam para ter em seus *sites*, porém, com maiores capacidades. Desta forma, pessoas de negócios são capazes de pesquisar na internet para encontrar um funcionário que se adapta às suas necessidades [67].

O DOAC foi projetado para ser compatível com o vocabulário FOAF. A Figura 3.10 mostra um exemplo de um documento DOAC utilizado em conjunto com FOAF:

```
<foaf:Person>
   <foaf:name>Ramon Antonio Parada</foaf:name>
   <foaf:mbox rdf:resource="mailto:rap@ramonantonio.net" />
   <foaf:homepage rdf:resource="http://ramonantonio.net/" />
   <doac:experience>
      <doac:VolunterExperience>
         <doac:title>Website Mantainer</doac:title>
         <doac:organization>Equus Zebra</doac:organization>
         <doac:start-date>2000-06-15</doac:start-date>
         <doac:end-date>2000-06-15</doac:end-date>
      </doac:VolunterExperience>
   </doac:experience>
   <doac:education>
      <doac:Degree>
         <doac:title>Systems Engineer</doac:title>
         <doac:organization>University of a Corunha</doac:organization>
         <doac:start-date>2000-10-01</doac:start-date>
         <doac:end-date>2006-06-15</doac:end-date>
      </doac:Degree>
   </doac:education>
   <doac:skill>
      <doac:LanguageSkill>
         <doac:language>es</doac:language>
         <doac:reads rdf:resource="http://ramonantonio.net/doac/0.1/#nativelevel"/>
         <doac:writes rdf:resource="http://ramonantonio.net/doac/0.1/#nativelevel"/>
         <doac:speaks rdf:resource="http://ramonantonio.net/doac/0.1/#nativelevel"/>
      </doac:LanguageSkill>
   </doac:skill>
   <doac:skill>
      <doac:LanguageSkill>
         <doac:language>en</doac:language>
         <doac:reads rdf:resource="http://ramonantonio.net/doac/0.1/#highlevel"/>
         <doac:writes rdf:resource="http://ramonantonio.net/doac/0.1/#highlevel"/>
         <doac:speaks rdf:resource="http://ramonantonio.net/doac/0.1/#highlevel"/>
      </doac:LanguageSkill>
   </doac:skill>
</foaf:Person>
```

Figura 3.10: *Exemplo de um documento DOAC [67].*

3.9 DOAP

DOAP é um vocabulário XML/RDF que descreve projetos de *software*. Os seus principais objetivos são [41]:

- Descrição de um projeto de *software* e seus recursos associados em nível internacional, incluindo os participantes e os recursos da Web;
- Ferramentas básicas para permitir a fácil criação e do consumo de tais descrições;
- Interoperabilidade com outros projetos populares de metadados Web (RSS, FOAF, Dublin Core);
- Fácil importação de projetos em diretórios de *software*;
- O intercâmbio de dados entre diretórios de *software*;
- Mantenedores de pacotes para os distribuidores.

A Figura 3.11 mostra um exemplo de um documento DOAP.

```
<Project xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns="http://usefulinc.com/ns/doap#">
  <name>DOAP</name>
    <homepage rdf:resource="http://usefulinc.com/doap" />
    <created>2004-05-04</created>
    <shortdesc xml:lang="en">
      Tools and vocabulary for describing community-based software projects.
    </shortdesc>
    <description xml:lang="en">
      DOAP (Description of a Project) is an RDF vocabulary and associated set of tools for
      describing community-based software projects.  It is intended to be an interchange
      vocabulary for software directory sites, and to allow the decentralized expression of
      involvement in a project.
    </description>
    <mailing-list rdf:resource="http://lists.usefulinc.com/mailman/listinfo/doap-interest" />
    <release>
      <Version>
        <name>unstable</name>
        <created>2005-07-12</created>
        <revision>0.1</revision>
      </Version>
    </release>
    <license rdf:resource="http://usefulinc.com/doap/licenses/GPL" />
    <repository>
      <SVNRepository>
        <location rdf:resource="http://svn.usefulinc.com/svn/repos/trunk/doap/" />
        <browse rdf:resource="http://svn.usefulinc.com/cgi-bin/viewcvs.cgi/trunk/doap/" />
      </SVNRepository>
    </repository>
</Project>
```

Figura 3.11: *Exemplo de um documento DOAP.*

3.10 Dublin Core

Dublin Core é um esquema de metadados que visa descrever objetos digitais, tais como: videos, sons, imagens, textos e *sites* na Web. Aplicações de Dublin Core utilizam XML e o RDF. Este é o vocabulário básico que deve ser utilizado para descrever documentos. A Wikipédia está planejando derivar algumas propriedades dele [50].

O Dublin Core possui quinze elementos de metadados: *Title* (Título), *Creator* (Criador), *Subject* (Assunto), *Description* (Descrição), *Publisher* (Publicador), *Contributor* (Contribuidor), *Date* (Data), *Type* (Tipo), *Format* (Formato), *Identifier* (Identificador), *Source* (Origem), *Language* (Idioma), *Relation* (Relação), *Coverage* (Abrangência) e *Rights* (Direitos).

Cada elemento Dublin Core é opcional e pode ser repetido. Não há ordem no Dublin Core para apresentar ou usar os elementos. Após a especificação do original com 15 elementos, foi iniciado um processo onde estão sendo desenvolvidos novos termos.

A Figura 3.12 apresenta um exemplo de um documento Dublin Core.

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dc="http://purl.org/dc/elements/1.1/">
   <rdf:Description rdf:about="http://media.example.com/audio/guide.ra">
      <dc:creator>Rose Bush</dc:creator>
      <dc:title>A Guide to Growing Roses</dc:title>
      <dc:description>
         Describes process for planting and nurturing different kinds of
         rose  bushes.
      </dc:description>
      <dc:date>2001-01-20</dc:date>
   </rdf:Description>
</rdf:RDF>
```

Figura 3.12: *Exemplo de um documento Dublin Core [ ].*

3.11 Music Ontology

O **Music Ontology** é uma tentativa de ligar todas as informações sobre os artistas musicais, álbuns e faixas em conjunto, de MusicBrainz para MySpace. A meta é a de expressar todas as relações entre a informação musical a fim de ajudar as pessoas a encontrar alguma coisa sobre música e músicos [69].

A internet mudou a indústria da música. Os sistemas de compartilhamento P2P (*peer-to-peer*) como o Napster permitiram que os usuários compartilhassem qualquer música de seu computador com milhões de outras pessoas.

Comunidades de redes sociais *online* como o MySpace despertou um foco muito forte de interesses musicais. Milhões de usuários têm sido capazes de esculpir o seu nicho musical, ignorando as gravadoras e apelando diretamente para seu público-alvo.

Enquanto MySpace fornece um mecanismo para que os artistas gerem um novo conteúdo com facilidade, bases de dados livres como MusicBrainz continuam com o modelo mais tradicional de classificação de música, arquivando milhões de artistas, álbuns e faixas.

Serviços de recomendação musical, como Pandora tentam igualar os interesses musicais de usuários para novas faixas através de marcação (folksonomia) e comparação algorítmica. Imitando mais de perto o modelo das gravadoras, empresas como a Apple já estão vendendo faixas individuais de um dolar com o iTunes. Assim, a indústria da música foi completamente alterada.

A Music Ontology é dividida em 3 níveis de expressividade:

- **Nível 1**: fornece um vocabulário para descrever informação editorial (músicas/artistas/lançamentos etc.);
- **Nível 2**: fornece um vocabulário para expressar o fluxo de criação musical (composição, arranjos, performace, gravação etc.);
- **Nível 3**: fornece um vocabulário complexo de eventos para expressar, por exemplo, o que acontece durante uma performace particular.

3.12 Considerações sobre o Capítulo

Ao longo deste capítulo, foram apresentadas as principais ontologias que estão sendo utilizadas na Web Semântica. As duas primeiras, FOAF e SIOC, são utilizadas diretamente no arcabouço Airetama. As outras ontologias descritas neste capítulo são recomendadas pelo W3C e podem servir como expansões futuras para o repositório semântico do Airetama.

O interessante em utilizar a ontologia FOAF é que a maioria dos recursos que podem ser acessados na Web possui algum tipo de relacionamento com pessoas. Assim, esta ontologia serve como ponto de partida onde outras ontologias de qualquer domínio podem ser acopladas de forma extensível.

O uso da ontologia SIOC foi facilitado pelo fato dela ser projetada para ser integrada com a FOAF. O mesmo modelo de integração pode ser utilizado para incluir novas ontologias, enriquecendo ainda mais a descrição da informação.

No próximo capítulo, são descritos os trabalhos relacionados com o arcabouço Airetama.

CAPÍTULO 4

Trabalhos Relacionados

Os três primeiros trabalhos apresentados neste capítulo (Coopractice, SemantiCore e fGrin) são de iniciativa brasileira. Isto demontra que, apesar da Web Semântica ser uma área nova[1], é uma área que desperta bastante interesse na comunidade científica, em especial aqui no Brasil.

Em seguida, é apresentada breve descrição de alguns dos trabalhos que estão relacionados com o arcabouço Airetama: OntoShare, Twine, Semantic MediaWiki, DBpedia, Swicki, WordNet, sua variante WordNet.PT, Cyc e Powerset. Estes trabalhos estão relacionados com o projeto Airetama pois: utilizam semântica nas suas bases de dados; grande parte utiliza sistemas multiagentes; e a maioria também utiliza redes sociais, wikis, ou outros tipos de grupos de pessoas que possibilitam interação colaborativa.

Existe uma infinidade de outros trabalhos ligados com a Web Semântica. Uma descrição detalhada dos portais semânticos Esperonto, OntoWeb, Empolis K42 e Modeca ITM pode ser encontrada em [57]. KA2 é bastante similar ao OntoShare e pode ser encontrado em [9]. Trabalhos similares ao Cyc são o Wolfram Alpha [93] e o Hakia [19]. Outra iniciativa brasileira, chamada Ubick [72], que é um *framework* baseado em agentes de *software* para computação ubíqua.

4.1 Coopractice

O **Coopractice** é uma arquitetura multiagente de manipulação do conhecimento. A arquitetura é composta de um conjunto de agentes que são responsáveis por filtrar, analisar e indexar o conteúdo trocado na comunidade de prática virtual, bem como o conteúdo externo relacionado ao domínio do assunto [80].

A arquitetura é formada por dois módulos principais: o módulo de organização de conteúdo, e o módulo de solução de problemas.

[1] A Web Semântica possui menos de 10 anos, já que a apresentação oficial ao mundo foi feita em 2001 por Tim Berners-Lee [11].

Estes dois módulos não são completamente independentes, possuindo como estruturas em comum a base de conhecimento da comunidade e a ontologia de domínio. Nesta arquitetura, um agente atua como um membro da comunidade, procurando por oportunidades de colaboração ao responder perguntas enviadas por outros membros da comunidade.

Como memória do agente, são utilizadas as próprias mensagens trocadas na comunidade de prática, ou de outras comunidades sobre o mesmo domínio, considerando também a indicação de material externo sobre o assunto tratado, como livros e endereços com conteúdo escrito que possa ser catalogado pelo agente.

O módulo de organização de conteúdo (MOC) é responsável pela captação e indexação de todo o conteúdo que fará parte da base de conhecimento da comunidade. Os agentes pertencentes a este módulo devem captar, organizar o conhecimento de maneira semi-estruturada, descartar informações irrelevantes e também buscar informações externas, que possam ser úteis à biblioteca de conhecimento da comunidade de prática.

O módulo de solução de problemas (MSP) contém os agentes que fazem a interface com os membros da comunidade de prática. Esses agentes tem como funcionalidade receber os questionamentos enviados à comunidade, para fazer análises no conhecimento pré-estruturado já organizado pelo MOC, e apoiado pela ontologia de domínio, e selecionar as melhores mensagens ou documentos relacionados à mensagem originalmente submetida.

Além do MOC e do MSP existe um agente externo, que não se encontra em nenhum desses módulos. Ele será o responsável por receber o questionamento da comunidade e enviar a resposta obtida pelo MSP. Este agente é o "agente de interface com a comunidade" (AIC).

Os mecanismos de associação de informações da arquitetura multiagente são proporcionados por uma ontologia de domínio, que tem como objetivo contextualizar a informação trocada, além da possibilidade de aprimoramento da pergunta enviada pelo membro da comunidade através da expansão dos termos associados a conceitos da ontologia.

4.2 SemantiCore

SemantiCore é um *framework* para a criação de aplicações na Web Semântica baseadas em agentes de *software*. [43].

Foi desenvolvido, em meados de 2002, a arquitetura Web Life, com o objetivo de permitir o desenvolvimento de sistemas multiagentes para a Web, incorporando os requisitos necessários à criação da Web Semântica.

O SemantiCore, apresentado inicialmente em 2004, surgiu a partir de uma extensão na arquitetura Web Life. O SemantiCore é estruturado como um *framework* que abstrai a plataforma de implementação e provê primitivas para a criação de aplicações organizadas em um conjunto de agentes que realizam suas tarefas no ambiente Web [43].

Posteriormente, uma nova versão foi criada, denominada SemantiCore 2006, onde agentes de *software* podem ser descritos através de uma estrutura ontológica. Representar os agentes em uma estrutura ontológica permite, por exemplo, que os agentes sejam instanciados a partir de sua representação.

Uma das principais inovações presentes na nova versão diz respeito ao modelo de arquitetura do agente. Os agentes do SemantiCore possuem uma estrutura orientada a componentes, onde cada componente contribui para uma parte essencial do funcionamento do agente. São quatro os componentes básicos: o sensorial, o decisório, o executor e o efetuador.

A Figura 4.1 mostra a arquitetura de um agente no SemantiCore.

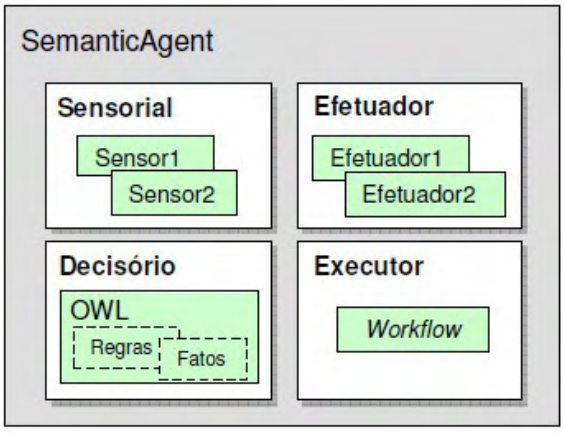


Figura 4.1: *Arquitetura de um agente no SemantiCore [43].*

O componente sensorial percebe e captura recursos que trafegam pelo ambiente. Este componente contém uma série de sensores definidos pelo desenvolvedor (cada sensor captura um tipo diferente de objeto do ambiente).

O componente decisório encapsula o mecanismo de tomada de decisão do agente. Este mecanismo é um dos pontos de flexibilidade do SemantiCore.

O componente executor contêm os planos de ação que serão executados pelo agente e o mecanismo necessário para controlar a execução das ações.

O encapsulamento de dados em mensagens para transmissão no ambiente é feito pelo componente efetuador. Da mesma forma que o componente sensorial, o componente efetuador armazena uma série de efetuadores, onde cada efetuador é responsável por publicar um tipo diferente de objeto no ambiente.

4.3 fGrin

fGrin é um *framework* baseado em sistemas multiagentes para formação de grupos de interesse a partir de uma base semântica. A partir de qualquer base de conhecimento anotada em DAML+OIL, o fGrin utiliza um sistema multiagente para formar grupos de pessoas que possuem interesses em comum [3].

Tem-se uma base de conhecimento que segue uma ontologia de um domínio específico. Na especificação dessa ontologia, encontram-se conceitos associados a pessoas, conceitos associados a interesses e relacionamentos que associam os conceitos referentes a pessoas aos conceitos associados aos interesses.

Existem pessoas, na base de conhecimento, que possuem interesse específico em comum, tais como: interesse em colecionar carros, em efetuar uma ação comunitária, em comprar determinado produto, em aprender determinada tecnologia ou em trocar informações a respeito de algum assunto geral ou específico. Com essas informações, pode-se construir o perfil de uma pessoa, utilizando-se uma heurística de definição de perfil.

Cada uma das pessoas recebe um agente pessoal, criado pelo agente coordenador, que tem a função de definir o seu perfil de acordo com o seu nível de interesse. A comunicação entre os agentes pessoais é muito importante na formação dos grupos, pois é a partir dela que os agentes verificam se as pessoas por eles representadas possuem um grau de afinidade desejado para que elas possam participar do mesmo grupo de interesse.

Um dos pontos em comum com o Airetama, é que o fGrin também utilizou a ferramenta JADE para a criação dos agentes e a ferramenta Jena para o acesso e recuperação da base semântica.

4.4 OntoShare

Ontoshare é um ambiente de gestão de conhecimento baseado em ontologias, voltado para Comunidades Virtuais de Prática. O perfil de cada usuário é gerado como um conjunto de conceitos ontológicos[2], para expressar seus interesses [34].

No modelo de Ontoshare, cada unidade de informação compartilhada está armazenada na forma de ontologias. Ontoshare é utilizado para armazenar, encontrar e organizar informações de usuários. O perfil do usuário é atualizado conforme a sua utilização no sistema, procurando refinar o perfil do usuário para criar um modelo melhor de seus interesses.

[2]Classes geradas em RDFS.

As anotações RDF são feitas de forma semi-autómática. Tendo em vista que a gerência de relacionamentos entre conceitos consome muito tempo, o Ontoshare fornece um sistema de evolução de suas ontologias. O Ontoshare encoraja o compartilhamento de informação entre Comunidades Virtuais de Prática. Os recursos e informações são marcadas utilizando metadados.

As ontologias são definidas utilizando RDF Schema (RDFS) e povoadas utilizando *Resource Description Framework* (RDF).

A Figura 4.2 mostra a estrutura ontológica do Ontoshare.

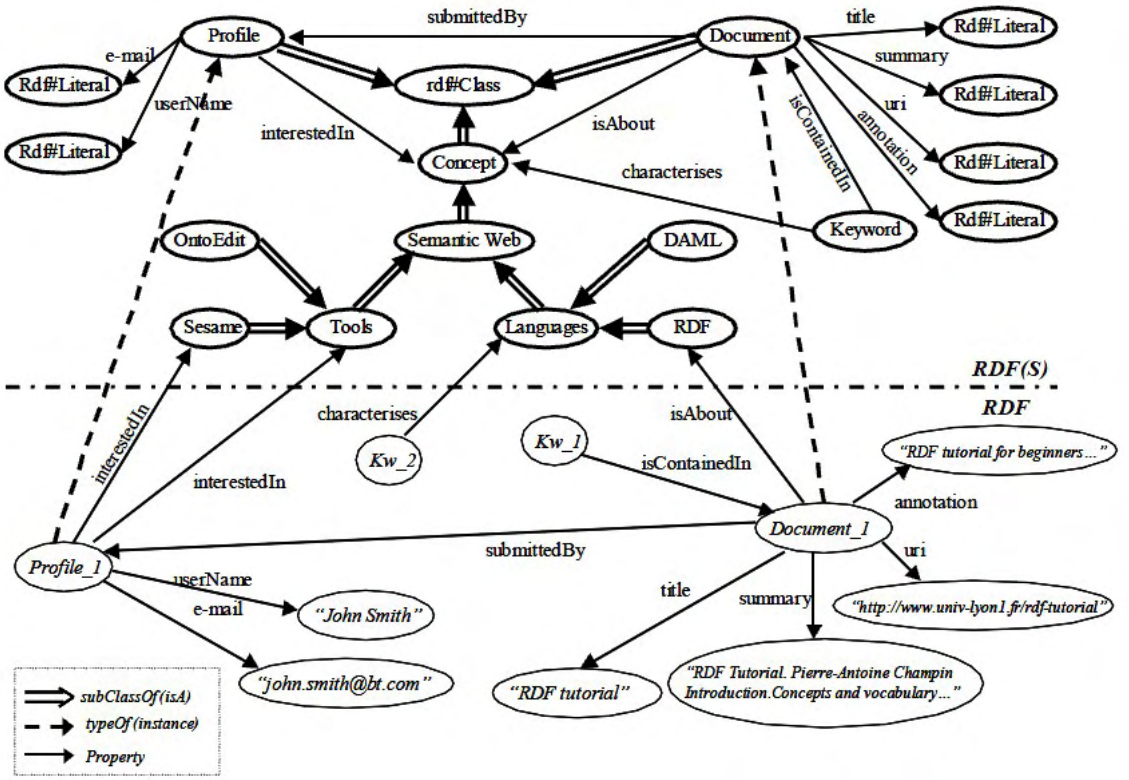


Figura 4.2: *Estrutura ontológica do Ontoshare [34].*

A classe *Concept* e suas subclasses representam os interesses da comunidade de prática. No exemplo da Figura 4.2, a subclasse sendo utilizada é a *Semantic Web*.

A classe *Document* representa a informação compartilhada. Um documento armazenado por um usuário possui as propriedades descritas na definição da classe *Document*: um URI, um sumário, um título, um conjunto de palavras-chave etc.

A classe *Profile* representa as informações do usuário, tais como: dados pessoais e os conceitos de seu interesse. No exemplo da Figura 4.2, pode-se observar um usuário que é instância da classe *Profile* com o valor "John Smith" para a propriedade *userName*, o valor "john.smith@bt.com" para a propriedade *e-mail* e o valor "Sesame" para a propriedade *interestedIn*.

4.5 Twine

Twine é um projeto criado pela empresa Radar Networks, que permite explorar conteúdo de *sites* de computação social, permitindo a gerência de diversos conteúdos, de viagens a filmes [76].

Os usuários podem utilizar o Twine para manter atualizados seus interesses. Twine é uma forma de colecionar conteúdo *online* (videos, fotos, artigos, páginas Web, produtos), e organizá-lo por tópicos.

Twine utiliza semântica para aprender, de forma automática, sobre os interesses, conexões e recomendações dos usuários.

Enquanto os atuais *sites* de recomendações de viagem forçam as pessoas a percorrerem longas listas de comentários e observações deixadas por outros, o Twine pesa e classifica todos os comentários e encontra, por exemplo, o hotel adequado para um usuário.

Assim como o Airetama, toda a informação no Twine é expressa em um conjunto de tuplas RDF[3]. No entanto, no Twine o URL ao ser acessado retorna uma página HTML, já no Airetama é retornada uma página JSP. A Figura 4.3 ilustra um exemplo de um documento RDF sobre "Jurassic Park", onde o autor é "Michael Crichton", e a data de atualização é "07/09/2006".

