Desenvolvendo aplicações baseadas em agentes com o SemantiCore

Mauricio S. Escobar, Ana Paula Lemke, Luis H. Ries, Marcelo B. Ribeiro

Programa de Pós-Graduação em Ciência da Computação – Faculdade de Informática –
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Av. Ipiranga, 6681 – 90619-9000 – Porto Alegre – RS – Brazil
{mauricio.escobar, ana.lemke, luis.ries, marcelo.blois}@pucrs.br

Abstract*. This paper describes the main features of the SemantiCore framework, an agent infrastructure to develop multi-agent systems (MAS). A look at an application illustrates the SemantiCore potential as an infrastructure for the development of agent applications. It is also presented some works that extending the SemantiCore functionalities. These extensions aim to support the development of mobile and Semantic Web applications, and knowledge management capabilities to the agents.*

Resumo. *Este artigo descreve as principais características do framework SemantiCore, uma infra-estrutura de agentes para o desenvolvimento de sistemas multiagentes (SMAs). Para ilustrar o potencial do SemantiCore como uma infra-estrutura de desenvolvimento de agentes, uma aplicação é apresentada. Também são apresentados alguns trabalhos que estendem as funcionalidades do SemantiCore. Estas extensões visam dar suporte ao desenvolvimento de aplicações móveis e para a Web Semântica, além de agregar aos agentes a capacidade de gerenciamento de conhecimento.*

1. Introdução

O desenvolvimento de sistemas multiagentes (SMAs) é uma área em franca expansão devido a sua adaptação a resolução de problemas de natureza distribuída [Luger 2002] e, em especial, a possibilidade de aplicação na construção de sistemas para a Web. O tema, historicamente restrito ao ambiente de pesquisa de Inteligência Artificial, passou a influenciar outras áreas da computação, como a área de Engenharia de Software [Jennings e Wooldridge 1999].

A identificação de novos requisitos para a construção de SMAs motivou o surgimento de ferramentas conceituais e computacionais que incorporam agentes de software como entidades fundamentais para a resolução de problemas complexos com distribuição de processamento [Odell 2002]. Em especial, a necessidade da definição de uma plataforma de prototipação e implementação de SMAs tornou-se evidente [Bigus e Bigus, 2001].

Algumas iniciativas foram tomadas por parte de instituições de pesquisa e por consórcios de empresas para a criação de uma infra-estrutura padrão multiagentes, que servisse de base para a criação de sistemas multiagentes interoperáveis. Uma das primeiras plataformas criadas com este objetivo foi o FIPA [Poslad *et al*. 2000]. Com o

uso de uma forma padrão de comunicação e a presença de entidades básicas de gerenciamento das funções da plataforma, a FIPA criou um modelo capaz de mapear as principais funções envolvidas na interação entre agentes de uma sociedade. Embora não tenha atingido plenamente o seu objetivo de se tornar uma plataforma padrão, o modelo da FIPA serviu como base para a implementação de diversas plataformas multiagentes.

Entre as plataformas disponíveis [JADE 2009, MadKit 2005, OpenCybele 2005, Jason 2009], algumas objetivam dar suporte a criação de SMAs genéricos, enquanto outras concentram esforços em determinados tipos de aplicação. Um dos tipos de aplicação para o qual há poucos estudos é o desenvolvimento de SMAs para a Web. A idéia de uso da Web como um ambiente onde agentes se relacionam foi proposta claramente por Berners Lee *et al*. (2001).

Com o objetivo de permitir o desenvolvimento de SMAs para a Web, incorporando os requisitos necessários à criação da Web Semântica, foi desenvolvida, em meados de 2002, a arquitetura Web Life [Ribeiro 2002]. Nessa época, falava-se muito sobre a Web Semântica, mas existiam poucas ferramentas que permitiam (ou ao menos objetivavam) que as suas características fossem de fato desenvolvidas. O Jena [JENA 2009], por exemplo, foi disponibilizado no final de 2001.

O SemantiCore, apresentado inicialmente em 2004 [Blois e Lucena 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.

O objetivo deste artigo é apresentar em linhas gerais a arquitetura do SemantiCore. A arquitetura do SemantiCore possui diversos pontos de flexibilidade que permitem a sua extensão, visando a adição de novas funcionalidades. Neste sentido, são apresentados alguns trabalhos que estendem o SemantiCore, permitindo a sua utilização em diferentes áreas, como a Web Semântica e a computação móvel.

