Universidade Federal de Juiz de Fora
Programa de Mestrado em Modelagem Computacional

Uma Arquitetura orientada a agentes para gerenciamento, busca e recuperação de artefatos científicos

Por
Luiz Felipe Carvalho Mendes

JUIZ DE FORA, MG - BRASIL
AGOSTO DE 2009

UMA ARQUITETURA ORIENTADA A AGENTES PARA GERENCIAMENTO, BUSCA E RECUPERAÇÃO DE ARTEFATOS CIENTÍFICOS

Luiz Felipe Carvalho Mendes

DISSERTAÇÃO SUBMETIDA AO PROGRAMA DE PÓS GRADUAÇÃO EM MODELAGEM COMPUTACIONAL DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.SC.) EM MODELAGEM COMPUTACIONAL.

Aprovada por:

Profa. Regina Maria Maciel Braga Villela, D.Sc.
(Orientadora)

Profa. Fernanda Cláudia Alves Campos, D.Sc.
(Co-Orientadora)

Prof. Ciro de Barros Barbosa, D.Sc.

Profa. Rosa Maria Esteves Moreira da Costa, D.Sc.

JUIZ DE FORA, MG - BRASIL
AGOSTO DE 2009

Mendes, Luiz Felipe Carvalho

Uma Arquitetura orientada a agentes para gerenciamento, busca e recuperação de artefatos científicos/ Luiz Felipe Carvalho Mendes; Orientador: Prof. Regina Maria Maciel Braga Villela. -- 2008.

82 f.

Dissertação (Mestrado em Modelagem Computacional) – Faculdade de Engenharia Civil, Universidade Federal de Juiz de Fora, Juiz de Fora, 2009.

1. e-Science. 2. Agentes 3. Web Semântica. 4. Modelagem Computacional. 5. Ontologias I. BRAGA, Regina Maciel. II. Uma Arquitetura orientada a agentes para gerenciamento, busca e recuperação de artefatos científicos

AGRADECIMENTOS

Agradeço a Deus por existir.

A família Carvalho, Aladim, Mendes por terem apoiado toda a minha caminhada até este momento. Em especial, minha mãe Adriana, meu padrasto Rogério, meus irmãos Luis Rogério Heitor que sempre fizeram de tudo para que eu pudesse continuar meus estudos e aguentaram o mau humor nos finais de semana em Ouro Branco. Meu pai Jorge Luiz que fez o que pôde também para que eu pudesse chegar até aqui. Meus avós Paulo e Jandira que estão mais para pais por todo o apoio recebido até hoje e sempre na torcida pelos meus resultados. E minha namorada Janaína, uma heroína em aguentar tempos tão turbulentos ao meu lado, seu amor foi reconfortante em vários momentos e sem ele não seria possível terminar esta etapa. Meus sogros Cida e Ruy com palavras de sabedoria em momentos difíceis e não poderia esquecer os tios Edgar e Cristina, pois sem eles minha história em Juiz de Fora não haveria sequer começado!

Meus amigos de Ouro Branco, Vitinho, Topo, Pedro, Rodrigo, Moita, Ju, Túlio e Patrícia que nos churrascos e peladas sempre podíamos esquecer a correria deste curso. Aos amigos da UFJF e agregados que se tornaram bem mais que colegas de sala e "namoradas de amigo", Trolhão, Paty, Tato, Mel, Motoboy, Samuca, Débora e Claudinho. Como dizem, amigos de verdade são os irmãos que nós escolhemos ter!

À SertaOggi, mais precisamente meus chefes Bráz e Ricardo e o ex-chefe Siegmar. Pois sem a tolerância deles nada disso teria sido possível. Obrigado pela compreensão e confiança durante todo o tempo, agora é hora de contarem comigo para o que precisar! Rumo ao "céu de brigadeiro"! Ao Horastor, claro, sempre quebrando "árvores"pra mim na empresa mesmo sendo uma mala sem alça!

Agradeço também a minha orientadora Regina, pela paciência e dedicação e aos trancos, dados no momento certo para que eu pudesse continuar o curso. Não posso esquecer também da co-orientação da professora Fernanda, com grande contribuição para a conclusão deste trabalho.

Aos ex-alunos das disciplinas que lecionei que hoje alguns viraram amigos e que sempre nos ensinam algo de novo transformando a função de professor em algo ainda mais nobre e humano.

Meus agradecimentos a todos os professores e alunos, que comigo, formaram a segunda turma do Mestrado em Modelagem Computacional da Universidade Federal de Juiz de Fora.

Por fim, meus agradecimentos a todos aqueles que, direta ou indiretamente, contribuíram para o desenvolvimento desse trabalho.

Resumo da Dissertação apresentada à UFJF como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UMA ARQUITETURA ORIENTADA A AGENTES PARA GERENCIAMENTO, BUSCA E RECUPERAÇÃO DE ARTEFATOS CIENTÍFICOS

Luiz Felipe Carvalho Mendes

Agosto/2009

Orientador : Regina Maria Maciel Braga Villela

A Computação Científica é uma área multidisciplinar que extrapola o uso docomputador como simples máquina onde pesquisadores escrevem textos,apresentações ou guardam resultados de experimentos e análises. Com aevolução das pesquisas, e o uso cada vez maior de recursos computacionais aplicados aos experimentos e simulações realizados pelos grupos de pesquisa, é notório que os grupos de pesquisa se encontram em um novo patamar da computação científica representado através do conceito de e-Science. A e-Science está intimamente ligada ao processo de modelagem computacional. A importância da modelagem computacional na computação científica está no fato de que através dela é possível estudar o desempenho de grandes sistemas computacionais sem que seja necessário implementá-los na realidade.

Este trabalho tem como objetivo propor uma arquitetura de busca e recuperação de artefatos científicos, gerados dentro dos projetos de pesquisa relacionados à computação científica, baseada em agentes inteligentes. A arquitetura proposta é chamada SASAgent e conta ainda, com um estudo de caso para analisar a viabilidade da mesma. Resultados preliminares sugerem que a arquitetura proposta é promissora para atender a requisitos encontrados em projetos de e-Science.

Abstract of Dissertation presented to UFJF as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

UMA ARQUITETURA ORIENTADA A AGENTES PARA GERENCIAMENTO, BUSCA E RECUPERAÇÃO DE ARTEFATOS CIENTÍFICOS

Luiz Felipe Carvalho Mendes

Agosto/2009

Supervisor: Regina Maria Maciel Braga Villela

Scientific Computing is a multidisciplinary field that goes beyond the use of computer as machine where researchers write simple texts, presentations or store analysis and results of their experiments. Considering the increase use of computer resources applied to experiments and simulations, it is notable that the research groups are at a new level of scientific computing, well represented through the domain of e-Science. The e-Science concept is closely related to the process of computational modeling. The significance of computational modeling in scientific computing is the fact that it is possible to study the performance of larger scientific models without the necessity of implement them in reality.

This work aims to propose an architecture, based on intelligent agents, to search and recover scientific artifacts, generated within research projects related to scientific computing,. The SASAgent architecture is proposed and a case study is detailed in order to examine the feasibility of SASAgent. Preliminary results suggest that the proposed architecture is promising to meet requirements found in e-Science projects.

Lista de Figuras

2.1 Arquitetura do myGrid (CHRIS *et al.*, 2003) 11

2.2 Arquitetura do Globus Toolkit (FOSTER, 2005) 15

2.3 Arquitetura em três camadas vista como serviços (ROURE *et al.*, 2003) 18

2.4 Workflow do cenário especificado (ROURE *et al.*, 2003) 20

3.1 Interação do agente e o ambiente (RUSSEL e NORVIG, 1995) 27

3.2 Mapeamento dos conceitos de agentes para o framework ASYNC (SARDINHA, 2005) 36

3.3 Visão geral da arquitetura do MathWS Broker (MATOS e *et al.*, 2007) 40

3.4 Arquitetura FUSION (KOURTESIS e PARASKAKIS, 2008) 41

3.5 Arquitetura do SemWIQ (LANGEGGER *et al.*, 2008) 45

3.6 Inclusão e integração de uma ontologia (RODRIGUES, 2005) 47

3.7 Arquitetura do matching semântico (RODRIGUES, 2005) 48

4.1 Plataformas e containeres no JADE 4.1 52

4.2 As fases da metodologia MaSE (WOOD e DELOACH, 2000) 57

4.3 Arquitetura Reativa (WOOD e DELOACH, 2000) 59

4.4 Diagrama de interações entre agentes (PADGHAM e WINIKOFF, 2003) 64

5.1 Diagrama de casos de uso Geral 81

5.2 Diagrama de casos de uso: específico 1 81
5.3 Diagrama de casos de uso: específico 2 82
5.4 Diagrama de casos de uso: específico 3 82
5.5 Diagrama de casos de uso: específico 4 83
5.6 Diagrama de Organização Geral 83
5.7 Diagrama de Organização Facilitador 84
5.8 Modelo de Agentes UserAgent 85
5.9 Modelo de Agentes QueryServiceAgent 85
5.10 Modelo de Agentes InvokerAgent 86
5.11 Modelo de Agentes GatewayAgent 86
5.12 Modelo de Tarefas e Objetivos Nível Macro 87
5.13 Modelo de Tarefas e Objetivos de Gerenciamento 88
5.14 Modelo de Tarefas e Objetivos - Atender às necessidades 89
5.15 Modelo de Tarefas e Objetivos Prover Artefatos Científicos 90
5.16 Interação para o registro nas "páginas amarelas" 91
5.17 Troca de mensagens entre o DFAgent e UserAgent 92
5.18 Interação para o registro das "páginas brancas" 92
5.19 Troca de mensagens entre o AMSAgent e o UserAgent 93
5.20 Interações para a busca dos artefatos científicos 93
5.21 Troca de mensagens entre UAgent, GAgent e QSAgent 94
5.22 Interação entre o UAgent e QSAgent para a busca dos artefatos . . . 94
5.23 Troca de mensagens entre o UAgent e QSAgent 95
5.24 Interação entre os agentes: UserAgent e InvokerAgent 96
5.25 Troca de mensagens entre os agentes: UserAgent e InvokerAgent . . . 97

5.26 Modelo de ambiente da plataforma 98

5.27 Arquitetura da plataforma JADE (BELLIFEMINE, 2003) 98

5.28 Arquitetura geral da plataforma *SASAgent* 99

5.29 Interação dos agentes com a base semântica 101

5.30 OWL-S no seu mais alto nível de abstração (OWLS, 2004) 102

5.31 Uso de serviços web semânticos na plataforma 104

6.1 Nível topo da ontologia SASOntology 108

6.2 Classe ScientificArtifact e subclasses da ontologia SASOntology . . . 109

6.3 Classe Publication da ontologia SASOntology 110

6.4 Classe ConceptualModel da ontologia SASOntology 110

6.5 Classe ComputationalTool da ontologia SASOntology 111

6.6 Classe ContextConcepts da ontologia SASOntology 111

6.7 Classes InterestArea e Organization da ontologia SASOntology 112

6.8 Importação do arquivo WSDL para a geração do esqueleto em OWL-S 114

6.9 Esqueleto em OWL-S do serviço Web OLS após leitura do WSDL . . 115

6.10 Associação de instâncias de pesquisadores da ontologia *SASOntology* ao *ServiceProfile* 116

6.11 Componentes do SOR (LU *et al.*, 2007) 117

6.12 Diagrama de Sequência para a publicação dos serviços Web 118

6.13 Formulário de publicação de serviços Web 119

6.14 Consulta em SPARQL para localizar os serviços Web 120

6.15 Tela de resultados da busca por serviços Web 121

6.16 Atuação do InvokerAgent exibindo as operações disponíveis do serviço 122

6.17 Atuação do InvokerAgent exibindo os parâmetros da operação selecionada . . . 123

6.18 Diagrama de sequência para a busca e invocação dos serviços Web . . 124

6.19 Formulário de inserção de artigos utilizando anotações semânticas . . 125

6.20 Formulário de inserção de *profiles* . . . 126

6.21 Formulário de inserção de *profiles* . . . 127

6.22 Consulta em SPARQL para o profile do João . . . 127

6.23 Tela de resultado da pesquisa por artigos . . . 128

Lista de Tabelas

6.1 Algumas operações disponíveis do serviço Web OLS 113

6.2 Operações disponíveis do serviço Web PICR 117

Lista de Abreviaturas

DAME	Distributed Aircraft Maintenance Environment
NSF	National Science Foundation
RDF	Resource Description Framework
RDQL	RDF Data Query Language
SQL	Structured Query Language
OQL	Object Query Language
LHC	Large Hadron Collider
CERN	European Organization for Nuclear Research
WLCG	Worldwide LHC Computing Grid
GT	Globus Toolkit
IA	Inteligência Artificial
BDI	Beliefs, Desires and Intentions
UML	Unified Modelling Language
OWL	Ontology Web Language
OWLS	Semantic Markup for Web Services
UDDI	Universal Description, Discovery and Integration
WSDL	Web Service Description Language
SAWSDL	Semantic Annotations Web Service Description Language
SPARQL	SPARQL Protocol and RDF Query Language
AUML	Agent UML
JADE	Java Agent DEvelopment Framework
FIPA	The Foundation for Intelligent Physical Agents
RUP	Ration Unified Process
SMA	Sistemas Multiagentes
MAS	Multi-Agent Systems
SASAgent	Scientific Artifact Search Agent
SASOntology	Scientific Artifact Search Ontology

Sumário

1 Introdução **2**

1.1 Motivação 2

1.2 Objetivos 3

1.3 Estrutura do Trabalho 6

2 *e-Science* **7**

2.1 Conceitos Básicos 7

2.2 Projetos de e-Science 10

2.2.1 myGrid 10

2.2.2 WorldWide LHC Computing Grid (WLCG) 12

2.3 Sistemas de Workflow para Suporte a e-Science 13

2.3.1 Globus Toolkit 13

2.3.2 Taverna 15

2.4 O Futuro da e-Science 17

2.5 Uso de Serviços Web e Agentes em e-Science 22

3 Sistemas Baseados em Agentes **25**

3.1 Conceitos Básicos 25

3.2 Taxonomia de Agentes 28

3.3 Modelos para a Construção de Agentes e seus Comportamentos . . . 29
3.3.1 Modelo BDI - *Beliefs, Desires and Intentions* 29
3.3.2 Modelo 5C 30
3.4 Problemas e Limitações do Uso de Agentes 32
3.5 Trabalhos Relacionados 34
3.5.1 *Frameworks* para Construção de Aplicações Semânticas Baseadas em Agentes 34
3.5.1.1 *SemantiCore* 34
3.5.1.2 ASYNC - Um *Framework* para Construção de Agentes Inteligentes 35
3.5.1.3 Considerações 37
3.5.2 Semântica em Serviços 38
3.5.2.1 *MathWS* 38
3.5.2.2 *Fusion* 39
3.5.2.3 OWLS-MX 41
3.5.2.4 Considerações 43
3.5.3 Infra-Estrutura para Apoio a *e-Science* 44
3.5.3.1 *Semantic Web Integrator and Query Engine* (SemWIQ) 44
3.5.3.2 *Matching* Semântico de Recursos Computacionais em Ambientes de Grade com Múltiplas Ontologias . . . 46
3.5.3.3 Considerações 48

4 Desenvolvimento de Sistemas Multiagentes 49

4.1 Fundamentação 49
4.2 Metodologias de Desenvolvimento de Sistemas Multiagentes 53
4.2.1 MASUP - *Multi-Agent Systems Unified Process* 53

4.2.1.1 Requisitos 53
4.2.1.2 Análise 54
4.2.1.3 Projeto 55
4.2.1.4 Considerações 55
4.2.2 MaSE - *Multiagent Systems Engineering* 56
4.2.2.1 Captura dos objetivos 58
4.2.2.2 Aplicação dos Diagramas de Casos de Uso 58
4.2.2.3 Refinamento dos Papéis 58
4.2.2.4 Criação das Classes dos Agentes 58
4.2.2.5 Diagrama de Classes de Comunicação 59
4.2.2.6 Arquitetura dos Agentes 59
4.2.2.7 Diagramas de Distribuição 59
4.2.2.8 Considerações 60
4.2.3 Tropos 60
4.2.3.1 Requisitos Iniciais 61
4.2.3.2. Requisitos Finais 61
4.2.3.3 Projeto Arquitetural 61
4.2.3.4 Projeto Detalhado 62
4.2.3.5 Considerações 62
4.2.4 Prometeus 62
4.2.4.1 Especificação de Sistema 63
4.2.4.2 Projeto Arquitetural 64
4.2.4.3 Projeto Detalhado 65
4.2.4.4 Considerações 66

4.2.5 MAS-CommonKADS 66
4.2.5.1 Fase de Conceituação 67
4.2.5.2 Fase de Análise 67
4.2.5.3 Fase de Projeto e Protótipo 68
4.2.5.4 Considerações 68
4.2.6 INGENIAS 68
4.2.6.1 Modelo de Agentes 70
4.2.6.2 Modelo de Interação 70
4.2.6.3 Modelo de Metas e Objetivos 71
4.2.6.4 Modelo de Organização 72
4.2.6.5 Modelo de Ambiente 73
4.2.6.6 Considerações 74
4.2.7 Comparação entre as Metodologias 74

5 *SASAgent*: Arquitetura Orientada a Agentes para Gerenciamento, Busca e Recuperação de Artefatos Científicos 77

5.1 Introdução 77
5.2 Projeto e Especificação 79
5.2.1 Diagramas de Caso de Uso 80
5.2.2 Modelo de Organização 81
5.2.3 Modelo de Agentes 84
5.2.4 Modelo de Tarefas e Objetivos 86
5.2.5 Modelo de Interação 89
5.2.6 Modelo de Ambiente 92
5.3 Arquitetura 94

5.3.1 O Framework para Agentes 95
5.3.2 Base de Dados Semântica 101
5.3.3 Uso de Serviços Web Semânticos no *SASAgent* 102
5.3.4 Integração Agentes-Semântica-Usuários 104

6 Estudo de Caso: Busca por Artefatos no Domínio de e-Science 106

6.1 A Ontologia *SASOntology* 107
6.1.1 Objetivos da *SASOntology* 107
6.1.2 Visão Geral da *SASOntology* 108
6.1.2.1 Classe *ScientificArtifact* 108
6.1.2.2 Classe *ContextConcepts* 110
6.2 Cenário Um - Busca por Serviços Web Semânticos 112
6.2.1 Anotações Semânticas dos Serviços Web em OWL-S 113
6.2.2 Publicação dos Serviços Web 117
6.2.3 Busca e Invocação dos Serviços 118
6.3 Cenário Dois - Busca por Publicações Científicas Relacionadas ao Domínio de *e-Science* 120
6.3.1 Publicação dos Artigos 121
6.3.2 Busca e Visualização dos Artigos 121
6.4 Discussões 123

7 Considerações Finais 129

7.1 Trabalhos Futuros 131

A Ontologia *SASOntology* 133

Capítulo 1

Introdução

1.1 Motivação

A Computação Científica é uma área multidisciplinar que extrapola o uso do computador como simples máquina onde pesquisadores escrevem textos, apresentações ou guardam seus resultados de experimentos e análises. Com a evolução das pesquisas, e o uso cada vez maior de recursos computacionais aplicados aos experimentos e simulações realizados pelos grupos de pesquisa, é notório que os grupos de pesquisa se encontram em um novo patamar da computação científica muito bem representado através do conceito de e-Science. Este conceito pode ser aplicado em qualquer área que utilize recursos computacionais em apoio às suas pesquisas como Bioinformática, Física e Engenharias.

Alguns autores consideram o conceito de e-Science bem abrangente, sendo encarado como todo um aparato tecnológico e conceitual em apoio às pesquisas científicas (WALTON e BARKER, 2004), (PEARCE, 2007). Enquanto outras vertentes definem e-Science como um conceito mais simples, sendo quase um sinônimo de grids computacionais aplicados às pesquisas científicas como o myGrid[1] , DAME (Distributed Aircraft Maintenance Environment)[2] (JACKSON *et al.*, 2005), AstroGrid[3] e Gridcast[4] citados em (PEARCE, 2007).

[1] http://www.mygrid.org.uk
[2] http://www.cs.york.ac.uk/dame/
[3] http://www.astrogrid.org/
[4] http://www.besc.ac.uk/

A e-Science está intimamente ligada ao processo de modelagem computacional. Ela envolve o uso de algoritmos já conhecidos ou que foram desenvolvidos juntamente com técnicas de simulação, manipulação de dados, mineração de dados, dentre outros processos, em que o modelo é um dos resultados da própria pesquisa, sendo interpretado como um processo que filtra, transforma, aglutina e gera dados (SBC, 2006). A importância da modelagem computacional na computação científica está no fato de que através dela é possível estudar o desempenho de grandes sistemas computacionais sem que seja necessário implementá-los na realidade ou ainda, no âmbito industrial, executando experimentos virtuais em detrimento da necessidade da construção de uma instalação física para tal (SBC, 2006).

Os produtos gerados além dos próprios modelos computacionais e matemáticos são algoritmos e ferramentas computacionais (SBC, 2006) que são desenvolvidos no intuito de resolver os problemas enfrentados. O grande problema é que muitas vezes é difícil a reutilização destes algoritmos e/ou ferramentas, pois estão amarrados a uma determinada tecnologia como FORTRAN (J3, 2006) ou C (KERNIGHAN e RITCHIE, 1988). Além disso, podem até mesmo não "existir"para a comunidade científica se o produto gerado não for exposto de alguma maneira que possa ser aproveitado. Os dados gerados por estas pesquisas precisam de mecanismos que facilitem a interação, a interoperabilidade e o compartilhamento entre grupos de pesquisa que estejam trabalhando cooperativamente ou mesmo que tenham como domínio a mesma área de interesse, aproveitando ao máximo os recursos que a Web pode proporcionar no âmbito do trabalho colaborativo.

1.2 Objetivos

Pelo cenário descrito na seção anterior é possível observar o ambiente extremamente heterogêneo no qual os grupos de pesquisa, sejam eles de mesmo domínio ou não, precisam trabalhar, apesar de todos os recursos computacionais que possuem à disposição. Neste ambiente heterogêneo, são gerados ainda os artefatos científicos, que em uma definição própria, são os produtos finais gerados pela pesquisa científica. São gerados artefatos que vão desde os mais simples como: publicações científicas e relatórios técnicos, até artefatos mais complexos, como modelos matemáticos, grandes bases de dados (seqüenciamento de DNA, por exemplo), algoritmos e programas

em linguagens estruturadas como C ou serviços Web.

A proposta de desenvolvimento da arquitetura **SASAgent**, apresentada nesta dissertação, visa realizar a busca semântica, recuperação e utilização de artefatos científicos, através do uso de anotações semânticas sobre os mesmos e o uso de mecanismos inteligentes que atuarão sobre essas anotações. Para que este objetivo seja alcançado, são utilizados dois recursos importantes: ontologias e agentes aplicados ao conceito de e-Science.

Sistemas baseados em agentes, sistemas multi-agentes e agentes inteligentes são sistemas que necessitam de uma autonomia maior, que muitas vezes precisam "tomar decisões"por conta própria e que, principalmente, queiram depender cada vez menos do usuário para encontrar e processar informações que sejam importantes para realizar determinada tarefa, exigem técnicas mais sofisticadas para serem desenvolvidos e implementados.

O papel dos agentes na proposta apresentada é representar os usuários com seus respectivos perfis para que possam encontrar o conteúdo desejado dentre os artefatos científicos de acordo com os parâmetros fornecidos na construção de cada perfil. Além disto, os agentes tem papel importante na busca dos artefatos e o mais importante, na invocação de serviços Web semânticos que estarão disponíveis.

(GRUBER, 1993) define ontologia como sendo uma especificação explícita de uma conceitualização. É uma descrição de conceitos e relacionamentos que podem existir para uma determinada comunidade. Em um refinamento da definição apresentada acima, (GUARINO e GIARETTA, 1995) definem ontologia como um registro parcial e explícito de uma conceitualização de uma linguagem utilizada para uma base de conhecimento, conceitualização esta que depende do propósito desejado para a ontologia.

Na proposta apresentada, as ontologias têm também um papel importante em todo o processo, pois é através delas que os artefatos científicos serão anotados semanticamente, sendo possível buscá-los e recuperá-los além de descrever semanticamente serviços Web que poderão ser invocados a qualquer momento.

Desta forma, podemos destacar como principal objetivo da **SASAgent** (1) o gerenciamento, (2) busca e (3) recuperação de artefatos científicos gerados pelos grupos de pesquisa de domínios relacionados à modelagem computacional, tais como

Engenharia, Física, Química e Bioinformática, através do uso de agentes inteligentes e uma base de conhecimento semântica a partir do uso de ontologias.