```
<rdf:RDF
xmlns:basic="http://www.radarnetworks.com/2007/09/12/basic#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:s="http://www.w3.org/2000/01/rdf-schema#">
<rdf:Description about="http://www.twine.com/item/1sj51123-2g5">
<s:label>Jurassic Park</s:label>
<basic:author>Michael Crichton</basic:author>
<basic:manufacturer>Arrow Books Ltd</basic:manufacturer>
<basic:tag>Science Fiction</basic:tag>
<rdf:type rdf:resource="http://www.radarnetworks.com/2007/09/12/basic#Book"/>
<radar:wasCreatedBy rdf:resource="http://www.twine.com/user/lew"/>
<app:hasComment rdf:resource="http://www.twine.com/item/1slbszfg-2x9"/>
</rdf:Description>
<rdf:Description about="http://www.twine.com/item/1slbszfg-2x9">
<rdf:type rdf:resource="http://www.radarnetworks.com/shazam#Comment"/>
<basic:description>One of my favorites</basic:description>
<radar:wasCreatedBy rdf:resource="http://www.twine.com/user/lew"/>
</rdf:Description>
</rdf:RDF>
```

Figura 4.3: *Exemplo de um documento RDF que gera uma página HTML no Twine [76].*

[3]Mais conhecidas como triplas RDF.

4.6 Semantic MediaWiki

Semantic MediaWiki é uma extensão do MediaWiki[4] que ajuda a pesquisar, organizar, marcar, exibir, avaliar, e compartilhar conteúdos wikis. Enquanto os wikis tradicionais contêm apenas texto que computadores não podem entender nem avaliar, Semantic MediaWiki adiciona anotações semânticas que permitem o wiki funcionar como um banco de dados colaborativo [56].

Semantic MediaWiki introduz algumas marcações adicionais nos textos wikis, permitindo que os usuários adicionem anotações semânticas às wikis. Embora pareça muito mais complicado utilizar anotações semânticas, elas permitem simplificar a estrutura da wiki, ajudando usuários a encontrar mais informações em menos tempo, e melhora a consistência da wiki.

Algumas das principais vantagens de utilizar Semantic MediaWiki são:

- **Geração automática de listas**: Wikis tendem a conter muitas listas agregadas. Um exemplo de uma lista da Wikipédia é: "Lista de áreas metropolitanas na Espanha por população". Estas listas são propícias a erros, desde que elas tenham que ser atualizadas manualmente;
- **Várias visualizações de informação**: são fornecidos diversos formatos e extensões que permitem personalizar visualizações de resultados semânticos, mapas semânticos, calendários, históricos, gráficos, entre outros;
- **Aperfeiçoamento da estrutura de dados**: Wikis tendem a utilizar categorias para estruturas de dados. Da mesma forma que as listas, as categorias podem ser atualizadas de forma automática;
- **Busca de informações**: usuários podem consultar informações específicas executando seus próprios métodos de busca;
- **Consistência de linguagem**: Wikis tradicionais frequentemente possuem uma grande quantidade de redundância de dados, que podem gerar inconsistências. Por exemplo, a população de Edimburgo é escrita nas Wikipédias Inglesa, Alemã e Francesa. Com os dados armazenados semanticamente, pode ser feita uma consulta que fornece o mesmo dado independentemente da linguagem utilizada.
- **Reuso externo**: os dados, uma vez criados de forma semântica, são mais fáceis de exportar via formatos como CSV, JSON e RDF.

[4] MediaWiki é um *software* para a execução de wikis (enciclopédias *online* e colaborativas) baseados na Web, utilizado nos mais populares wikis do mundo, sendo o mais conhecido a Wikipedia.

4.7 DBpedia

DBpedia é um esforço comunitário para extrair informação estruturada daWikipédia e torná-la disponível naWeb. Os artigos da Wikipédia consistem em sua maioria em textos livres, mas também possuem diversos tipos de informações estruturadas: tabelas de “infobox”, informação de categorização, imagens, coordenadas geográficas, e *links* de páginas Web externas. Essas informações estruturadas podem ser extraídas da Wikipédia e podem servir de base para permitir que a DBpedia faça consultas mais expressivas que a Wikipédia [5].

O papel das bases de conhecimento tem representado o grande aumento de inteligência na Web. Atualmente, as bases de conhecimento abrangem somente domínios específicos, e são criadas por grupos relativamente pequenos de engenheiros de conhecimento. A atualização destas bases de conhecimento representam um grande custo, mas ao mesmo tempo, Wikis tem tido uma taxa de crescimento cada vez maior.

A base de conhecimento da DBpedia atualmente descreve milhões de informações, incluindo milhares de pessoas, lugares, álbuns de música, filmes, e companhias. Esta base de conhecimento é criada através de triplas RDF.

A base de conhecimento da DBpedia possui diversas vantagens sobre outras bases: abrange diversos domínios; representa o real comum acordo da comunidade; é atualizada automaticamente a cada mudança da Wikipédia; e é completamente multilinguística.

Alguns exemplos de consultas expressivas que podem ser feitas nesta base de conhecimento são: “Quais são as cidades em Nova Jérsei que possuem mais de 10.000 habitantes?” ou “Quais são os músicos italianos do século 18”.

4.8 Swicki

Swicki é um motor de busca colaborativo, baseado em redes sociais, grupos de interesse e de confiança. Usuários podem criar e personalisar seus portais de busca. A cada busca efetuada, o Swicki torna suas buscas mais relevantes e sob medida para seus usuários da comunidade [61].

Seu mecanismo de busca utiliza um método chamado busca vertical, que trabalha para que os termos procurados possam ser relevantes e contextualizados. Também possui uma ferramenta de pesquisa que além de utilizar a relevância e contexto, também utiliza os interesses dos usuários.

Em cada busca feita por um usuário, é possível efetuar um voto contrário ou a favor deste resultado, tornando-o mais ou menos relevante quanto outro, determinando a sua posição entre os resultados. Assim, o Swicki aprende com o comportamento dos usuários através de notícias e frequência de buscas.

4.9 WordNet

WordNet é um grande banco de dados léxico de língua inglesa. Substantivos, verbos, adjetivos e advérbios são agrupados em conjuntos de sinônimos cognitivos chamados synsets, cada um expressando um conceito distinto [64].

Embora o WordNet não utilize ontologias[5], ele é citado neste trabalho devido à sua grande importância em diversos projetos da Web Semântica, como por exemplo a DBpedia.

Os *synsets* são interligados semanticamente e lexicamente, e essa rede de relacionamentos pode ser utilizada em um navegador Web. O WordNet também é disponível para *download*, sendo uma ferramenta útil para linguística computacional e processamento de linguagem natural.

O significado dos *synsets* é definido por glossários (sentenças). Um exemplo típico de *synset* é apresentado na Figura 4.4.

```
good, right, ripe --
  "most suitable or right for a particular purpose";
  "a good time to plant tomatoes";
  "the right time to act";
  "the time is ripe for great sociological changes".
```

Figura 4.4: *Exemplo de um synset [64].*

A maioria dos *synsets* é conectada com outros *synsets* através de relações semânticas. As principais relações são:

- **Sinônimo**: X é um sinônimo de Y se todo X é Y e todo Y é X;
- **Hiperônimo**: Y é um hiperônimo de X se todo X é um (tipo de) Y (*canine* é um hiperônimo de *dog*);
- **hipônimo**: Y é um hipônimo de X se todo Y é um (tipo de) X (*dog* é um hipônimo de *canine*);
- **holônimo**: Y é um holônimo de X se X é uma parte de Y (*building* é um holônimo de *window*);
- **merônimo**: Y é um merônimo de X se Y é uma parte de X (*window* é um merônimo de *building*).

Tanto substantivos quanto verbos são organizados em hierarquias, definidas por hiperônimos. Um exemplo típico de uma hierarquia é ilustrado na Figura 4.5.

[5] Alguns pesquisadores consideram a taxonomia do WordNet como uma ontologia fraca, mas neste trabalho uma representação de dados é considerada ontologia somente se possuir os componentes descritos no final da Seção 2.2.5.