O restante deste artigo está organizado da seguinte forma: na Seção 2 é apresentado o *framework* SemantiCore e a descrição de uma aplicação desenvolvida; na Seção 3 são apresentados trabalhos que estendem as funcionalidades do SemantiCore. Por fim, na Seção 4 são traçadas algumas considerações, seguidas das referências bibliográficas utilizadas ao longo do texto.

2. O Framework SemantiCore

Conforme descrito em Blois e Lucena (2004), o SemantiCore é um *framework* que provê uma camada de abstração sobre serviços de distribuição e uma definição interna de agente capaz de oferecer aos desenvolvedores uma abstração de alto nível para a construção de SMAs.

O *framework* SemantiCore é dividido em dois modelos: o modelo do agente (*SemanticAgent*) e o modelo do ambiente. Os dois modelos dispõem de pontos de flexibilidade (*hotspots*) permitindo aos desenvolvedores associar diferentes padrões, protocolos e tecnologias. O modelo do agente possui uma estrutura orientada a componentes, onde cada componente contribui para uma parte essencial do funcionamento do agente, agregando todos os aspectos necessários a sua

implementação. São quatro os componentes básicos de um agente no SemantiCore: o sensorial, o decisório, o executor e o efetuador.

Para perceber e capturar os recursos que trafegam pelo ambiente, o agente possui o componente *sensorial*. Este componente contém uma série de sensores definidos pelo desenvolvedor (cada sensor captura um tipo diferente de objeto do ambiente) que são verificados toda vez que é percebido um objeto no ambiente. Se um ou mais sensores forem ativados, os objetos são encaminhados para processamento em outros componentes.

O componente *decisório* encapsula o mecanismo de tomada de decisão do agente. O mecanismo decisório presente no componente é um dos pontos de flexibilidade do *framework*. Visando permitir o desenvolvimento de aplicações voltadas a Web Semântica, o componente decisório opera sobre ontologias (em termos de fatos e regras) o que torna necessário o uso de uma linguagem apropriada para a definição semântica dos dados, como OWL.

A saída gerada pelo componente decisório deve ser uma instância de uma ação (*Action*). As ações mapeiam todos os possíveis comandos que um agente deve entender para trabalhar de forma apropriada e podem ser aplicadas tanto aos elementos do agente quanto aos elementos do ambiente. O desenvolvedor pode definir suas próprias ações através de uma extensão da classe *Action* (*hotspot*) presente no *framework*. No SemantiCore, as ações são vistas como processos de computação, cujo modelo é apresentado por Ferber em [Ferber 1999]. O componente *executor* contém os planos de ação que serão executados pelo agente e trabalha com o mecanismo de *workflow*. Este mecanismo é necessário para o controle das transições de atividades dentro de um processo do *workflow*.

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. Uma das características do SemantiCore é a abstração da plataforma de software e do protocolo de comunicação, possibilitando ao desenvolvedor da aplicação enviar e receber mensagens usando diferentes padrões, como Web Service SOAP [Gudgin *et al*. 2002] e FIPA ACL [FIPA 2002].

Para que um agente possa atuar, é necessário que ele esteja situado em um ambiente. No SemantiCore, o ambiente onde os agentes atuam possui algumas entidades administrativas, como o Controlador de Ambiente (*Environment Controller*) e o Gerente de Ambiente (*Environment Manager*). O Controlador de Ambiente é responsável por registrar os agentes, pela recepção de agentes móveis vindos de outros ambientes e também pelas características de segurança. O Gerente de Ambiente representa uma ponte com outros ambientes que estão distribuídos.

No SemantiCore é possível criar ambientes e agentes distribuídos. Em ambientes distribuídos, a primeira instância a entrar em execução é considerada a instância principal e contém, portanto, o Controlador de Ambiente. Como exemplo de ambiente distribuído cita-se uma praça de compra e venda de ações, onde há clientes e vendedores operando em máquinas diferentes.

A distribuição de agentes permite que os componentes do agente estejam espalhados nas diferentes partes do ambiente. Com esta distribuição é possível, por exemplo, balancear a carga computacional, colocando componentes que necessitam de maior poder computacional em máquinas de maior porte. No SemantiCore, a localização do agente distribuído é armazenada de acordo com a localização de seu componente sensorial, isto porque é através deste componente que o agente recebe informações do ambiente.