Este objetivo principal pode ser decomposto em um conjunto de objetivos secundários:

- Implementação de uma camada que se constitui de um sistema multiagentes que servirá como mediadora entre a base de conhecimento, os artefatos científicos e os pesquisadores que desejam utilizar a arquitetura. Este sistema consistirá das seguintes características:
  - Agentes responsáveis pela interface com o usuário final;
  - Agentes responsáveis por inserir, buscar e recuperar os artefatos científicos e conseqüentemente acessar a base de dados semântica;
  - Agentes responsáveis por interagir diretamente com alguns artefatos científicos como os Web Services;
  - Agentes responsáveis pelo monitoramento dos artefatos;
- Existência de uma base de conhecimento que através do uso de ontologias represente o conhecimento necessário para dar suporte ao gerenciamento dos artefatos permitindo buscas, cadastro e monitoramento dos mesmos;
- Independência do domínio de aplicação dos grupos de pesquisa, ou seja, a **SASAgent** deve servir a qualquer grupo de pesquisa que gere artefatos científicos que atendam à demanda de áreas como Engenharia, Bioinformática e Física, dentre outras;
- Dar suporte a diferentes papéis que o usuário final (cientista) do sistema pode assumir: ora como consumidor de artefatos, ora como provedor dos mesmos, permitindo ainda, que o usuário tenha a capacidade de montar seu próprio perfil, com as principais preferências em relação às instituições de pesquisa, artefatos, áreas de interesse, dentre outros pontos.

1.3 Estrutura do Trabalho

Esta dissertação está organizada em sete capítulos. O Capítulo 2 apresenta o contexto da e-Science, fazendo uma revisão dos conceitos associados. O Capítulo 3 fornece uma visão geral do conceito de agentes, principais técnicas de modelagem e trabalhos relacionados a esta dissertação.

O Capítulo 4 apresenta diversas metodologias de desenvolvimento de software baseado em agentes, com ênfase na escolhida para a proposta apresentada: INGENIAS (SANZ, 2002).

O Capítulo 5 descreve a arquitetura do sistema multi-agente SASAgent, responsável pela busca, recuperação e utilização dos artefatos científicos. Incluem-se nesta arquitetura os agentes existentes, suas interações, comportamento e uso de recursos externos. O Capítulo 6 apresenta o estudo de caso que implementa uma parte do problema da busca e recuperação dos artefatos científicos e apresenta, ainda, a ontologia SASOntology que serve de apoio à arquitetura proposta. Finalmente, o Capítulo 7 apresenta as conclusões e sugere trabalhos futuros.

Capítulo 2

e-Science

2.1 Conceitos Básicos

Experimentação científica é uma das formas usadas pelos cientistas para investigar fenômenos, tendo como objetivo adquirir novos conhecimentos ou corrigir e integrar conhecimentos previamente estabelecidos. Nas últimas décadas, com o avanço da computação, os experimentos científicos passaram a utilizar ferramentas computacionais para facilitar a sua execução. A utilização dessas ferramentas computacionais foi motivada também pelo próprio aumento da complexidade dos experimentos que, em alguns casos, eram inviáveis de serem executados sem algum apoio computacional (MARINHO *et al.*, 2009). Desta forma, esforços como, por exemplo, do Office of Cyberinfrastructure pertencente a NSF (National Science Foundation)[1] (NSF, 2006) e do UK e-Science Programme[2] (RCUK, 2007), para o desenvolvimento de uma infraestrutura de apoio a execução destes experimentos científicos ganharam força no contexto das comunidades científicas.

Estes pesquisadores consideram que uma infraestrutura para o apoio a ciência deve englobar as seguintes características: a) o processamento de grandes massas de dados, b) o compartilhamento de informações e a coordenação deste processo, c) a simulação de experimentos através da criação, gerenciamento e compartilhamento de workflows científicos, d) o uso de serviços (Web Services) e aplicações desenvol-

[1] http://www.nsf.gov

[2] http://www.rcuk.ac.uk/escience

vidos por diversos grupos de pesquisas, e) uso de metodologias que possam auxiliar no desenvolvimento de software científico para que possam ser reutilizados e que possam ter qualidade para facilitar a manutenção, f) a correção e a escalabilidade do software. As pesquisas relacionadas a especificação e implementação destas características fazem parte do contexto de e-Science (WALTON e BARKER, 2004) (PEARCE, 2007) (FOX e WALKER, 2003).

Atualmente, os grupos de pesquisa não podem mais trabalhar de maneira isolada, uma vez que suas pesquisas são realizadas em domínios correlatos e, por este motivo, o compartilhamento de dados e aplicações também é necessário para a continuidade de suas pesquisas. As interações entre os grupos de pesquisa vão além de uma simples troca de e-mails. Ela evolui para o compartilhamento de ferramentas (sejam elas aplicações isoladas ou serviços Web), intercâmbio de dados científicos sobre os experimentos realizados e até mesmo os processos que foram utilizados para chegar aos resultados obtidos (workflows científicos).

No entanto, questões importantes como: como implementar e coordenar o compartilhamento em ambientes heterogêneos, ou como disponibilizar aplicações para que outros pesquisadores possam utilizá-las, além de como desenvolver aplicações para que possam ser compartilhadas ou ainda como conseguir processar grandes massas de dados e informações dentro de um tempo hábil de resposta são objetos de pesquisa.

Os agentes de software têm um papel importante neste contexto de e-Science. Eles podem assumir o papel de aplicações que podem ser compartilhadas e desta forma precisam de características de agentes como, por exemplo, mobilidade, como também podem fazer o papel de coordenadores, na descoberta, busca e execução de serviços Web.

Os grids (BERSTIS, 2002) (FOSTER, 2002) também podem contribuir com a infraestrutura para permitir tanto o processamento maciço de dados quanto o processamento distribuído de tarefas, mesmo entre grupos de pesquisas distantes geograficamente, como também podem prover um conjunto de serviços de um determinado grupo de pesquisa para a utilização por terceiros.

Considere, por exemplo, um grupo de pesquisa que está no Brasil, e trabalha com modelos matemáticos para representar seus problemas. Um destes modelos

(traduzidos para alguma linguagem de programação) precisa de um método que foi implementado por outro grupo de pesquisa. Este método pode ser implementado como um serviço web, sendo que o pesquisador no Brasil só conhece seus parâmetros de entrada e de saída. Deste modo, o pesquisador no Brasil não precisa desenvolver a sua própria versão do método, podendo esta versão ser suscetível a erros (pois não foi ele que criou o método), e ainda distribui o processamento das suas tarefas fazendo uso de um grid computacional.

Neste contexto, outro componente importante desta infraestrutura seria um repositório de serviços Web, onde cada serviço poderia receber uma descrição completa sobre sua proveniência, métodos disponíveis e resultado esperado, e, em um ambiente onde agentes fossem utilizados, anotações semânticas relacionando o serviço a algum conceito em uma ontologia. A busca por estes serviços seria realizada por agentes inteligentes de modo que os mesmos pudessem localizar os serviços mais adequados à requisição de um pesquisador, realizando a invocação, a coordenação e a interação deste serviço e outros que por ventura possam depender dele e compor um workflow.

Alguns autores consideram o conceito e-Science de maneira mais abrangente como todo um aparato tecnológico e conceitual em apoio às pesquisas científicas (WALTON e BARKER, 2004) (PEARCE, 2007).

(FOX e WALKER, 2003) citam de outros trabalhos, que a e-Science é realizada pelo uso rotineiro de recursos computacionais distribuídos pelos cientistas dos projetos. Em uma visão mais ampla, e-Science pode ser usada como uma metodologia por cientistas de maneira individual, que se torna mais efetiva quando é usada para permitir a colaboração global, envolvendo um grande número de pessoas e recursos em larga-escala. A e-Science torna os trabalhos colaborativos mais produtivos, quebrando as barreiras para a comunicação e conseqüentemente para as interações entre os envolvidos, fazendo ainda com que recursos sejam mais acessíveis tanto por pessoas quanto para outros sistemas computacionais (FOX e WALKER, 2003).

(DE ROURE *et al.*, 2002) destaca que e-Science oferece uma visão promissora de como as tecnologias dos computadores e de comunicação podem dar suporte e melhorar os processos científicos, permitindo assim, gerar, analisar, compartilhar e discutir seus pontos de vista, experimentos e resultados de uma maneira mais

efetiva.

2.2 Projetos de e-Science

2.2.1 myGrid

O myGrid[3] é um projeto de e-Science aplicado a bioinformática liderado pela Universidade de Manchester no Reino Unido (PEARCE, 2007). Tem como objetivo principal desenvolver um workbench virtual para que possa ajudar os cientistas a utilizar recursos compartilhados, como por exemplo, análise de dados do projeto Genoma. Este workbench visa dar suporte ao processo científico de investigação experimental, acúmulo de evidências e assimilação de resultados; uso de informações compartilhadas por outros cientistas; colaboração científica, entre outros.

O myGrid utiliza uma conjunto de ontologias para representar os metadados. Basicamente, no contexto do myGrid, uma ontologia provê uma taxonomia de termos, conceitos, relacionamentos e restrições sobre determinado domínio. Neste caso em específico, os termos e conceitos estão relacionados com objetos como serviços, workflows e dados. As ontologias são descritas em OWL (OWL, 2004).

Por utilizar ontologias e ainda anotações semânticas, o myGrid pode ser caracterizado como um grid semântico (ZHUGE, 2005) (SURE *et al.*, 2005). Os serviços e workflows disponíveis no sistema são descobertos semanticamente. Estes mesmos serviços e workflows são publicados em um registro próprio, federado, que é baseado em RDF permitindo assim uma maior flexibilidade e provê suporte a metadados adicionais. A busca por estes recursos é feita através de informações como localização, custo, qualidade do serviço, disponibilidade entre outras, e utiliza a linguagem RDQL, uma linguagem de consulta própria para o RDF (FOX e WALKER, 2003).

O myGrid permite ao usuário construir descrições semânticas dos serviços em termos de alguma ontologia e anexá-las a registros de serviços novos ou já existentes. Há ainda serviços de notificação que dão suporte ao monitoramento do progresso do workflow e do registro de serviços e notificações sobre atualizações do banco de dados (FOX e WALKER, 2003). Analisando as características semânticas do myGRID,

[3]http://www.mygrid.org.uk/

podemos considerar que o suporte semântico ainda é incipiente. O conceito de ontologia utilizado não inclui características importantes como uso de regras lógicas.

O acesso a dados pode ser feito via o componente de processamento de consultas que provê uma interface que dá suporte a consultas distribuídas sobre qualquer fonte de dados compatível com SQL e ODMG OQL (FOX e WALKER, 2003).

Na figura 2.1 pode ser visto um esquema da arquitetura do myGrid.

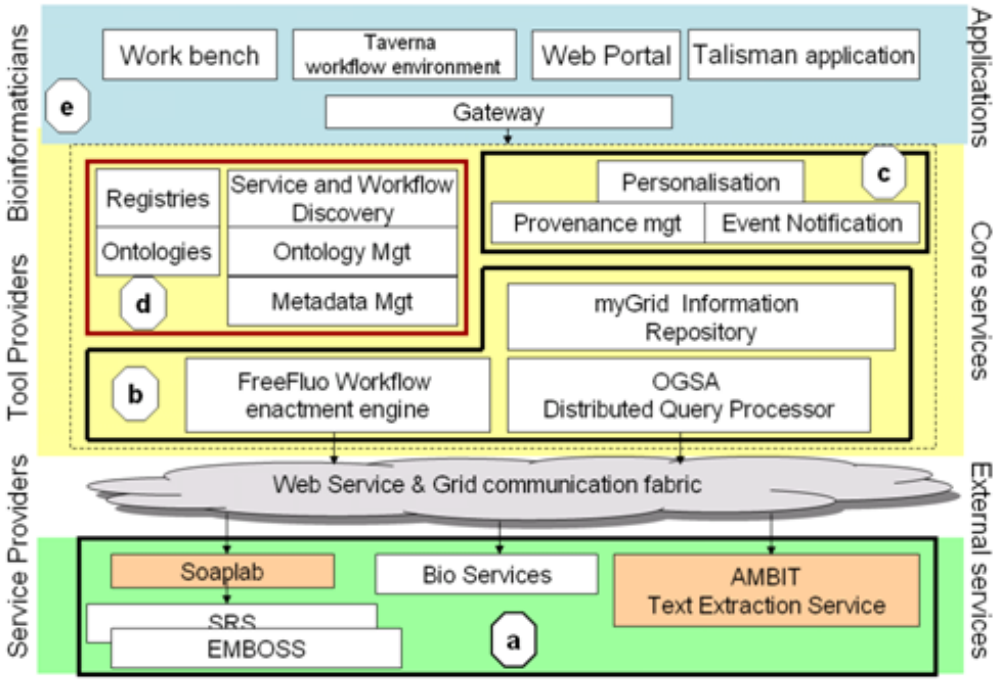

Figura 2.1: Arquitetura do myGrid (CHRIS *et al.*, 2003)

Os agentes, dentro do contexto de grids, podem assumir o papel dos serviços que são oferecidos (AVILA-ROSAS *et al.*, 2002), podem fazer o papel de personalizar, comunicar e fazer negociações como descrito em (MOREAU *et al.*, 2003). Em (MOREAU *et al.*, 2003) é descrito que o agente pode representar um usuário do sistema myGrid. Ele pode cuidar da personalização do uso de um serviço ou da composição de vários deles. Suponha que o usuário do sistema utilize um serviço disponível, pois sabe que a geração dos seus resultados é quase sempre muito correta, portanto, esta preferência por este serviço pode ser vista pelo agente no momento que controla o workflow de serviços que serão executados.

Além disto, o agente pode ainda ser o ponto de comunicação entre os serviços disponíveis dentro do myGrid e o usuário final. Ele é uma espécie de intermediário

que recebe as requisições originárias dos serviços para que o usuário possa entrar com os dados ou pode ainda receber, por exemplo, notificações sobre mudanças nos bancos de dados remotos. Essas mensagens serão enviadas aos usuários somente quando eles estiverem disponíveis dentro do grid.

2.2.2 WorldWide LHC Computing Grid (WLCG)

O acelerador de partículas LHC (Large Hadron Collider) localizado no CERN (European Organization for Nuclear Research) próximo a Genebra, Suiça, é hoje o maior instrumento científico do mundo. Quando entrar em operação de maneira regular, serão produzidos por volta de 15 Petabytes (15 milhões de Gigabytes) de dados anualmente que centenas de cientistas espalhados pelo mundo poderão acessar e analisar.

Para prover uma infraestrutura que permita o armazenamento, o acesso e a análise de toda essa informação foi criado o projeto WLCG[4] (Worldwide LHC Computing Grid) (http://lcg.web.cern.ch/LCG/) para que toda a comunidade científica que estuda a física de alta energia possa ter acesso ao projeto LHC.

Os dados recolhidos do experimento são divididos em uma arquitetura de quatro camadas por todo o mundo. A camada 0 (Tier-0) é responsável pelo backup das informações coletadas que ficam gravadas em uma fita dentro do próprio CERN. Após um processamento inicial, os dados são distribuídos para uma série de centros pertencentes a camada 1 (Tier-1), centros estes com computadores de alto desempenho e capacidade de armazenamento que provê suporte ao grid. Os dados providos pela camada 1 tornam-se disponíveis para a camada 2 (Tier-2), que são centros de pesquisa que podem armazenar uma quantidade menor de dados e oferecer poder computacional suficiente para a realização de tarefas de análise bastante específicas. A última camada, a camada 3 (Tier-3) é composta por clusters locais de algum laboratório de uma universidade ou até mesmo um computador pessoal de onde os cientistas, consumidores finais dos dados, têm acesso a toda a informação coletada pelo LHC.

A estrutura física do projeto compreende recursos computacionais de mais de

[4] http://lcg.web.cern.ch/LCG

140 centros em 33 países, o que representa um poder de processamento de 100.000 CPUs para processar, analisar e armazenar os dados produzidos pelo LHC. Além disto, os dados estarão disponíveis para 5000 cientistas espalhados por mais de 500 institutos de pesquisa no mundo que participam do experimento.

O middleware necessário para prover toda esta infraestrutura precisa, além de permitir acesso aos recursos do grid, controlar e monitorar sua performance. Ele precisa garantir um balanceamento de carga otimizado entre os diversos centros, para isto precisa monitorar os recursos disponíveis de cada um: os dados que estão sendo armazenados e a situação corrente das redes que conectam toda a camada. Além disso, há duas questões importantes relacionadas a segurança: autorização e autenticação, que precisam estar cobertas pelo middleware do grid. A infraestrutura, portanto, utiliza vários projetos como: Globus (FOSTER, 2005), Condor (THAIN *et al.*, 2005), Virtual Data Toolkit[5] e o gLite Toolkit (BURKE *et al.*, 2009) pertencente ao projeto EGEE[6] (pan-European Grid infrastructure).

2.3 Sistemas de Workflow para Suporte a e-Science

2.3.1 Globus Toolkit

A ferramenta é um dos produtos que foram gerados através da iniciativa The Globus Alliance[7] , um conglomerado de universidades, centros de pesquisa e pesquisadores que trabalham no desenvolvimento de ferramentas de código aberto para computação distribuída, grids¸ arquiteturas orientadas à serviços e e-Science. Como exemplo das entidades envolvidas podemos citar: University of Chicago, University of Edinburgh, National Center for Supercomputing Applications (NCSA), Northern Illinois University, Royal Institute of Technology, UnivaUD e University of Southern California.

O Globus Toolkit (GT) é desenvolvido desde a década de 90 para dar suporte ao desenvolvimento de aplicações computacionais orientadas a serviços. Os componentes principais, como qualquer framework, visam atender a requisitos como:

[5]http://vdt.cs.wisc.edu
[6]http://www.eu-egee.org/
[7]http://www.globus.org/

segurança, acesso a determinado recurso, gerenciamento dos recursos, transferência de dados, descoberta de recursos dentre outros. Todo este framework é usado para desenvolver aplicações distribuídas e prover infraestruturas para grids computacionais em larga escala (FOSTER, 2005).

Dentre os diversos recursos podemos citar de maneira mais detalhada as seguintes capacidades do framework:

- Um conjunto de implementações de serviços que incluem gerenciamento de recursos, transferência de dados, descoberta de serviços e conceitos relacionados;
- Ferramentas para construir novos serviços que podem ser desenvolvidos em Java, C ou Python;
- Uma infraestrutura de segurança poderosa e padronizada para autenticação e autorização dentro da infraestrutura;
- APIs em diferentes linguagens e programas em linha de comando para acessar vários destes serviços;
- Documentação detalhada destes vários componentes, interfaces e como eles podem ser usados no desenvolvimento de aplicações.

A figura 2.2 ilustra como a arquitetura da ferramenta é organizada para atender aos requisitos das aplicações desenvolvidas sobre o framework. Um primeiro conjunto de serviços implementados representa as funções que podemos chamar de estruturais, que compõem a base da infraestrutura do kit. Entre eles podemos destacar: GRAM (gerenciador de execução), GridFTP, RFT, OGSA-DAI (todos responsáveis pelo acesso e transporte dos dados), RLS, DRS (ambos responsáveis pelo gerenciamento de replicação), Index, Trigger, WebMDS (encarregados pelo monitoramento e busca), MyProxy, Delegation, SimpleCA (gerenciamento de credenciamento) e GTCP (gerenciamento de instrumentos) (FOSTER, 2005). A maioria dos serviços citados foi desenvolvida em Java, mas alguns foram implementados em outras linguagens como C ou utilizam outros protocolos.

Três repositórios podem ser usados para hospedar os serviços desenvolvidos pelos usuários escritos em Java, Python e C respectivamente. Estes repositórios, além de

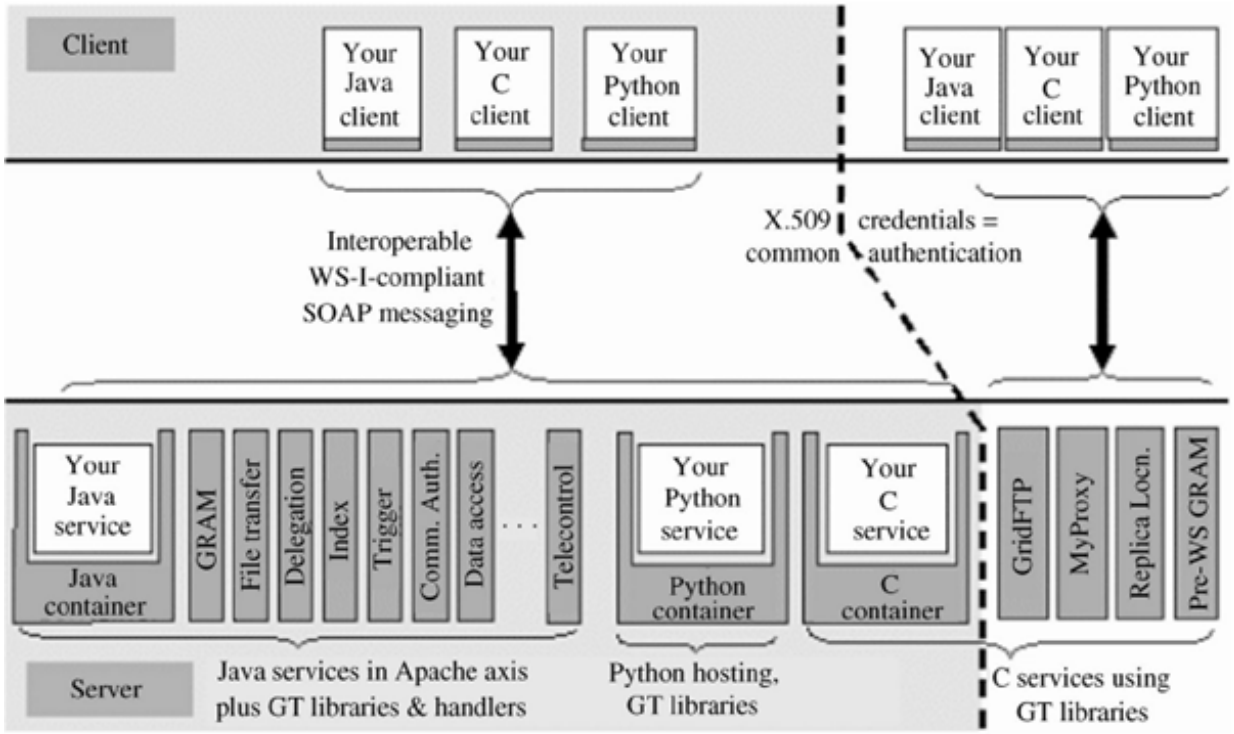

Figura 2.2: Arquitetura do Globus Toolkit (FOSTER, 2005)

hospedar os serviços, provêem implementações de segurança, gerenciamento, descoberta, gerenciamento de estados e outros mecanismos frequentemente necessários na construção de serviços. Estes repositórios estendem ambientes já conhecidos de código livre que hospedam serviços para que possam oferecer novas funcionalidades: Web Service Resource Framework (WSRF) (FOSTER *et al.*, 2005), WS-Notification e WS-Security (WELCH, 2004).

Além disto, um conjunto de bibliotecas disponíveis permite que aplicações cliente desenvolvidas em Java, C ou Python, invoquem operações dos serviços que se encontram nos repositórios do framework, sejam eles da própria infraestrutura ou dos serviços disponibilizados por outros usuários (FOSTER, 2005).

2.3.2 Taverna

Para contextualizar o desenvolvimento da ferramenta Taverna (OINN *et al.*, 2004) é importante entender o domínio na qual ela está inserida. Basicamente, o Taverna é uma ferramenta para composição e execução de workflows científicos

relacionados à bioinformática.

Estes workflows estão diretamente relacionados aos chamados experimentos in silico, que são experimentos realizados através do uso coordenado de ferramentas computacionais e repositórios de dados para que uma hipótese possa ser testada, derivar uma conclusão ou buscar por padrões (STEVENS *et al.*, 2003). Grande parte destas ferramentas computacionais está sendo desenvolvida como serviços Web, portanto, a criação, o gerenciamento e execução destes workflows científicos tentam orquestrar o funcionamento destes serviços, com suas respectivas entradas e saídas de dados e ainda o acesso às bases de informações muitas vezes distribuídas e escritas em formatos diferentes (OINN *et al.*, 2004).

A ferramenta foi desenvolvida a partir da necessidade identificada dentro do projeto myGrid, apresentado anteriormente, que é prover um middleware de serviços para dar suporte a experimentos in silico relacionados a bioinformática que utilizam grandes fontes de dados e recursos computacionais distribuídos.

O Taverna, portanto, provê uma interface gráfica para a criação e execução de workflows que representam experimentos in silico em bioinformática. Para a ferramenta, o conceito de workflow é um grafo de processadores, onde cada processador transforma um conjunto de dados de entrada em um novo conjunto de dados de saída (OINN *et al.*, 2004). Todos estes workflows são representados utilizando a linguagem Scufl.

Scufl (Simple conceptual unified flow language) é uma linguagem conceitual de alto nível, baseada em XML, onde cada passo de processamento de um workflow representa uma tarefa atômica. Existem três entidades principais para a construção dos workflows:

- **Processadores:** Um processador é uma transformação que toma um conjunto de dados de entrada e produz um conjunto de dados de saída. Ele sempre possui um nome que o identifica e um conjunto de portas de entrada e saída. Além disto, durante a execução de um workflow, cada processador possui um status corrente da sua execução: inicializando, aguardando, rodando, terminado, falho ou abortado;
- **Links de dados:** Realizam a mediação do fluxo de dados entre uma fonte de

dados e o seu consumidor. A fonte pode ser uma saída de um processador ou uma entrada do workflow. O consumidor destas informações pode ser a porta de entrada de um processador ou ainda a saída de um workflow;

- **Restrições de coordenação:** uma restrição de coordenação une dois processadores e controla suas execuções. Este nível de controle é exigido quando um há um processo onde cada passo precisa ser executado em uma determinada ordem e ainda não há dependência direta de dados entre eles.