```
dog, domestic dog, Canis familiaris
  => canine, canid
    => carnivore
      => placental, placental mammal, eutherian, eutherian mammal
        => mammal
          => vertebrate, craniate
            => chordate
              => animal, animate being, beast, brute, creature, fauna
                => ...
```

Figura 4.5: *Exemplo de uma hierarquia no WordNet [64].*

4.10 WordNet.PT

A **WordNet.PT** é uma base de dados de conhecimento linguístico do Português que pode ser utilizada em várias áreas da Linguística Computacional e da Engenharia da Linguagem, tais como: tradução automática, sistemas de busca e de extração de informação, sistemas periciais, aplicações para o ensino do Português, e dicionários [63].

A base de dados é organizada de acordo com o modelo geral da EuroWordNet, uma base de dados multilíngue que integra WordNets de várias línguas européias.

Uma WordNet é uma rede léxico-conceitual estruturada em torno de um conjunto de relações. O significado de uma dada unidade é deduzido da sua posição relativa no complexo de relações especificadas na rede, possuindo também informações complementares (glosas e etiquetas).

A unidade básica de uma WordNet é o conceito. Cada conceito corresponde a um nó da rede e é representado pelo conjunto das expressões lexicais que lhe correspondem, sejam elas atômicas ou complexas.

A atual versão da WordNet.PT inclui diversos termos, expressões nominais (nomes comuns e nomes próprios), verbais e adjetivais, relativos aos seguintes subdomínios semânticos: atividades profissionais e artísticas; alimentos; áreas geográficas e político-administrativas; instituições; instrumentos; meios de transporte; obras de arte; saúde e atividades médicas; seres vivos; vestuário e meios de comunicação.

O projeto desenvolve-se no quadro geral da EuroWordNet, contando com o apoio científico do seu coordenador, Dr. Piek Vossen, bem como da Dra. Christiane Fellbaum, responsável, juntamente com o Prof. Doutor George Miller, pela WordNet de Princeton, a “mãe” de todas as WordNets.

4.11 Cyc

Cyc é um *software* que combina ontologias com bases de conhecimento, juntamente com um motor de raciocínio e uma interface de linguagem natural [58].

Para fomentar o crescimento da comunidade de pesquisa, é disponibilizada uma licença gratuita que pode ser utilizada para fins não comerciais. Além disso, o núcleo de ontologias Cyc também está disponibilizado para domínio público.

A base de conhecimento Cyc é representada por uma grande quantidade de conhecimento humano: fatos, princípios básicos, além de heurísticas para raciocínio sobre objetos e eventos da vida cotidiana.

A representação é feita através da linguagem formal CycL. A base de conhecimento é constituída por um vocabulário da Cycl e de suas declarações sobre o relacionamentos destes termos. Cyc não é um sistema baseado em frames, a base de conhecimento é baseada em declarações e regras.

Atualmente, a base de conhecimento do Cyc possui aproximadamente duzentos mil termos e dezenas de declarações envolvendo cada termo. Milhões de novos termos não-atômicos e declarações são criados automaticamente, gerados por processos de inferência.

No entanto, a maior parte dos estudos é concentrada no processamento de linguagem natural. A seguir, é dado um exemplo de como é feito o processamento de linguagem natural:

- Fred viu o avião voando sobre Zurique;
- Fred viu as montanhas voando sobre Zurique.

Embora as sentenças sejam similares, humanos não possuem dificuldade em reconhecer que “voando” se refere ao avião, e não à montanha. Sistemas de processamento de linguagens naturais tradicionais tem dificuldade em resolver ambiguidades sintáticas deste tipo. No entanto o Cyc sabe que aviões podem voar, mas montanhas não, assim ele é capaz de rejeitar interpretações que não façam sentido.

Para poder alcançar estes resultados, o processador de linguagem natural do Cyc (chamado de Cyc-NL), é decomposto em 3 componentes: analisador léxico, analisador sintático, e interpretador semântico.

Atualmente o Cyc está sendo melhorado para que futuramente seja utilizado para:

- Traduções de máquinas;
- Reconhecimento de fala;
- Interfaces de usuário.

4.12 Powerset

Powerset desenvolve um motor de busca com linguagem natural para a Internet, e é uma companhia recentemente comprada pela Microsoft para concorrer com a Google.

O objetivo é encontrar respostas utilizando processamento de linguagem natural para questões feitas por usuários. Por exemplo, quando um usuário faz uma busca do tipo “qual é o estado americano que possui o maior imposto?”, os motores de busca convencionais ignoram a questão e efetuam a busca baseada nas palavras-chave “estado”, “americano” e “imposto”. Por outro lado, Powerset tenta utilizar processamento de linguagem natural para entender a natureza da questão e retornar as páginas que possuam a resposta [68].

Assim, Powerset pretende inovar utilizando uma busca direcionada à linguagem natural, segundo um processamento de dados com base na estrutura semântica da língua, determinada pela valoração das palavras contextualizadas dentro do enunciado. Segue o princípio de que as palavras encontram-se interligadas e não possuem significados apenas em si mesmas, mas na sua relação entre as outras que compõem o enunciado.

Desta forma, para melhorar a relevância dos resultados de busca, a tendência dos atuais motores de busca é incorporar semântica de forma gradual a estas buscas. Um exemplo bastante conhecido são os operadores OR e AND do famoso Google. No entanto, embora as consultas “Livro de Criança” e “Livro para Criança” sejam distintas, os motores de busca atuais interpretam ambas, a princípio, como “Livro Criança”.

Portanto, utilizando o processamento de linguagem natural, é possível que o sistema consiga interpretar as consultas de forma mais próxima ao seu real significado, tornando os resultados mais relevantes e direcionados ao real interesse dos usuários.

4.13 Considerações sobre o Capítulo

Ao longo deste capítulo, foram descritos os trabalhos relacionados com o arcabouço Airetama. Os quatro primeiros trabalhos apresentados (Coopractice, SemantiCore, fGrin e Ontoshare) estão diretamente relacionados com o arcabouço Airetama, pois utilizam ontologias para o armazenamento de informações que formam a base de conhecimento, e são baseados em sistemas multiagentes, onde os agentes são responsáveis pelas funcionalidades.

No entanto, o arcabouço Airetama se destaca em relação a estes trabalhos relacionados, pois é o único que utiliza a linguagem SPARQL (e SPARQL/Update) em conjunto com as triplas RDF para o gerenciamento dos dados. Além disso, cada trabalho utiliza uma ontologia própria, em contraste com o Airetama, que utiliza as ontologias FOAF e SIOC, recomendadas e divulgadas pelo W3C.

CAPÍTULO 5

Projeto do Arcabouço

Um arcabouço é um conjunto de classes inter-relacionadas, que objetiva formar uma estrutura básica para auxiliar na construção de aplicações inseridas em um domínio particular. O uso desta estrutura visa minimizar o esforço para o desenvolvimento de aplicações, pois permite que o desenvolvedor se abstraia de preocupações com a definição da arquitetura do sistema [78].

A Figura 5.1 ilustra a diferença entre uma aplicação que reutiliza classes (Aplicação 1), e uma aplicação construída através de um arcabouço (Aplicação 2). No primeiro caso, o desenvolvedor deve interligar e reutilizar componentes de *software* isolados. No segundo caso, é utilizada uma estrutura completa de classes inter-relacionadas.

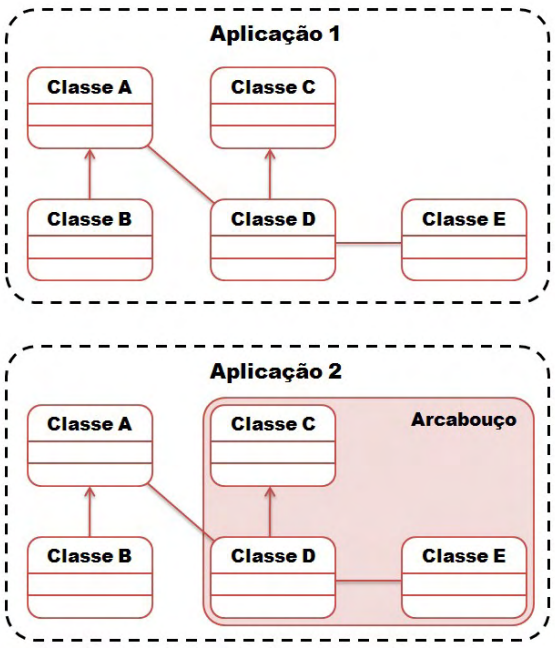


Figura 5.1: *Conjunto de Classes Versus Arcabouço.*

Assim, um arcabouço pode ser visto como uma estrutura genérica, flexível e adaptável, que fornece uma estrutura de classes genérica para o desenvolvimento de aplicações em um domínio [74].

Projeto de Software (*Software Design*) é a parte da Engenharia de Software que se encarrega de transformar os resultados da Análise de Requisitos em um documento ou conjunto de documentos capazes de serem interpretados diretamente pelo programador.

O projeto e análise do arcabouço consistiu na identificação dos atores e seus respectivos papéis, no levantamento de requisitos funcionais e não funcionais, e finalmente, na definição da sua arquitetura.

Os principais critérios para a elaboração do projeto do Airetama foram o baixo acoplamento e a alta coesão, que são conceitos amplamente difundidos e consagrados na Engenharia de Software.

É importante buscar baixo acoplamento para: facilitar o entendimento do código; facilitar a manutenção do código; evitar efeitos colaterais de futuras mudanças; e facilitar o teste de uma entidade (classe, método, módulo) isolada.

Quanto menor a diversidade de assuntos abordados por uma entidade, maior a sua coesão. Uma entidade bem focada em um determinado aspecto do sistema, é uma entidade bem coesa. Sistemas onde os aspectos estão difusos são pouco flexíveis e dificultam extensões, modificações e reutilizações.

5.1 Atores

Um ator na Engenharia de Software representa um papel executado por um usuário ou por outro sistema que interaja com o sistema que está sendo modelado. O ator não vai representar a pessoa e sim o papel que essa pessoa encena. Dessa forma, pode ser que um usuário possa interagir com o sistema com dois ou mais papéis. Além disso, cada classe de usuário define o nivel de acesso às funcionalidades. A seguir, são descritas as classes de usuários que interagem com o Airetama.

- **Visitante**: pode acessar apenas as áreas abertas que não precisam de autenticação no sistema;
- **Usuário**: um visitante, após efetuar autenticação no sistema, torna-se um usuário. Um usuário, ao cadastrar-se em uma comunidade e ser aceito, torna-se membro dessa comunidade;
- **Membro**: é um usuário que participa (está cadastrado) em uma comunidade;
- **Administrador**: um usuário ao criar uma comunidade torna-se administrador da mesma. Assim, administrador é um usuário que possui permissões exclusivas para a manutenção de uma comunidade;
- **Desenvolvedor**: é um usuário que deseja desenvolver uma ferramenta e torná-la disponível para uma comunidade.

5.2 Requisitos Funcionais

O modelo de Casos de Uso foi proposto por Ivar Jacobson[1] como um instrumento para descrição dos requisitos funcionais de um sistema computacional. A construção do Modelo de Casos de Uso corresponde a uma das fases iniciais de um projeto de *software* pois envolve a determinação dos usos que o sistema terá.

A Figura 5.2 apresenta o diagrama de casos de uso do projeto Airetama.

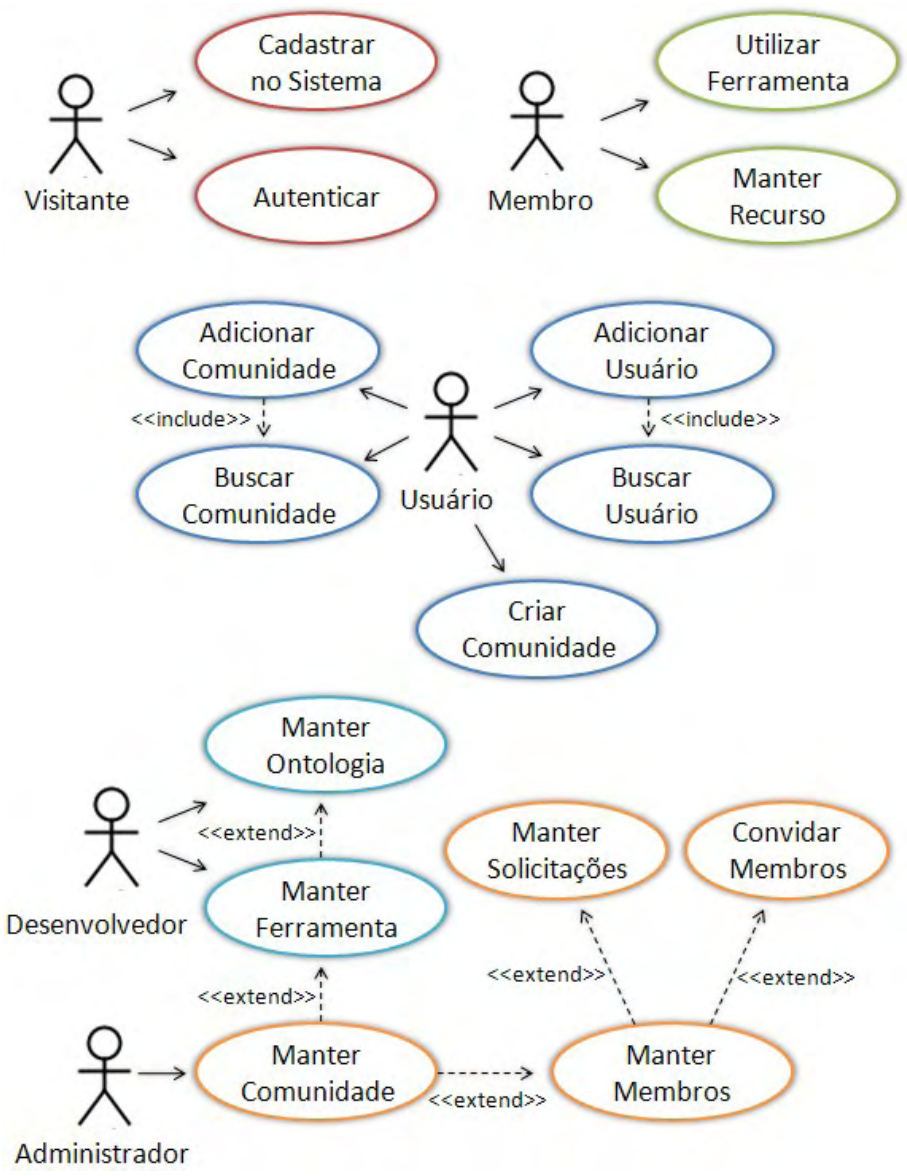


Figura 5.2: *Diagrama de Casos de Uso.*

A seguir serão descritas, de forma breve, cada funcionalidade do diagrama de Casos de Uso.

[1]Também inventou o diagrama de sequência, desenvolveu os diagramas de colaboração, de estado, e transições. Foi um dos principais colaboradores para a criação da linguagem UML (*Unified Modeling Language*).

- **Cadastrar no Sistema**: um visitante, inicialmente, terá como única opção efetuar o seu cadastro no sistema. Logo após o cadastro, o visitante poderá acessar a funcionalidade Autenticar;
- **Autenticar**: um visitante fornece seu *login* e senha que serão validados para verificar se o visitante se tornará um usuário, tendo assim acesso ao sistema;
- **Utilizar Ferramenta**: ao pertencer em uma comunidade o membro poderá utilizar as ferramentas disponibilizadas pela mesma;
- **Manter Recurso**: deve ser permitido o cadastro, exclusão e alteração de recursos em uma comunidade cuja natureza dependerá da ferramenta utilizada;
- **Adicionar Comunidade**: é o envio de uma solicitação de um usuário para poder tornar-se membro de uma comunidade. Antes de enviar uma solicitação à comunidade, o caso de uso Adicionar Comunidade deve chamar o caso de uso Buscar Comunidade, para poder identificá-la. Se a solicitação do usuário for aceita, ele torna-se membro da comunidade e um relacionamento entre o membro e a comunidade é criado;
- **Buscar Comunidade**: é uma listagem com as comunidades cadastradas no sistema;
- **Adicionar Usuário**: um usuário poderá adicionar um relacionamento com outros usuários;
- **Buscar Usuário**: é uma listagem com os usuários cadastrados do sistema;
- **Criar Comunidade**: um usuário deve ser capaz de cadastrar uma nova comunidade no sistema;
- **Manter Ontologia**: um desenvolvedor deve ser capaz de efetuar a manutenção (criação, alteração e exclusão) de ontologias para serem utilizadas por comunidades;
- **Manter Ferramenta**: o desenvolvedor deve ser capaz de gerenciar (criar, alterar e excluir) ferramentas, para serem utilizadas em comunidades. Um desenvolvedor interessado em incorporar uma ferramenta ao sistema deve ser capaz de acoplá-la utilizando a interface definida na arquitetura;
- **Manter Comunidade**: um usuário ao cadastrar uma comunidade, torna-se administrador dela. Um administrador poderá efetuar a exclusão, alteração e manutenção da comunidade. Também deve ser permitida a manutenção das relações entre a comunidade, suas ferramentas e seus usuários;
- **Manter Membros**: o administrador da comunidade deve poder remover membros da comunidade, além de modificar suas permissões;
- **Manter Solicitações**: o administrador da comunidade deve ser capaz de aceitar ou rejeitar solicitações de cadastros de membros na comunidade;
- **Convidar Membros** o administrador da comunidade também deve ser capaz de convidar usuários para poderem ser membros da comunidade.

5.3 Requisitos Não Funcionais

A elaboração dos requisitos não funcionais[2] foi feita de acordo com os atributos de qualidade especificados no Modelo de Qualidade da Norma ISO 9126. Esta norma define 6 atributos de qualidade: Funcionalidade, Confiabilidade, Usabilidade, Eficiência, Manutebilidade e Portabilidade.

As adaptações feitas foram as seguintes: como Funcionalidade foi definido o requisito Segurança; como Confiabilidade foi definido o requisito Tolerância a Falhas; o atributo Usabilidade não foi utilizado; como Eficiência foram definidos o requisitos Balanceamento de Carga, Redução do Fluxo de Dados e Paralelismo; como Manutebilidade e Portabilidade foi definido o requisito Flexibilidade. Outros requisitos que não estavam definidos nesta norma foram Simplicidade, Modularidade e Semântica.

- **Segurança**: a arquitetura deve permitir a autenticação de usuários;
- **Disponibilidade**: tolerância a falhas, com o objetivo de manter os serviços do sistema o maior tempo possível. Novos agentes devem assumir responsabilidades de agentes que venham a falhar, através da redundância de agentes ou clonagem dos mesmos[3];
- **Flexibilidade**: característica de o sistema manipular uma porção crescente de trabalho de forma uniforme, ou estar preparado para crescer. Deve ser permitido substituir e implementar novos agentes de diferentes habilidades em um ambiente heterogêneo e distribuído;
- **Balanceamento de Carga**: capacidade de distribuir tarefas entre os agentes;
- **Redução no Fluxo de Dados**: somente soluções parciais em alto nível devem ser transmitidas para outros agentes, ao invés de dados brutos para um lugar central;
- **Paralelismo**: capacidade de resolução mais rápida de problemas;
- **Simplicidade**: a capacidade de agregar novos agentes deve ser o mais simples possível;
- **Modularidade**: é a modularização das ferramentas com a utilização de agentes de *software*;
- **Semântica**: a arquitetura deve permitir o acesso, armazenamento, processamento e comunicação dos dados de forma semântica entre os agentes, além da utilização de ontologias e metadados.

[2] Também conhecidos como requisitos de qualidade de *software*.
[3] Chamado também de serviço de replicação.

5.4 Arquitetura

Esta seção é dedicada à apresentação da arquitetura do Airetama, baseada nos requisitos abordados no início deste capítulo. Inicialmente são descritos os componentes da arquitetura, sendo detalhados o Portal, o Sistema Multiagentes e o Repositório Semântico, como ilustrado na Figura 5.3.

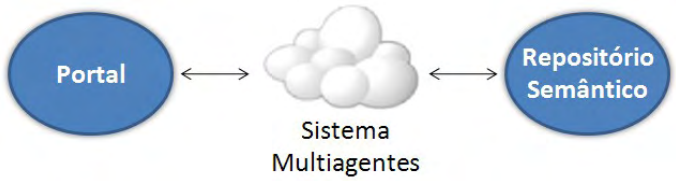


Figura 5.3: *Componentes da Arquitetura.*

Esta arquitetura também pode ser comparada com um padrão MVC (*Model-View-Controller*), onde o Portal representa a camada de visão, o Sistema Multiagente representa a camada controladora onde são processados os dados, e o Repositório Semântico representa a camada de persistência.

5.4.1 Portal

Toda a comunicação dos usuários com o arcabouço é feita através do Portal. Como pode ser visto na Figura 5.4, esta comunicação é feita por meio de um navegador Web em um computador cliente e um Servidor Web.

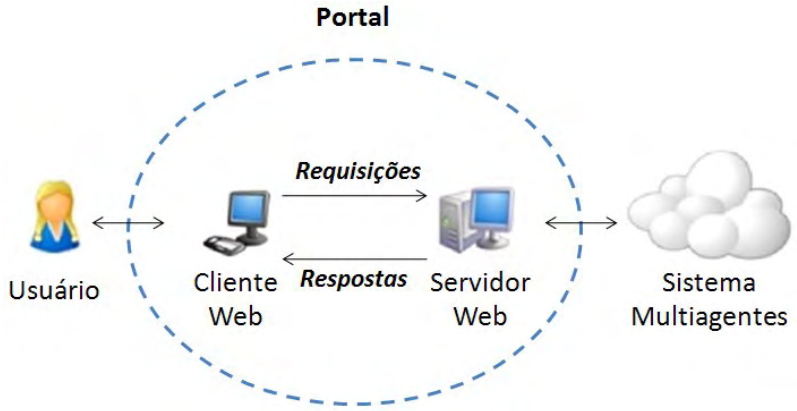


Figura 5.4: *Componentes do Portal.*

O Portal serve como ponto de entrada para as requisições de serviços feitas pelos membros das comunidades. Estas requisições são encaminhadas para as páginas dinâmicas do servidor Web, que as redireciona para o Sistema Multiagente, onde são processadas.

A implementação do Portal foi feita utilizando a linguagem Java, juntamente com a tecnologia JSP (*Java ServerPages*) para a criação das páginas dinâmicas.

5.4.2 Sistema Multiagentes

Como pode ser visto na Figura 5.5, o Sistema Multiagentes é composto por 3 sociedades de agentes: Agentes de Membros, a Agentes Controladores, e Agentes de Ferramentas.

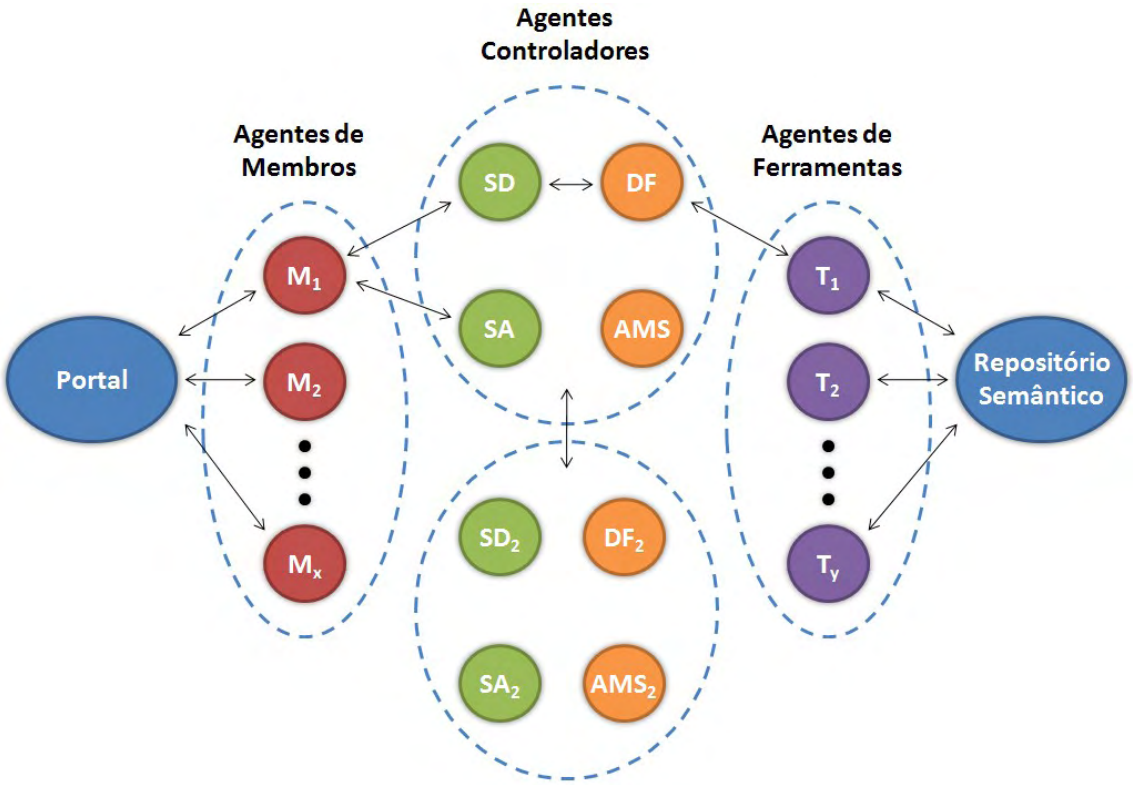


Figura 5.5: *Sociedades de Agentes.*

Além das 3 sociedades de agentes (de membros, controladores e ferramentas), há uma quarta sociedade de agentes que é uma réplica da sociedade de agentes controladores. Esta réplica deve estar em outra máquina e tem o objetivo de assumir a responsabilidade da sociedade de agentes controladores, caso esta por algum motivo fique indisponível.

Segundo a Seção 2.4.2 (Arquiteturas de Sociedades de Agentes) deste trabalho, esta arquitetura pode ser classificada como direta (*Message Passing*), heterogênea, aberta e baseada em leis.

Ainda, segundo a Seção 2.4.3 (Arquiteturas Distribuídas), esta arquitetura também pode ser comparada com uma Arquitetura Baseada em Objetos, onde cada agente é uma forma específica de objeto.

A sociedade de agentes de membros irá alocar um agente para cada membro que estiver ativo na comunidade. Assim, cada agente desta sociedade poderá requisitar serviços da sociedade de agentes de ferramentas.

A sociedade de agentes controladores possui os agentes responsáveis pelo gerenciamento e coordenação de todos os agentes da arquitetura. Estes agentes são:

- **Scheduler (SD)**[4]: responsável por encaminhar e distribuir as requisições (solicitações) de agentes de membros para os agentes de ferramentas;
- **Security Agent (SA)**[5]: responsável pelos requisitos de segurança dos agentes;
- **Agent Management System (AMS)**[6]: é o agente que supervisiona toda a plataforma, controla o ciclo de vida dos agentes, além de manter os estados, identificadores e nomes de agentes, chamado de serviço de páginas brancas (*white-pages*).
- **Directory Facilitator (DF)**[7]: é responsável pelo serviço de páginas amarelas (*yellow-pages*), usado por qualquer agente que queira registrar serviços ou procurar outros serviços disponíveis.

A sociedade de agentes de ferramentas é responsável por agrupar as aplicações que disponibilizam serviços para os agentes membros. Diversas ferramentas poderão ser implementadas, tais como: Fórum Semântico, Wiki Semântica, Bate-papo Semântico, Biblioteca Digital Semântica, Indexadores, Motores de Busca, Motores de Inferência (*Reasoners*) etc.

Um desenvolvedor pode criar uma ferramenta para ser adicionada ao arcabouço. Para efetuar o acoplamento desta ferramenta no arcabouço, o desenvolvedor deve criar uma classe que represente o agente da ferramenta e que herde as características da classe *ToolAgent*. Esta classe *ToolAgent* é uma classe abstrata Java que implementa os atributos e métodos de um agente na sociedade Agentes de Ferramentas, e representa o "contrato" a ser utilizado entre o arcabouço e o desenvolvedor de ferramentas.

Assim, esta nova classe poderá herdar o métodos de comunicação *receive()* e *send()* que permitem a ferramenta interagir com o arcabouço. As assinaturas destes métodos são:

- **public String receive(String sender, int performative, String content)**: *sender* é o emissor da mensagem, *performative* é uma constante indicando o tipo da mensagem, e *content* é o conteúdo XML em forma de String da mensagem.
- **public void send(String sender, int performative, String content)**: é o segundo método de comunicação e possui os mesmos parâmetros do método *receive()*.

Para exemplificar o uso da classe abstrata, é apresentada no Código 5.1 a classe *PrimeTool.java*.

[4]Escalonador.
[5]Agente de Segurança.
[6]Agente de Gerenciamento do Sistema.
[7]Diretório Facilitador.

Código 5.1 – Classe PrimeTool.java