No SemantiCore há dois tipos de barramentos para a comunicação: o de controle e o de dados. No barramento de controle trafegam as mensagens de controle do SemantiCore, escritas em um formato proprietário e fixo. No barramento de dados trafegam as mensagens de dados que são trocadas entre os diferentes agentes. Os componentes sensorial e efetuador de cada agente estão ligados ao barramento de dados, podendo assim, receber e enviar mensagens de e para outros agentes. Ambos os barramentos possibilitam a implementação de mecanismos de segurança, visando garantir a integridade na troca de mensagens.

2.1. Criando um agente no SemantiCore

Um agente no SemantiCore deve necessariamente estender a classe *SemanticAgent* (Tabela 1). Essa classe define que o agente deva iniciar a sua execução através do método *setup*. Durante o *setup* o desenvolvedor pode criar sensores, fatos, regras, efetuadores, ações, planos de ação e objetivos para os agentes. Todas essas estruturas são criadas através da utilização de classes do SemantiCore, formando assim, o modelo de agente. A Tabela 1 mostra um exemplo de código extraído de um agente responsável por verificar as clínicas que possuem qualificação “excelente”.

Tabela 1. Exemplo de criação de agente

```
 1 public class ClinicCheckerAgent extends SemanticAgent {
 2  protected void setup(){
 3     addSensor(new OWLSensor());
 4     OntModel clinicSchema = OWLUtil.ontModelFromFile("./clinics.owl");
 5     setDecisionEngine(new InferenceJenaEngine(clinicSchema));
 6     SimpleFact sf1 = new SimplesFact("?clinic", "rdf:type", "#Clinic");
 7     SimpleFact sf3 = new SimplesFact("?clinic", "#qualification",
 8 "'excelent'");
 9     ComposedFact cf1 = new ComposedFact(sf1, sf2);
10     SimpleFact consequence = new SimpleFact("?clinic", "#accepted",
11 "'true'");
12     Rule clinicRule1 = new Rule("rule1", cf1, consequence);
13     addRule(clinicRule1);
14 }}
```

A classe *SemanticAgent* possui métodos que permitem a adição de todos os elementos que um agente pode ter no SemantiCore. O método *addSensor* permite ao agente utilizar o sensor *OWLSensor* já definido (linha 3). O método *setDecisionEngine* (linha 5) é usado para definir o mecanismo de inferência a ser usado para a manipulação de ontologias. Este método permite a criação de agentes com diferentes métodos de tomada de decisão. Durante o *setup* também é possível criar fatos e regras. Através das classes *SimpleFact* e *ComposedFact* é possível criar diferentes padrões de fatos que devem ser associados ao mecanismo de tomada de decisão do agente, ou utilizados como parte de uma regra. O método *setup* é executado somente quando o agente é criado no ambiente. Uma vez iniciado, o agente basicamente executa um laço com 4 operações: sentir o ambiente, decidir de acordo com as informações sentidas, executar

ações de acordo com a decisão tomada, e, publicar informações de volta no ambiente. Este ciclo de vida é automaticamente gerenciado pelo SemantiCore.

2.2. Exemplo de aplicação que utiliza o SemantiCore

Em um mundo cada vez mais competitivo, criar redes de empresas com objetivos comuns tende a reduzir incertezas, principalmente em processos de inovação. Entretanto, a identificação de empresas com interesses relacionados – e a aproximação destas - é uma tarefa complexa. Os gerentes ou administradores necessitam de conhecimento consistente e atualizado sobre as competências das empresas, seus interesses e os relacionamentos com outras empresas para poder sugerir prováveis associações entre elas.

Para facilitar a coleta de informações de diferentes empresas e verificar uma provável associação entre elas, foi desenvolvido o SIMNET - um sistema baseado em agentes projetado para simular a criação de arranjos institucionais entre empresas de acordo com um modelo previamente projetado. O protótipo do SIMNET foi desenvolvido utilizando o *framework* SemantiCore.

No SIMNET, cada empresa é representada por um agente e há um agente centralizador (*broker*), por meio do qual o administrador pode indicar novas demandas. Cada agente institucional possui seu próprio mecanismo de tomada de decisão, que é responsável por analisar as demandas e decidir se é benéfico ou não a participação da instituição em um arranjo (baseado nos interesses da instituição representada pelo agente).