2.4 O Futuro da e-Science

Apesar de todo o suporte que a e-Science é capaz de prover para a comunidade científica, muitos desafios ainda estão para serem resolvidos. Principalmente no que diz respeito à facilidade de uso dos artefatos gerados, automação de tarefas, personalização de funcionalidades, entre outras características.

Similarmente ao conceito de Web Semântica (BERNERS-LEE *et al.*, 2001), onde o uso de conteúdos semanticamente ricos, estensíveis pelos computadores, buscas mais inteligentes e utilização de agentes para automação de várias tarefas, existe também o conceito de infraestruturas de apoio às ciências semânticas, que visam apoiar a e-Science de uma maneira mais rica. (ROURE *et al.*, 2003) propõe uma arquitetura para uma futura infraestrutura de e-Science baseada em serviços, onde vários stakeholders de áreas distintas do processo científico, na figura de agentes de software, provêem serviços uns aos outros, sobre vários "contratos"possíveis. O foco principal deste tipo de arquitetura está na aquisição do conhecimento vindo dos stakeholders e como este conhecimento passa a ser utilizado dentro do ambiente. A infraestrutura computacional nesta arquitetura pode ser dividida basicamente em três camadas conceituais (ROURE *et al.*, 2003):

- **Dados:** Esta camada é a responsável em como recursos computacionais são alocados, agendados e executados e ainda o modo que os dados trafegam entre os vários recursos de processamento. Essa camada deve ter a capacidade de lidar com um grande volume de dados, prover redes muito rápidas para o trafego de informações e disponibilizar os vários recursos disponíveis;

- **Informação:** Basicamente, esta camada lida com o modo que as informações são representadas, armazenadas, acessadas, compartilhadas e mantidas. Aqui, a informação tem um significado, e não é meramente uma seqüência de bytes sem sentido. Por exemplo, a caracterização de um valor inteiro pode ser associada como representação de uma temperatura de um processo reativo, uma string pode ser o nome de algum indivíduo, etc;

- **Conhecimento:** Aqui, o foco é o modo com o conhecimento é adquirido, usado, exibido, publicado e mantido para dar suporte aos cientistas para atingir seus objetivos e metas particulares. O conhecimento nesta camada é entendido como uma informação aplicada para atingir determinado objetivo, resolver um problema ou tomar uma decisão. Por exemplo, o reconhecimento por parte de um operador de uma usina que, no contexto que ele está inserido, a temperatura atingida para uma reação requer a parada do processo.

Um dos motivos para esta divisão em camadas conceituais é que se pode adequar a arquitetura do sistema de acordo com a necessidade do domínio. Podem haver situações onde o mais importante seja a manipulação de grandes volumes de dados, enquanto em outros casos, serviços relacionados ao conhecimento podem ser importantes. A divisão em camadas também auxilia nas fases de análise e projeto do desenvolvimento. E é importante lembrar ainda que, em uma visão orientada a serviços, estes serviços podem ser relacionados a todas as camadas. A figura 2.3 representa uma visão orientada a serviços para uma arquitetura de camadas.

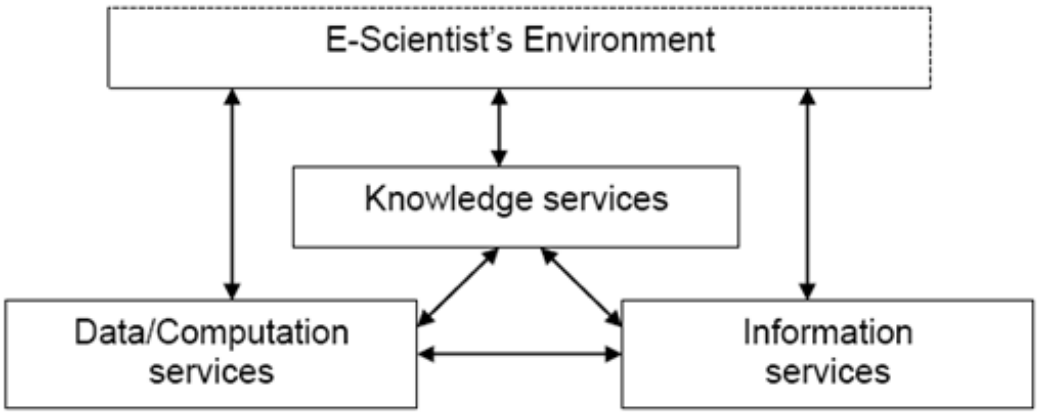

Figura 2.3: Arquitetura em três camadas vista como serviços (ROURE *et al.*, 2003)

(ROURE *et al.*, 2003) apresenta ainda um cenário para elucidar o que seria o

futuro da e-Science com esta arquitetura com foco no conhecimento e na semântica. Não é um cenário fictício e abrange, em uma escala mais restrita, situações que já acontecem na comunidade científica de modo geral. Neste cenário, podemos imaginar que temos uma amostra para determinado experimento que é identificado por um ID (identificador). O técnico responsável pelo experimento entra no sistema e as informações sobre aquela amostra aparecem. Algumas configurações para o experimento são confirmadas e a amostra vai para um analisador. Esse analisador automaticamente é executado e a saída desta análise é gravada juntamente com os parâmetros utilizados nas simulações e análises. A análise feita automaticamente chama atenção de um cientista da companhia, ou de outros grupos de pesquisa, que investigam os resultados do experimento a partir do seu escritório. Ele refaz o experimento com aquela amostra utilizando outros parâmetros próprios a partir de outros experimentos semelhantes e ainda grava em vídeo todo o experimento. A partir do resultado obtido, ele ainda revê o experimento pela gravação que foi feita e aperta um botão de "consulta"para que o sistema sumarize os resultados daquela amostra e envia automaticamente para outros cientistas da área. Os cientistas interagem dizendo que aqueles resultados obtidos parecem ser únicos. O cientista que conseguiu os resultados publica os dados do experimento para que outros cientistas possam acessar. E estes dados, caso sejam julgados relevantes, podem ser inseridos em uma grande base de dados internacional. Outros cientistas que não interagiriam automaticamente, mas, por terem um perfil de pesquisa semelhante ao experimento realizado, são avisados sobre a disponibilização de todas as informações do experimento.

Neste cenário, que pode ser descrito na figura 2.4, existem vários requisitos aplicados a uma infraestrutura de e-Science.

As características do cenário que se enquadram em uma infraestrutura de e-Science são (ROURE *et al.*, 2003):

- **Armazenamento:** O sistema deve ter a capacidade de armazenar e processar um grande volume de dados dentro de um tempo hábil;
- **Propriedade:** Diferentes stakeholders precisam ter controle sobre seus conteúdos ao mesmo tempo permitindo o acesso de terceiros às suas informações em uma política de termos e usabilidade;

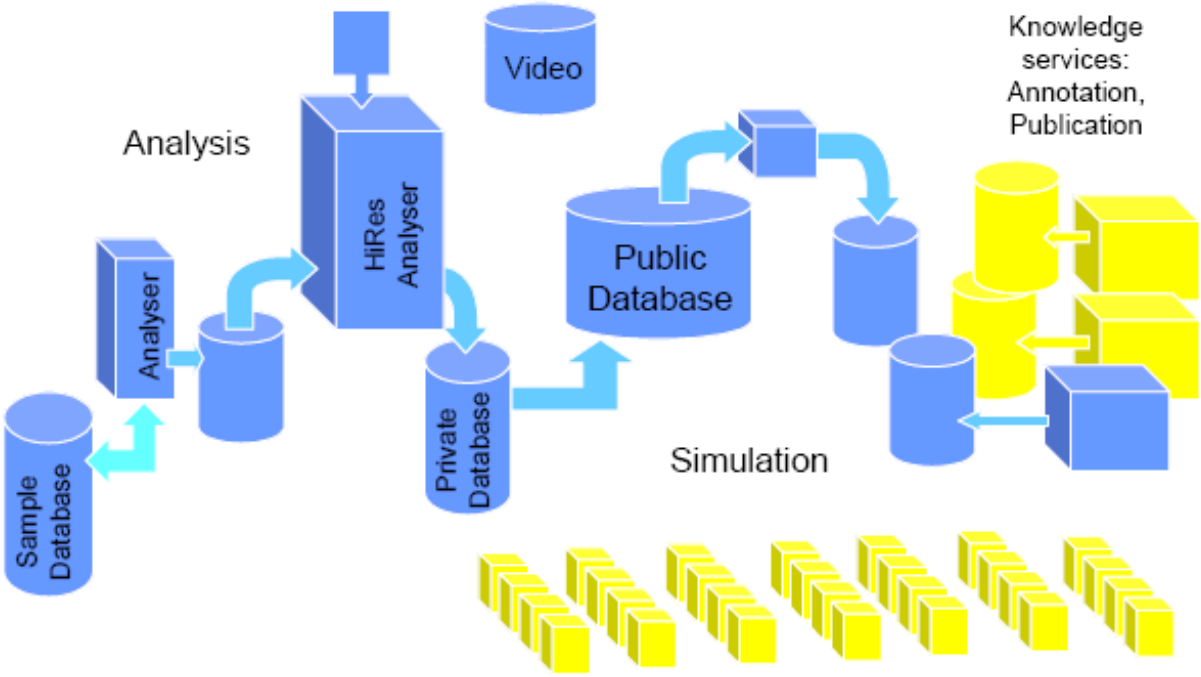

Figura 2.4: Workflow do cenário especificado (ROURE *et al.*, 2003)

- **Proveniência:** Informações suficientes devem ser armazenadas para um futuro reuso ou análise dos dados;
- **Transparência:** O acesso do usuário aos recursos deve ser transparente independente da localização dos dados, do usuário e dos recursos disponíveis no grid;
- **Comunidades:** Usuários devem ter possibilidade de formar comunidades da prática com critérios de aceitação de membros e regras de operação;
- **Fusion:** Interoperabilidade entre os dados armazenados, vindos de fontes distintas para atender a demanda do usuário;
- **Serviços de conferência:** Possibilidade de visualizar os artefatos que estão sob discussão e interação com outros membros;
- **Anotações:** A partir da publicação dos dados de maneira pública e seguindo as regras de termos e uso, anotações podem ser feitas no experimento para poder enriquecer seu conteúdo;
- **Workflow:** Possibilidade de descrição dos processos para possibilitar automatização de tarefas;

- **Notificação:** Envio de notificações aos usuários a partir de experimentos feitos que casem com seus interesses;
- **Suporte à decisão:** O sistema deve ser capaz de fazer sugestões de melhorias para a tarefa em questão a partir das informações coletadas, armazenadas e analisadas;
- **Reserva de recursos:** Há a necessidade de facilitar a reserva de utilização de algum recurso. Isso se aplica aos equipamentos de laboratório, serviços de colaboração e agendamento de recursos para a simulação;
- **Segurança:** Deve haver autenticação, criptografia e requisitos de privacidade, com múltiplas organizações envolvidas e todos estes recursos devem ter o mínimo de intervenção manual;
- **Confiabilidade:** O sistema aparenta ser confiável, mas na prática ele precisa lidar com falhas e exceções em todos os níveis, inclusive modelado no workflow;
- **Vídeo:** Vídeos ao vivo e armazenados tem papel importante, especialmente quando são enriquecidos por metaconteúdos associados de maneira temporal;
- **Laboratório moderno:** Por exemplo, o equipamento detecta a amostra (via código de barra ou outro identificador), o cientista pode utilizar dispositivos portáteis para acessar as informações e visualizar a amostra;
- **Conhecimento:** Serviços relacionados ao conhecimento são muito importantes no processo de e-Science. Encontrar artigos, encontrar pessoas, encontrar experimentos anteriores (que podem envolver até mesmo inferência), anotações na análise feita e configuração do laboratório virtual de acordo com o perfil do cientista;
- **Escalabilidade:** O sistema deve suportar o crescimento evolucionário do conteúdo e incluir novas técnicas de processamento quando disponíveis. Além do crescimento da infraestrutura física, capacidade de processamento, largura de banda, armazenamento e complexidade dos relacionamentos entre as informações.

Muitas destas questões apresentadas acima já estão sendo discutidas e resultados interessantes estão sendo apresentados (GARCÍA-SANCHÉZ *et al.*, 2009). No

entanto, muito trabalho ainda é necessário para que todas estas questões sejam tratadas de maneira satisfatória. Particularmente, no contexto de dados semânticos, pesquisas relacionadas a ontologias, serviços web semânticos e agentes de software parecem promissoras.

2.5 Uso de Serviços Web e Agentes em e-Science

SOA (Service-Oriented Architecture) (GROUP, 2007) é uma proposta de arquitetura baseada em serviços. Um serviço é uma representação lógica de alguma atividade de negócio que vai ser realizada e possui uma saída específica como, por exemplo: verificar cartão de crédito de um cliente, prover dados sobre o tempo, dentre outros possíveis serviços. Estes serviços ainda podem ser compostos com outros serviços, além de sua implementação ser transparente para os clientes que utilizam o serviço em questão (HARDING, 2007).

No contexto de e-Science, a idéia principal é que tenhamos nesta arquitetura entidades que provêem serviços umas para as outras sobre diversos tipos de "contratos". O interessante na visão de uma arquitetura de serviços - define-se serviços nesse caso como uma caracterização abstrata e encapsulada de algum conteúdo ou capacidade de processamento - é que há uma modelagem conceitual de relacionamentos entre essas entidades, serviços e conteúdo, evitando dessa forma que sejam construídos workflows de maneira ad-hoc já que essa modelagem é independente de tecnologia (ROURE *et al.*, 2003).

Potenciais serviços no cenário de e-Science são, por exemplo: reconhecimento automático da amostra por parte do equipamento e sua configuração automática, a gravação das informações do experimento em bases de dados, a localização de recursos computacionais para dar suporte ao experimento, encontrar cientistas que tenham publicado trabalhos na mesma área, notificação automática por parte do sistema quando algum resultado particular é encontrado (via SMS, e-mail, etc).

Em uma arquitetura de serviços também é importante a questão de propriedade dos serviços oferecidos. Portanto, cada serviço tem um proprietário, que define quem pode usá-lo, se há cobrança pelo seu uso e como ele se relaciona com outros serviços. Na outra ponta, temos quem usa o serviço e para reger essa utilização do serviço

é implementado um "contrato"entre ambas as partes (ROURE *et al.*, 2003). Cada proprietário do serviço terá um ou mais agentes agindo sobre o serviço. O papel dos agentes nessa situação está relacionado ao gerenciamento do acesso ao serviço e ao cumprimento do contrato digital entre o proprietário e o consumidor do serviço. Além disso, os agentes podem agir na localização de serviços e manipulação dos resultados para a apresentação e recebimento.

A interação entre os agentes deve ser alcançada através de uma interoperabilidade semântica entre eles, já que os agentes podem possuir modelos individuais de comportamento, utilizar diferentes linguagens de comunicação e ainda serem implementados em tecnologias diferentes. Talvez essa interação e, principalmente, esta interoperabilidade possam ser alcançadas através do uso de ontologias que modelem os relacionamentos entre os diversos tipos de agentes, realizem inferências para derivar outros relacionamentos a partir dos já existentes, entre outras funcionalidades.

Os agentes podem interagir entre si das mais variadas formas. Desde uma simples troca de informações até cooperação e coordenação para realizar determinada ação e atingir algum objetivo traçado. (ROURE *et al.*, 2003) descreve que as interações dos agentes podem ser de duas naturezas distintas. Primeiro, os agentes estão relacionados aos contratos entre os provedores e consumidores dos serviços. É possível até mesmo atingir alguma forma de negociação entre os agentes de maneira automatizada desde que os agentes possuam a característica de serem autônomos. Para que essa negociação possa ser projetada três questões precisam ser levadas em consideração: o protocolo de negociação (o conjunto de regras que governam a interação), o objeto de negociação (características que serão analisadas para a negociação como preço, termos e condições, qualidade, etc) e modelos de tomada de decisão para os agentes (modelagem das ações que devem ser tomadas na negociação para que o agente possa atingir seu objetivo).

Quando sistemas com agentes, que possuem autonomia, são pró-ativos e, inteligentes, interagem entre si, estes podem se deparar com um alto grau de incerteza na tomada de decisão. O que pode vir a ser uma solução para este tipo de agentes é a definição de um conjunto de leis que regem estas interações. Em (GATTI *et al.*, 2006) é citado um framework que é utilizado na criação dessa governança por leis nas interações entre os agentes. O framework utilizado é o XMLaw que juntamente com seu software M-Law responsável pela aplicação e execução das leis, controlam

toda a definição das leis e sua aplicação na prática.

Dentro do cenário descrito em (ROURE *et al.*, 2003) alguns agentes importantes foram identificados: agentes que reportam resultados, agentes que monitoram as análises experimentais, agentes que configuram parâmetros para o experimento, agentes que notificam e buscam cientistas do mesmo ramo, entre outros. Em uma alternativa mais interessante, os agentes seriam vistos como agentes de domínio, isto é, com funcionalidades já agregadas ao domínio que ele pertence, podendo desta forma reutilizar o conhecimento adquirido quando novas aplicações requisitarem.

Por exemplo, agentes do domínio de comércio eletrônico poderiam ter funcionalidades comuns como negociação automatizada segundo uma parametrização, ou verificação do limite de cartão de crédito do cliente, entre outros. O uso de ontologias como o descrito em (SILVA *et al.*, 2006) para modelar agentes de domínio é uma boa alternativa. Neste trabalho, as ontologias modelam o domínio do comércio eletrônico de modo geral, conjuntamente com as leis que devem reger a negociação. Ao mesmo tempo, estas ontologias devem ser flexíveis de forma que informações mais específicas daquelas entidades que estão negociando no papel dos agentes possam ser adicionadas. Uma representação semelhante poderia ser modelada para o domínio de e-Science, incluindo conceitos relacionados a publicações, autores, grupos de trabalho envolvidos, de um determinado domínio de aplicação.

Capítulo 3

Sistemas Baseados em Agentes

3.1 Conceitos Básicos

Sistemas baseados em agentes, sistemas multi-agentes, agentes inteligentes são várias facetas possíveis e, atualmente, muito pesquisadas no desenvolvimento de software, com características que vão além das necessidades convencionais de uma interface com usuário, um banco de dados relacional e uma interação com o usuário via uma interface.

Existem necessidades de áreas específicas da ciência de desenvolver sistemas capazes de tomar decisões de maneira autônoma, baseados em informações coletadas do ambiente no qual ele está inserido. Informações estas coletadas por sensores de temperatura, de ar, pressão ou movimento, para citar alguns exemplos. Ou ainda, que estes sistemas possam auxiliar o homem quando ele é o responsável em tomar alguma decisão baseada em um processamento realizado.

Na grande maioria das situações, o que é fácil de identificar é a necessidade de desenvolvimento de softwares que possam depender cada vez menos da interação com o usuário, quando este possui um objetivo claro de encontrar e processar informações que sejam importantes para realizar uma determinada tarefa, ou, ainda, que consigam adequar seu funcionamento em função do perfil que o usuário possui.

Novas necessidades foram trazidas também pelo advento da Web Semântica (BERNERS-LEE *et al.*, 2001) e de e-Science introduzido por John Taylor em um

encontro sobre o tema em Londres, no ano de 2001. Taylor percebeu que diversas áreas da ciência estavam ficando cada vez mais próximas em modos de colaboração e interdisciplinaridade (HEY e TREFETHEN, 2005). Essas novas áreas levaram a uma nova estrutura que precisa ser desenvolvida para auxiliar no acesso a bases de dados heterogêneas e distribuídas, busca por conteúdos científicos, ferramentas computacionais desenvolvidas por grupos de pesquisa geograficamente distribuídos entre outros artefatos, que auxiliem no desenvolvimento de novos experimentos científicos, simulações de experimentos já existentes e, ainda, a possibilidade de composição dos mesmos.

Todos estes conceitos podem ser inseridos em um contexto importante chamado Modelagem Computacional. Em ((BRESCIANI e *et al.*, 2004) é dito que cada vez mais o software precisa operar em diferentes plataformas, heterogêneas, sem recompilações, com o mínimo de informações sobre o ambiente operacional e os usuários. São aplicações robustas, com autonomia e que exigem pró-atividade por parte do software desenvolvido. Para que estas aplicações possam ser classificadas como agentes, devem possuir ainda outras características como: **habilidade social**, **reatividade** e **pró-atividade** além da já citada autonomia. Há também alguns atributos extras requeridos em situações mais específicas onde o agente está sendo utilizado como: **mobilidade** e **rationale** (RUSSEL e NORVIG, 1995), (RAO *et al.*, 1995).

Existem na literatura muitas definições do que são agentes. No contexto da área de Inteligência Artificial (IA) o conceito de agentes inteligentes é muito explorado (RUSSEL e NORVIG, 1995), (RAO *et al.*, 1995). No contexto de Web-Semântica são sistemas multi-agentes (ESCOBAR e LEMKE, 2006) (SHARON e *et al.*, 2002) com federações e/ou sociedades de agentes que interagem entre si para realizar determinada tarefa, ou mais simplesmente, agentes de software. A diferença básica entre um agente inteligente e um agente de software simples é que os agentes inteligentes são capazes de raciocinar sobre os elementos percebidos e escolher a melhor forma de agir de acordo com essa percepção (TAVARES, 2004).

Ainda em (TAVARES, 2004), um agente pode funcionar de maneira autônoma e isolado na máquina do usuário ou em uma federação de agentes, construindo sistemas multi-agentes. A idéia fundamental, independente do tipo de agente, é a delegação. Ao software é delegada alguma autonomia de decisão. O agente ou um grupo de

agentes recebe uma ou mais tarefas a serem executadas e através do reconhecimento do ambiente e da percepção do problema vai em busca desta solução, realizando ações compatíveis de maneira autônoma deixando ao usuário somente o trabalho de requisitar a solução e aguardar por ela.

Em (SANTOS, 2006) algumas características importantes dos agentes são citadas como: **adaptação** (capacidade de aprendizado e através dele melhorar suas habilidades na resolução dos problemas que ele se propõe a resolver); **aprendizado** (se o agente é capaz de aprender sobre o ambiente que ele se encontra, ou sobre o(s) usuário(s) e outros agentes que interagem com ele); **composição** (composição de agentes no ambiente) e **mobilidade** (habilidade de migrar de um host para outro, perceber novos ambientes).

Já em (RUSSEL e NORVIG, 1995), o agente é um objeto que pode ser visto como um perceptor dentro do ambiente que está inserido através de sensores e ainda um ator que age através de seus atuadores. Os agentes possuem seus próprios "efetores"para agir no seu ambiente. Além disto, há o conceito de racionalidade para agentes, onde um agente racional é classificado como tal, quando realiza uma ação que seja considerada correta dentro dos seus objetivos e metas. A figura 3.1 ilustra a atuação do agente no ambiente.

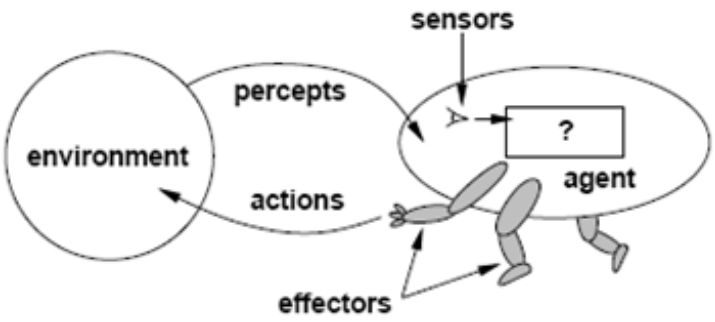

Figura 3.1: Interação do agente e o ambiente (RUSSEL e NORVIG, 1995)

O conceito de "fazer a coisa certa"é a ação realizada pelo agente que teve mais sucesso ao ser executada. Porém, é necessário decidir "como"e "quando"o sucesso do agente é avaliado. O "como"basicamente é a partir da ação executada pelo agente o quão próximo ele ficou do que era esperado. Já o "quando", é o sucesso avaliado dentro de um período de tempo, por exemplo, a medida do que foi alcançado dentro deste período.

Para (RUSSEL e NORVIG, 1995) um agente racional ideal se traduz em: "Para cada seqüência possível de percepção, um agente racional ideal deve realizar a ação que maximizará a sua medida de performance, baseada em evidências providas pela sua percepção juntamente com o conhecimento nativo que o agente possui".

Outras características pertinentes aos agentes foram citadas em (TAVARES, 2004) como:

- Agentes podem operar sem controle direto de humanos ou de outros agentes. Em outras palavras, são autônomos;
- Agentes podem agir em sociedade com humanos e com outros agentes, eles se comunicam;
- Agentes podem reagir a várias formas de estimulação dos domínios, eles são reativos;
- Agentes podem por si mesmo tomar decisões para ajustar-se a metas definidas, eles são pró-ativos.

3.2 Taxonomia de Agentes

Há na literatura, muitas classificações dos agentes baseados nos seus propósitos, na sua arquitetura, tecnologia, etc. Em (RUSSEL e NORVIG, 1995) são apresentados diferentes papéis e aliada a cada papel a complexidade que um agente pode assumir.