```
1  public class PrimeTool extends ToolAgent {
2
3    @Override
4    public String receive(String sender, int type, String content) {
5      boolean prime = isPrime(Integer.parseInt(content));
6
7      String answer = "";
8
9      if(prime) {
10       answer = content+" is prime!";
11     }
12     else {
13       answer = content+" is not prime!";
14     }
15
16     return answer;
17   }
18
19   private boolean isPrime(int number) {
20     for(int i = 2; i < number/2; i++) {
21       int resultado = number % i;
22
23       if(resultado == 0)
24         return false;
25     }
26     return true;
27   }
28 }
```

A linha 1 demostra que a classe *PrimeTool* utiliza a classe abstrata *ToolAgent*. Na linha 3, a anotação *Override* serve para indicar que o método *receive()* foi sobreescrito. A idéia geral no uso da classe abstrata *ToolAgent*, é que o desenvolvedor da ferramenta apenas precisa se preocupar com quais são os dados de entrada e de saída do seu agente.

Finalmente, para a implementação do Sistema Multiagentes, foi utilizado o *framework* JADE (*Java Agent DEvelopment Framework*).

5.4.3 Repositório Semântico

As ferramentas que são acopladas ao arcabouço devem ser o mais independentes possível, no entanto, elas devem compartilhar a mesma base de dados. O Repositório Semântico é o local onde estes dados são guardados.

A linguagem HTML não fornece semântica para a informação. O padrão XML possui maior flexibilidade, mas somente pode ser utilizado se houver um pleno consenso no significado de seus vocabulários. Assim, foram escolhidas a linguagem RDF em conjunto com RDFS e OWL para o armazenamento de informações, pois permitem a utilização destas informações de forma semântica e consensual.

A Figura 5.6 apresenta uma breve comparação entre as principais linguagens que são utilizadas na Web.

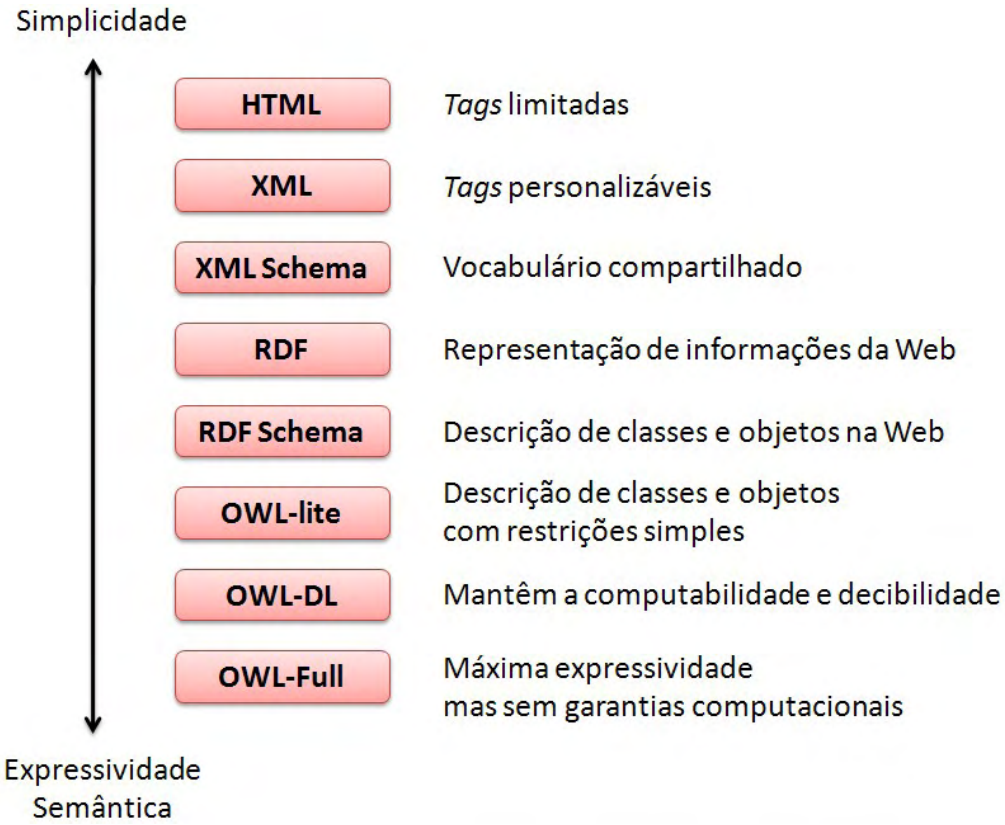


Figura 5.6: *Comparação entre linguagens utilizadas na Web.*

Como pode ser observado na Figura 5.7, o Repositório Semântico é composto pelo Repositório FOAF e pelo Repositório SIOC.

A representação ontológica dos membros e comunidades foi feita utilizando, respectivamente, o vocabulário FOAF e o vocabulário SIOC, que são complementares. Além disso, o Repositório Semântico deve ser capaz de anexar novas ontologias conforme for necessário. Para a implementação do Repositório Semântico, foi utilizado o *framework* Jena.

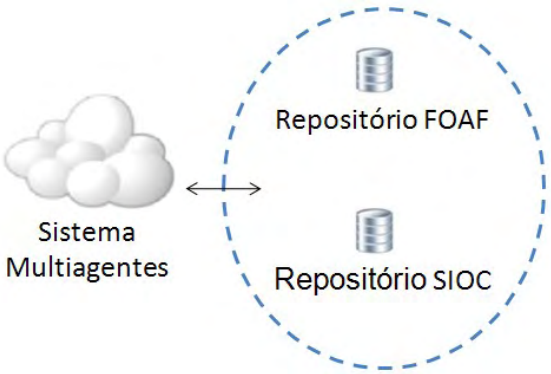


Figura 5.7: *Repositório Semântico.*

A linguagem utilizada na implementação do Airetama foi Java, pelo fato desta ser orientada a objetos, ter portabilidade (independente de plataforma), sintaxe simples (baseada em C e C++), distribuída, aberta, *multithreaded*, segura, robusta, além de ser bastante utilizada, difundida e documentada.

CAPÍTULO 6

Desenvolvimento do Arcabouço

Neste capítulo, são descritos os principais passos executados na implementação do arcabouço Airetama. O desenvolvimento apresentado neste capítulo é uma aplicação dos conceitos que foram descritos no Capítulo 2, e é baseado na análise e projeto do Capítulo 5.

Ao início do capítulo, são enumeradas as tecnologias utilizadas para o desenvolvimento do arcabouço. Em seguida, é feita uma breve apresentação das plataformas Jena e JADE, que foram as principais ferramentas utilizadas durante a implementação. O Jena é utilizado para a implementação dos grafos RDF com ontologias, e o JADE para a implementação dos agentes.

Vale enfatizar que o uso de ontolgias e agentes são os dois pilares da Web Semântica. Informações mais detalhadas sobre Jena e JADE podem ser encontradas nos relatórios técnicos disponibilizados no *site* do Airetama que se encontra no SourceForge [2].

Na sequência, é descrita a implementação do Portal e do Repositório Semântico. Em seguida, é apresentado o processo de autenticação, cadastro de usuários, além da solicitação e convite de usuários. Finalmente, é descrita a busca de usuários, e a implementação dos agentes.

6.1 Tecnologias Empregadas

Para o desenvolvimento do Airetama foram utilizados os seguintes *softwares*: JADE versão 3.6; Jena 2.6.3; ARQ 2.8.4; JDK 1.6.0_19; Eclipse EE 1.2.2; Apache Tomcat 6.0; Commons FileUpload 1.2.1; Adobe Fireworks CS4; e Sistema Operacional Windows XP Professional SP3 (também foram efetuados testes no Linux).

6.2 Jena

Jena é uma ferramenta Java *open-source* para construção de aplicações baseadas em Web Semântica, originada no *HP Labs Semantic Web Programme*. O *framework* Jena inclui [52]:

- Uma API RDF;
- Leitura e escrita RDF em RDF/XML, N3, N-Triplas e Turtle;
- Uma API OWL;
- Armazenamento em memória e persistência;
- Motor de consultas SPARQL;
- Mecanismos de raciocínio pré-construídos, e permite adicionar mecanismos de raciocínio de terceiros.

Para o desenvolvimento do Airetama foram utilizados principalmente: a API RDF; o armazenamento persistente através do TDB, que será descrito posteriormente; e o motor de consultas SPARQL.

Como já foi exposto no Capítulo 2, o RDF (*Resource Description Framework*) é uma linguagem utilizada para descrever recursos na Web. Todas as informações da Web podem ser representadas utilizando esta linguagem.

A definição precisa do que é um recurso ainda está em debate: pode ser uma pessoa, um livro, uma página Web etc. No entanto, em termos práticos, um recurso é tudo aquilo que pode ser identificado. Como pode ser observado na Figura 6.1, no Jena, o mesmo é representado graficamente por uma elipse.

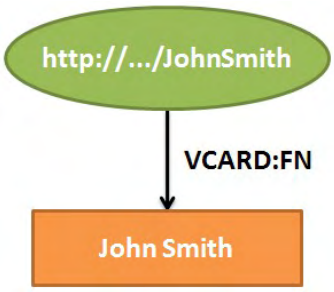


Figura 6.1: *Um recurso, propriedade e literal RDF.*

Uma declaração (*Statement*) expressa um fato sobre um recurso. Como já foi indicado anteriormente no Capítulo 2, uma declaração pode ser chamada de tripla pois possui três partes: sujeito, predicado e objeto.

A Figura 6.1 ilustra uma declaração, onde o sujeito é o recurso "http://.../JohnSmith", o predicado é a propriedade "VCARD:FN", e o objeto é o literal "John Smith".

Vale ressaltar que o objeto de uma tripla pode ser o sujeito de outra tripla, desta forma, inúmeras triplas podem ser interligadas formando um grafo.

Tim Berners-Lee redefine em 2007 a Web Semântica com o conceito que ele chamou de GGG (*Giant Global Graph*, ou Grafo Global Gigante). A discussão que ele iniciou sobre o tema pode ser vista no seu *blog* pessoal em [12].

Recursos possuem propriedades, que são representadas por arcos, rotulados com o nome da propriedade. Cada propriedade possui um valor que, normalmente, é um literal. Um literal é uma *String* (cadeia ou sequência de caracteres) representada por um retângulo.

No Jena, as interfaces que representam recursos, propriedades e literais são chamadas *Resource*, *Property* e *Literal* respectivamente.

Através da utilização de um modelo RDF os dados podem ser representados de uma forma descentralizada e distribuída. Um modelo é representado como um grafo contendo declarações, e é implementado no Jena através da interface *Model*.

A implementação de um modelo pode ser feita baseada em um modelo de memória ou banco de dados relacional (persistência). O Código 6.1 mostra implementação baseada em memória.

Código 6.1 – Persistência de um modelo em memória