Cada agente institucional manifesta seu interesse em uma demanda através do envio de uma mensagem para o agente *broker*. Com base nas respostas recebidas dos agentes que representam as instituições, o agente *broker* apresenta arranjos institucionais que poderiam ser criados em função da demanda fornecida.

A Figura 1 apresenta o gráfico gerado pelo protótipo desenvolvido com o SemantiCore. À esquerda da tela, podem ser visualizadas todas as empresas cadastradas para a simulação. O gráfico a direita mostra a intensidade dos possíveis relacionamentos entre as empresas cadastradas em decorrência de uma demanda composta pelas áreas “Engenharia de Software” e “Java”. A Empresa 4 é o centro do gráfico porque possui o maior somatório de competências e interesses na demanda. A distância entre uma determinada empresa e a empresa central indica a intensidade de um relacionamento provável entre elas. As linhas cheias indicam os relacionamentos já existentes entre duas empresas.

3. Extensões do SemantiCore

Nesta seção serão descritos alguns trabalhos que estendem o *framework* SemantiCore. Alguns trabalhos visam adicionar novas funcionalidades aos agentes, enquanto outros buscam permitir a sua aplicabilidade a outras áreas, como no caso da computação móvel. Através da flexibilidade presente nas estruturas que formam os agentes SemantiCore, é possível que sejam criados e adicionados novos componentes aos agentes. Através da adição de novos componentes, um agente pode adquirir outras habilidades, como a organização de seu conhecimento.

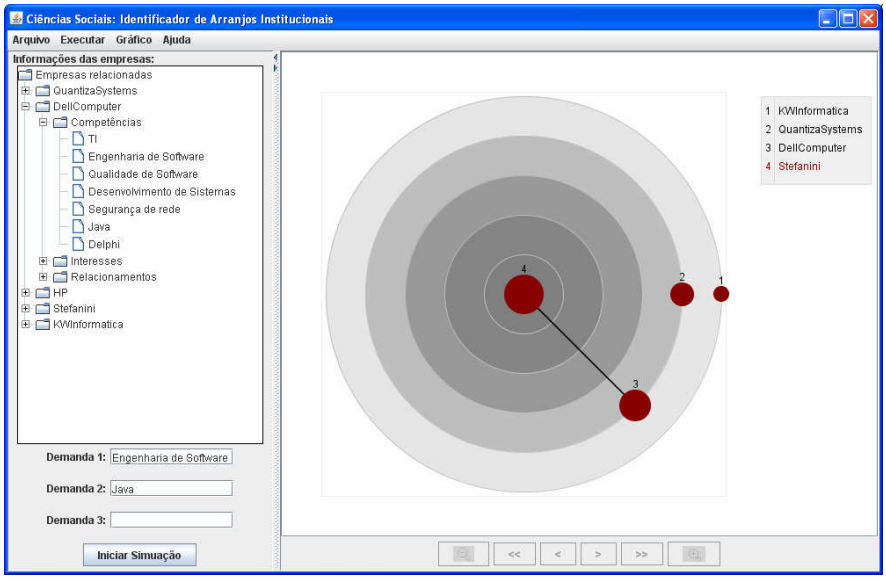


Figura 1: Interface com o usuário do agente *broker*.

3.1. Semantic Web Plugin

O Semantic Web Plugin (SWP) [Escobar 2009] é uma arquitetura dividida em dois modelos. Um dos modelos integra o SemantiCore a navegadores (SWP-cliente) e o outro a servidores Web (SWP-servidor), permitindo a criação de ambientes onde os agentes de software vivem, estendendo a Web, sem interferir na sua estrutura atual. Estes ambientes alteram o atual paradigma *request-response* utilizado na Web para um paradigma híbrido *request-response* / *peer-to-peer*. Os agentes vivendo em máquinas clientes em ambientes associados a navegadores Web são capazes de comunicar-se com outros agentes em outros ambientes e processar conteúdos anotados semanticamente em páginas Web.