Os agentes são separados em quatro tipos: **agentes de reflexo simples** (é o mais simples dos agentes, utilizando um conjunto simples de regras do tipo condição-ação como base para tomada de decisão), **agentes que rastreiam o mundo** (requerem uma representação do mundo em estados possíveis para que os agentes possam tomar suas decisões), **agentes baseados em metas** (agentes são baseados em metas que devem ser alcançadas baseadas em uma seqüência de ações que devem ser tomadas) e **agentes baseados na utilidade** (onde as ações também são baseadas em metas, porém, contém heurísticas que possibilitam o agente "raciocinar"sobre qual caminho seguir caso haja mais de um possível para atingir determinada meta).

Já (NWANA, 1996), classifica os agentes em sete classes distintas: **agentes colaborativos** (ênfase na cooperação e autonomia), **agentes de interface** (ênfase em autonomia e aprendizado), **agentes móveis** (componentes de software capazes de navegar em uma rede como na Web interagindo com servidores externos, buscando informações para seus usuários, etc), **agentes de informação/Internet** (utilizados em sites de busca), **agentes reativos** (agem sobre o princípio de estímulo/resposta respondendo as mudanças do ambiente o qual está inserido), **agentes híbridos** (possuem mais de uma característica em um único agente) e **agentes inteligentes**.

3.3 Modelos para a Construção de Agentes e seus Comportamentos

Na literatura podem ser encontrados diversos modelos para o desenvolvimento e documentação de agentes, entretanto, para fins dessa dissertação, destacamos a seguir dois modelos: BDI (RAO *et al.*, 1995) e 5C (VAN AART, 2005). O modelo BDI é o modelo clássico de desenvolvimento de agentes inteligentes e norteia o desenvolvimento de aplicações desde quando foi idealizado e deu origem, inclusive, a diversas metodologias exclusivas para o desenvolvimento de software baseados em agentes. O modelo 5C já possui aplicações comerciais desenvolvidas baseadas em seus conceitos e é, em parte, semelhante ao BDI.

3.3.1 Modelo BDI - *Beliefs, Desires and Intentions*

O modelo BDI - Beliefs, Desires and Intentions - é um modelo bastante popular para agentes inteligentes (RAO *et al.*, 1995). Possui sua base na Filosofia e oferece uma teoria lógica que define atitudes mentais de Beliefs (Fatos), Desire (Metas) e Intentions (Seqüência de ações escolhida para se chegar à meta) (WINIKOFF e *et al.*, 2001).

Existem, inclusive, várias metodologias de desenvolvimento de sistemas baseados em agentes que modelam os conceitos (ou pelo menos parte deles) do modelo BDI: Tropos (BRESCIANI e *et al.*, 2004), AUML - Agents Unified Modeling Language (ODELL *et al.*, 2000), MESSAGE (EVANS *et al.*, 2001), Prometeus (PADGHAM e

WINIKOFF, 2003), entre outras.

Os conceitos centrais do modelo BDI são:

- ***Beliefs:*** Informações sobre o ambiente; informativo;
- ***Desires:*** Metas para serem alcançadas, possivelmente com prioridades associadas a cada objetivo; motivacional;
- ***Intentions:*** A estratégia corrente escolhida para alcançar os objetivos; deliberativo;
- ***Plans:*** Meios de atingir estados futuros do ambiente. Intuitivamente, os planos de ações são uma especificação abstrata tanto dos meios para atingir certos desejos (desires) quanto as opções disponíveis para os agentes. Cada plano tem um corpo descrevendo as ações primitivas ou sub-objetivos que precisam ser atingidos para o plano obter sucesso e o plano ainda possui uma condição de invocação que especifica os eventos que disparam a execução do plano e por último, existe uma condição de contexto que especifica em qual situação o plano é aplicável.

3.3.2 Modelo 5C

O modelo 5C é um framework conceitual baseado na generalização dos diferentes tipos de capacidades dos agentes que podem ser encontrados na literatura. Este modelo foi criado com o intuito de servir como um guia para o projeto de desenvolvimento de aplicações baseadas em agentes inteligentes tanto protótipos quanto comerciais (VAN AART, 2005).

O desenvolvimento destas aplicações mostrou que as restrições funcionais e técnicas podem ser representadas ao longo de cinco dimensões, usando a noção de separação de conceitos. Dependendo dos requisitos da aplicação pode-se focar na funcionalidade que necessita de mais atenção, sem, porém, afetar o projeto como um todo. Já existem aplicações que foram desenvolvidas seguindo este modelo: Intelligent Freight Planner - IFP (Planejamento de transporte de cargas inteligente) e uma aplicação multiagente que é executada entre diversas companhias de seguro para facilitar os pagamentos de indenizações.

Este modelo 5C pode ser dividido, portanto, em cinco dimensões: competência, comunicação, autonomia, planejamento e ambiente. Estas dimensões são enquadradas em modelos onde cada modelo é responsável por um tipo particular de funcionalidade que o agente exige.

Competência: Um agente é projetado para executar uma ou um pequeno número de tarefas. Podemos enxergar uma tarefa como a oferta de um serviço. Este agente pode executar suas tarefas que já fazem parte da sua própria natureza, ou, ainda, tem a possibilidade de consultar um sistema legado, por exemplo, para que possa executar seus serviços.

Autonomia: Um agente tem uma identidade incluindo a descrição do seu nome, objetivo, estado atual, papel organizacional, capacidades que possui e localização computacional. Mesmo que um agente se mova de um servidor para outro é necessário assumir que ele ainda continua sendo o mesmo agente. Complementando seu comportamento e autonomia, o agente precisa ter as seguintes capacidades: introspecção, reflexão, gerenciamento do seu ciclo de vida e um plano de instruções.

Planejamento: O planejamento está ligado à autonomia dos agentes. A autonomia de um agente é o que torna um agente de software inteligente tão diferenciado (M.GASPARI e MOTTA, 1994)(WOOLDRIDGE, 2009). Enquanto um sistema especialista é projetado para realizar uma tarefa de maneira racional, ancorado em um ambiente fixo que determina o comportamento do processo, a racionalidade de um agente é estendida para um número de decisões a serem tomadas: como fazer, quando fazer, com qual freqüência e quanto tempo levará.

Ambiente de execução: Os agentes de software estão localizados em um ou mais ambientes (NWANA, 1996) onde eles podem interagir com outros atores como: outros agentes, pessoas, sistemas legados e fontes de informação. Em um modelo mais simples, menos dinâmico, o agente mantém ligações com todos os atores externos que ele precisa interagir. Porém, quando o agente se insere em um ambiente complexo, móvel e imprevisível ele necessita de novas habilidades.

3.4 Problemas e Limitações do Uso de Agentes

O uso de agentes no desenvolvimento de aplicações faz parte dos mais diversos campos de pesquisa no contexto da Ciência da Computação, oferecendo recursos interessantes principalmente no desenvolvimento de aplicações distribuídas e complexas e ainda em domínios que possuem uma composição bastante heterogênea dos seus componentes e onde existam interesses que estejam em conflito (ELAMY, 2005).

Características dos agentes como: autonomia, pró-atividade e comportamento baseado em objetivos permitem a estas aplicações complexas a execução com o mínimo de intervenção humana. Além disto, os agentes podem contar ainda com habilidades como aprendizado e raciocínio, que, aplicados a outras técnicas da Inteligência Artificial, permitem a estas aplicações desenvolver suas tarefas de maneira otimizada, inteligente e confiável (GARCÍA, 2007).

Apesar de todos estes avanços, a tecnologia de agentes ainda não é muito utilizada nas grandes corporações e suas necessidades de aplicações distribuídas, complexas e inteligentes. Poucos são os casos que se transformaram em plataformas de desenvolvimento corporativas e poucas são as instituições privadas que utilizam a tecnologia. A plataforma JADE (BELLIFEMINE, 2003), é uma exceção, pois foi criada e continua sendo desenvolvida e mantida pela Telecom Itália, uma rara exceção dentro do desenvolvimento de sistemas baseado em agentes.

Dentre os aspectos que mais contribuem para a não disseminação da tecnologia em aplicações comerciais esteja a falta de uma metodologia de desenvolvimento padrão para sistemas multiagentes, assim como é a UML para o desenvolvimento orientado à objetos. Podemos destacar ainda a dificuldade na comunicação entre agentes situados em plataformas diferentes em servidores remotos. O problema surge quando uma determinada mensagem que precisa ser enviada necessita passar pelos limites de uma instituição para alcançar seu destino. A dificuldade é notória, pois as principais plataformas multi-agentes fazem uso de padrões proprietários como RMI (Remote Method Invocation), ou o protocolo proposto pela OMG para o CORBA, IIOP (Internet Inter-Orb Protocol) na troca de mensagens entre os agentes. Sendo assim, as mensagens não conseguem chegar ao seu destino principalmente por serem barradas na ação dos firewalls que estão quase sempre configurados para aceitarem

somente conexões http (GARCÍA, 2007).

Outra dificuldade encontrada é relativa à falta de semântica nos processos de comunicação. Apesar da linguagem FIPA-ACL prever esta situação e definir todo o processo comunicativo entre os agentes, as plataformas que implementam este padrão não o implementam de maneira satisfatória (LOUIS e MARTINEZ, 2005). Portanto, é necessário que exista um mecanismo que consiga lidar com a questão da interoperabilidade entre as diversas plataformas e agir na "tradução"das mensagens trocadas pelos seus respectivos agentes (GARCÍA, 2007).

Em (MAS, 2005) alguns pontos são destacados para que a tecnologia de agentes possa dar um salto de utilização na comunidade em geral:

- Obter uma definição cartesiana do conceito de agente. Um agente deve se identificar como uma entidade computacional com forma e comportamento bem definidos e de maior nível de abstração que os elementos usuais na programação (classes, funções e procedimentos, por exemplo) e deve ter associado ainda um padrão arquitetônico que especifique sua estrutura externa (interfaces), interna (componentes de controle) e o comportamento computacional necessário para funcionar;

- Alcançar a compatibilidade com as tecnologias e plataformas computacionais relacionadas. Um agente não deve depender de um padrão proprietário para que mesmo advindo de plataformas diferentes, dois agentes possam se comunicar sem maiores problemas. Uma das formas de se contornar este problema é seguir os mesmos passos da OWL para a representação de ontologias, ou os padrões estabelecidos relacionados a serviços Web (WSDL, OWL-S, UDDI, etc);

- Desenvolvimento de ferramentas e metodologias eficazes para auxiliar no desenvolvimento destas aplicações com requisitos bem diferentes dos usuais;

- Produzir aplicações do tipo: baratas, úteis, robustas e "inteligentes".

3.5 Trabalhos Relacionados

Com o objetivo de contextualizar este trabalho e apresentar os diversos desafios encontrados em utilizar uma arquitetura baseada em agentes para apoiar o desenvolvimento de novas aplicações em e-Science e Web Semântica foram analisados diversos trabalhos que de alguma forma estão ligados aos conceitos de **agentes**, ***e-Science*** e ***Web Semântica***. Os trabalhos foram divididos em três categorias: (i) trabalhos que envolvem a implementação de frameworks que sirvam como base para o desenvolvimento de aplicações baseadas em agentes voltadas para a Web, independente da sua natureza (ii) trabalhos envolvendo a semântica na busca e recuperação de serviços Web e (iii) trabalhos relacionados a infra-estruturas de apoio a *e-Science*.

3.5.1 *Frameworks* para Construção de Aplicações Semânticas Baseadas em Agentes

3.5.1.1 *SemantiCore*

O trabalho inicial relacionado a este framework foi proposto em (BLOIS e LUCENA, 2004). Esta proposta foi expandida e atualizada (ESCOBAR e LEMKE, 2006) e tem como objetivo principal permitir o desenvolvimento de aplicações baseadas em agentes na Web Semântica.

O SemanticCore é estruturado como um framework que abstrai a plataforma de implementação e prevê primitivas para aplicações organizadas em um conjunto de agentes que realizam suas tarefas no ambiente Web. Os agentes de software podem ser descritos através de uma estrutura ontológica. Portanto, é um framework conceitual com o intuito de nortear o desenvolvimento de aplicações para a Web Semântica baseada em agentes independente da implementação escolhida para a plataforma de agentes, padrões de comunicação, dentre outros detalhes.

Como características gerais, podemos citar que o framework é dividido em dois modelos: o modelo do agente (SemanticAgent) e o modelo do domínio semântico. Os dois modelos dispõem de pontos de flexibilidade (hotspots) permitindo aos desenvolvedores associar diferentes padrões, protocolos e tecnologias.

Basicamente são quatro os componentes do modelo de representação do agente: sensorial, decisório, executor e efetuador. O componente sensorial é responsável em captar os recursos que trafegam no ambiente ao qual o agente está inserido e reage toda vez que algum objeto semântico é percebido. O componente decisório encapsula o mecanismo de tomada de decisão do agente. Um dos pontos de flexibilidade encontrados no framework se encontra neste componente. Este componente basicamente trabalha em cima de uma ontologia descrita em linguagens como OWL (OWL, 2004), onde são especificados fatos e regras que vão guiar o agente na sua tomada de decisão e sempre ter como saída uma ação (Action). O desenvolvedor tem liberdade para definir suas próprias ações através de uma extensão da classe Action contida no framework, por isto, este é um ponto de flexibilidade do projeto. Os planos de ação que são gerados a partir do processo decisório são executados pelo componente Executor utilizando para isto uma estrutura de workflow para controlar as transições das ações que são realizadas.

O último componente do modelo do agente são os efetuadores. Eles são responsáveis pelo encapsulamento dos dados em mensagens para transmissão. Cada agente pode ter um ou mais efetuadores, cada um sendo responsável em publicar um tipo de objeto no ambiente. Como o framework é independente de tecnologia, as mensagens podem ser enviadas em diferentes padrões como SOAP ou FIPA ACL, que por sinal é outro ponto de flexibilidade deixando a cargo do desenvolvedor usar o padrão que julgar melhor.

3.5.1.2 ASYNC - Um *Framework* para Construção de Agentes Inteligentes

O ASYNC (Agent´s SYNergistic Cooperation) (SARDINHA *et al.*, 2003) é um framework orientado a objetos que permite construir agentes de software em ambientes distribuídos (SARDINHA, 2005). Ele foi concebido através de um desenvolvimento bottom-up a partir de três aplicações que foram desenvolvidas para identificar os pontos em comum entre elas. As ferramentas são: (i) uma ferramenta (SARDINHA, 2001)(MILIDIU *et al.*, 2001) que utiliza agentes reativos e evolutivos para identificar interdependência de produtos para promoção em um mercado varejista; (ii) uma ferramenta (BEVILACQUA e SARDINHA, 2001) baseada em agentes para criar grupos de consumo, e (iii) uma ferramenta (RIBEIRO, 2001) de negociação

entre agentes.

O projeto deste framework orientado a objetos separa em agentes duas abstrações importantes: ações internas e protocolos de comunicação (SARDINHA, 2005). O primeiro remete as ações e tarefas que os agentes precisam realizar independente da ação de outros agentes. Já a segunda abstração é relativa ao modo de interação e de comunicação entre os agentes que compõem o sistema. Através destas duas abstrações, um plano de ação de um determinado agente pode ser construído por meio de várias ações e ainda definir os protocolos de comunicação que serão usados nas suas interações com outros agentes. Para desenvolver um sistema baseado em agentes utilizando o framework ASYNC, os principais passos do mapeamento são elucidados na figura 3.2.

Conceitos de Agentes	**Classes do ASYNC instanciadas**
O ambiente e os recursos	(i) Classe Principal para o ambiente; (ii) Classes para os recursos.
Agente	(i) Uma classe concreta que herda de *Agent*, e implementa *AgentInterface*; (ii) Uma classe concreta que herda de *InteractionProtocols*.
Ações internas do Agente	Métodos incluídos na classe concreta de *Agent*.
Protocolos de Interação	Método incluídos na classe concreta de *InteractionProtocols*.
Formato das Mensagens nos protocolos de interação	(i) Classe concreta que implementa *AgentMessage*, se a troca de mensagens é síncrona; (ii) Classe concreta que implementa *AgentBlackBoardInfo*, se a troca de mensagens é assíncrona.

Figura 3.2: Mapeamento dos conceitos de agentes para o framework ASYNC (SARDINHA, 2005)

O framework é composto por duas classes abstratas (Agent e InteractionProtocols), duas classes concretas e finais (ProcessMessageThread e AgentCommunicationLayer) e três interfaces (AgentMessage, AgentBlackBoardInfo e AgentInterface) (SARDINHA, 2005). Iremos descrever de maneira sucinta estes principais componentes.

A classe abstrata Agent é responsável por fornecer as funcionalidades básicas do agente como: iniciar, parar, executar ações, etc. A interface AgentInterface é responsável por transformar subclasse de Agent em uma thread. A classe abstrata InteractionProtocols define as várias maneiras que o agente pode interagir com outros agentes do sistema. Portanto, qualquer código que remeta a comunicação deverá estar em subclasses desta classe abstrata. A classe concreta e final ProcessMessageThread é responsável por processar as mensagens recebidas pelo agente.

A interface AgentMessage deverá ser implementada pela classe que definirá o formato da mensagem, por exemplo: FIPA ACL, KQML, etc. A segunda interface, AgentBlackBoardInfo, deve ser implementada pela classe que especifica o formato das informações no ambiente do sistema. A classe concreta e final AgentCommunicationLayer especifica como será a camada de comunicação do agente por onde as mensagens trafegarão (SARDINHA, 2005). Ela fornece uma infra-estrutura para o envio de mensagens síncronas ou pode implementar uma arquitetura de blackboard que é uma espécie de "mural de recados"para os agentes procurarem mensagens que lhe pertençam quando não ocorre o envio direto.

3.5.1.3 Considerações

Os trabalhos descritos SemantiCore (BLOIS e LUCENA, 2004) e ASYNC (SARDINHA *et al.*, 2003) consideram, prioritariamente a adição de semântica para a construção de sistemas multiagentes. No entanto, não são relacionadas diretamente ao contexto de e-Science e nem lidam diretamente com questões como composição semântica de serviços web, que é uma questão preponderante no contexto de e-Science. O SASAgent, proposto nessa dissertação, esta inserido no contexto do framework ASOW-Science (MATOS *et al.*, 2009), que é baseado na tecnologia de agentes e permite a composição de aplicações científicas.

3.5.2 Semântica em Serviços

3.5.2.1 *MathWS*

Em (MATOS e *et al.*, 2007) é proposta uma arquitetura orientada a serviços para registro, descoberta e execução de serviços Web matemáticos. O registro da descrição semântica do serviço é feito através do armazenamento em banco de dados, dos conceitos inferidos na ontologia. A descoberta usa uma máquina de inferência para recuperar serviços que combinem com os conceitos fornecidos pelo cliente. A execução utiliza a descrição WSDL contida na ontologia para acessar o serviço, bem como realiza a conversão dos parâmetros de conceitos semânticos para os tipos de dados esperados pelo serviço.

Para agregar semântica aos serviços é utilizada a ontologia OWL-S (Ontology Web Language for Services) (OWL-S, 2004) que provê os blocos básicos para codificar a descrição semântica dos serviços, construída naturalmente sobre a OWL. A OWL-S está estruturada de maneira a responder três tipos de questões:

- O que o serviço provê a seus clientes? As capacidades e características do serviço são expressas através de um profile e todo serviço registrado no MathWS possuirá um ServiceProfile que o descreve;
- Como o serviço é usado? O modelo do processo do serviço é capturado pela classe ServiceModel. O modelo detalha o conteúdo semântico das requisições, as condições em que os resultados são fornecidos e, quando necessário, o processo passo-a-passo que gera estes resultados. Uma espécie de workflow para o funcionamento do serviço;

As ontologias no contexto do MathWS são um sub-conjunto das ontologias desenvolvidas pelo projeto MONET (MONET, 2006). Estas ontologias são utilizadas tanto na descrição das características e capacidades dos serviços matemáticos quanto na realização de consultas executadas pelos clientes. Basicamente utilizam-se três ontologias: GAMS (Guide to Available Mathematical Software) (GAMS, 2006), Problem (MATOS e *et al.*, 2007) e Algorithm (MATOS e *et al.*, 2007).

A GAMS é um serviço oferecido pelo National Institute for Science and Technology (NIST) que provê um índice online dos softwares matemáticos disponíveis,

classificados de acordo com o problema que eles resolvem. A ontologia Problem foi projetada para representar um problema matemático específico. Um problema pode ser descrito em termos de suas entradas e saídas, pré-condições e pós-condições. E finalmente a ontologia Algorithm serve de referência para a implementação de algoritmos pelos serviços matemáticos. Ela descreve algoritmos conhecidos em computação científica. Neste trabalho específico ela foi estendida para representar algoritmos para resolução numérica de sistemas lineares e não-lineares.

A arquitetura do MathWS pode ser dividida em três camadas principais e vista na figura 3.3:

- Camada de mediação: representada pelo broker MathWS. Ele próprio está implementado como um serviço Web. O broker possui três componentes principais: ClientManager (responsável pela interação dos clientes com o broker), o StoreManager (que armazena os registros dos serviços Web) e o ExecuteManager (responsável pela codificação dos parâmetros e execução do serviço Web);
- Camada cliente: implementa uma interface para a interação do usuário com o broker;
- Camada de serviços: abrange os serviços Web que implementam ou encapsulam os algoritmos matemáticos.

3.5.2.2 *Fusion*

Este trabalho apresenta o FUSION (KOURTESIS e PARASKAKIS, 2008), uma ferramenta para registros de serviços Web semânticos, desenvolvida no contexto do projeto FUSION[1] e liberada como uma ferramenta open source para uso da comunidade científica. O objetivo principal é promover a integração de processos de negócio e interoperabilidade dentro do ambiente corporativo através de uma abordagem semântica para aplicações de negócio orientadas a serviços.

A abordagem faz uso de três tecnologias já conhecidas no contexto de serviços Web e Web Semântica: UDDI para armazenar e recuperar informações sintáticas

[1] http://www.fusion-strep.eu/

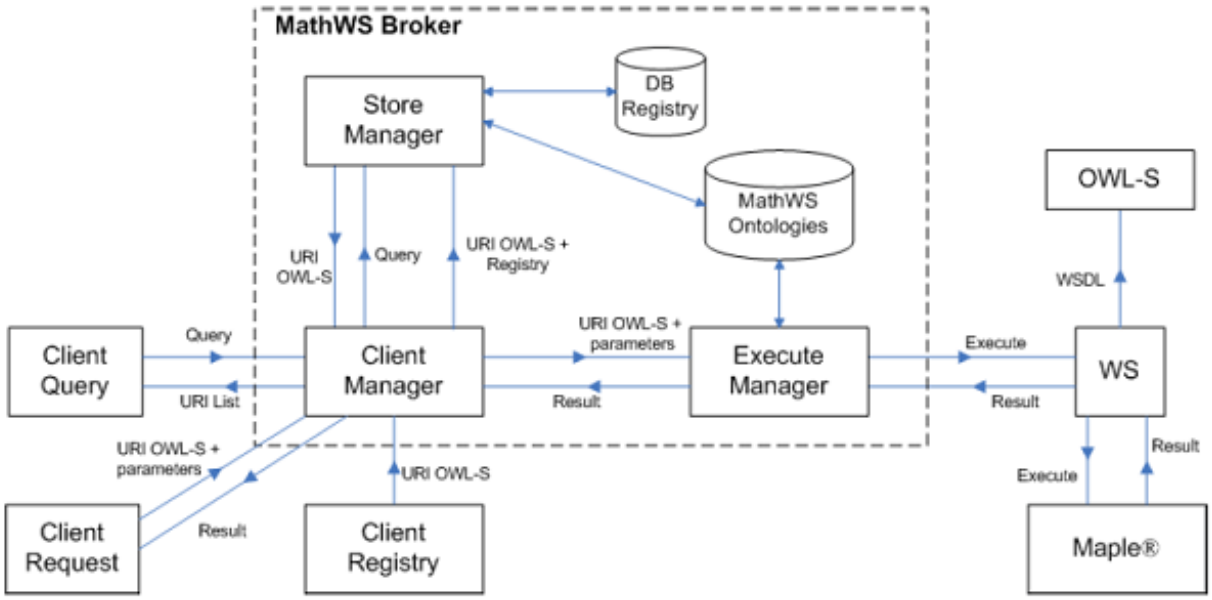

Figura 3.3: Visão geral da arquitetura do MathWS Broker (MATOS e *et al.*, 2007)

e semânticas sobre os serviços e seus provedores, SAWSDL (Semantic Annotations for WSDL) (KOPECKY *et al.*, 2007) para a criação das descrições anotadas semanticamente das interfaces dos serviços e OWL para a modelagem das características dos serviços relacionadas ao seu domínio de aplicação para que possa permitir a realização de inferências.

A arquitetura do FUSION baseia-se em propriedades funcionais e propriedades não-funcionais para fazer o cruzamento de informações para localizar os serviços que atendem a um determinado critério definido pelo usuário. As propriedades funcionais quando são analisadas para se encontrar um serviço Web normalmente são: entrada de dados, saída de dados, pré-condições e efeitos gerados após o processamento das informações. O FUSION trabalha no nível de análise da entrada e saída de dados buscando uma interoperabilidade entre os dados, quer dizer que, ele avalia se o cliente do serviço forneceu toda a informação necessária que o serviço precisava e se o cliente obteve toda a informação que necessitava com saída.