```
1  import com.hp.hpl.jena.rdf.model.*;
2  import com.hp.hpl.jena.vocabulary.*;
3
4  public class Tutorial01 extends Object {
5    static String personURI = "http://somewhere/JohnSmith";
6    static String fullName = "John Smith";
7
8    public static void main (String args[]) {
9      Model model = ModelFactory.createDefaultModel();
10     Resource johnSmith = model.createResource(personURI);
11     johnSmith.addProperty(VCARD.FN, fullName);
12   }
13 }
```

Como pode ser visto no Código 6.1, inicialmente são criadas as definições representadas como constantes (linhas 5 e 6). Em seguida, é feita a criação de um modelo vazio (linha 9), através do método *createDefaultModel()* da classe *ModelFactory*, que é responsável por implementar modelos. O recurso “John Smith” é criado utilizando-se a classe constante “VCARD” que representa todas as definições do esquema “VCARD”. Finalmente, é criada a propriedade através do método *addProperty()* (linha 11).

6.3 JADE

A programação orientada a agentes é um paradigma de *software* relativamente novo e traz conceitos da Inteligência Artificial, em conjunto com Sistemas Distribuídos. JADE é uma importante ferramenta, pois possibilita a criação de sistemas que utilizam este paradigma, promovendo o desenvolvimento de sistemas baseados em agentes e sistemas multiagentes.

FIPA (*The Foundation for Intelligent Physical Agents*) é uma organização de padronização IEEE cujo objetivo é promover tecnologias baseadas em agentes e interoperabilidade destes padrões com outras tecnologias. Atualmente as especificações FIPA representam a maior atividade de padronização conduzida na área de tecnologias de agentes.

A organização FIPA é importante para o entendimento do JADE já que atualmente JADE é considerado o principal *framework open-source* que implementa as especificações FIPA. A missão oficial da FIPA é "A promoção de tecnologia e especificações de interoperabilidade que facilitem a comunicação entre sistemas de agentes inteligentes no contexto comercial e industrial moderno" [15].

JADE (*Java Agent DEvelopment framework*) é uma ferramenta que possibilita desenvolver aplicações baseadas em agentes conforme as especificações da FIPA, e seu principal objetivo é simplificar e facilitar o desenvolvimento de sistemas multiagentes garantindo um padrão e interoperabilidade.

JADE pode ser distribuído através de diversas máquinas com sistemas operacionais diferentes e toda sua configuração, gerenciamento e depuração pode ser feita através de ferramentas gráficas remotas (em tempo de execução).

A arquitetura de comunicação é flexível e eficiente, já que o modelo de comunicação FIPA foi implementado completamente. O mecanismo de transporte se adapta a qualquer situação escolhendo o protocolo mais adequado.

A maioria dos protocolos de interação FIPA foram implementados pelo JADE, e os protocolos de gerenciamento de ontologias já foram completamente implementados.

O ponto mais fraco do JADE é o fato dele não possuir mecanismos de "inteligência", planejamento ou raciocínio. No entanto, JADE consegue interagir de forma relativamente fácil com implementações em Java, Prolog, sistemas especialistas ou motores de inferência tais como JESS ou JADEX.

Diversos *plug-ins* para o JADE vem sendo desenvolvidos. O WADE (*Workflows and Agents Development Environment*), é uma das principais extensões do JADE, e permite a execução de tarefas de agentes de acordo com um *workflow*.

Outro *add-on* bastante conhecido é o WSDC (*Web Service Dynamic Client*), que integra a ferramenta JADE com a tecnologia de Web Services.

O desenvolvimento do JADE se iniciou em 1998 pela Telecom Italia (antiga CSELT), motivados pela necessidade de validar as primeiras especificações FIPA. JADE tornou-se *open-source* em 2000 e tem sido distribuído pela Telecom Italia sob a licença LGPL. Diferentemente da GPL, a LGPL não impõe nenhuma restrição no *software*, inclusive permite o uso comercial [47] e [59].

No JADE, todas as tarefas, trabalhos (*jobs*) ou ações de um agente são chamados de comportamentos (*behaviours*). Estas tarefas são implementadas por meio da classe *Behaviour* ou alguma de suas subclasses. Como estas classes são abstratas, o implementador não as pode instanciar diretamente. Em vez disso, o implementador deve criar uma nova classe (que irá representar uma tarefa específica) que herde as capacidades de alguma destas classes de tarefas. Em outras palavras, a classe precisa ser herdada primeiro para depois poder ser instanciada.

Uma tarefa no JADE é basicamente um *Event Handler* (fragmento de programa que reage a eventos ocorridos). Formalmente, um evento corresponde a uma mudança de estado, em termos práticos corresponde a um recebimento de mensagem ou interrupção de tempo [81].

Se uma tarefa irá ser executada diversas vezes, sem parar, então esta deve ser implementada utilizando a classe *CyclicBehaviour*. Esta classe implementa uma tarefa que funciona como um *loop* infinito. O Código 6.2 mostra um agente que exibe a mensagem "‘Olah"’ ininterruptamente e utiliza esta classe.

Código 6.2 – Arquivo AgenteCyclicBehaviour.java

```
1  import jade.core.Agent;
2  import jade.core.behaviours.*;
3  public class AgenteCyclicBehaviour extends Agent {
4    protected void setup() {
5      addBehaviour(new TarefaCyclicBehaviour());
6    }
7  }
8  class TarefaCyclicBehaviour extends CyclicBehaviour {
9    public void action() {
10     System.out.println("olah!");
11   }
12 }
```

O método *setup()* (na linha 4) sempre é invocado quando um agente é iniciado. Vale ressaltar que cada *Behaviour* possui um método *action()* com comportamento diferente. No caso do *CyclicBehaviour* o método *action()* (na linha 9), é invocado continuamente.

6.4 Portal

O Portal serve como ponto de entrada para as requisições de serviços feitas pelos membros das comunidades. Estas requisições são encaminhadas para as páginas dinâmicas do servidor Web, que as redireciona para o Sistema Multiagente, onde são processadas. Além disso, o Portal também serve para apresentar a saída das requisições.

A Figura 6.2 mostra um exemplo de um documento FOAF utilizando um formato externo, e a Figura 6.3 de um equivalente utilizando um formato *inline*.

```
<Foaf:name df:datatype=
"http://www.w3.org/2001/XMLSchema#string">
John Doe</foaf:name>
<Foaf:homepage df:resource="http://johndoe.org"/>
<Foaf:knows rdf:resource=
"http://wiki.ontoworld.org/index.php/Jane_Doe"/>
<rdf:type rdf:resource=
"http://xmlns.com/foaf/0.1/Person"/>
```

Figura 6.2: *Vocabulário FOAF utilizando um formato externo*

```
[[name::John Doe]] has the homepage
[[foaf:homepage::http://johndoe.org]].
His best friend is [[foaf:knows::Jane Doe]].
[[Category:Person]]
```

Figura 6.3: *Vocabulário FOAF inline*

Para o projeto Airetama, foi feita a escolha de utilizar o formato externo ao invés de *inline*, devido à complexidade adicional da segunda opção, além da aparente falta de vantagens em misturar metadados com XHTML.

Esta segunda opção é mais dificil de ser implementada, pois exige a utilização de algum mecanismo, como por exemplo GRDDL [49], para que as triplas RDF sejam extraídas da página XHTML. Também exige que seja utilizada uma forma de combinar as triplas RDF com uma página XHTML, como por exemplo o RDFa [1], o que atualmente deve ser feito manualmente.

Assim, para que os agentes utilizem os dados de forma direta, sem nenhum tipo de conversão, foi escolhido o tipo de vocabulário em um formato externo.

A princípio teve-se a ideia de utilizar páginas XHTML com XSL e XSLT para a visualização dos dados. XSL (*EXtensible Stylesheet Language*) é uma linguagem para formatação de documentos XML, e serve para transformar um documento XML em outro tipo de documento (utilizando o XSLT). XSLT é uma linguagem utilizada para criar documentos XSL que, por sua vez, definem a apresentação dos documentos XML.

Como os dados são armazenados através de triplas RDF, e o RDF é escrito através de XML, então tudo que é possível fazer com um documento XML, também é possível com um RDF.

No entanto, em vez de utilizar páginas XHTML com XSL e XSLT, são utilizadas páginas JSP em conjunto com CSS. Há uma diferença importante entre XSLT e CSS: o documento XSL pode adicionar ou esconder conteúdo à apresentação do documento XML no *browser*. Mas, as desvantagens da primeira abordagem são: dificuldade de implementar regras de negócio, necessidade de aprender uma nova linguagem, pelo uso maior de memória, e pela impossibilidade de alterar valores de variáveis.

Na Figura 6.4 é mostrado o Portal do arcabouço Airetama, implementado utilizando páginas JSP e CSS.

Figura 6.4: *Portal do Airetama.*

6.5 Repositório Semântico

A esquemagem (*Schemagen*) fornecida pelo Jena é um processo utilizado para converter um vocabulário OWL, DAML ou RDFS em uma classe Java (arquivo .class) que contém constantes estáticas para os termos do vocabulário. RDFS, OWL e DAML são bastante convenientes para definir um vocabulário ou ontologia. Entretanto, para o desenvolvimento de aplicações, frequentemente é conveniente acessar os termos de um vocabulário diretamente através de um código Java [39].

No entanto, o processo de esquemagem no Jena ainda é um processo complicado que consiste na execução de *scripts*. Desta forma, foi decidido que a esquemagem seria feita de forma manual. Um dos principais inconvenientes encontrados na esquemagem do Jena foi o fato dela criar constantes para cada termo, em vez de objetos. O ideal seria criar um objeto para cada termo, e um método para cada propriedade.

Assim, foram criadas manualmente duas classes principais: a classe *Person* para a representação do termo *Person* da ontologia FOAF, e a classe *Community* para representação do termo *Community* da ontologia SIOC. Os métodos relacionados às classes também foram criados manualmente. Um exemplo é o método *memberOf()* que equivale à propriedade *member_of* da ontologia SIOC e possui indiretamente como domínio (*domain*) o termo *Person* da ontologia FOAF. Este método é descrito com detalhes na Seção 6.8.

Vale ressaltar que o domínio de uma propriedade indica quais termos podem utilizá-la. No exemplo dado anteriormente, significa que a propriedade *member_of* pode ser utilizada por termos do tipo *Person*. Além do mais, o alcance (*range*) da propriedade *member_of* inclui o tipo *Community*. Então, pode-se observar que a propriedade *member_of* conecta o termo *Person* da ontologia FOAF com o termo *Community* da ontologia SIOC, criando uma "ponte" que interliga as duas.

Existem muitas semelhanças entre objetos e ontologias, mas também há algumas diferenças: as ontologias são mais expressivas que a orientação a objetos. Por exemplo, ontologias permitem a herança entre propriedades e a orientação a objetos não.

Desta forma, pode-se propor uma nova evolução nas linguagens de programação, onde a programação orientada a objetos será substituída por uma programação orientada a ontologias. Esta seria uma mudança interessante, pois tornaria a esquemagem desnecessária, já que o suporte a ontologias seria dado nativamente pela linguagem de programação.

Outro método importante da classe *Person* é o *select()* mostrado no Código 6.3, que faz o mapeamento entre o termo *Person* da ontologia FOAF e a própria classe. A classe *Community* faz o mesmo tipo de conversão.

Código 6.3 – Método *select()*