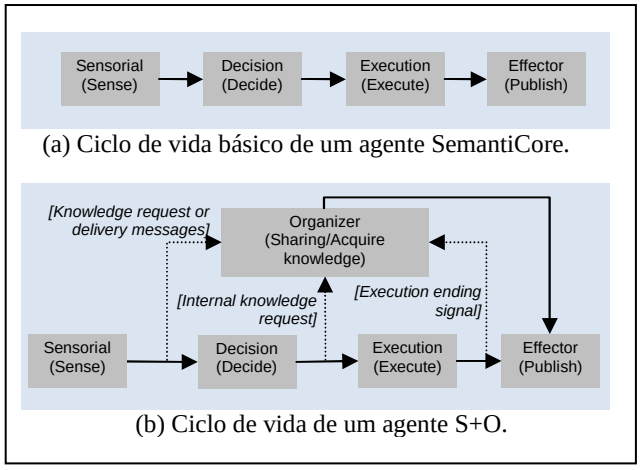
O SWP-cliente possui mecanismos que realizam a captura de anotações semânticas em OWL das páginas Web, extraindo estas informações das demais informações da página, e o seu envio para processamento pelos agentes no ambiente. O SWP-servidor adiciona informações na resposta a uma requisição HTTP, permitindo que um SWP associado ao navegador possa automaticamente reconhecer o ambiente presente no SWP-servidor. Após esse reconhecimento, os ambientes podem conectar-se e os agentes podem comunicar-se diretamente.

3.2. SemantiCore+Ontowledge

O Ontowledge é um *framework* desenvolvido em Java para dar suporte a agentes que não possuem conhecimento (ou não possuem o conhecimento adequado) para atingir um determinado objetivo. O SemantiCore+Ontowledge, ou simplesmente S+O, é o *framework* resultante da integração do Ontowledge ao SemantiCore. Conforme pode ser visto na Figura 2, agentes desenvolvidos com o S+O possuem cinco componentes básicos, que são os quatro componentes da distribuição padrão e o componente *Organizador*.

Sempre que um agente com o Ontowledge integrado não possui o conhecimento necessário para atingir um objetivo, um *objeto de conhecimento* é recuperado e o seu

conteúdo (o conhecimento encapsulado no objeto) é automaticamente adicionado a arquitetura do agente. Um objeto de conhecimento encapsula todos os elementos requeridos para resolver determinado problema, definição similar a "*knowledge source*" dada por Ferber em [Ferber 1999]. Os objetos de conhecimento são representados com ontologias. Através da verificação da satisfação alcançada com o uso de um objeto de conhecimento, agentes com o Ontowledge integrado também são capazes de avaliar seus próprios desempenhos. Assim, os agentes se tornam capazes de identificar quando o conhecimento corrente não está sendo satisfatório para atingir determinado objetivo e de adquirir novos conhecimentos para a sua substituição.



(a) Ciclo de vida básico de um agente SemantiCore.

(b) Ciclo de vida de um agente S+O.

Figura 2. Componentes de um agente SemantiCore.

3.2.1 Desenvolvimento de aplicações com o SemantiCore+Ontowledge

A primeira aplicação desenvolvida com o S+O foi baseada no exemplo para a Web Semântica descrito por Berners-Lee e co-autores [Berners-Lee *et al*. 2001]. Na aplicação desenvolvida, os agentes de Pete e Lucy (dois irmãos que precisam agendar sessões de fisioterapia para a mãe) não "nascem" com o conhecimento necessário para atingir o objetivo "Selecionar Clínicas" e, devido a diferentes políticas e critérios de seleção, acabam selecionando objetos de conhecimento diferentes para a realização do objetivo citado. Desta forma, os agentes identificam conjuntos diferentes de clínicas, visto que utilizam diferentes estratégias para localizá-las.

Outra aplicação foi desenvolvida para mostrar como os agentes podem identificar e substituir o conhecimento obsoleto com base na análise do histórico de execução. A aplicação simula a interação entre agentes que tem como objetivo a venda de carros (vendedores de uma concessionária) e agentes que buscam por ofertas de carros (clientes). Os agentes que representam os vendedores precisam estar atentos as mudanças no mercado para que possam adaptar suas estratégias de vendas e atingir o objetivo "vender carro" de maneira mais satisfatória (alcançando maiores lucros). O desenvolvimento das aplicações foi feito na linguagem Java (J2SE 5.0) (ambiente de desenvolvimento Eclipse) e as ontologias foram criadas e processadas utilizando a API do Jena (versão 2.4).

3.3. SemantiCore Mobile

Dispositivos móveis e embarcados se caracterizam por possuírem menor capacidade computacional e uma série de limitações, tais como processamento, memória e consumo de energia. Para que o SemantiCore pudesse ser utilizado neste tipo de dispositivo, foi necessária uma reformulação em sua arquitetura, reduzindo os recursos computacionais necessários para a sua execução. À versão simplificada do SemantiCore, deu-se o nome de SemantiCore *Mobile*.