Este controle que o FUSION realiza é implementado em dois níveis distintos:

- Nível de mensagem: o objetivo aqui é determinar a razão entre o que o serviço pode prover e o que o cliente pode prover para a invocação do serviço nesta via de mão dupla. No caso mais simples, em uma operação atômica e não tran-

sacional do serviço Web corresponde à verificação da mensagem de requisição e/ou resposta da operação que foi invocada;

- Nível de domínio: o objetivo é determinar se os dados que compõem os parâmetros trocados entre o cliente e o serviço possuem as mesmas propriedades quando compartilham de um mesmo modelo de dados.

De maneira geral, a arquitetura do FUSION (figura 3.4) possui um servidor UDDI independente do módulo que faz registro semântico dos serviços Web. Ele fornece duas APIs especializadas ao cliente para que ele publique e descubra funções. Estas APIs são ainda, responsáveis pela transformação do WSDL em SAWSDL, processamento da ontologia em OWL e operações de inferência sobre a mesma. O repositório UDDI permanece inalterado e o que realmente muda é esta camada entre o cliente que deseja invocar um serviço e o repositório. Esta camada é que contém a infra-estrutura necessária para permitir a busca, registro e invocação de serviços Web semânticos sem que precise alterar o repositório UDDI.

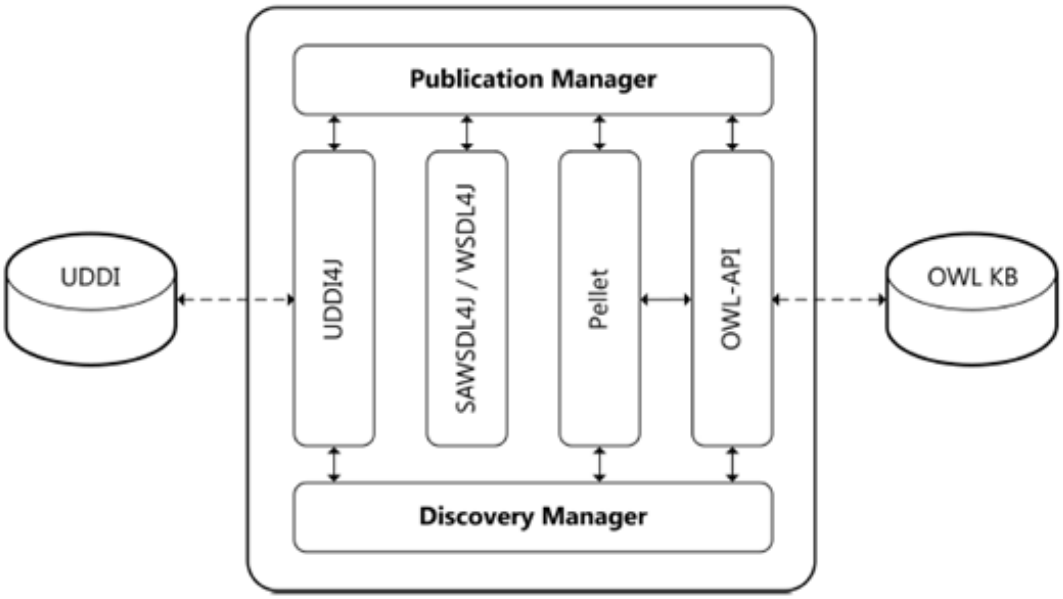

Figura 3.4: Arquitetura FUSION (KOURTESIS e PARASKAKIS, 2008)

3.5.2.3 OWLS-MX

O OWLS-MX é um mecanismo para se encontrar serviços Web baseado em regras semânticas (OWL-S) e de recuperação de informação, tornando-se uma estru-

tura híbrida pois além de utilizar lógica relacional na sua pesquisa, aplica filtros de similaridade vindos da área de recuperação de informação (KLUSCH *et al.*, 2006).

Esta proposta computa o grau de similaridade semântica para um determinado par de serviços aplicando sucessivamente cinco diferentes filtros: **EXACT**, **PLUG IN**, **SUBSUMES**, **SUBSUMED-BY** e **NEAREST-NEIGHBOR**. Os três primeiros são baseados em regras lógicas, enquanto que os dois últimos são filtros híbridos que requerem um processamento adicional para verificação de similaridade.

Os serviços que se encaixam na categoria **EXACT** são aqueles cuja assinatura de entrada e saída se enquadra perfeitamente a requisição que está sendo realizada. Este enquadramento é verificado através de regras lógicas formais para comprovar esta exatidão semântica.

A categoria **PLUG IN** possui uma regra de relaxamento que o diferencia da categoria **EXACT** no que diz respeito à entrada de dados da requisição realizada. Um serviço qualquer S, se encaixa nesta categoria quando ele necessita de uma entrada de informação menor do que a requisição está fornecendo. Isto garante, no mínimo, que o serviço S é executável com a entrada de informação fornecida pela requisição desde que os conceitos da entrada escritos em OWL possam ser mapeados de maneira equivalente às mensagens de entrada escritas em WSDL com seus correspondentes tipos de dados. Além do mais, a saída do serviço S pode ser bem mais completa do que aquela esperada pela requisição do usuário semelhante ao conceito de um componente, tornando este serviço capaz de atender a diversos tipos de requisições.

O filtro **SUBSUMES** é mais fraco que o **PLUG IN** no que diz respeito a extensão da saída do serviço S, que é mais específica do que a saída da requisição do usuário. Como conseqüência, o conjunto de serviços que atendem a este filtro é mais extenso.

Já na categoria **SUBSUMES-BY**, o filtro seleciona os serviços cuja saída é mais genérica do que a requisitada pelo usuário. O foco na saída é diretamente relacionado ao conceito de "pai"para evitar que serviços com saídas muito genéricas sejam retornados nas consultas. As restrições de relaxamento também podem variar de acordo com a perspectiva do usuário fazendo com que este filtro seja mais flexível ou não, dependendo do que se deseja filtrar.

A proposta também avalia quando um serviço S não atende uma requisição R. Esta situação ocorre quando os filtros anteriores, baseados em regras lógicas, não são atendidos. E chama-se **LOGIC-BASED FAIL**.

Por último, o filtro que utiliza princípios da área de recuperação da informação como grau de similaridade é o NEAREST-NEIGHBOR. Ele avalia três situações ao mesmo tempo: se a entrada do serviço S é mais genérica do que a entrada que a requisição está exigindo, se a saída da requisição é mais genérica do que a saída gerada pelo serviço e ainda verifica se o grau de similaridade entre um serviço S e uma requisição R é maior ou igual a um determinado coeficiente.

3.5.2.4 Considerações

Das propostas apresentadas relacionadas à adição de semântica a serviços, MathWS (MATOS e *et al.*, 2007), FUSION (KOURTESIS e PARASKAKIS, 2008) e OWLS-MX (KLUSCH *et al.*, 2006), podemos considerar como a mais completa a abordagem FUSION, que além de considerar a questão da semântica relacionada a serviços, está relacionada a um contexto científico. O SASAgent pode ser considerado como uma evolução do broker MathWS, onde questões relativas a composição e busca de serviços web semânticos foi aprimorada com a abordagem de sistemas multiagentes. A evolução também fica clara na independência de domínio proposta neste trabalho frente ao MathWS que é voltado especificamente para serviços Web matemáticos sem contemplar outros artefatos científicos como publicações e serviços Web de outras áreas. A proposta OWLS-MX é a mais limitada, uma vez que trata de somente da busca por serviços web semânticos.

A proposta SASAgent, comparada com estas abordagens tenta ir além, considerando o contexto da computação científica, onde não há trabalhos tão abrangentes, uma vez que propõe uma busca semântica direcionada, além da possibilidade de composição e ativação dos serviços. Além disto, utiliza conceitos já estabelecidos como agentes e ontologias para a representação semântica dos artefatos científicos para criar novos mecanismos de busca e recuperação. É facilmente integrável a outras plataformas multi-agentes que utilizem o padrão FIPA para a comunicação dos agentes e é capaz de atender a qualquer projeto de pesquisa independente do domínio no qual estejam inseridos. É extensível através da ontologia SASOntology e prevê

ainda integração com ontologias próprias de domínio, que os grupos de pesquisa possam decidir fazer uso.

3.5.3 Infra-Estrutura para Apoio a *e-Science*

3.5.3.1 *Semantic Web Integrator and Query Engine* (SemWIQ)

O SemWIQ apresenta um sistema que foi desenvolvido com foco no compartilhamento de dados nas comunidades científicas. Mais precisamente, o sistema é desenvolvido como parte do projeto Austrian Grid para permitir que os dados distribuídos possuam acesso de maneira transparente aos cientistas que os compartilham (LANGEGGER *et al.*, 2008).

O projeto principal é chamado Semantic Data Access Middleware for Grids (G-SDAM) porém o sistema foi desacoplado do projeto principal, uma vez que ele pode servir a outras aplicações que visam utilizar os recursos disponíveis da Web Semântica. A arquitetura foi desenvolvida baseada em quatro princípios básicos: os dados científicos estão geralmente estruturados, são geograficamente distribuídos, armazenados em formatos heterogêneos e normalmente é restrito somente a pessoas autorizadas.

A implementação do SemWIQ utiliza a biblioteca Jena. O sistema é baseado ainda em uma abordagem de mediadores e wrappers e sua arquitetura pode ser vista na figura 3.5. Um cliente estabelece uma conexão com o mediador e requisita dados submetendo consultas em SPARQL (1). O tradutor (2), que faz parte da biblioteca Jena, calcula um plano de execução por um otimizador (3). Este otimizador analisa a consulta e procura pelo catálogo as fontes de dados relevantes para o usuário. A saída do otimizador é um plano global para a execução da consulta que é transmitido ao componente de busca (4). Este plano global é executado pelos wrappers disponíveis (5).

Qualquer fonte de dados precisa de um wrapper capaz de converter a consulta original escrita em SPARQL para o formato específico daquela fonte a não ser que a fonte de dados já dê suporte nativo a consultas escritas em SPARQL. Geralmente um wrapper é colocado o mais próximo possível da fonte de informação para obter melhor desempenho na execução das consultas. Porém, nem sempre isto é possível,

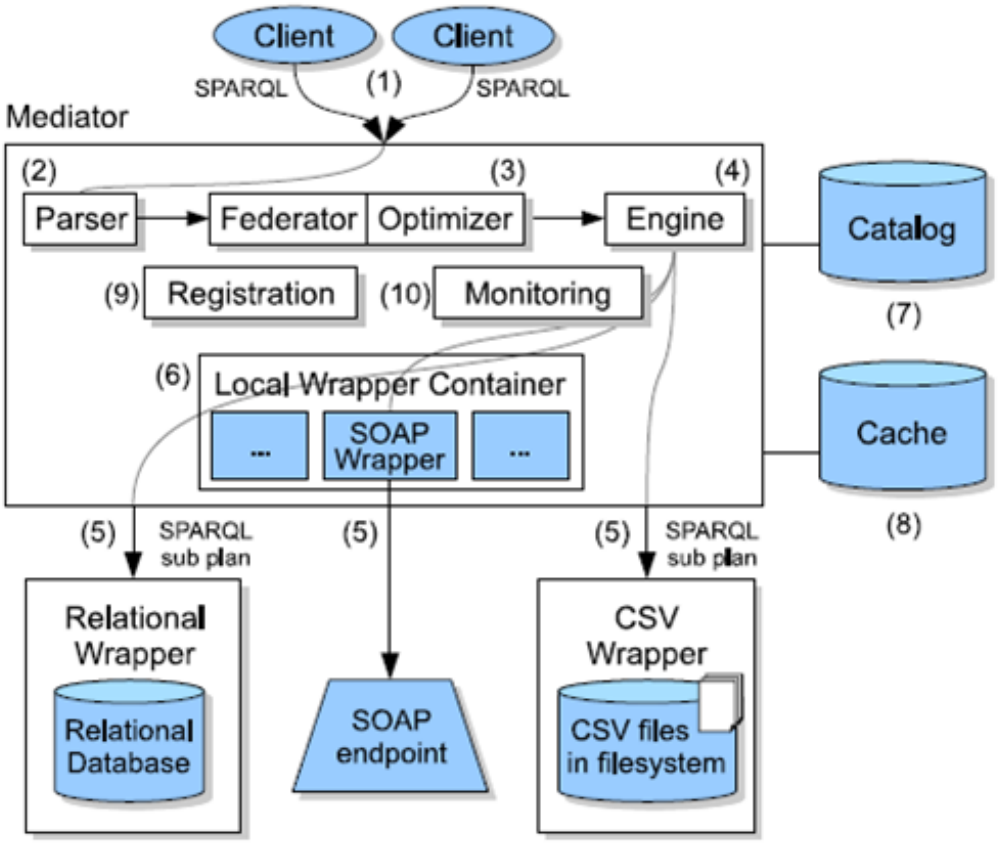

Figura 3.5: Arquitetura do SemWIQ (LANGEGGER *et al.*, 2008)

principalmente quando não temos controle sobre a base de dados remota ou é uma estrutura mais complexa como um serviço Web. Neste caso, o mediador provê um container de wrappers (6). O catálogo (7) armazena descrições e estatísticas sobre as bases de dados registradas e um cache local em RDF (8) será usado para armazenar tuplas recorrentes nos resultados das consultas. O registro de uma base de dados é realizado enviando uma requisição HTTP POST para o mediador junto com um documento RDF anexado (9). Ele especifica a URI do seu ponto de acesso e contém metadados que podem conter informações da origem dos dados e o contato do pesquisador, por exemplo. A saída da base de dados ainda não está implementada, porém, a partir do momento que o mediador não consegue mais acesso àquela base de dados específica, ela automaticamente é retirada do catálogo. Finalmente, o componente de monitoramento (10) atualiza as estatísticas sobre as bases de dados registradas.

3.5.3.2 *Matching* Semântico de Recursos Computacionais em Ambientes de Grade com Múltiplas Ontologias

O trabalho propõe uma infra-estrutura que permite compartilhar recursos computacionais de diversos provedores diferentes de maneira transparente ao usuário através da integração de ontologias. A ontologia global criada a partir da integração das ontologias locais de cada provedor de recursos é que permitirá o usuário navegar e pesquisar pelos recursos disponíveis no ambiente de grid computacional (RODRIGUES, 2005).

Para a integração das diversas ontologias algumas questões devem ser analisadas. A união de conceitos de duas ontologias distintas pode ser baseada em vários aspectos: o primeiro deles, na sintaxe, portanto, dois conceitos com a mesma sintaxe são equivalentes. O segundo aspecto leva em consideração sintaxes distintas, mas que significam a mesma coisa como casos de diferença entre letras maiúsculas e minúsculas. Um terceiro aspecto: conceitos que possuam propriedades que possuem uma mesma nomenclatura também são unidos. O quarto e último aspecto é a união de conceitos baseados em um dicionário de sinônimos. Além disto, para que seja mantida a consistência da ontologia global, um sistema de regras também foi utilizado. A contribuição secundária do trabalho é uma ferramenta gráfica que incorpora tanto a visualização da ontologia global como a consulta a esta mesma ontologia e recuperação dos elementos que satisfazem esta consulta. O trabalho é uma extensão de (HEINE *et al.*, 2004).

Todo o mecanismo foi projetado para trabalhar como um broker, onde os provedores de recursos cadastram seus recursos via broker e este é o responsável pela união das ontologias enviadas por cada um dos provedores. O broker foi projetado como um serviço de grade que possui os métodos: getOntology, addOntology e removeOntology.

O sistema possui uma ontologia base, onde nesta existe um conceito chamado **recurso** com uma propriedade de nome localização que será utilizada para armazenar as informações de **localização** dos recursos das ontologias que são enviadas pelos provedores de recursos.

Na inclusão de uma ontologia oito passos são realizados para que além de inserida, ela componha a ontologia global nos padrões corretos para o acesso transparente ao

usuário. Os passos são ilustrados na figura 3.6.

A junção da nova ontologia à ontologia global durante o processo de verificação de sinônimos ainda possui três fases distintas: uma pesquisa é feita, dentre as classes da ontologia, por classes que tenham a mesma escrita, porém com alguma diferença com relação as letras maiúsculas e minúsculas. Ao encontrar estas classes equivalentes utiliza-se a propriedades equivalentClass da linguagem OWL. Além disto, uma verificação de sinônimos é realizada para que possa ser novamente aplicada a propriedade equivalentClass. Por exemplo, queremos unir os conceitos **Computador** e ***Computer*** ou ainda **Memória** e ***Memory***.

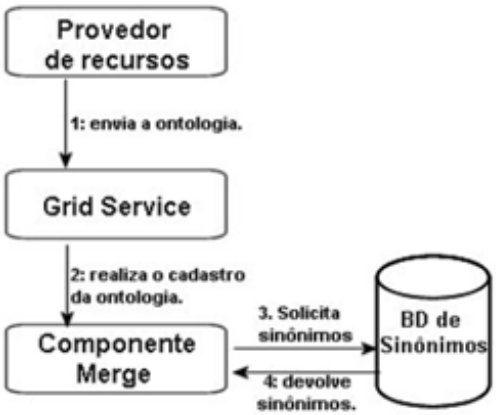

Figura 3.6: Inclusão e integração de uma ontologia (RODRIGUES, 2005)

Por último, e não menos importante, apesar de não ter relação direta com o processo de integração, é a descrição da localização física dos recursos do provedor como, por exemplo, um endereço IP ou coordenadas para um GPS, dentre outros. Esta informação é extremamente necessária, pois durante a pesquisa do usuário, existem recursos que estão geograficamente dispersos e dependendo do recurso, a localização é de extrema importância para o desempenho de uma tarefa.

O matching semântico, que é a busca pelos recursos utilizada pelo usuário final, é realizado através de uma ferramenta gráfica. Esta ferramenta permite visualização da ontologia global tanto em forma de árvore e em forma de grafo como a visualização das propriedades individuais de cada classe (figura 3.7).

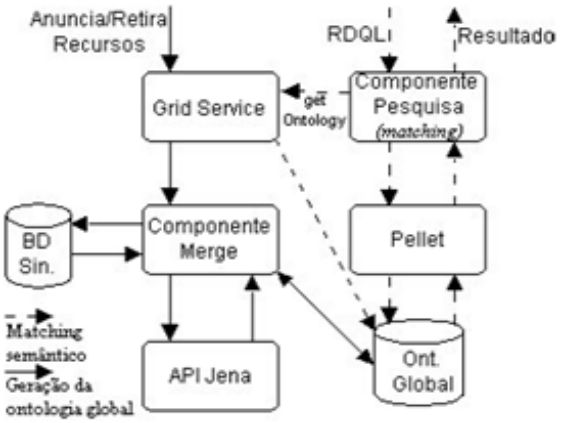

Figura 3.7: Arquitetura do matching semântico (RODRIGUES, 2005)

3.5.3.3 Considerações

Considerando as abordagens relacionadas ao apoio a e-Science, o SASAgent tem uma proposta semelhante as apresentadas: SemWIQ (LANGEGGER *et al.*, 2008) e (RODRIGUES, 2005). No entanto, inclui, além do armazenamento e busca semântica, a possibilidade de composição dos serviços web encontrados. Além disso, as abordagens pesquisadas possuem ênfase maior na criação de arquiteturas de mediação para lidar com a heterogeneidade das fontes. O *SASAgent* permite a busca por artefatos científicos, utilizando uma arquitetura SOA, além de permitir a invocação e possibilidade de composição dos serviços.

Capítulo 4

Desenvolvimento de Sistemas Multiagentes

4.1 Fundamentação

Não há nenhum consenso sobre uma tecnologia específica para o desenvolvimento de agentes. Nas atividades conceituais que lidam com o ciclo de vida de desenvolvimento, por exemplo, muitas metodologias estão sendo estudadas e criadas como AUML (ODELL *et al.*, 2000), MaSE (WOOD e DELOACH, 2000) , Prometeus (PADGHAM e WINIKOFF, 2003), MAS-CommonKADS (IGLESIAS e GARIJO, 2005), Gaia (WOOLDRIDGE *et al.*, 2000), entre outras.

No que diz respeito às plataformas de desenvolvimento, protocolos de interação e outros assuntos relacionados à implementação dos agentes, duas iniciativas tem tido maior destaque na literatura: a plataforma JADE (Java Agent Development Framework) (BELLIFEMINE, 2003), e a organização FIPA que está ligada ao IEEE. A organização FIPA pesquisa a padronização de protocolos que controlam a comunicação entre os agentes, transporte, gerenciamento e metamodelos para a arquitetura das aplicações, com o intuito de favorecer o desenvolvimento de aplicações baseadas em agentes. A comunicação entre os agentes é o cerne da iniciativa e é largamente utilizada como padrão para a comunicação em diversas aplicações espalhadas pelo mundo, pois já foi aprovada e recomendada pelo IEEE como padrão de comunicação.

Este padrão é a FIPA ACL (Agent Communication Language).

Uma das especificações FIPA que ainda está em fase experimental, é relativa às tecnologias que permitam agentes gerenciar explicitamente ontologias criadas para que os mesmos possam se comunicar, mesmo vindo de plataformas diferentes, porém inseridos em um mesmo domínio. Para que isto possa ser feito é descrito um serviço que será provido por um agente específico chamado Ontology Agent (OA) que gerenciará as ontologias, dentro de uma comunidade de agentes, através das seguintes funções:

- Descobrir ontologias públicas;
- Manter um conjunto de ontologias;
- Integrar conteúdo semântico;
- Responder às consultas semânticas;
- Permitir o compartilhamento de uma ontologia com o intuito de facilitar a comunicação entre os agentes.

Essa especificação não previne a existência de agentes, cujos conceitos de uma ou mais ontologias estejam implícitos no seu próprio código-fonte, como estruturas de uma linguagem de programação normal, fazendo com que o acesso a este conhecimento não seja alcançado pelo *Ontology Agent*. Ao mesmo tempo, o que é especificado neste modelo são as interfaces de comunicação entre os agentes de modo a prover suporte ao uso das ontologias, porém, a implementação interna e as capacidades que serão dadas ao agente são de responsabilidade do programador.

Para tentar manter a independência de plataforma, a especificação do serviço não trabalha com alguma linguagem de descrição de ontologias em específico. A especificação tem o papel de dizer o modo que os agentes irão acessar o serviço de ontologias. Há somente um único formalismo para especificar o serviço chamado FIPA-Meta-Ontology que permite a comunicação de conhecimento entre os agentes (FIPA, 2004).

O *framework* JADE é totalmente implementado em Java (GOSLING, 2005). Ele simplifica o processo de implementação de sistemas multiagentes através de

um middleware que está de acordo com as especificações FIPA e através de um conjunto de ferramentas gráficas que suportam o *debugging* e a implementação. A plataforma de agentes pode ainda ser distribuída entre várias máquinas (que não necessariamente sejam do mesmo sistema operacional) e o controle da configuração de cada máquina pode ser feita remotamente via interface gráfica. Os agentes podem ainda mover-se entre as diversas máquinas quando e como for requerido. Uma visão geral da tecnologia pode ser vista em (BELLIFEMINE, 2003). Os princípios básicos sobre os quais a plataforma foi desenvolvida são:

- Interoperabilidade: JADE atende às especificações FIPA, portanto, os agentes JADE podem se relacionar com agentes de outras plataformas desde que estejam definidos ou implementados no mesmo padrão;
- Uniformidade e Portabilidade: JADE provê um conjunto de APIs homogêneas que são independentes da rede e da versão da linguagem Java;
- Facilidade de uso: a complexidade da plataforma está oculta através de APIs simples e intuitivas de se manipular.

Além dos princípios básicos, a plataforma JADE inclui três elementos principais (BELLIFEMINE, 2003): (1) conjunto de pacotes que proporcionam uma série de componentes para o desenvolvimento dos agentes; (2) um ambiente de execução que fornece os serviços básicos necessários para a execução dos agentes; e (3) um conjunto de ferramentas gráficas que permitem administrar e monitorar a atividade dos agentes ativos do sistema. Cada instância do ambiente de execução chama-se container e oferece uma serie de serviços de acordo com a especificação FIPA, como o registro, o sistema de troca de mensagens e o serviço de autenticação. Em um mesmo ambiente ou plataforma, podem existir diversos containeres, porém, todo ambiente terá sempre um container principal onde os outros containeres deverão se registrar.

Ainda pertencentes à plataforma JADE e contidos sempre nos containeres principais de cada instância em execução, existem dois agentes específicos, sempre inicializados de forma automática:

- AMS (*Agent Management System*): proporciona um serviço de páginas brancas, assegurando-se que todo agente dentro da plataforma tenha um nome

único. Ele representa a autoridade na plataforma supervisionando o acesso e o uso da mesma;

- DF (*Directory Facilitator*): proporciona um serviço de páginas amarelas, onde este agente pode encontrar outros agentes através dos serviços que realizam. A figura 4.1 exibe esta arquitetura.

A plataforma JADE é um software de código aberto, contando com um grande número de colaboradores tanto da indústria quanto da academia, levando a numerosos projetos baseados nesta plataforma (GARCÍA, 2007), tornando-o um projeto mais maduro e, consequentemente, levando à sua escolha para este trabalho.