```
1  public void select() {
2    Model m = TDBFactory.createModel(DBDIRECTORY);
3    String queryString =
4      " PREFIX foaf: <"+FOAFNS+"> "+
5      " SELECT ?member ?firstName ?surname ?sha1 ?birthday "+
6      "   ?gender ?depiction "+
7      " WHERE { "+
8      "   ?member foaf:mbox \""+this.getMbox()+"\". "+
9      "   OPTIONAL { ?member foaf:firstName ?firstName } . "+
10     "   OPTIONAL { ?member foaf:surname ?surname } . "+
11     "   OPTIONAL { ?member foaf:sha1 ?sha1 } . "+
12     "   OPTIONAL { ?member foaf:birthday ?birthday } . "+
13     "   OPTIONAL { ?member foaf:gender ?gender } . "+
14     "   OPTIONAL { ?member foaf:depiction ?depiction } . "+
15     " }";
16   Query query = QueryFactory.create(queryString);
17   QueryExecution qe = QueryExecutionFactory.create(query, m);
18   ResultSet results = qe.execSelect();
19   if(results.hasNext()) {
20     QuerySolution node = (QuerySolution) results.next();
21     this.setAbout(node.get("member")==null ? "" :
22         node.get("member").toString());
23     this.setFirstName(node.get("firstName")==null ? "" :
24         node.get("firstName").toString());
25     this.setSurname(node.get("surname")==null ? "" :
26         node.get("surname").toString());
27     this.setSha1(node.get("sha1")==null ? "" :
28         node.get("sha1").toString());
29     this.setBirthday(node.get("birthday")==null ? "" :
30         node.get("birthday").toString());
31     this.setGender(node.get("gender")==null ? "" :
32         node.get("gender").toString());
33     this.setDepiction(node.get("depiction")==null ? "" :
34         node.get("depiction").toString());
35   }
36   qe.close();
37   m.close();
38 }
```

No Código 6.3, o método *select()* é responsável por efetuar uma consulta SPARQL no modelo TDB, utilizando como parâmetro o *e-mail* do usuário, para retornar todos seus outros atributos (*firstName*, *surname*, *sha1*, *birthday*, *gender* e *depiction*).

Como foi descrito anteriormente, o conjunto de triplas forma um grafo, que é implementado no Jena através da interface *Model*. Pode-se observar no Código 6.3, na linha 2, que o método estático *createModel()* da classe *TDBFactory* recebe como parâmetro uma *String* contante (com o caminho do local onde é armazenado o modelo), e retorna o modelo indicado pela constante ou um modelo novo caso não exista.

A classe *TDBFactory* faz parte do componente TDB que pode ser anexado ao Jena. TDB é um gerenciador de triplas RDF (RDF Store), que serve para armazená-las, consultá-las através da linguagem SPARQL, ou modificá-las através da especificação SPARQL/Update. O subsistema TDB é otimizado para alta performace, pois permite utilizar bilhões de triplas RDF [31].

Ainda no Código 6.3, na linha 3 à 15, foi criada uma consulta SPARQL que seleciona diversas variáveis (indicadas pelos nomes iniciados com o símbolo "?"). O requisito para que esta consulta retorne as variáveis é que o atributo *mbox* tenha sido preenchido anteriormente, pois ele é o parâmetro de busca da consulta (como pode ser observado na linha 8). O método *execSelect()*, da linha 18, é responsável por executar consultas SPARQL e retorna um *ResultSet*, que é acessado para preencher o próprio objeto que invocou o método *select()*, através dos métodos *setters* (como pode ser observado na linha 21 à 33).

O método *execSelect()* consegue efetuar consultas apenas na linguagem SPARQL. Entretanto, para efetuar modificações em triplas RDF, deve-se utilizar a linguagem SPARQL/Update. Esta linguagem também permite: inserir novas triplas RDF em um grafo, remover triplas RDF de um grafo, executar um grupo de operações de atualizações em uma única execução, criar novos grafos RDF e remover grafos RDF [73].

O Código 6.4 mostra o método *delete()* da classe *Person*, que é um exemplo que utiliza a linguagem SPARQL/Update. Este método remove por completo uma pessoa do modelo. O acesso ao modelo é feito da mesma forma que no Código 6.3, utilizando o método *createModel()* (linha 2). Na linha 6 à 8, o comando *DELETE* indica a remoção da tripla inteira, removendo o sujeito (*?s*), o predicado (*?p*) e o objeto (*?o*). Na linha 9 à 13, o comando *WHERE* indica as condições de remoção, onde somente as triplas que tenham como objeto o *e-mail* do objeto (linha 12) serão deletadas. Em vez de utilizar o método *execSelect()* que executa consultas na linguagem SPARQL, para executar uma *query* com a linguagem SPARQL/Update, deve-se utilizar o método *parseExecute()* da classe *UpdateAction* (linha 15).

Código 6.4 – Método *delete()*

```
1  public void delete() {
2    Model m = TDBFactory.createModel(DBDIRECTORY);
3    String queryString =
4      " PREFIX foaf: <"+Constants.FOAFNS+"> \n"+
5      " PREFIX rdf: <"+Constants.RDFNS+"> \n"+
6      " DELETE { "+
7      "   ?s ?p ?o . "+
8      " } "+
9      " WHERE {"+
10     "   ?s ?p ?o . "+
11     "   ?s rdf:type foaf:Person . "+
12     "   ?s ?p \""+this.getMbox()+"\" . "+
13     " }";
14   GraphStore graphStore = GraphStoreFactory.create(m) ;
15   UpdateAction.parseExecute(queryString, graphStore) ;
16   m.close();
17 }
```

Além de poder efetuar consultas em triplas RDF com a linguagem SPARQL, e modificações nestas triplas com a linguagem SPARQL/Update, o Jena também permite que consultas externas sejam feitas nestas triplas RDF. O Joseki é um motor HTTP do Jena que fornece um ponto de acesso (denominado *endpoint*) [20] para estas consultas.

O Airetama não utiliza o Joseki, devido às suas restrições de segurança, onde usuários externos ao Airetama não devem ter acesso às triplas RDF. No entanto, futuramente, o uso de *endpoints* através do Joseki, poderá ser implementado, para integrar o Airetama com outras aplicações externas.

O Código 6.5 mostra uma consulta no Jena que utiliza o *endpoint* da DBPedia [5]. O exemplo faz uma consulta que retorna os *softwares* desenvolvidos por organizações fundadas na Califórnia. As linhas 2 e 3 mostram os prefixos que indicam onde estão os termos e recursos da DBPedia. Na linha 6, o predicado “a” da tripla equivale ao predicado “rdf:type”, indicando que a variável *?company* deve ser uma instância do termo *Organisation*. O acesso ao endpoint da DBPedia é dado através do método *sparqlService()* na linha 13.

Código 6.5 – Efetuando consulta SPARQL através de um *endpoint*

```
1  String queryString =
2    " PREFIX dbp: <http://dbpedia.org/ontology/> "+
3    " PREFIX dbpr: <http://dbpedia.org/resource/> "+
4    " SELECT * "+
5    " WHERE { "+
6    "  ?company a dbp:Organisation . "+
7    "  ?company dbp:foundationPlace dbpr:California . "+
8    "  ?product dbp:developer ?company . "+
9    "  ?product a dbp:Software "+
10   " } ";
11 Query query = QueryFactory.create(queryString);
12 QueryExecution qe = QueryExecutionFactory.
13   sparqlService("http://dbpedia.org/sparql", query);
14 ResultSet results = qe.execSelect();
```

Para analisar as triplas RDF no Repositório Semântico foi criada uma página JSP que exibe todas as triplas armazenadas no mesmo, que pode ser vista no Código 6.6. O método *ResultSetFormatter* na linha 6 é responsável por apresentar o resultado no formato de triplas. A Figura 6.5 apresenta um trecho com 4 triplas do Repositório Semântico.

Código 6.6 – Código utilizado para *debugar* triplas RDF

```
1  <%
2  String queryString = " SELECT * WHERE {?s ?p ?o} ";
3  Model m = TDBFactory.createModel(Constants.DBDIRECTORY);
4  Query query = QueryFactory.create(queryString);
5  QueryExecution qe = QueryExecutionFactory.create(query, m);
6  ResultSet results = qe.execSelect();
7  ResultSetFormatter.out(System.out, results, query);
8  m.close();
9  %>
```

```
-------------------------------------------------------------------------
| s                                | p             | o                  |
=========================================================================
| airetama:akira@yahoo.com.foaf.rdf | foaf:firstName | "Akira"           |
| airetama:akira@yahoo.com.foaf.rdf | foaf:surname   | "Sato"            |
| airetama:akira@yahoo.com.foaf.rdf | foaf:mbox      | "akira@yahoo.com" |
| airetama:akira@yahoo.com.foaf.rdf | rdf:type>      | foaf:Person       |
```

Figura 6.5: *Trecho com triplas RDF do Repositório Semântico*

6.6 Autenticação

A autenticação de usuários é feita diretamente no Portal através de variáves de sessão do JSP e funções Hash.

Os aplicativos Web têm um problema intrínseco que os programas *standalone* nunca tiveram: que é como manter um valor disponível em todo o aplicativo. Em um programa *standalone* basta colocar o valor em uma variável global. Já os programas Web sofrem uma séria deficiência: como o protocolo HTTP não mantém uma conexão entre as chamadas das diferentes páginas, é preciso encontrar algum mecanismo para passar informações de uma página para outra.

Uma forma de manter informações em páginas Web é através do uso de campos *hidden*, no entanto a passagem de parâmetros implica em sérios problemas de segurança. Desta forma, o mecanismo escolhido para manter a autenticação de usuários foram as variáveis de sessão.

As variáveis de sessão são únicas para cada usuário, e acessíveis apenas para ele. Quando o usuário abandona a página ou o tempo limite de sessão expira (normalmente definido como 20 minutos), fecham-se automaticamente as variáveis de sessão do usuário.

Uma função Hash recebe um valor de um determinado tipo e retorna um código para ele. O processo é unidirecional e impossibilita descobrir o conteúdo original a partir do Hash. A função Hash escolhida para o Airetama foi a SHA-1 (*Secure Hash Algorithm*), desenvolvida pelo NIST e NSA (*National Security Agency*). A função SHA-1, considerada sucessora da MD5, é usada numa grande variedade de aplicações e protocolos de segurança, incluindo: TLS, SSL, PGP, SSH, S/MIME e IPSec.

A Figura 6.6 mostra o exemplo de uma função Hash que recebe uma cadeia de caracteres como entrada e retorna seu respectivo Hash.

```
SHA1("The quick brown fox jumps over the lazy dog")
 = 2fd4e1c6 7a2d28fc ed849ee1 bb76e739 1b93eb12
```

Figura 6.6: *Exemplo de função Hash.*

Mesmo uma pequena mudança na mensagem resultará, graças ao efeito avalanche (se apenas um dos bits for alterado, muitos bits do resultado serão afetados), em um Hash completamente diferente.

Vale resaltar que, não são as senhas que são armazenadas no Repositório Semântico, mas sim os seus respectivos valores Hash. Assim, da mesma forma que ocorrem nos bancos, se um usuário esquecer a senha, esta não poderá ser recuperada, mas somente recriada.

O Código 6.7 mostra um trecho onde é feita a autenticação do usuário. Na linha 1 é invocado o método *isValid()* do objeto *member*. Se a autenticação for válida, são atribuídos os valores para as variáveis de sessão, tornando-os acessíveis globalmente.

Código 6.7 – Trecho de código onde é feita a autenticação do usuário.

```
1  if (member.isValid()) {
2    session.setAttribute("USER_EMAIL", email);
3    session.setAttribute("USER_PASSWORD", Security.SHA1(password));
4    ...
5  }
```

Se a autenticação for válida, o usuário será redirecionado para sua página inicial, como pode ser observado na Figura 6.7. Caso contrário, o usuário será redirecionado para a página de *login*, mostrada na Figura 6.4.

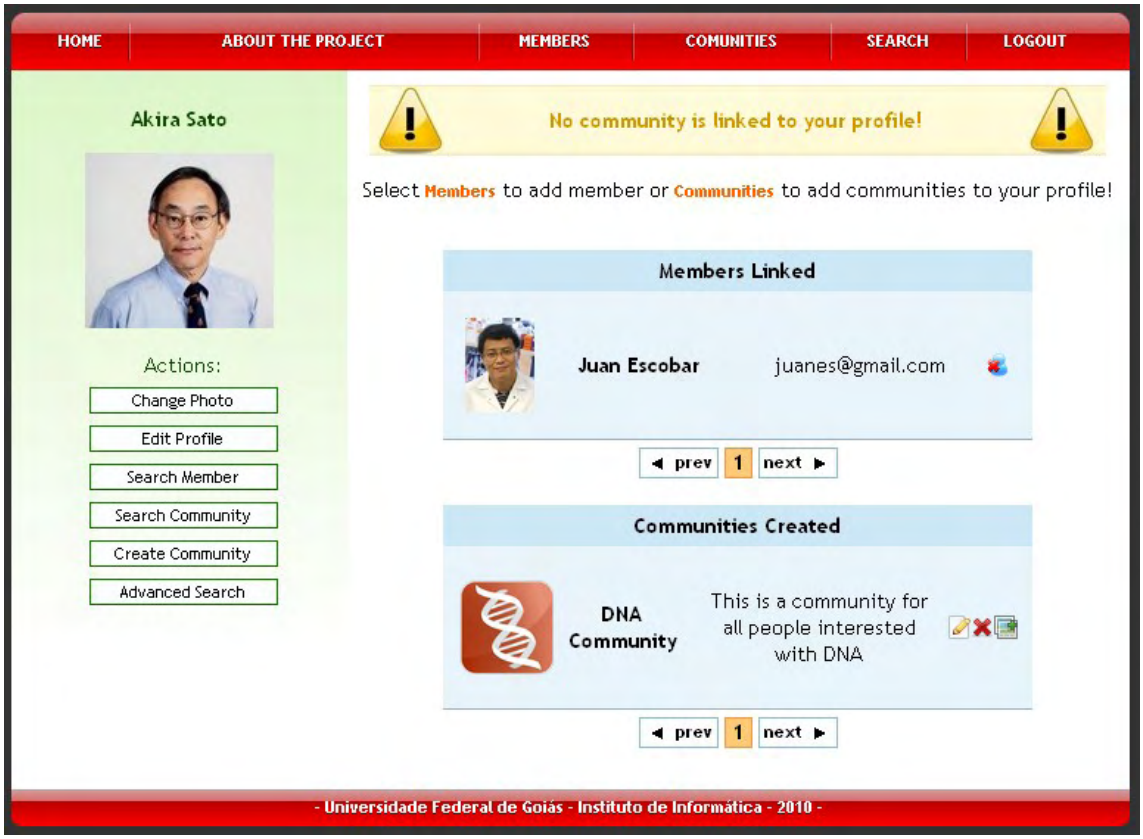


Figura 6.7: *Tela inicial após acesso de membro.*

Após o acesso, o usuário pode efetuar diversas ações, como por exemplo: modificar sua foto, atualizar o seu perfil, procurar um usuário, procurar uma comunidade, criar uma comunidade ou efetuar uma busca personalisada. Além disso, é na sua página inicial que são mostradas mensagens de aviso, suas conexões com outros usuários e conexões com comunidades.

6.7 Cadastro de Usuários

Para o cadastro de um novo usuário, é apresentado um formulário JSP que deve ser preenchido pelo mesmo. Após o usuário preencher o formulário, é criado um objeto *Person*. Este objeto somente pode ser instanciado se forem passados todos os atributos pelo construtor, garantindo assim que o objeto possua todos os atributos preenchidos com os dados do formulário (como por exemplo nome, *e-mail* e data de nascimento). Assim, somente após o objeto estar corretamente preenchido que o formulário JSP invoca o método *insert()* deste objeto, que é apresentado na Figura 6.8.

Código 6.8 – Método *insert()*

```
1  public void insert() {
2    Model m = TDBFactory.createModel(DBDIRECTORY);
3    m.setNsPrefix("foaf", FOAFNS);
4    m.setNsPrefix("rdf", RDFNS);
5    String URI = AIRETAMANS + "members/" + this.getMbox()+".foaf.rdf";
6    Resource r = m.createResource(URI);
7    r.addProperty(m.createProperty(FOAFNS+"firstName"),
8      this.getFirstName());
9    r.addProperty(m.createProperty(FOAFNS+"surname"),
10     this.getSurname());
11   r.addProperty(m.createProperty(FOAFNS+"mbox"), this.getMbox());
12   r.addProperty(m.createProperty(FOAFNS+"sha1"), this.getSha1());
13   r.addProperty(m.createProperty(FOAFNS+"gender"), this.getGender());
14   r.addProperty(m.createProperty(FOAFNS+"birthday"),
15     this.getBirthday());
16   r.addProperty(m.createProperty(FOAFNS+"depiction"),
17     m.createResource(AIRETAMANS+"images/member.jpg"));
18   r.addProperty(m.createProperty(RDFNS+"type"),
19     m.createProperty(FOAFNS+"Person"));
20   m.close();
21 }
```

Como pode ser observado no Código 6.8, o recurso com o URI do usuário é criado (linha 6) através do método *createResource()*. Em seguida são adicionadas as propriedades ao recurso através do método *addProperty()*, onde o primeiro parâmetro indica o predicado que é criado com o método *createProperty()* e o segundo parâmetro indica o objeto que é acessado através dos métodos *getters*.

Para a alteração do perfil do usuário, é utilizado o mesmo formulário do cadastro, só que ao invés de invocar o método *insert()*, é chamado o método *update()*.

6.8 Solicitação e Convite de Membros

Existem três situações possíveis de conexão entre um usuário e uma comunidade. A primeira situação ocorre quando um membro deseja pertencer a uma comunidade e lhe envia uma solicitação, como pode ser observado na Figura 6.8.

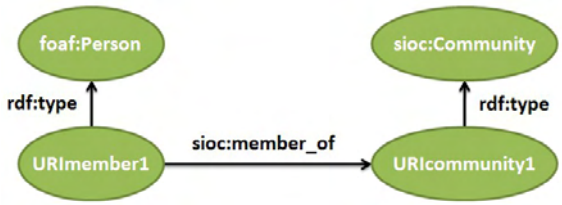


Figura 6.8: *Solicitação de membro.*

A segunda situação é inversa, pois ocorre quando uma comunidade deseja que um membro faça parte dela, então um administrador desta comunidade envia um convite ao membro, como pode ser observado na Figura 6.9.

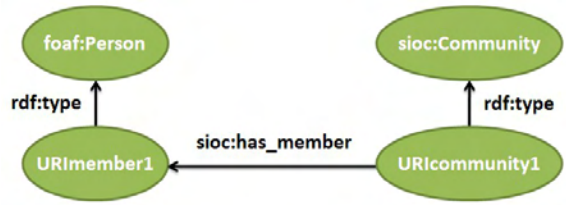


Figura 6.9: *Convite de membro.*

A terceira situação ocorre quando o convite ou a solicitação foi confirmada, e a conexão é bi-direcional, como pode ser observado na Figura 6.10.

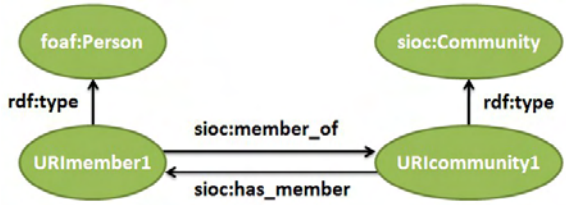


Figura 6.10: *Membro confirmado.*

O Código 6.9 mostra o método *memberOf()*, responsável por conectar um membro em uma comunidade. Outros métodos tais como: *hasMember()*, *administratorOf()* e *hasAdministrator()* funcionam de forma similar.

Código 6.9 – Conectando um membro com uma comunidade.