O SemantiCore *Mobile* é um *framework* para auxiliar no desenvolvimento de aplicações multiagentes para dispositivos móveis e embarcados [Escobar *et al*. 2007]. Por estender o SemantiCore, o SemantiCore *Mobile* permite a criação de agentes, a criação do ambiente e a troca de mensagens entre agentes do mesmo ou de diferentes ambientes.

3.3.1 Desenvolvimento de aplicações com o SemantiCore *Mobile*

A fim de avaliar a viabilidade do SemantiCore *Mobile* para ambientes móveis, duas aplicações foram desenvolvidas: *Servidor de Notícias* e *Guia Turístico*. Na aplicação *Servidor de Notícias*, dois agentes se comunicam entre si para disponibilizar informações (notícias) de interesse do usuário em qualquer hora e lugar: um agente de notícias (AgNoticias) localizado em um servidor é responsável pela divulgação de notícias no ambiente, enquanto um agente pessoal do usuário (AgUsuario) é responsável por armazenar as preferências do usuário. A interação entre esses agentes ocorre da seguinte maneira: (i) AgUsuario detecta a existência de um novo ambiente com notícias; (ii) AgNoticias solicita o perfil do usuário (AgUsuario só envia as informações relacionadas a notícia), filtra as notícias de acordo com as preferências do usuário e envia as notícias selecionadas; (iii) AgUsuario recebe as notícias e realiza um processo de tomada de decisão para verificar se realmente essas notícias atendem as preferências do usuário; por fim, (iv) AgUsuario apresenta as informações para o usuário. A implementação da aplicação foi realizada utilizando a API Java Micro Edition (JME), e para a sua execução foram utilizados dois celulares, sendo eles um Sony Ericsson W810i, contendo o ambiente Usuário, e, um Nokia 6230i, contendo o ambiente Notícias.

A aplicação *Guia Turístico*[1] consiste em auxiliar os usuários a encontrar determinados locais em um ambiente, de acordo com seus interesses. Para isso, considera-se que cada ambiente possui um robô guia, que terá o papel de recepcionar os visitantes e guiá-los no ambiente até o seu local de preferência. Primeiramente, o usuário deve informar os seus locais de interesse para o seu Agente Pessoal localizado no celular. Após o usuário entrar no ambiente, o *Agente Controlador do Ambiente* detectará o celular, e solicitará ao *Agente Pessoal* as suas preferências. De posse das preferências, o *Agente Controlador do Ambiente* envia uma tarefa para o *Agente Guia Turístico*, localizado no robô guia, para que este guie o usuário através do ambiente. Utilizando seu sensor de proximidade, o *Agente Guia Turístico* desvia-se de obstáculos e guia o usuário até a localização desejada. Para a realização do cenário, foram

[1] Para informações acesse: http://semanticore.pucrs.br/

utilizados um celular (Sony Ericsson W810i), dois computadores pessoais e um robô Lego Mindstorms NXT.

4. Considerações finais e trabalhos futuros

Este artigo apresentou as principais características do *framework* para desenvolvimento de sistemas multiagentes SemantiCore. Para mostrar como o SemantiCore pode ser utilizado no desenvolvimento de aplicações baseadas em agentes, uma aplicação foi descrita.

Vários trabalhos vêm sendo desenvolvidos com o objetivo de acrescentar novas funcionalidades ao modelo de agente e ao modelo de ambiente providos pelo SemantiCore. Entre os trabalhos que visam acrescentar novas funcionalidades aos agentes, foi citado o *framework* Ontowledge, cuja integração ao SemantiCore resulta em um novo componente para os agentes (componente Organizador).

Com a identificação de novos requisitos para agentes de software e o aperfeiçoamento das tecnologias existentes, o SemantiCore certamente passará por outras modificações e extensões. Em relação à arquitetura dos agentes, cita-se o desenvolvimento de uma extensão para a criação de agentes adaptativos.