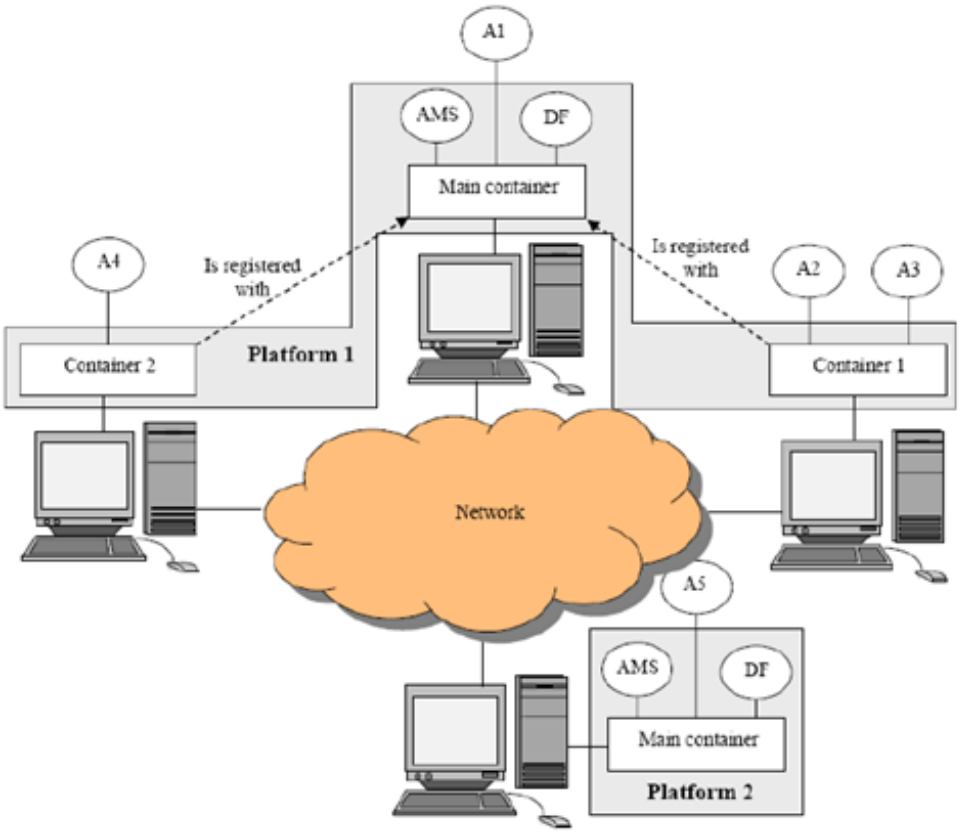

Figura 4.1: Plataformas e containeres no JADE 4.1

4.2 Metodologias de Desenvolvimento de Sistemas Multiagentes

No desenvolvimento das aplicações baseadas em agentes, algumas questões devem ser ressaltadas, pois, apesar de ter um ciclo de vida semelhante ao de um software convencional, pontos adicionais devem ser verificados durante todo o processo. É necessário haver alguma metodologia que modele o papel dos agentes dentro do software, como os agentes se relacionam com as outras partes do sistema, para que tipos de estrutura de dados esses papéis serão mapeados, modelos comportamentais também são importantes e na implementação propriamente dita, qual paradigma será utilizado, qual framework de desenvolvimento, por exemplo, além de saber quais protocolos de comunicação e formato de mensagens serão usados para a interação entre agentes e entre os agentes e o software. Apresentamos nesta seção algumas das principais metodologias para desenvolvimento de softwares baseados em agentes, com o objetivo de identificar qual destas é a mais adequada para o desenvolvimento da arquitetura *SASAgent*.

4.2.1 MASUP - *Multi-Agent Systems Unified Process*

MASUP (BASTOS e RIBEIRO, 2005) é uma extensão do RUP (RUMBAUGH *et al.*, 1999) que foca no desenvolvimento de sistemas multiagentes. O principal objetivo da metodologia é identificar sistematicamente a aplicabilidade de uma solução de agentes durante a fase de modelagem. Possui três grandes fases: requisitos, análise e projeto, sendo que a identificação da utilização de agentes ou não acontece nas fases de Análise e Projeto através de uma heurística aplicada sobre os diagramas gerados (SANTOS, 2006). Por ser uma metodologia baseada no RUP e, conseqüentemente, compatível com ele, as partes do sistema que não utilizam agentes podem ser modeladas normalmente pelas técnicas comuns do RUP.

4.2.1.1 Requisitos

Na fase de análise de requisitos, é necessário identificar os atores e as funcionalidades do sistema (SANTOS, 2006). Da mesma forma como no RUP, o principal

modelo gerado é o modelo de casos de uso. A primeira atividade, portanto, tem a função de capturar os requisitos por meio dos casos de uso. O passo seguinte é gerar, a partir dos casos de uso identificados, os diagramas de atividades correspondentes. Essa metodologia propõe a apresentação dos objetos consumidos e gerados pelas atividades nos diagramas de atividades, pois, com base nestes objetos é feita a análise dos papéis dos agentes quando a solução multiagente é a indicada (RIBEIRO e SILVA, 2005).

É importante salientar que nesta fase ainda não é possível confirmarmos a utilização ou não de agentes de software no sistema.

4.2.1.2 Análise

Nesta fase é feita uma revisão dos diagramas de atividades gerados no projeto de forma a descobrir quais atividades envolvem mecanismos de tomada de decisão, que deverão ser implementados diretamente no sistema e que na modelagem são realizados por algum ator, de modo que assim certas tomadas de decisões, que seriam feitas pelos atores, possam ser automatizadas através de agentes (RIBEIRO e SILVA, 2005).

De maneira mais detalhada, conjuntamente com a revisão dos diagramas de atividade, é especificado um diagrama de classes para se identificar quais serão as classes necessárias nos casos de uso identificados (SANTOS, 2006).

Esta fase pode ser dividida basicamente em 5 atividades distintas: revisão dos diagramas de atividades, identificação dos papéis dos agentes, especificação dos papéis dos agentes, identificação dos agentes e representação da sociedade de agentes.

Na revisão dos diagramas de atividades, as classes identificadas no diagrama de classes são incluídas nos diagramas de atividades gerados como objetos que são produzidos e consumidos em cada atividade.

A identificação dos papéis dos agentes é feita através da inspeção de cada diagrama de atividade através de uma heurística própria do modelo MASUP no sentido de encontrar papéis candidatos a agentes entre os objetos representados. Na modelagem, para marcar que determinado objeto pode ser um papel de um agente, são utilizados estereótipos (*stereotype*) da UML. Cada papel identificado, na fase de

especificação dos papéis dos agentes, deve ser detalhado em termos de atribuições e restrições.

Na atividade de identificação de agentes, o projetista pode agregar vários papéis livremente a um mesmo agente, porém, recomenda-se que um agente só assuma papéis que sejam do mesmo domínio do conhecimento. A representação da sociedade de agentes possui um diagrama próprio para representar os relacionamentos entre os agentes dentro do sistema. Onde é possível especificar os papéis que cada agente assume naquele relacionamento da sociedade, cardinalidade e a qual classe o agente pertence.

4.2.1.3 Projeto

A fase de projeto pode ser dividida em 2 atividades: criação dos cenários de interação entre os agentes e especificação das classes dos agentes.

Nos cenários de interação são modeladas as interações entre os agentes que ocorrem em determinado momento durante a execução do sistema. Fornecem os atos de comunicação que o agente é capaz de reconhecer como válidos para atender a requisição de outro agente. Essa especificação dos cenários é feita em diagramas de seqüência estendidos da UML específicos para agentes. Em resumo, os cenários de interação definem os protocolos de comunicação usados pelos agentes que são compostos de interfaces de interação.

A especificação das classes dos agentes complementa as classes especificadas no diagrama de classes com as interfaces de interação geradas nos cenários de interação para que um agente possa se comunicar com outros agentes.

4.2.1.4 Considerações

Podemos considerar que a metodologia MASUP tem como ponto de destaque sua forte integração com o RUP/ UML, utilizando modelos e diagramas amplamente conhecidos pela comunidade de desenvolvimento de software. No entanto, a forma como os agentes são modelados pode ser considerada incompleta, uma vez que não permite a modelagem e identificação dos diferentes tipos de agentes. Desta forma, a metodologia não permite identificar nas fases de análise e projeto do desenvolvi-

mento, que tipo de agentes o sistema deverá desenvolver, o que ajuda posteriormente na escolha da plataforma de desenvolvimento. Além disso, como alguns dos modelos do RUP devem ser estendidos, não existe ferramental específico para apoio à metodologia, além de não ter integração alguma com plataformas para desenvolvimento de agentes, como é o caso do JADE.

4.2.2 MaSE - *Multiagent Systems Engineering*

Engenharia de Sistemas Multi-Agentes (Multiagent Systems Engineering) (WOOD e DELOACH, 2000) foca no auxílio ao projetista nos requisitos iniciais, análise, projeto e implementação de sistemas multiagentes. MaSE é similar às metodologias de desenvolvimento de software tradicionais, mas orientada ao desenvolvimento de sistemas multiagentes. A metodologia é dividida em duas fases: Análise e Projeto. Porém, esta metodologia já pressupõe que o software a ser desenvolvido já irá contar com uma arquitetura de agentes.

A MaSE foi uma das primeiras metodologias a serem desenvolvidas para tentar adequar a Engenharia de Software a esta nova classe de aplicações baseadas em agentes. A metodologia possui um escopo bem definido para tentar facilitar a utilização da metodologia somente aos tipos de sistemas que ela é capaz de ser aplicada. Em ((WOOD e DELOACH, 2000) são listados quatro pontos que limitam o escopo de utilização da metodologia.

Primeiramente, o sistema que será criado é fechado e todas as suas interfaces externas são encapsuladas por um agente que participa dos protocolos de comunicação do sistema. Em segundo lugar, a metodologia não considera sistemas dinâmicos, onde agentes podem ser criados, destruídos ou movidos durante a execução. Terceiro ponto, a interação entre os agentes é de um-para-um. Apesar de que, há a possibilidade de se formar um conjunto de mensagens ponto para fazer o papel da comunição multicast. E finalmente, os sistemas modelados pela MaSE não devem ser muito grandes, sendo que o ideal é que sejam modeladas até dez classes de agentes. Entretanto, esta não é uma restrição pesada, só indica que sistemas muito grandes ficam difíceis de serem validados ou verificados através da metodologia quando o número de classes extrapola o limite.

A figura 4.2 mostra as fases da metodologia MaSE. Basicamente são duas fases

principais: Análise e Projeto. Essas duas fases podem ser decompostas ainda em outras sete sub-fases: Captura dos Objetivos, Aplicação dos Diagramas de Caso de Uso, Refinamento dos Papéis, Criação das Classes dos Agentes, Elaboração da Comunicação entre os Agentes, Definição da Arquitetura dos Agentes e Projeto do Sistema. A metodologia gera durante todas essas fases e sub-fases nove artefatos que são: Hierarquia das Metas, Diagrama de Casos de Uso, Diagramas de Seqüência, Modelo dos Papéis, Diagrama de Tarefas Concorrentes, Diagrama de Classes dos Agentes, Diagrama de Classes de Comunicação, Arquitetura dos Agentes e Diagrama de Distribuição.

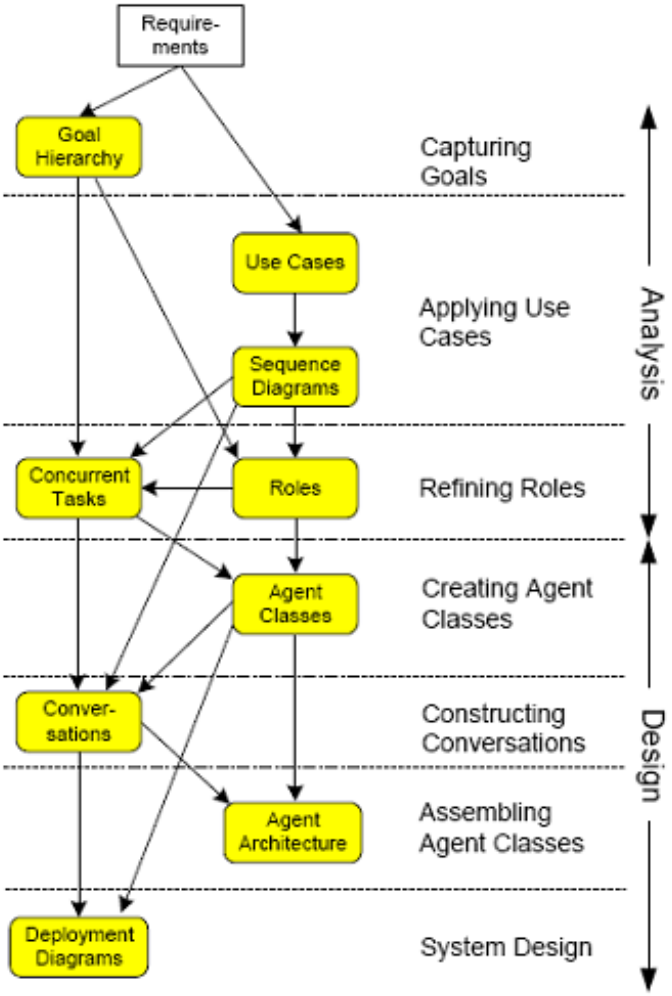

Figura 4.2: As fases da metodologia MaSE (WOOD e DELOACH, 2000)

4.2.2.1 Captura dos objetivos

Pode ser dividido em duas fases: identificação e estruturação dos objetivos. Os objetivos são identificados através do conjunto de requisitos. Lembrando que os objetivos definidos são sempre objetivos ao nível de sistema, do que ele deve abranger. Os requisitos que são a base para a definição dos objetivos podem incluir documentos técnicos, situações expostas por usuários ou especificações governamentais formais. Depois de identificados eles precisam ser estruturados de maneira que possam ser utilizados nas próximas fases da metodologia MaSE. Basicamente, é construído um diagrama de hierarquia destes objetivos, onde essa hierarquia é construída sobre a importância de cada objetivo.

4.2.2.2 Aplicação dos Diagramas de Casos de Uso

Nesta fase, os diagramas de casos de uso elaborados a partir dos requisitos iniciais do sistema são utilizados para a geração do diagrama de seqüência que representa uma seqüência de mensagens entre os vários papéis dos agentes.

4.2.2.3 Refinamento dos Papéis

O objetivo desta fase é transformar os objetivos que foram estruturados na fase anterior em papéis dos agentes. Papéis são blocos de construção usados para definir as classes dos agentes e capturar os objetivos durante a fase de projeto. Os papéis são descrições abstratas de uma função esperada para cada entidade e encapsula os objetivos levantados. O caso geral é mapear os objetivos nos papéis através de uma cardinalidade um-para-um, ou seja, cada objetivo se transforma em um papel dentro do modelo. Porém, caso seja necessário, pode-se combinar os objetivos para se construir um único papel.

4.2.2.4 Criação das Classes dos Agentes

As classes de agentes são identificadas a partir dos papéis que foram definidos na fase anterior. O produto desta fase é exatamente um diagrama de classes dos agentes que modela, além das classes, toda a comunicação entre os agentes. No

diagrama de classes gerado irá constar além do nome da classe do agente, qual o papel que aquela classe foi derivada.

4.2.2.5 Diagrama de Classes de Comunicação

Nesta fase, é definido um protocolo de coordenação entre dois agentes. Especificamente, uma "conversa"entre dois agentes consiste de dois diagramas de classes de comunicação um para quem inicia a comunicação e outro para quem responde a requisição. Um diagrama de classes de comunicação é um par de máquinas de estados finitas que define os estados das "conversas"entre os dois agentes participantes.

4.2.2.6 Arquitetura dos Agentes

A representação interna dos agentes é criada nesta fase. A metodologia prevê cinco arquiteturas possíveis para o agente: modelo BDI, reativa, planning, baseada em conhecimento e uma arquitetura definida pelo usuário. Cada modelo de arquitetura contém um conjunto específico de componentes. Como exemplo, na figura 4.3 temos a arquitetura reativa que prevê um Controller, MessageInterface, RuleContainer e Effectors.

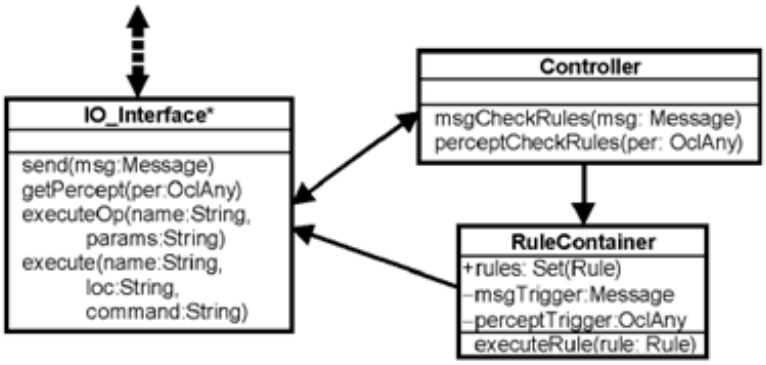

Figura 4.3: Arquitetura Reativa (WOOD e DELOACH, 2000)

4.2.2.7 Diagramas de Distribuição

Os diagramas de distribuição são usados para definir um sistema baseado nas classes dos agentes definidos nas fases anteriores da metodologia. Diagramas de dis-

tribuição definem parâmetros do sistema como: número atual, tipos e as localizações dos agentes dentro do sistema.

4.2.2.8 Considerações

Um dos pontos fortes da metodologia MaSE é permitir a identificação dos tipos de agentes ainda na fase de projeto, o que é um diferencial em relação à metodologia MASUP. No entanto, conforme ressaltado pelos próprios autores, não é uma metodologia adequada para sistemas dinâmicos, onde agentes são criados e destruídos, como é o caso do *SASAgent*. Além, disso, não privilegia habilidade social, que é uma característica preponderante para o *SASAgent*. Outras questões importantes não identificadas na metodologia MaSE é a facilidade de ferramental e integração com plataformas de desenvolvimento de agentes.

4.2.3 Tropos

A metodologia Tropos (BRESCIANI e *et al.*, 2004) foca no levantamento dos requisitos permitindo um melhor entendimento do ambiente envolvido na operação do sistema. Ela se baseia no framework i* (YU, 1995) que descreve atores, objetivos e as dependências entre os atores. A metodologia Tropos pode ser dividida em cinco fases distintas: Requisitos Iniciais, Requisitos Finais, Projeto Arquitetural, Projeto Detalhado e Implementação.

Os conceitos-chave da metodologia são (BRESCIANI e *et al.*, 2004):

- Ator: modela uma entidade que possui objetivos estratégicos e intenções dentro do sistema. Um ator representa um agente físico, social ou de software como se fosse um papel;
- Objetivo: representa os interesses estratégicos dos atores. Os objetivos podem ser divididos em duas classes: pesados e leves. Os objetivos leves não possuem uma definição ou critérios que possam decidir se eles foram satisfeitos ou não;
- Plano: representa em nível abstrato, o caminho para se fazer alguma coisa. A execução do plano pode ser o meio para satisfazer determinado objetivo;

- Recurso: representa uma entidade física ou informacional;
- Dependência: representa a relação entre dois atores, indicando que um ator depende, de alguma maneira, do outro para atingir determinado objetivo, executar algum plano ou entregar algum recurso;
- Capacidade: representa a habilidade de um ator em definir, escolher e executar um plano para alcançar determinado objetivo, considerando certas condições do ambiente que está inserido e a presença de eventos específicos.

4.2.3.1 Requisitos Iniciais

Consiste em identificar e analisar os stakeholders e as suas intenções. Stakeholders são modelados como atores sociais que dependem um dos outros para que os objetivos possam ser alcançados, planos serem executados e recursos serem utilizados. As intenções são modeladas como os objetivos, que através de análises são decompostos em objetivos mais refinados (BRESCIANI e *et al.*, 2004).

4.2.3.2. Requisitos Finais

Os requisitos finais focam no que o sistema será dentro do ambiente operacional, levantando suas funções relevantes e qualidades. O que o sistema será é representado como um ator que possui um número de dependências com os outros atores da organização. Essas dependências definem os requisitos funcionais e não-funcionais do sistema (BRESCIANI e *et al.*, 2004).

4.2.3.3 Projeto Arquitetural

Define a arquitetura global do sistema em termos de sub-sistemas (atores), interconectados através de um fluxo de dados e controle (dependências). De maneira bem resumida, são definidos três passos para esta fase:

- Passo 1: definição da arquitetura organizacional. Novos atores (incluindo sub-atores) são introduzidos no sistema;

- Passo 2: identificação das capacidades necessárias dos atores para atingir seus objetivos e planos;
- Passo 3: definição de um conjunto de tipos de agentes e atribuir a eles uma ou mais diferentes capacidades.

4.2.3.4 Projeto Detalhado

Esta fase lida com a especificação interna dos agentes, os seus objetivos e capacidades, assim como a especificação mais detalhada da comunicação entre os agentes. Abordagens práticas para essa atividade são geralmente propostas dentro de plataformas específicas de desenvolvimento, pois tem relação estrita com a tecnologia envolvida na implementação do sistema e dos agentes.

4.2.3.5 Considerações

A metodologia TROPOS é uma das mais metodologias mais conhecidas no Brasil para modelagem de agentes, tendo diversos grupos de pesquisa no Brasil e no mundo que trabalham em pesquisas relacionadas ao desenvolvimento da TROPOS. Possui diversas fases e tenta abranger diversos tipos de agentes. Como envolve diversos grupos de pesquisa, possui amplo ferramental para o seu desenvolvimento. No entanto, por ser uma metodologia muito detalhada e complexa, muitas vezes é de difícil aplicação, principalmente quando temos um conjunto de agentes cujas habilidades sejam mais simples. Possui integração com plataformas de desenvolvimento de agentes, o que é uma característica que simplifica, de certa forma, a passagem entre as diversas fases de desenvolvimento.

4.2.4 Prometeus

Prometeus é uma metodologia que define um processo detalhado para especificar, projetar e implementar sistemas de agentes inteligentes. A idéia principal é que deve ser fácil de usar por especialistas e por usuários comuns. A metodologia é focada em sistemas que usam agentes baseados no modelo BDI. Ela pode ser dividida em três fases: Especificação de Sistema, Projeto Arquitetural e Projeto Detalhado

(PADGHAM e WINIKOFF, 2003).

4.2.4.1 Especificação de Sistema

Esta fase de modelagem tem o papel de identificar como os agentes irão interagir com o ambiente no qual estão inseridos. Dois conceitos, que precisam ser definidos, são de fundamental importância neste processo: percepções e ações. As percepções e os eventos que ocorrem dentro do sistema podem ser confundidos, por isto, é importante distinguir bem o que são eventos e o que são percepções. Segundo (PADGHAM e WINIKOFF, 2003), um evento pode ser definido como alguma ocorrência significativa para os agentes, enquanto percepções são conjuntos de dados disponíveis para os agentes utilizarem na tomada de suas decisões.

Em resumo, o que deve ser feito nesta fase é a especificação, por parte do desenvolvedor, de o quê um agente do sistema deve fazer de uma maneira geral - as funcionalidades do sistema. Essas funcionalidades ajudarão no entendimento do sistema.

Quando se definem as funcionalidades é importante também definir quais são as informações que são requeridas e quais informações são produzidas a partir das funcionalidades. A descrição das funcionalidades contém um nome, uma descrição curta em linguagem natural, uma lista de ações, uma lista de percepções relevantes, dados utilizados e gerados e uma breve descrição das interações com outras funcionalidades (PADGHAM e WINIKOFF, 2003).

Nesta fase também temos a criação de cenários de casos de uso. A parte central de um cenário de caso de uso no Prometheus é a seqüência de passos de um exemplo do sistema ao executar uma operação. O modelo de caso de uso contém um número de identificação, uma descrição breve em linguagem natural, um campo opcional chamado contexto que indica quando o cenário que está sendo modelado pode acontecer, o cenário em si, um resumo de todas as informações utilizadas nos vários passos e uma lista de pequenas variações (PADGHAM e WINIKOFF, 2003).

4.2.4.2 Projeto Arquitetural

O papel principal desta fase está em identificar quais agentes devem existir de acordo com as interações existentes e a lista de funcionalidades levantadas na fase anterior. O processo de identificação destes agentes está intimamente ligado ao agrupamento de funcionalidades. Este agrupamento pode acontecer por conta de diversas funcionalidades utilizarem um mesmo fluxo de dados, ou porque possuem interações relevantes entre elas. O fato é que precisamos buscar agentes que tenham uma forte coerência entre si.

Um dos artefatos gerados nesta fase é um diagrama de interações entre agentes que simplesmente liga cada agente com os outros agentes que interagem entre si. A figura 4.4 mostra um exemplo deste diagrama (PADGHAM e WINIKOFF, 2003).