```
1  public void memberOf(String idCommunity) {
2    Model m = TDBFactory.createModel(DBDIRECTORY);
3    String queryString =
4      " PREFIX foaf: <"+FOAFNS+">"+
5      " PREFIX sioc: <"+SIOCNS+">"+
6      " INSERT { "+
7      "   <"+AIRETAMANS+"members/"+this.getMbox()+".foaf.rdf> "+
8      "   sioc:member_of "+
9      "   <"+AIRETAMANS+"communities/"+idCommunity+".sioc.rdf> "+
10     " } ";
11   GraphStore graphStore = GraphStoreFactory.create(m) ;
12   UpdateAction.parseExecute(queryString, graphStore) ;
13   m.close();
14 }
```

O Código 6.10 mostra o método *notMemberOf()*, responsável por desconectar um membro de uma comunidade. Outros métodos tais como: *notHasMember()*, *notAdministratorOf()* e *notHasAdministrator()* funcionam de forma similar.

Código 6.10 – Desconectando um membro de uma comunidade.

```
1  public void notMemberOf(String idCommunity) {
2    Model m = TDBFactory.createModel(DBDIRECTORY);
3    String queryString =
4      " PREFIX foaf: <"+FOAFNS+">"+
5      " PREFIX sioc: <"+SIOCNS+">"+
6      " DELETE { "+
7      "   <"+AIRETAMANS+"members/"+this.getMbox()+".foaf.rdf> "+
8      "   sioc:member_of "+
9      "   <"+AIRETAMANS+"communities/"+idCommunity+".sioc.rdf> "+
10     " } ";
11   GraphStore graphStore = GraphStoreFactory.create(m) ;
12   UpdateAction.parseExecute(queryString, graphStore) ;
13   m.close();
14 }
```

O Código 6.10 não utiliza o comando *WHERE* como o Código 6.4, pois irá remover uma única tripla.

6.9 Busca de Usuários

Da mesma forma que um usuário pode se conectar com uma comunidade, também pode se conectar com outro usuário. Este relacionamento é feito através da propriedade *foaf:knows*, como pode ser observado na Figura 6.11. Outros tipos de relacionamentos podem ser vistos na Seção 3.3.

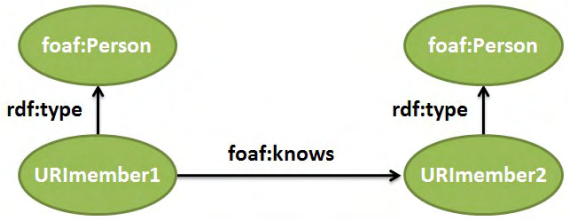


Figura 6.11: *Relacionamento entre usuários em um grafo RDF.*

O Código 6.11 mostra como pode ser feita a busca de usuários que tenham a ligação *foaf:knows*. É importante enfatizar que a busca é feita utilizando o emparelhamento de triplas (que pode ser visto nas linhas 9 à 11).

Código 6.11 – Consulta SPARQL para selecionar membros conhecidos.

```
1  public List<Person> selectKnows(int page) {
2    Model m = TDBFactory.createModel(DBDIRECTORY);
3    List<Person> members = new ArrayList<Person>();
4    int limit = PAGINATELIMIT; int offset = PAGINATELIMIT*(page-1);
5    String queryString =
6      " PREFIX foaf: <"+FOAFNS+">"+
7      " SELECT ?email "+
8      " WHERE { "+
9      "   ?member foaf:knows ?know ."+
10     "   ?member foaf:mbox \""+this.getMbox()+"\" ."+
11     "   ?know foaf:mbox ?email ."+
12     " } ORDER BY ?firstName LIMIT "+limit+" OFFSET "+offset;
13   Query query = QueryFactory.create(queryString);
14   QueryExecution qe = QueryExecutionFactory.create(query, m);
15   ResultSet results = qe.execSelect();
16   while(results.hasNext()) {
17     String result = results.next().get("email").toString();
18     Person member = new Member(result);
19     members.add(member);
20   }
21   return members;
22 }
```

No Código 6.11 também é possível ver o uso da paginação com os comandos *LIMIT* e *OFFSET* (na linha 12). Após a execução da consulta, a variável *results* é percorrida gerando uma lista de objetos do tipo *Person*. É importante notar que não são todos os dados que são retornados pela consulta mas apenas o atributo *?email*. Esta variável *?email* é utilizada para instanciar um novo objeto (linha 18), onde o construtor é encarregado de preencher todos os outros atributos.

A Figura 6.7 mostra uma página que utiliza o método *selectKnows()*, mostrando algumas informações do usuário "Juan Escobar" que está conectado com o usuário "Akira Sato".

A Figura 6.12 mostra os membros que não estão ligados ao usuário "Akira Sato". Para adicioná-los basta clicar no ícone "add user", que o mesmo irá invocar o método *knows()*, bastante similar ao *memberOf()* detalhado na Seção 6.8. A única diferença é que ao invés de utilizar a propriedade *sioc:member_of* é utilizada a propriedade *foaf:knows*.

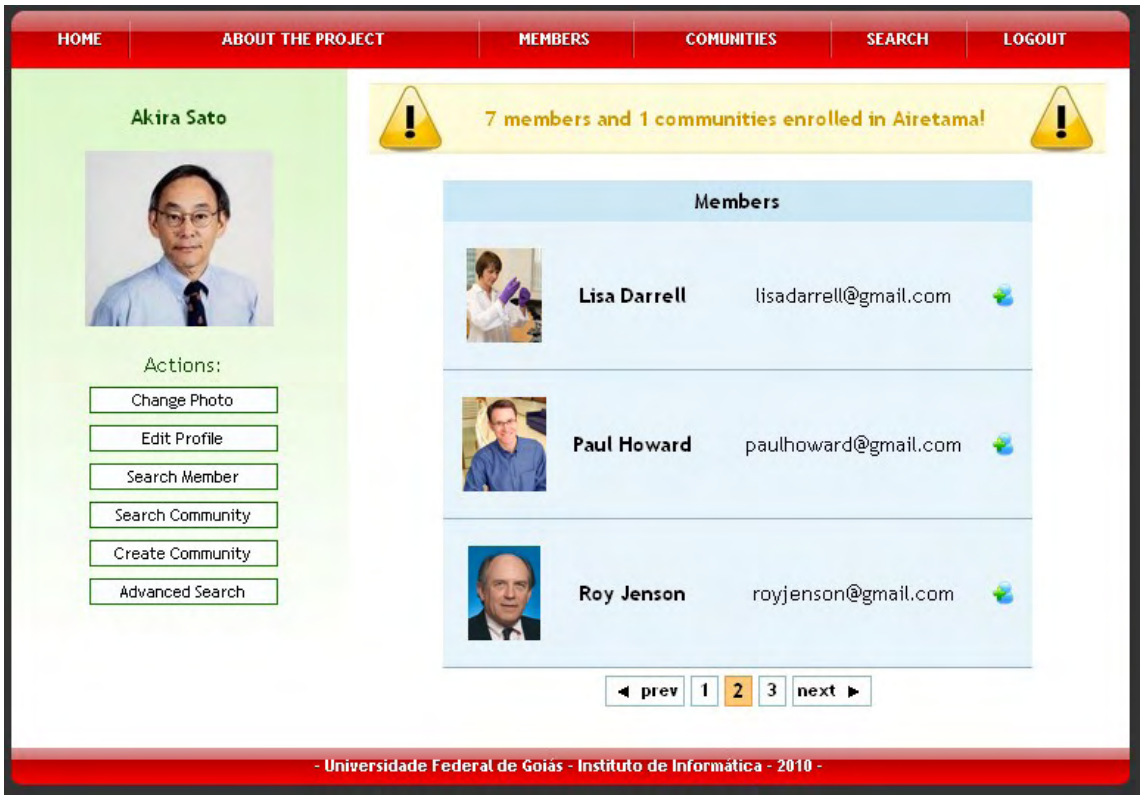


Figura 6.12: *Busca de membros.*

A busca de comunidades, tanto as que estão conectadas com o usuário, quanto as que não estão, é feita utilizando o mesmo padrão.

6.10 Agentes

A comunicação entre o Portal e o Sistema Multiagentes foi feita utilizando o JadeGateway, que é uma classe que fornece uma ponte de comunicação entre a plataforma JADE e uma aplicação externa, em especial *Servlets* e páginas JSP [53].

JadeGateway é uma classe *singleton* que possui apenas métodos estáticos. Desta forma, não é necessário instanciar uma classe para invocar os seus métodos. *Singleton* é um padrão de projeto de *software*, que garante a existência de uma única instância de uma classe, mantendo um ponto global de acesso ao seu objeto.

O Código 6.12 mostra um exemplo de utilização da classe JadeGateway. O *Main Container* é criado através do método *createMainContainer()* (linha 6), os agentes podem ser instanciados através do método *createNewAgent()*, e a classe pode ser iniciada através do método *init()* (linha 13).

Código 6.12 – Exemplo de utilização do JadeGateway.