O estudo de agentes adaptativos na Web Semântica é justificado, pois, sendo a Web Semântica um ambiente dinâmico e complexo, torna-se necessário a criação de agentes que se adaptem automaticamente às mudanças, cumprindo satisfatoriamente as metas definidas. O contexto de comércio eletrônico é um exemplo para esta proposta, no qual um agente conhecedor de um tipo de negócio, após cessar seus lucros, modificaria sua estrutura interna para trocar o seu ramo de atividade. Também, estuda-se a integração do SemantiCore a uma metodologia para a modelagem do comportamento interno de agentes, como o MASUP [Bastos e Ribeiro, 2005]. Isso permitiria, por exemplo, a geração automática de código SemantiCore, tendo-se como base a modelagem do sistema definida pelo desenvolvedor.

Referências

Bastos, R. M., Ribeiro, Marcelo Blois (2005) “MASUP: An Agent-Oriented Modeling Process for Information Systems”. In: Software Engineering for Multi-Agent Systems III: Research Issues and Practical Applications. Berlin: Springer-Verlag.

Berners-Lee, Tim; Hendler, James; Lassila, Ora (2001) “The Semantic Web” In: Scientific American, May 2001.

Bigus, J. P.; Bigus, J. (2001) “Constructing intelligent agents using java”. 2.ed. New York: John Wiley & Sons, 2001. 432p.

Blois, M.; Lucena, Carlos (2004) “Multi-Agent Systems and the Semantic Web – The SemanticCore Agent-based Abstraction Layer”. Proceedings of Sixth International Conference on Enterprise Information Systems ICEIS 2004. Porto: INSTICC, 2004. v.4. p.263 – 270.

Escobar, M.; Ries, L. H.; Lemke, A. P.; Ribeiro, M. B.. SemantiCore Mobile - Permitindo o Desenvolvimento de Aplicações. In: I Workshop on Pervasive and

Ubiquitous Computing - WPUC, 2007, Gramado. Anais WPUC. Porto Alegre: SBC Editora, 2007.

Escobar, Mauricio da Silva. “Um Modelo de integração de tecnologias e padrões para execução de aplicações baseadas em agentes para a Web Semântica”. 2009. Dissertação (Mestrado em Ciência da Computação) – Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2009.

Ferber, J. (1999) “Multi-agent systems: an introduction to distributed artificial Intelligence”. Oxford: Addison-Wesley, 1999. 528p.

FIPA. FIPA ACL Message Structure Specification. Desenvolvido por: The Foundation for Intelligent Physical Agents, 2000-2002. Capturado em <http://www.fipa.org/specs/fipa00061/>. Abril 2006.

Gudgin, M.; Hadley, M.; Mandelsohn, N.; Moreau, J. e Nielsen, H. (2002). “SOAP Version 1.2”, 2002-2006. Capturado em: <http://www.w3c.org/2000/xp/Group/2/06/LC/soap12-part1.html>. Abril 2006.

JADE - Java Agent DEvelopment Framework (2009). Desenvolvido por: Tilab. Capturado em: <http://jade.tilab.com/>. Novembro 2009.

Jason - A Java-based interpreter for an extended version of AgentSpeak. Capturado em: <a Java-based interpreter for an extended version of AgentSpeak> Abril 2009.

JENA “A Semantic Web Framework for Java”. Capturado em: <http://jena.sourceforge.net/>, Abril, 2009.

Jennings, N.; Wooldridge, M. (1999) “Agent-oriented software engineering”. In: European Workshop on Modelling Autonomous Agents in a Multi-Agent World, 9, 1999, Valência, Espanha. Anais. 1999, p. 1-7.

Luger, George F. (2002) “Inteligência Artificial: estruturação e estratégias para a solução de problemas complexos”. Bookman, 4º edição.

MadKit - a Multi-Agent Development Kit (2005). Capturado em: <http://www.madkit.org/>. Novembro 2005.

Odell, J. (2002) “Objects and agents compared. Journal of Object Technology”, v.1, n.1, p. 41-53.

OpenCybele (2005). Desenvolvido por: Intelligent Automotion Inc. Capturado em: <http://www.opencybele.org/>. Novembro 2005.

Poslad, S., Buckle, P., Hadingham, R. (2000) “FIPA-OS: the FIPA agent Platform available as Open Source”. In: International Conference on the Pratical Application of Intelligent Agents and Multiagent Technology (PAAM 2000), 5, 2000, Manchester, Inglaterra. Anais. 2000. p 355-368.

Ribeiro, Marcelo Blois (2002) “Web Life: Uma arquitetura para a implementação de sistemas multi-agentes para a Web”. Tese (doutorado) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 204p.