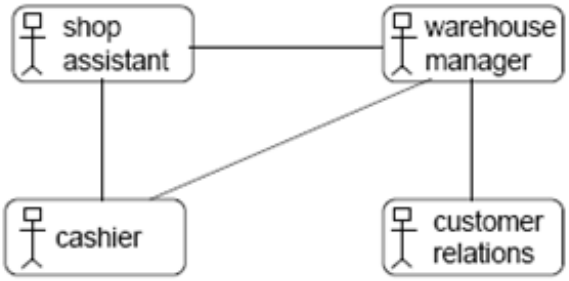

Figura 4.4: Diagrama de interações entre agentes (PADGHAM e WINIKOFF, 2003)

Após a definição de quais agentes deverão existir, é necessário descrever ainda em um alto nível de abstração (assim como foram descritas as funcionalidades) informações importantes e relevantes sobre os agentes como: cardinalidade, tempo de vida do agente, criação/destruição do agente durante a execução do sistema, inicialização do agente entre outras. Basicamente o descritor do agente conterá as seguintes informações: Nome, Descrição, Cardinalidade, Funcionalidades incluídas, Dados que são lidos, Dados que são gerados, Interações ativas.

Nesta fase ainda temos a definição de dados distribuídos (informações compartilhadas ao mesmo tempo por vários agentes, manter consistências, etc), o diagrama de visão geral do sistema que mostra funcionalidades, agentes, dados entre outras coisas, os diagramas de interação (diagrama de seqüência) e por um diagrama de protocolos que utiliza a notação da AUML como padrão.

4.2.4.3 Projeto Detalhado

O foco desta fase está no desenvolvimento da estrutura interna de cada um dos agentes e como eles irão realizar suas tarefas dentro do sistema (PADGHAM e WINIKOFF, 2003).

Esta fase está intimamente ligada as plataformas de implementação de agentes como PRS, dMARS, JAM ou JACK. Principalmente ao mapeamento do modelo BDI através dos planos de execução definidos pelo usuário, disparados pelos objetivos ou eventos. E apesar de termos várias plataformas diferentes, a metodologia tem condições de detalhar todo este mapeamento do modelo para a implementação.

O que será definido nesta fase são as capacidades (módulos dentro dos agentes), eventos internos, planos e estruturas de dados mais detalhadas. É utilizado um processo de refinamento sucessivo sobre a definição destas capacidades dos agentes.

Em suma, cada capacidade identificada deve ser descrita por um descritor que contém informações sobre as interfaces externas desta capacidade (quais eventos servem como entrada para o processamento e quais os eventos são gerados como saída). Contém também uma descrição em linguagem natural da funcionalidade principal, um nome descritivo único, informações sobre as interações com outras capacidades, ou inclusões de outras capacidades, e referências aos dados que são lidos e/ou gravados por aquela capacidade.

Há também um diagrama que representa uma visão geral do agente e das suas estruturas internas. O diagrama mostra as capacidades dos agentes em um nível de abstração mais alto e o fluxo dos eventos ou tarefas, entre estas capacidades.

Um último artefato gerado nesta fase, à partir de toda representação interna dos agentes, das suas capacidades, interações, eventos e planos, é o plano individual propriamente dito, o plano de eventos e os descritores de dados. Estas descrições é que irão guiar diretamente as implementações dos agentes.

Basicamente, os descritores dos planos provêem um identificador, o tipo de evento disparado, o passo-a-passo do plano em si, assim como uma descrição em linguagem natural e uma especificação do contexto, indicando quando este plano deve ser usado e uma listagem dos dados que serão lidos e gravados.

Os descritores dos eventos precisam de um propósito para o evento e todos os dados que o evento utiliza. Há ainda duas características que podem ser descritas: se o evento está coberto (quando existe sempre pelo menos um plano que leva a um resultado verdadeiro no contexto) e se o evento é não-ambíguo (quando existe no máximo um plano aplicável àquele evento).

E por último, os descritores dos dados devem especificar os campos e métodos de todas as classes usadas para armazenamento de informações dentro do sistema. Opcionalmente pode ser construído também um dicionário de dados para garantir um uso consistente dos nomes utilizados desde as fases iniciais do processo de modelagem.

4.2.4.4 Considerações

A metodologia Prometeus possui um bom detalhamento de todas as fases do desenvolvimento. No entanto, por ser uma metodologia muito detalhada, é de difícil aplicação, além disso, não possui integração com a plataforma JADE.

4.2.5 MAS-CommonKADS

A metodologia MAS-CommonKADS foi desenvolvida baseada em uma outra metodologia chamada CommonKADS (SCHREIBER e et al., 1999) da área de desenvolvimento de sistemas baseados em conhecimento, englobando os aspectos relevantes ao desenvolvimento de sistemas multiagentes.

O modelo CommonKADS puro tem como ponto central o Modelo de Experiência (além de outros modelos) que visa modelar o conhecimento de resolução de problemas empregado por um agente para realizar uma tarefa.

A partir de algumas considerações feitas e análise de alguns problemas na aplicação direta do método CommonKADS no desenvolvimento de sistemas multi-agentes, surgiu a proposta MAS-CommonKADS. A metodologia define sete modelos principais: Modelo de Agentes, Modelo de Organização, Modelo de Tarefas, Modelo de Experiência, Modelo de Comunicação, Modelo de Coordenação e Modelo de Design. Estes modelos são gerados dentro de três fases distintas: Fase de Conceituação, Fase

de Análise e Fase de Projeto e Protótipo.

4.2.5.1 Fase de Conceituação

Esta fase tem como objetivo levantar uma primeira descrição do problema e do sistema que será desenvolvido. Basicamente, são realizadas as seguintes atividades: descrição do problema, identificação dos atores, identificação e descrição dos casos de uso, definição dos diagramas de casos de uso e dos diagramas de seqüência de mensagens.

4.2.5.2 Fase de Análise

Na fase de Análise se encaixam os modelos de Agentes, Tarefas, Conhecimento, Organização, Coordenação e Comunicação.

O modelo de agentes descreve os agentes que participam da solução dos problemas, que são descritos nas planilhas de CRC (Classes-Responsabilidades-Colaborações), nos diagramas de casos de uso interno (extensão do conceito de casos de uso do UML para agentes humanos e agentes de software), nos diagramas de seqüência de mensagens relativas aos casos de uso internos, nas planilhas de Agentes e na tabela de Distribuição de Tarefas e Agentes.

O Modelo de Tarefas é um modelo elaborado na fase de Análise e permite mostrar a decomposição funcional do sistema. Para representar a decomposição das tarefas foram construídos diagramas de fluxo de dados que permitem mostrar de forma mais explícita a relação entre as tarefas e as informações. A descrição das tarefas se completou na identificação e descrição de seus objetivos. Esse detalhamento contém os parâmetros de entrada e saída, as condições de ativação e finalização, além da descrição do processamento e tipo de objetivo.

O Modelo de Conhecimento foi, também, definido na fase de Análise, sendo composto das Estruturas de Domínio, Inferências e de Tarefas. A Estrutura de Domínio representa o conhecimento declarativo do problema, descrevendo as entidades relevantes do domínio e as relações entre essas entidades. A Estrutura de Inferência foi especificada para os agentes inteligentes mediante a definição das inferências que são realizadas para a resolução do problema e da estrutura que relaciona as inferências e

os papéis do conhecimento. A Estrutura de Tarefas representa de forma procedural as primitivas definidas na estrutura de inferência.

O Modelo de Organização construído permitiu analisar as relações estruturais entre os agentes, tanto de software quanto humanos, que interagem com o sistema. Finalmente, o último modelo elaborado na fase de Análise foi o Modelo de Coordenação cujo principal objetivo foi definir as interações entre agentes, permitindo um estudo mais aprofundado das interações homem-máquina e máquina-máquina. Para cada interação foi definido um Diagrama de Fluxo de Eventos, descrevendo como os agentes processam as mensagens recebidas e como enviam a mensagem através dos Diagramas de Transição de Estado na notação do UML.

4.2.5.3 Fase de Projeto e Protótipo

O Modelo de Design tem como propósito a descrição dos componentes que cumprem os requisitos dos modelos de análise e que levam em conta os requisitos funcionais.

4.2.5.4 Considerações

A metodologia MAS-CommonKADS é uma das mais conhecidas da comunidade (na sua versão anterior) e ainda a mais usada. No entanto, por ser bastante complexa, é de difícil aplicação, além de não ser de fácil interação com as plataformas atuais de desenvolvimento de agentes, como é o caso de JADE.

4.2.6 INGENIAS

O grupo de pesquisa chamado grasia! (http://grasia.fdi.ucm.es) desenvolveu a metodologia INGENIAS (SANZ, 2002) a partir de uma análise das metodologias de desenvolvimento de sistemas multi-agentes já existentes, onde verificou que nem todos os aspectos eram cobertos no momento de desenvolver a aplicação. Entretanto, as fases de extração dos objetivos, definição e verificação dos protocolos de comunicação ou execução das tarefas, por exemplo, eram contemplados em parte pelas metodologias analisadas.

INGENIAS foi baseada em conceitos da engenharia como a definição de workflows, encapsulamento das funcionalidades através de papéis, grupos e organizações, e ainda, a possibilidade de modelar diferentes visões do sistema tanto estruturais como funcionais. A metodologia leva em consideração todo o ciclo de vida do desenvolvimento de um software: análise, projeto e implementação. Todos os esforços despendidos neste desenvolvimento são gerenciados através do Unified Software Development Process (USDP) (RUMBAUGH *et al.*, 1999).

A INGENIAS faz reuso de uma metodologia já existente, MESSAGE/UML (EVANS *et al.*, 2001), escolhida por apresentar uma visão bem próxima a da engenharia no que diz respeito ao projeto de um sistema multi-agente. A MESSAGE/UML define cinco tipos de visões do sistema baseadas em uma linguagem de meta-modelagem, visões estas, aplicadas ao longo de todo o ciclo de desenvolvimento para representar os aspectos concretos do sistema. A INGENIAS além de reusar as visões existentes na MESSAGE/UML, também a estende com uma nova visão (visão do ambiente), reconstrói as visões para atender ao modelo BDI (*Belief-Desire-Intention*), faz integração com o USDP e ainda provê ferramentas de modelagem, documentação do sistema e geração de código automática dos agentes para o projeto.

A abordagem geral da metodologia descrita para especificar um SMA é dividir o problema em aspectos mais concretos (PAVÓN *et al.*, 2005), que juntos formam diferentes visões do sistema. A idéia não é nova e a diferença para outras metodologias é como as visões são construídas e como são integradas no desenvolvimento do SMA. Para que as diversas visões sejam construídas e os modelos que as representam sejam gerados, uma linguagem de meta-modelo é utilizada, GOPRR (Graph, Object, Property, Relationship, and Role) (LYYTINEN e ROSSI, 1996). O meta-modelo funciona como uma gramática que especifica que estruturas existem e podem ser utilizadas na construção dos modelos que são efetivamente as visões que os engenheiros irão fazer. Na metodologia INGENIAS existem cinco meta-modelos que originaram as cinco visões para o desenvolvimento de um SMA:

- Modelo de Agentes;
- Modelo de Interação;
- Modelo de Metas e Objetivos;

- Modelo Organizacional;
- Modelo do Ambiente.

4.2.6.1 Modelo de Agentes

O modelo de agentes visa descrever os agentes em particular, excluindo, portanto, suas interações com outros agentes. O objetivo principal é descrever a funcionalidade do agente e o projeto do seu controle. Para atingir o objetivo, os seguintes aspectos são modelados e analisados:

- Responsabilidades: representadas pelas tarefas que o agente saberá executar e dos objetivos que se compromete a alcançar. Usa-se geralmente o termo role (papel) para agrupar as funcionalidades e as propriedades que aparecem com freqüência no projeto do agente;
- Comportamento: este termo denota várias interpretações nas diferentes metodologias existentes. A UML avalia comportamento como um conjunto de chamadas a procedimentos, MaSE e MESSAGE a troca de mensagens entre os agentes e até transições em máquinas de estado são avaliadas como "comportamento"de um agente. A INGENIAS relaciona comportamento ao controle do agente, mais precisamente, mediante quais mecanismos a execução das tarefas será assegurada dentro dos parâmetros estabelecidos.

4.2.6.2 Modelo de Interação

O papel das interações dentro de um SMA é fundamental, pois identifica dependências entre os componentes do sistema e contribuem para a especificação do comportamento dos mesmos assim como a funcionalidade associada. As interações determinam o comportamento dos agentes exibindo a reação durante suas atuações dentro do SMA. Como o comportamento é em função dos objetivos e tarefas a executar, fica explícito a ligação entre as interações e os objetivos e tarefas de cada agente. A especificação completa de uma interação deve cobrir os seguintes itens:

- Atores participantes: um ator deve mostrar porque está participando da interação. É coerente com um modelo de agente que se baseia no princípio da

racionalidade;

- Unidade de interação: a natureza da unidade de interação determina como o receptor deve ser processado. Pode ser simples como um passo da mensagem como pode ser complexa por adequar as interações de acordo com as capacidades do receptor;

- Ordem sobre as unidades de interação: as unidades de interação se organizam seguindo um protocolo padrão adequado para uma situação concreta. Este protocolo é responsável em organizar a interação e sua execução;

- Ações executadas na interação:

  - Critérios para decidir quando executar uma tarefa: para executar uma tarefa não é simplesmente o agente aceitá-la. Ele precisa ter autonomia de negar a solicitação, avaliar se pode abandonar uma tarefa já em execução e assumir esta nova ou, avaliadas estas situações, assumir efetivamente a tarefa;

  - Conseqüências da execução de uma tarefa: ao final da execução de uma tarefa esperam-se mudanças no estado do agente e do ambiente no qual ele está inserido;

- Definição do contexto de uma interação: o contexto consiste em mostrar o que ocorre com o sistema quando se inicia uma interação. Esta informação demonstra qual o estado necessário dos agentes para que sejam acionados durante a interação, indicando os atores participantes, os motivos que levaram estes agentes a participarem, e que objetivos são perseguidos pela interação;

- Modelo de controle: O controle assegura que a interação vai ser realizada da maneira que foi definida. Este controle deverá levar em consideração que várias interações podem ser realizadas em paralelo, assumindo assim um papel central na coordenação das interações.

4.2.6.3 Modelo de Metas e Objetivos

Este modelo tem como objetivo principal representar as motivações do SMA, definir as ações identificadas nos modelos de organização, interações e agentes e

como afetam estas ações e suas responsabilidades. Através deste modelo, têm-se parte da especificação em alto nível de como será o controle do agente. Trata-se de poder expressar quais são as conseqüências de executar as tarefas e o porquê de executá-las.

A associação de objetivos e tarefas é baseada no modelo clássico BDI. Segundo este mesmo modelo, os seres humanos tendem a focar suas ações seguindo uma ordenação concreta: Qual é a situação atual? O que está buscando (objetivos)? E como podem ser alcançados (intenções)? A intenção de associar as tarefas é por estar associada diretamente ao estado dos agentes e suas responsabilidades, por isto a ligação estreita com os objetivos.

Nesta metodologia, a tarefa tem dois papéis principais: transformar o estado global e de processo. A primeira está relacionada à representação das pré-condições e pós-condições dos agentes que permite o planning e tarefas de raciocínio. Como processo, ela representa uma estrutura pragmática e direta: um conjunto de instruções que devem ser executadas.

Os objetivos na metodologia são vistos como alternativas que se apresentam aos agentes em um dado momento. Para representá-las existem duas tendências:

- Objetivo como agregação: é o enfoque predominante. O objetivo é visto como uma descrição do estado do mundo que deve ser alcançado. Quando a descrição do estado do mundo se faz com um conjunto de predicados, o objetivo se converte em uma agregação destes predicados;
- Objetivos como entidades: é o enfoque usado no clássico BDI. Não se trata de agregações de predicados e sim de entidades auto-representativas. Seguindo o modelo BDI, trata-se de um desejo que queira satisfazer.

4.2.6.4 Modelo de Organização

O modelo de organização é equivalente à arquitetura de um sistema quando estamos lidando com SMA. O valor principal de um modelo de organização, como ocorre nas organizações humanas, são os fluxos de trabalho existentes. Do estudo destes fluxos surgem novas interações que refletem com detalhes como se coordena os participantes do fluxo. O modelo de organização também contribui com o modelo

de tarefas e objetivos identificando as tarefas relevantes para a organização assim como os objetivos que são perseguidos globalmente. Também define restrições no comportamento dos agentes mediante relações como a subordinação. Com estas restrições, o projetista assegura que alguns agentes obedecerão outros respeitando a prioridade na execução das tarefas relacionadas.

A partir da metodologia MESSAGE três pilares foram construídos para dar suporte aos modelos de organização:

- Distinção entre agentes e organizações: as organizações são compostas por agentes. As mesmas só existem mediante a existência de agentes que as compõem. Porém, os agentes podem existir sem as organizações. Como nas organizações humanas, os agentes sempre são os responsáveis finais por executar tarefas dentro de uma organização;
- A organização delega sua estruturação em uma estrutura organizacional: não há relações de agregação entre as organizações. Existem diferenças entre a Organização e a Estrutura Organizacional. A Organização tem um conjunto de atributos que não existem na Estrutura Organizacional. O ciclo de vida também é diferente. As Estruturas Organizacionais se criam e se destroem dentro de uma Organização sem que esta deixe de existir. Os agentes e os recursos sempre pertencem à Organização apesar de serem disponibilizados dinamicamente a diferentes Estruturas Organizacionais. Por último, se uma Organização é destruída, consequentemente todas as suas Estruturas Organizacionais também o são.
- Fluxo de trabalho: permite contextualizar a execução das tarefas e interrelacioná-las umas com as outras, independente de quem são os executores da mesma.

4.2.6.5 Modelo de Ambiente

Neste modelo o objetivo é gerar representações do ambiente que o SMA se encontra. Em (RUSSEL e NORVIG, 1995) mostra-se que os ambientes podem ser acessíveis (capacidade de perceber o ambiente) ou inacessíveis; determinísticos (dada uma codificação do estado do ambiente e uma ação executada pode-se saber o estado seguinte) ou não-determinísticos; estático ou dinâmico e, finalmente, discreto

ou contínuo. Isto mostra que o ambiente influencia diretamente na percepção do agente, suas ações e controle (SANZ, 2002).

Os mecanismos de percepção dos agentes são funções diretas de como se descrevem as entidades do sistema. A literatura mostra que há dois mecanismos básicos: pesquisa e notificação. Este meta-modelo de ambiente define as associações dos agentes com as entidades externas e o tipo de mecanismo para receber dados de retorno. As ações sobre o ambiente assumem o que são chamadas as operações que definem estas entidades.

A evolução do ambiente é a soma da evolução de seus componentes (recursos, entidades e agentes) influenciada pelas dependências entre eles (uma ação sobre um determinado recurso pode afetar outras entidades). Em suma, o modelo define a percepção dos agentes em termos dos elementos existentes no SMA. Também identifica os recursos do sistema e os responsáveis pelo seu gerenciamento.

4.2.6.6 Considerações

A metodologia INGENIAS é relativamente nova e ainda não muito utilizada. Tem como principal vantagem a disponibilidade de uma ferramenta de modelagem específica para os seus padrões que permite inclusive a integração com a plataforma de desenvolvimento de agentes JADE através da geração de código automática para o comportamento dos agentes a partir dos modelos desenvolvidos na ferramenta. Outro ponto importante a ser considerado é a modelagem do ambiente, onde são modeladas em detalhes as interfaces dos agentes com o ambiente externo. Este aspecto é particularmente interessante para o SASAgent, uma vez que utiliza serviços web semânticos interagindo diretamente com agentes e fazendo o papel destes em alguns casos. Neste contexto, a modelagem das interfaces se mostra preponderante.

4.2.7 Comparação entre as Metodologias

Há vários artigos publicados que fizeram análises criteriosas sobre diversas metodologias para desenvolvimento de sistemas baseadas em agentes como (SANTOS, 2006), (DAM e WINIKOFF, 2003) e uma avaliação específica para o método MAS-CommonKADS (WERNECK e et al., 2006).

O trabalho de (SANTOS, 2006) levou em consideração pontos como: definição do processo do ciclo de vida (incremental, iterativo ou evolucionário), utilização da UML como base já que é um padrão bem estabelecido e bem conhecido, captura das características específicas dos agentes (mobilidade, aprendizado, adaptação entre outros), estrutura organizacional (representação da sociedade dos agentes) entre outras características.

O trabalho apresentado em (DAM e WINIKOFF, 2003) levou em consideração quatro pontos principais que foram refinados: conceitos relacionados aos agentes, linguagem para modelagem (seja gráfica, textual ou ambas), processo (ciclo de vida de desenvolvimento), impacto no gerenciamento do processo (custos para adoção da metodologia, impacto sobre as práticas de negócio atuais da empresa, maturidade, etc) e características relacionadas aos aspectos técnicos do desenvolvimento (se a metodologia é aplicável a diversos domínios ou só a domínios específicos, independência de tecnologia, escalabilidade, etc).

Estas análises, no entanto, não ressaltam aspectos importantes na modelagem de agentes tais como:

- Identificação de tipos de agentes que comporão a aplicação. Este é um aspecto importante, uma vez que podemos ter modelos extremamente complexos de agentes e outros que são mais simples e que, portanto, não necessitam de metodologias mais elaboradas para sua modelagem;
- Modelagem da interação dos agentes com o ambiente. Esta característica se torna importante, principalmente no contexto da Web Semântica, onde o conceito de serviços web semânticos muitas vezes se confunde com o conceito de agentes.
- Integração com plataformas de desenvolvimento de agentes, uma vez que a complexidade no desenvolvimento de sistemas multi-agentes cresce a cada dia e a possibilidade de geração automática de código a partir de modelos especificados ajuda a reduzir esta complexidade.
- Modelagem da base de conhecimento a ser utilizada pelos agentes: novamente, com a Web Semântica, o uso de metadados semânticos aumenta em muito o conhecimento que o agente pode ter de seus objetivos e do ambiente. Assim,

utilizar metodologias que incluam em suas atividades modelos relacionados à base de conhecimento e sua integração com os agentes torna-se preponderante.

O que se pode concluir através dos comparativos e das questões levantadas acima é que não há nenhuma metodologia que se sobressaia sobre as outras e que tenha se tornado um padrão estabelecido ou mesmo padrão de utilização. Há pontos falhos em todas elas. A escolha depende exclusivamente do que o grupo conhece ou pretende desenvolver.

Capítulo 5

SASAgent: Arquitetura Orientada a Agentes para Gerenciamento, Busca e Recuperação de Artefatos Científicos

5.1 Introdução

A arquitetura para gerenciamento, busca e recuperação de artefatos científicos, denominada ***SASAgent*** (*Scientific Artefact Search Agent*) é baseada em três características: os agentes do sistema com seus respectivos papéis, a base de conhecimento que, através das ontologias, permitirá a realização de inferências lógicas extraindo novas informações que por ventura estejam implícitas e, finalmente, os artefatos científicos.

Os agentes que compõem a arquitetura são baseados em três características básicas: reatividade (reagir mediante a um estímulo: no caso as mensagens que chegam para cada agente), pró-atividade (ação por parte do agente sem que haja tido um evento que a tenha disparado, por exemplo, o monitoramento da disponibilidade dos artefatos) e por último a habilidade social (capacidade de o agente interagir com outros agentes da "sociedade"na qual ele está inserido).

A base de conhecimento é fundamentada no uso de ontologias. Optou-se por utilizar ontologias, uma vez que o uso de bancos de dados relacionais limitaria a representatividade e a capacidade de expressão necessária a execução do *SASAgent*. Desta forma, o conhecimento será representado na forma de uma ontologia.

É fato que os grupos de pesquisa, sejam eles de qualquer domínio da ciência, geram diversos produtos a partir das suas pesquisas que vão desde publicações (artigos, relatórios técnicos, livros, dissertações de mestrado, teses de doutorado), ferramentas computacionais (editores, validadores de arquivos, algoritmos otimizados, aplicações, Web Services) e ainda, modelos que representam uma parte do universo do domínio em questão. Estes modelos podem ser ontologias, modelos de banco de dados, diagramas UML ou ainda modelos específicos de um domínio como modelos representativos de fenômenos da física, matemática, química ou biologia. Para organizar estes artefatos científicos, permitindo sua utilização, compartilhamento, gerenciamento e pesquisa dos mesmos, é necessária uma arquitetura que seja expansível e fácil de ser integrada com outras aplicações.

Desta forma, podemos destacar como principal objetivo da *SASAgent* o gerenciamento, busca e recuperação de artefatos científicos gerados pelos grupos de pesquisa de domínios relacionados a modelagem computacional, tais como Engenharia, Física, Química e Bioinformática, através do uso de agentes inteligentes e uma base de conhecimento semântica a partir do uso de ontologias.

Este objetivo principal pode ser decomposto em um conjunto de objetivos secundários:

- Implementação de uma camada que se constitui de um sistema multiagentes que servirá como mediadora entre a base de conhecimento, os artefatos científicos e os pesquisadores que desejam utilizar a arquitetura. Este sistema consistirá das seguintes características:
  - Agentes responsáveis pela interface com o usuário final;
  - Agentes responsáveis por inserir, buscar e recuperar os artefatos científicos e conseqüentemente acessar a base de dados semântica;
  - Agentes responsáveis por interagir diretamente com alguns artefatos científicos como os Web Services;

  - Agentes responsáveis pelo monitoramento dos artefatos;

- Existência de uma base de conhecimento que através do uso de ontologias represente o conhecimento necessário para dar suporte ao gerenciamento dos artefatos permitindo buscas, cadastro e monitoramento dos mesmos;

- Independência do domínio de aplicação dos grupos de pesquisa, ou seja, a *SASAgent* deve servir a qualquer grupo de pesquisa que gere artefatos científicos que atendam a demanda de áreas como Engenharia, Bioinformática e Física dentre outras;

- Dar suporte a diferentes papéis que o usuário final (cientista) do sistema pode assumir: ora como consumidor de artefatos, ora como provedor dos mesmos, permitindo ainda, que o usuário tenha a capacidade de montar seu próprio perfil, com as principais preferências em relação às instituições de pesquisa, artefatos, áreas de interesse, dentre outros pontos.