```
1  <%
2    // Starting main container
3    Profile profile = new ProfileImpl();
4    profile.setParameter(Profile.PLATFORM_ID, "MyTestJadePlatform");
5    AgentContainer mainContainer =
6      Runtime.instance().createMainContainer(profile);
7
8    // Create dynamically some agent and start it immediately
9    mainContainer.createNewAgent("PrimeTool",
10     PrimeTool.class.getName(), null).start();
11
12   // Initializing Jade gateway
13   JadeGateway.init(MyGatewayAgent.class.getName(), null);
14 %>
```

Deve-se enfatizar que o JADE não restringe a arquitetura interna de cada agente, pois as estruturas internas dos agentes não são especificadas pela FIPA. Assim, a princípio pode ser utilizada qualquer uma das quatro arquiteturas tradicionais (Baseada em Lógica, Reativa, BDI, e baseada em Camadas) para a implementação dos agentes de ferramentas.

CAPÍTULO 7

Considerações Finais

Esta dissertação apresentou o arcabouço Airetama, cujo desenvolvimento permitiu aplicar conhecimentos de diversos campos relacionados com a Web Semântica, tais como: Comunidades Virtuais de Prática, Sistemas Multiagentes e Ontologias.

7.1 Contribuições

Uma das contribuições deste trabalho foi a proposta, definição, desenvolvimento e implementação de uma arquitetura genérica, extensível, flexível, escalável, semântica, baseada em agentes, aberta e livre.

A utilização de Comunidades Virtuais de Prática mostrou-se promissora, pois estas servem para contextualizar os conceitos manipulados pelos usuários em um domínio específico. Assim, cada comunidade delimita um "espaço conceitual" que pode ser utilizado pelos seus membros.

Além disso, o Airetama foi construído orientado a agentes, tornando-o altamente escalável. A utilização de agentes para implementação de ferramentas torna a inclusão, mudança ou atualização destas ferramentas mais fácil, pois agentes são mais modulares e independentes que o *software* convencional. Estas características incentivam que os desenvolvedores construam aplicações para o arcabouço. A utilização de agentes também possibilita que a requisição de um usuário permaneça sendo executada mesmo que ele não esteja *online*.

A vantagem do arcabouço Airetama incluir o Repositório Semântico, vem do fato do mesmo utilizar uma abordagem puramente ontológica para armazenamento dos dados, sem nenhuma utilização de banco de dados tradicionais, tornando as características de inteligência, interoperabilidade e integração um de seus principais focos.

O interessante em utilizar a ontologia FOAF é que a maioria dos recursos que podem ser acessados na Web possui algum tipo de relacionamento com pessoas. Assim, esta ontologia serve como ponto de partida onde outras ontologias de qualquer domínio podem ser acopladas de forma extensível.

O projeto Airetama também contribui na convergência de redes sociais, pois o perfil de um usuário ou de uma comunidade pode ser facilmente *linkado* ou importado.

O uso da ontologia SIOC foi facilitado pelo fato dela ser projetada para ser integrada com a FOAF. O mesmo modelo de integração pode ser utilizado para incluir novas ontologias, enriquecendo ainda mais a descrição da informação.

O uso de ontologias na base de dados do projeto contribui com o compartilhamento destes dados. Agentes de *software* podem utilizar estas informações de forma a satisfazer as requisições dos usuários. Desta forma, o uso de ontologias se mostra promissor para organizar as informações e compartilhar um vocabulário comum.

Mesmo supondo que todos os termos (conceitos e propriedades) possíveis estejam descritos através de ontologias, os grandes desafios atuais para o sucesso da Web Semântica que foram identificados neste trabalho são: Como transformar o desejo de um usuário em uma consulta semântica (como SPARQL) de forma automática? Como processar semanticamente as triplas consultadas? Como apresentar o resultado de uma consulta?

Apesar deste trabalho alcançar o objetivo descrito na Seção 1.2, algumas das diversas melhorias que podem ser feitas no arcabouço Airetama são listadas a seguir.

7.2 Trabalhos Futuros

Algumas das diversas melhorias que podem ser feitas no arcabouço Airetama são listadas a seguir:

- **Web Services Semânticos**: possibilitam o arcabouço se integrar com *softwares* externos e totalmente heterogêneos (por exemplo, implementados em outras linguagens);
- **Computação Pervasiva**: permite a comunicação do arcabouço com celulares e dispositivos móveis;
- **Computação Oportunista**: caso um membro da comunidade deseje disponibilizar recursos de memória e processador, os agentes dos membros podem ser migrados para o computador do usuário, tornando-o capaz de compartilhar recursos para o arcabouço;
- **Sistema de Recomendações**: todo o histórico de atividade de um membro poderá ser gerado pelo seu agente de forma semântica, possibilitando a recomendação ou sugestão de novos produtos ou serviços para o usuário;
- **Grades Semânticas**: uma abordagem semântica para a integração de recursos, aplicações e dados utilizando uma grade computacional;
- ***Crawlers*, *Robots* ou *Spiders* Semânticos**: conhecidos também na comunidade FOAF como *Scutters*, são agentes que processam páginas Web de forma semântica.

Referências Bibliográficas

[1] ADIDA, B.; BIRBECK, M.; 2008. **Rdfa primer**. Especificação disponível em http://www.w3.org/TR/xhtml-rdfa-primer, último acesso em setembro de 2010.

[2] ALARCÓN, J. A. B. L.; DE CARVALHO, C. L. **Airetama framework**. Disponível em http://sourceforge.net/projects/airetama, último acesso em setembro de 2010.

[3] ALBARELLO, A. B.; DE LUCENA, C. J. P.; 2004. **fgrin: Um framework baseado em sistemas multiagentes para formação de grupos de interesse a partir de uma base semântica**. Artigo disponível em ftp://ftp.inf.puc-rio.br/pub/docs/techreports/04_31_albarello.pdf, último acesso em setembro de 2010.

[4] ANTONIOU, G.; VAN HARMELEN, F.; 2008. **A Semantic Web Primer**. Cambridge, Massachusetts: The MIT Press, 2th edition.

[5] AUER, S.; BIZER, C.; IDEHEN, K. **Dbpedia**. Disponível em http://dbpedia.org, último acesso em setembro de 2010.

[6] BARKSDALE, J.; 1998. **Developing Multi-agent Systems With JADE**. Jossey-Bass.

[7] BELLIFEMINE, F.; CAIRE, G.; GREENWOOD, D.; 2007. **Developing Multi-agent Systems with JADE**. John Wiley & Sons Ltd.

[8] BELLIFEMINE, F.; CAIRE, G.; TRUCCO, T. **Jade programmer's guide**. Disponível em http://jade.tilab.com, último acesso em setembro de 2010.

[9] BENJAMINS, V. R.; FENSEL, D. **Community is knowledge in (ka)2**. Artigo disponível em http://ksi.cpsc.ucalgary.ca/KAW/KAW98/benjamins1, último acesso em setembro de 2010.

[10] BERNERS-LEE, T. **Weaving the web: The original design and ultimate destiny of the world wide web**. Harper Paperbacks, 1999.

[11] BERNERS-LEE, T.; 2001. **The semantic web**. Scientific American.

[12] BERNERS-LEE, T.; 2007. **Giant global graph - tim berners-lee personal blog**. Artigo disponível em http://dig.csail.mit.edu/breadcrumbs/node/215, último acesso em setembro de 2010.

[13] **Synthetic biology site**. Especificação disponível em http://syntheticbiology.org/Semantic_web_ontology/Examples.html, último acesso em setembro de 2010.

[14] **Rdf calendar - an application of the resource description framework to icalendar data**. Especificação disponível em http://www.w3.org/TR/rdfcal/#ns-gnd, último acesso em setembro de 2010.

[15] **Fipa - the foundation for intelligent physical agents**. Disponível em http://www.fipa.org, último acesso em setembro de 2010.

[16] **Friend of a friend (foaf) project site**. Disponível em http://www.foaf-project.org, último acesso em setembro de 2010.

[17] **Foaf vocabulary specification**. Especificação disponível em http://xmlns.com/foaf/spec, último acesso em setembro de 2010.

[18] **Goodrelations - the web ontology for e-commerce**. Especificação disponível em http://purl.org/goodrelations, último acesso em setembro de 2010.

[19] **Hakia search engine**. Disponível em http://www.hakia.com, último acesso em setembro de 2010.

[20] **Joseki - a sparql server for jena**. Disponível em http://joseki.sourceforge.net/, último acesso em setembro de 2010.

[21] **Linkedin official site**. Disponível em http://www.linkedin.com, último acesso em setembro de 2010.

[22] **Microformats official page**. Disponível em http://microformats.org, último acesso em setembro de 2010.

[23] **The open biological and biomedical ontologies**. Disponível em http://www.obofoundry.org, último acesso em setembro de 2010.

[24] **Ontoselect ontology library**. Disponível em http://olp.dfki.de/ontoselect, último acesso em setembro de 2010.

[25] **Schemaweb - directory of rdf schemas**. Disponível em http://www.schemaweb.info, último acesso em setembro de 2010.

[26] **Semantically-interlinked online communities project**. Especificação disponível em http://sioc-project.org, último acesso em setembro de 2010.

[27] **Sioc core ontology specification**. Especificação disponível em http://rdfs.org/sioc/spec, último acesso em setembro de 2010.

[28] **Skos simple knowledge organization system reference**. Especificação disponível em http://www.w3.org/TR/skos-reference, último acesso em setembro de 2010.

[29] **Arq - a sparql processor for jena**. Disponível em http://jena.sourceforge.net/ARQ, último acesso em setembro de 2010.

[30] **Swoogle - semantic web search**. Disponível em http://swoogle.umbc.edu, último acesso em setembro de 2010.

[31] **Tdb subsystem for jena**. Disponível em http://jena.sourceforge.net/downloads.html#TDB, último acesso em setembro de 2010.

[32] BRICKLEY, D.; 2003. **Basic geo vocabulary**. Especificação disponível em http://www.w3.org/2003/01/geo, último acesso em setembro de 2010.

[33] DA COSTA, M. T. C. **Uma arquitetura baseada em agentes para suporte ao ensino à distância**. Tese da UFSC disponível em http://www.eps.ufsc.br/teses99/thiry/, último acesso em setembro de 2010.

[34] DAVIES, J.; DUKE, A.; STONKUS, A.; 2002. **Ontoshare: Using ontologies for knowledge sharing**. Artigo disponível em http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-55/davies.pdf , último acesso em setembro de 2010.

[35] DAVIS, I.; VITIELLO, E. **Relationship: A vocabulary for describing relationships between people**. Especificação disponível em http://vocab.org/relationship/.html, último acesso em setembro de 2010.

[36] DE CARVALHO, C. L. **Dream web (dweb)**. Disponível em http://www.inf.ufg.br/∼cedric/DWeb.html, último acesso em setembro de 2010.

[37] DE SOUZA, E. M. S.; 1996. **Uma estrutura de agentes para assessoria na internet**. Dissertação da UFSC disponível em http://www.eps.ufsc.br/disserta96/eliane/index/index.htm#sumario, último acesso em setembro de 2010.

[38] DICKINSON, I. **The jena ontology api**. Especificação disponível em http://jena.sourceforge.net/ontology/index.html, último acesso em setembro de 2010.

[39] DICKINSON, I.; 2008. **Jena schemagen howto**. Tutorial disponível em http://jena.sourceforge.net/how-to/schemagen.html, último acesso em setembro de 2010.

[40] DRUMMOND, N.; HORRIDGE, M.; STEVENS, R.; WROE, C.; SAMPAIO, S.; 2007. **Pizza ontology**. Especificação disponível em http://www.co-ode.org/ontologies/pizza/2007/02/12/, último acesso em setembro de 2010.

[41] DUMBILL, E. **Description of a project**. Especificação disponível em http://trac.usefulinc.com/doap, último acesso em setembro de 2010.

[42] ERNST, S.; 2009. **Top 25 social networks re-rank**. Disponível em www.compete.com, último acesso em setembro de 2010.

[43] ESCOBAR, M.; LEMKE, A. P.; RIBEIRO, M. B.; 2006. **Semanticore 2006 – permitindo o desenvolvimento de aplicações baseadas em agentes na web semântica**. Artigo disponível em http://www.les.inf.puc-rio.br/seas2006/papers/X072.pdf, último acesso em setembro de 2010.

[44] FERNANDES, J. H. C. **Agentes inteligentes**. Artigo disponível em http://www.cic.unb.br/ jhcf/MyBooks/ciber/doc-ppt-html/ AgentesInteligentes.html, último acesso em setembro de 2010.

[45] FERREIRA, A. B. H. **Dicionário aurélio eletrônico - século xxi**. Editora Nova Fronteira, 1999.

[46] FRANKLIN, S.; GRAESSER, A.; 1996. **Is it an agent, or just a program? a taxonomy for autonomous agents**. Artigo disponível em http://www.msci.memphis.edu/ franklin/AgentProg.html, último acesso em setembro de 2010.

[47] **Gnu general public license (gpl)**. Disponível em http://www.gnu.org/licenses/licenses.html#GPL, último acesso em setembro de 2010.

[48] GRUBER, T.; 1992. **What is an ontology?** Artigo disponível em http://www-ksl.stanford.edu/kst/what-is-an-ontology.html, último acesso em setembro de 2010.

[49] HALPIN, H.; DAVIS, I.; 2007. **Grddl primer**. Especificação disponível em http://www.w3.org/TR/grddl-primer, último acesso em setembro de 2010.

[50] HILLMANN, D.; 2005. **Dublin core metadata userguide**. Guia do Usuário disponível em http://dublincore.org/documents/usageguide, último acesso em setembro de 2010.

[51] IEEE.; ISO/IEC.; 2007. **Systems and software engineering - recommended practice for architectural description of software-intensive systems**. Especificação disponível em http://www.iso-architecture.org/ieee-1471, último acesso em setembro de 2010.

[52] **Jena: A semantic web framework for java**. Disponível em http://jena.sourceforge.net, último acesso em setembro de 2010.

[53] KELEMEN, V.; 2006. **Simple example for using the jadegateway class**. Tutorial disponível em http://jade.cselt.it/doc/tutorials/JadeGateway.pdf, último acesso em setembro de 2010.

[54] KING, R. **Microformats draft specification**. Especificação disponível em http://microformats.org/wiki/hresume, último acesso em setembro de 2010.

[55] KOCH, M.; LACHER, M. S. **Integrating community services - a common infrastructure proposal**. Discrete Math, 2000.

[56] KROTZSCH, M.; VRANDECIC, D.; VOLKEL, M. **Semantic mediawiki**. Disponível em http://semantic-mediawiki.org, último acesso em setembro de 2010.

[57] LAUSEN, H.; STOLLBERG, M.; HERNÁNDEZ, R. L.; DING, Y.; HAN, S.-K.; FENSEL, D. **Semantic web portals – state of the art survey**. Institute for Computer Science University of Innsbruck, Austria, 1997.

[58] LENAT, D.; WITBROCK, M.; CURTIS, J.; MATUSZEK, C. **Cyc knowledge base**. Artigo disponível em http://cyc.com/cyc/technology/whatiscyc_dir/whatsincyc, último acesso em setembro de 2010.

[59] **Gnu lesser general public license (lgpl)**. Disponível em http://www.gnu.org/copyleft/lesser.html, último acesso em setembro de 2010.

[60] MAES, P.; 2005. **Modeling adaptative autonomous agents**. Artigo disponível em http://www.sci.brooklyn.cuny.edu/ sklar/teaching/f05/ alife/papers/maes-94modeling.pdf, último acesso em setembro de 2010.

[61] MARDER, S.; WILSON, G.; LEUNG, T.; MEYER, B.; MUTTER, A.; ZIMMERMAN, M. **Eurekster official site**. Disponível em http://www.eurekster.com, último acesso em setembro de 2010.

[62] MARKOFF, J. **What is web 3.0**. The New York Times, Califórnia, 2006.

[63] MARRAFA, P. **Wordnet - a lexical database for english**. Disponível em http://www.clul.ul.pt/clg/wordnetpt/index.html, último acesso em setembro de 2010.

[64] MILLER, G. A.; FELLBAUM, C.; TENGI, R.; WAKEFIELD, P.; LANGONE, H. **Wordnet - a lexical database for english**. Disponível em http://wordnet.princeton.edu, último acesso em setembro de 2010.

[65] MOTA, P. G.; 2007. **De rerum natura - biodiversidade: 300 anos de lineu**. Disponível em http://dererummundi.blogspot.com/2007/05/biodiversidade-300-anos-de-lineu.html, último acesso em setembro de 2010.

[66] O'REILLY, T.; 2005. **What is web 2.0**. Artigo disponível em http://oreilly.com/web2/archive/what-is-web-20.html, último acesso em setembro de 2010.

[67] PARADA, R. A.; 2008. **Doac vocabulary specification**. Especificação disponível em http://ramonantonio.net/doac/0.1, último acesso em setembro de 2010.

[68] PELL, B. **Powerset - microsoft corporation**. Disponível em http://www.powerset.com, último acesso em setembro de 2010.

[69] RAIMOND, Y. **Music ontology specification**. Especificação disponível em http://musicontology.com, último acesso em setembro de 2010.

[70] RHEINGOLD, H.; 1998. **The virtual community**. Disponível em http://www.rheingold.com/vc/book, último acesso em setembro de 2010.

[71] RUSSEL, S.; NORVIG, P. **Inteligência Artificial**. Campus, 2003.

[72] SANTANA, L. H. Z.; DO PRADO, A. F.; DE SOUZA, W. L. **Ubick: Um framework baseado em agentes de software para computação ubíqua**. Artigo disponível em http://www.lbd.dcc.ufmg.br:8080/colecoes/seas/2008/002.pdf, último acesso em setembro de 2010.

[73] SEABORNE, A.; MANJUNATH, G.; BIZER, C.; BRESLIN, J.; DAS, S.; 2008. **Sparql update - a language for updating rdf graphs**. Especificação disponível em www.w3.org/Submission/SPARQL-Update, último acesso em setembro de 2010.

[74] SILVA, R. P. **Suporte ao desenvolvimento e uso de frameworks e componentes**. PhD thesis, Instituto de Informática, UFRS, 2000.

[75] SIMMONS, J.; 2007. **Microformats vs. rdf: How microformats relate to the semantic web**. Artigo disponível em http://www.semanticfocus.com/blog/entry/title/microformats-vs-rdf-how-microformats-relate-to-the-semantic-web, último acesso em setembro de 2010.

[76] SPIVACK, N.; ERICKSON, S.; WISSNER, J. **Twine - radar networks**. Disponível em www.twine.com, último acesso em setembro de 2010.

[77] STONE, P.; VELOSO, M. **Multiagent systems: A survey from a machine learning perspective**. Springer Netherlands, 2004.

[78] TALIGENT.; 1995. **Leveraging object-oriented frameworks**. Artigo disponível em http://lhcb-comp.web.cern.ch/lhcb-comp/Components/postscript/leveragingoo.pdf, último acesso em setembro de 2010.

[79] TANENBAUM, A. S. **Sistemas Distribuídos - Princípios e Paradigmas**. Prentice-Hall, 2007.

[80] VARELLA, A. N. **Coopractice – comunidades de prática virtuais apoiadas por ontologias**. Universidade Federal do Rio de Janeiro, 2007.

[81] VAUCHER, J. **Jade tutorial and primer**. Tutorial disponível em http://www.iro.umontreal.ca/ ∼vaucher/Agents/Jade_Primer.zip, último acesso em setembro de 2010.

[82] VICTORETTE, G. W. D. B. **O processo de construção de ontologias baseado na modelagem uml**. Tutorial disponível em http://projetos.inf.ufsc.br/arquivos_projetos/projeto_712/uml2onto.pdf, último acesso em setembro de 2010.

[83] W3C. **Extensible markup language (xml)**. Artigo disponível em http://www.w3.org/XML, último acesso em setembro de 2010.

[84] W3C. **World wide web consortium (w3c)**. Artigo disponível em http://www.w3.org, último acesso em setembro de 2010.

[85] W3C.; 2004. **Owl web ontology language overview**. Especificação disponível em http://www.w3.org/TR/owl-features, último acesso em setembro de 2010.

[86] W3C.; 2004. **Owl web ontology language reference**. Especificação disponível em http://www.w3.org/TR/owl-ref, último acesso em setembro de 2010.

[87] W3C.; 2004. **Rdf primer**. Artigo disponível em http://www.w3.org/TR/rdf-primer, último acesso em setembro de 2010.

[88] W3C.; 2004. **Rdf primer — turtle version**. Especificação disponível em http://www.w3.org/2007/02/turtle/primer, último acesso em setembro de 2010.

[89] W3C.; 2004. **Rdf vocabulary description language 1.0: Rdf schema**. Especificação disponível em http://www.w3.org/TR/rdf-schema, último acesso em setembro de 2010.

[90] W3C.; 2004. **Web ontology language (owl)**. Especificação disponível em http://www.w3.org/2004/OWL, último acesso em setembro de 2010.

[91] WENGER, E.; 1998. **Communities of Practice - Learning, Meaning, and Identity**. Cambridge University Press, New York.

[92] WENGER, E.; 2001. **Supporting communities of practice - a survey of community-oriented technologies**. Artigo disponível em http://www.ewenger.com/tech, último acesso em setembro de 2010.

[93] WOLFRAM, S. **Wolfram alpha - computational knowledge engine**. Disponível em http://www.wolframalpha.com, último acesso em setembro de 2010.

[94] WOOLDRIDGE, M.; JENNINGS, N. R.; 1995. **Intelligent Agents: Theory and Practice**. Manchester Metropolitan University and Queen Mary & Westfield College.