5.2 Projeto e Especificação

No Capítulo 4 foram apresentadas diversas metodologias existentes na área de modelagem de sistemas baseados em agentes: MASUP (BASTOS e RIBEIRO, 2005), MaSE (WOOD e DELOACH, 2000), Prometeus (PADGHAM e WINIKOFF, 2003), MAS-CommonKADS (IGLESIAS, 1998), (IGLESIAS e GARIJO, 1999), (IGLESIAS e GARIJO, 2005) e INGENIAS (SANZ, 2002). Dentre todas elas, a metodologia INGENIAS foi escolhida para modelagem da arquitetura proposta pelos seguintes fatores: (1) as características dos agentes (reatividade, pró-atividade e habilidade social) que compõem a arquitetura se adequam às características oferecidas pela metodologia, (2) a existência de uma ferramenta própria de modelagem que atende ao seu metamodelo e (3) por permitir a geração de código dos agentes modelados na própria ferramenta diretamente para a plataforma JADE (*Java Agent Development Framework*), que foi a plataforma escolhida para gerenciar os agentes da arquitetura.

A arquitetura dos agentes projetada a partir das diretivas da metodologia INGENIAS representa os requisitos do sistema e considera o uso de seis modelos: diagramas de casos de uso (do mais geral para os mais específicos), modelo de organização (visão da estrutura geral do SMA incluindo os papéis dos agentes, relações

das características de cada um e o fluxo de trabalho), modelo de agente (representando as tarefas que os agentes devem realizar para alcançar seus objetivos), modelo de objetivos/tarefas (identifica todos os objetivos, tarefas gerais e específicas existentes na arquitetura que podem ser associadas aos agentes), modelo de interação (representando a interação dos agentes com seus respectivos papéis) e um modelo do ambiente externo (especificando as entidades externas existentes e como se relacionam com o SMA). Detalhamos a seguir os modelos resultantes da utilização da metodologia INGENIAS para a modelagem da arquitetura *SASAgent*.

5.2.1 Diagramas de Caso de Uso

Os diagramas de casos de uso da arquitetura proposta foram divididos em duas categorias: gerais e específicos. Para os diagramas específicos é importante salientar que os usuários (pesquisadores) se dividem em dois grupos distintos: consumidores e provedores. Estes dois grupos podem ser entendidos como papéis que cada pesquisador pode assumir dentro da plataforma.

No momento que um pesquisador busca por artefatos científicos que atendem a seus critérios de pesquisa e/ou ao seu perfil de preferências ele assume o papel do consumidor. Por outro lado, no momento que ele adiciona novos artefatos à plataforma, sejam eles publicações, algoritmos, bibliotecas ou serviços Web ele assume o papel do provedor de artefatos. Obviamente, não há restrições para o pesquisador assumir ora seu papel de consumidor, ora de provedor. Com isto não temos nunca um pesquisador que seja exclusivamente consumidor ou que seja exclusivamente provedor de artefatos científicos.

Os diagramas de uso identificados na arquitetura são:

- Geral: Os atores são os pesquisadores que ora agem como consumidores, ora agem como provedores dos artefatos científicos. O principal caso de uso engloba o gerenciamento, busca e recuperação destes artefatos. Dentre eles podemos citar: artigos publicados, eventos, livros, teses, dissertações, códigos de aplicações stand-alone, serviços Web, dentre outros (figura 5.1);
- Específico 1: Pesquisadores (consumidores) buscam por artefatos científicos que atendem aos seus critérios de pesquisa, preferências (através do seu perfil)

ou ainda através de inferência realizada semanticamente (figura 5.2);

- Especifico 2: Pesquisadores (consumidores) ao buscar por artefatos científicos e encontrar o serviço Web que seja do seu interesse em usar, por exemplo, tem condição de invocá-lo para que possa interagir com ele através das suas operações e parâmetros de entrada (figura 5.3);
- Específico 3: Pesquisadores (consumidores) constroem seu perfil de preferências através da escolha de projetos de pesquisa relacionados, pesquisadores de sua preferência, áreas de interesse cujos artefatos científicos estão contextualizados, dentre outros recursos descritos em uma ontologia (figura 5.4);
- Específico 4: Pesquisadores (provedores) publicam seus artefatos científicos gerados no seu próprio grupo de pesquisa na plataforma para que possam ser compartilhados com outros grupos e pesquisadores (figura 5.5).

Figura 5.1: Diagrama de casos de uso Geral

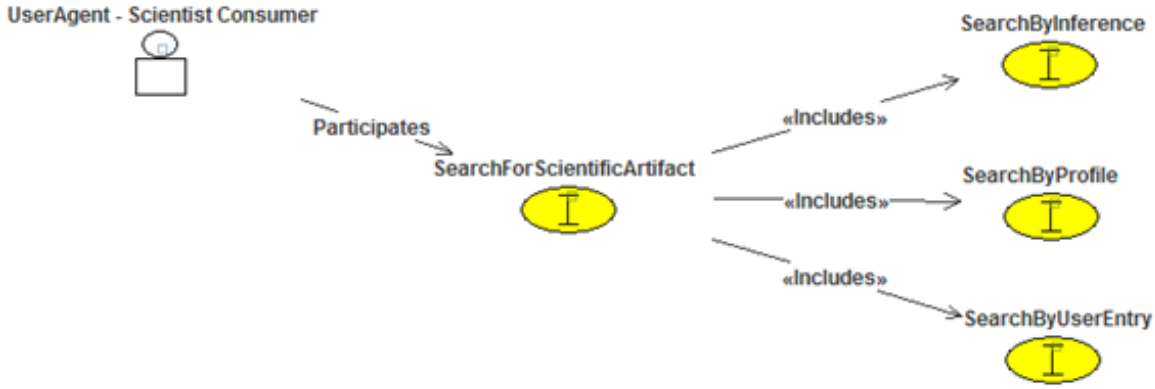

Figura 5.2: Diagrama de casos de uso: específico 1

5.2.2 Modelo de Organização

O diagrama de organização da metodologia INGENIAS é responsável por representar os diferentes componentes do sistema (agentes, papéis, recursos e aplicações),

Figura 5.3: Diagrama de casos de uso: específico 2

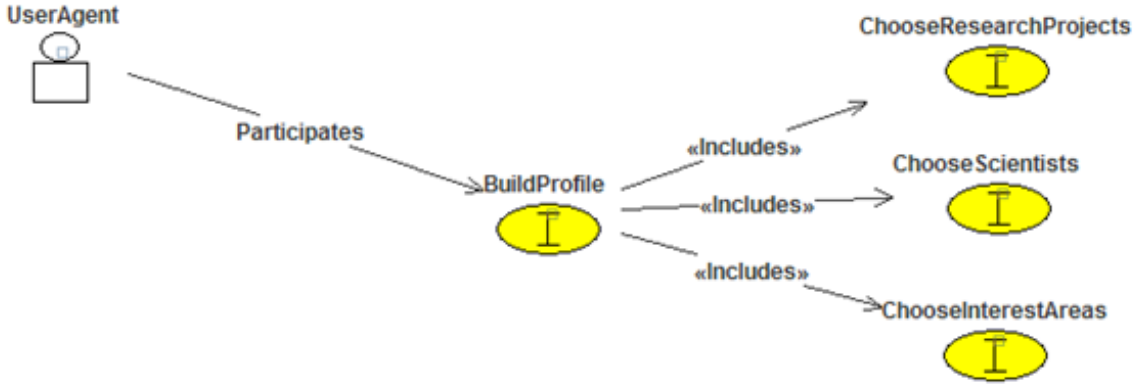

Figura 5.4: Diagrama de casos de uso: específico 3

a funcionalidade do sistema e possíveis restrições que porventura existem sobre as interações entre os agentes.

Na arquitetura proposta neste trabalho foram identificados dois pontos de vista que precisam ser representados, ambos a partir de uma mesma origem: um projeto de pesquisa *ResearchProjectGroup* (figura 5.6). Os cientistas pertencentes a um projeto de pesquisa possuem dois papéis distintos, porém não exclusivos, dentro da arquitetura. Ora podem agir como consumidores de artefatos científicos existentes (*ConsumerScientistGroup*), ora podem assumir o papel de provedores (*ProviderScientistGroup*) destes mesmos artefatos científicos.

Os objetivos principais da arquitetura, e, consequentemente dos projetos de pesquisa, são: prover e consumir artefatos científicos gerados pelos cientistas e pesquisadores, compartilhar e reutilizar conhecimento implícito ou explícito nestes artefatos. Dentre os diversos artefatos científicos que podem ser gerados, podemos citar: arti-

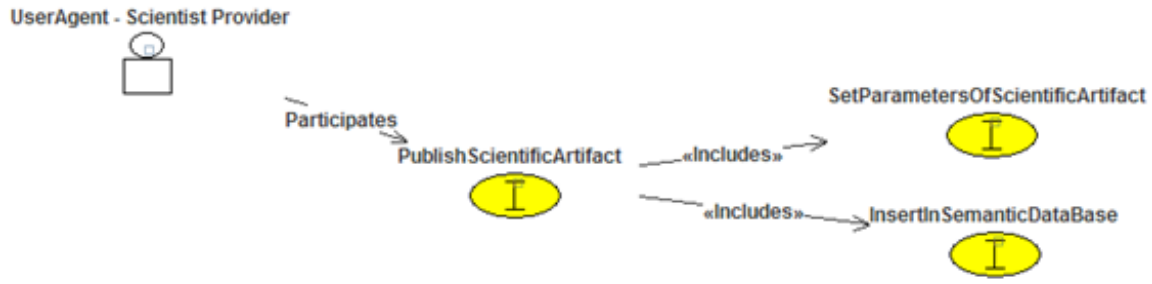

Figura 5.5: Diagrama de casos de uso: específico 4

gos, relatórios técnicos, modelos (biológicos, matemáticos, etc), eventos relacionados (congressos, *workshops*, simpósios, etc), serviços web e algoritmos que representam aplicações stand-alone, por exemplo.

É importante salientar que existe o grupo que representa os agentes do *framework* de suporte a implementação da arquitetura (agentes nativos da plataforma JADE). Estes agentes são os responsáveis pelo gerenciamento de toda a arquitetura e ainda responsáveis em fazer a interação entre o usuário final (pesquisador, cientista) através de uma interface amigável com o *SASAgent*.

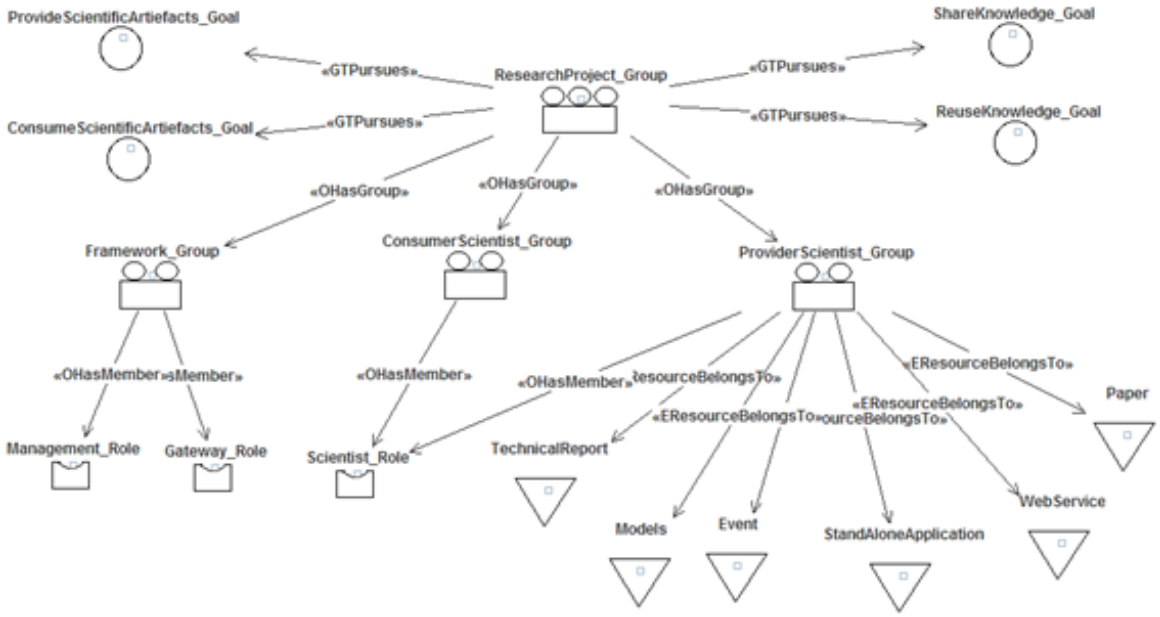

Figura 5.6: Diagrama de Organização Geral

Dentre os principais objetivos da arquitetura proposta está a busca pelos artefatos científicos, diretamente relacionada aos pesquisadores, no papel de consumidores destes artefatos. Para que este objetivo seja alcançado temos 3 papéis que os agen-

tes assumem durante todo o processo: *QueryServiceAgentRole, UserAgentRole* e *InvokerAgentRole*, apresentados na figura 5.7.

O primeiro é diretamente responsável por realizar a busca em si. A pesquisa é realizada sobre um repositório de dados, semanticamente construído, baseado em uma ontologia do domínio, onde todos os conceitos relevantes à arquitetura são modelados e especificados. Esta ontologia será detalhada nas seções posteriores. O *UserAgentRole* é a representação computacional do pesquisador ou cientista participante do projeto de pesquisa que está em busca de artefatos científicos que atendem a seus critérios. E por último, o *InvokerAgentRole* é um agente computacional que entra em ação quando o artefato científico desejado pelo pesquisador é um serviço Web. Através deste agente, o pesquisador consegue consumir qualquer serviço Web disponível na plataforma.

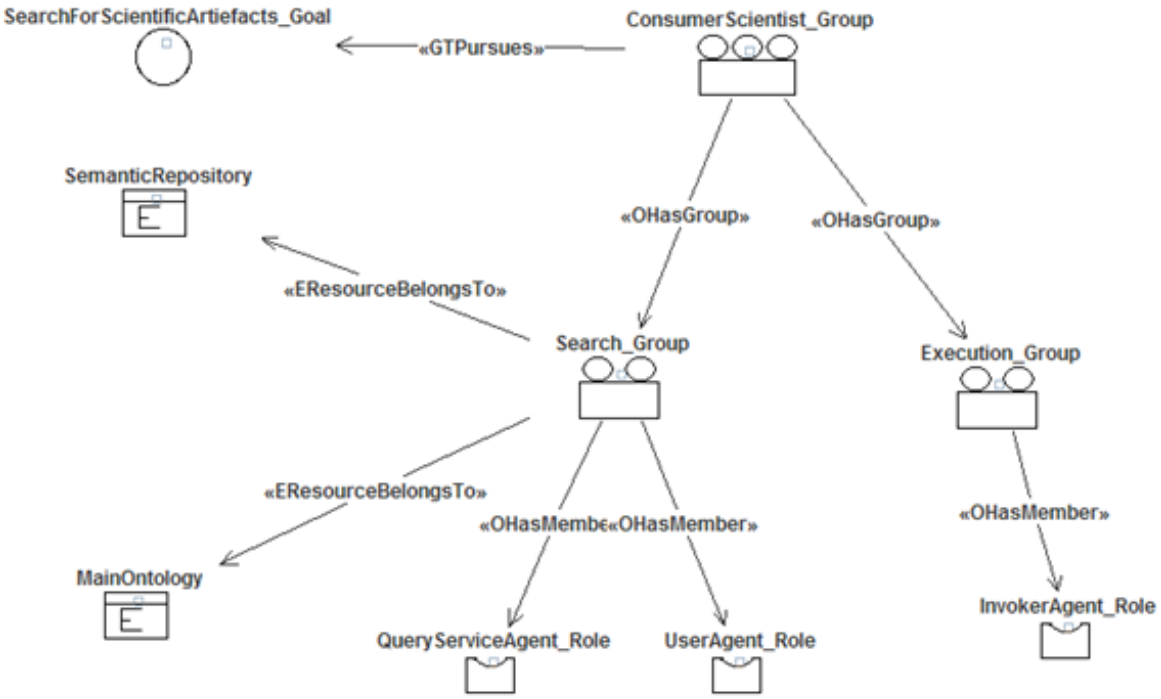

Figura 5.7: Diagrama de Organização Facilitador

5.2.3 Modelo de Agentes

O modelo de agentes prevê a representação dos agentes mais importantes da arquitetura de maneira particular. Seus papéis e seus objetivos desempenhados dentro da arquitetura são o foco destes modelos. Na arquitetura proposta podemos sepa-

rar quatro agentes que merecem destaque pela importância individual: *UserAgent*, *QueryServiceAgent*, *InvokerAgent* e *GatewayAgent*.

O *UserAgent* (figura 5.8) é o agente responsável em representar o pesquisador de um determinado projeto de pesquisa dentro da arquitetura. Este agente pode assumir dois papéis principais: provedor ou consumidor de artefatos científicos. Lembrando que não são papéis exclusivos, ou seja, um mesmo pesquisador pode prover novos artefatos científicos, como também pode ser consumidor de artefatos científicos existentes. Desta formal, estes são os objetivos do UserAgent: prover novos artefatos científicos e utilizar de artefatos já existentes e disponíveis na plataforma.

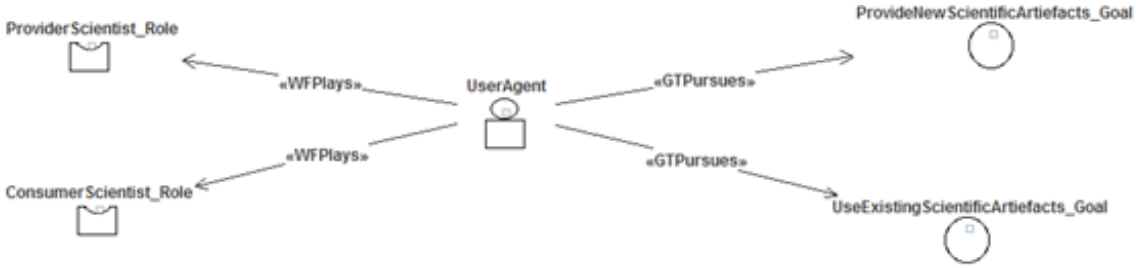

Figura 5.8: Modelo de Agentes UserAgent

Já o *QueryServiceAgent* interage diretamente com o *UserAgent* pois responde às suas solicitações de pesquisa pelos artefatos científicos que atendem a critérios determinados (figura 5.9). Os papéis do agente de busca podem ser três. O primeiro é pesquisar pelo profile do *UserAgent* onde estão as informações sobre instituições de pesquisa, áreas de interesse, pesquisadores preferidos, dentre outras que o pesquisador escolheu. O segundo papel é pesquisar através de inferências lógicas que permitem retornar artefatos científicos que não são tangíveis explicitamente. E por último, uma busca simples e livre através de palavras-chave.

Figura 5.9: Modelo de Agentes QueryServiceAgent

Na figura 5.10 temos a representação do *InvokerAgent*, agente responsável pela invocação de serviços Web que foram encontrados na busca realizada pelo *Query-*

ServiceAgent através dos critérios determinados pelo UserAgent. Este agente é importante uma vez que permite a comunicação com qualquer serviço Web que está publicado na plataforma de busca de maneira transparente ao usuário. A partir do arquivo WSDL que foi encontrado semanticamente na etapa anterior de busca, as operações disponíveis do serviço Web são lidas e oferecidas para que o usuário escolha a que melhor lhe atenda. Após a operação escolhida, os parâmetros de entrada para aquela operação são lidos e também são disponibilizados para o usuário para que o mesmo possa entrar com os valores desejados. A entrada dos valores é a última etapa e finaliza a invocação do serviço Web, obtendo como resposta a saída do serviço que foi gerada a partir da entrada fornecida pelo usuário.

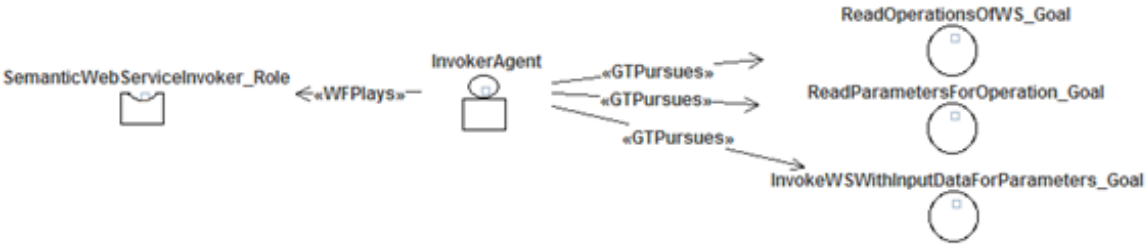

Figura 5.10: Modelo de Agentes InvokerAgent

Finalmente temos o *GatewayAgent* (figura 5.11), responsável pela interação usuário-plataforma. É através dele que todas as informações geradas pela interface da plataforma chegam até o *framework* de agentes para serem tratadas. Por isto, seu papel é ser a ponte entre o usuário final (pesquisador) e o *framework* de agentes e tem a missão de fazer essa ligação com sucesso.

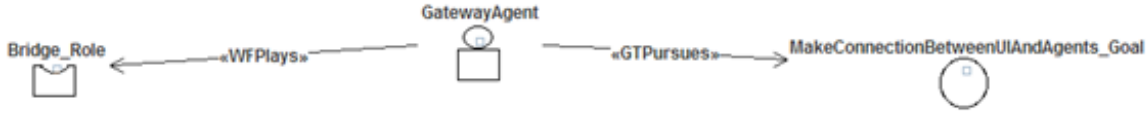

Figura 5.11: Modelo de Agentes GatewayAgent

5.2.4 Modelo de Tarefas e Objetivos

Os diagramas ou modelos de tarefas e objetivos representam as relações entre objetivos e tarefas além das suas estruturas internas. Permitem também indicar as entradas e saídas de cada tarefa e quais são os efeitos das mesmas tanto na arquitetura, de modo geral, como no estado de cada agente.

A figura 5.12 representa o modelo de tarefas e objetivos em nível macro. O objetivo primordial da arquitetura proposta é permitir o compartilhamento de conhecimento através de artefatos científicos (artigos, relatórios, implementações, serviços web, etc) entre grupos de pesquisa dispersos geograficamente e de áreas afins. Este objetivo principal pode ser decomposto em três outros objetivos que precisam ser atendidos pela plataforma: gerenciamento, obter os artefatos científicos necessários e prover novos artefatos científicos.

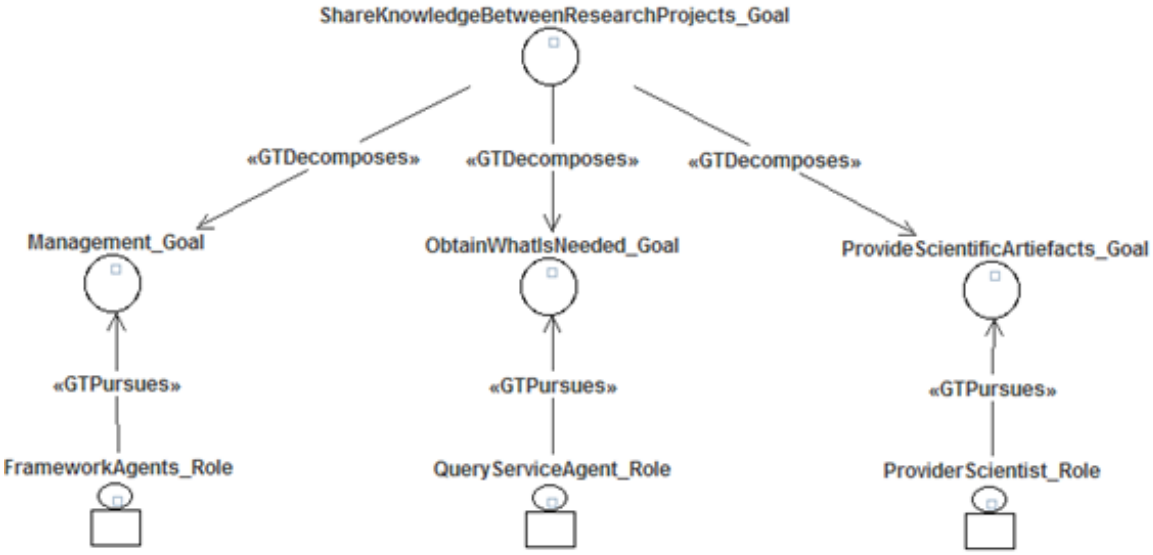

Figura 5.12: Modelo de Tarefas e Objetivos Nível Macro

Na figura 5.13 e apresentado o modelo de tarefas e objetivos que representa e detalha o objetivo de gerenciamento da arquitetura. Este objetivo é garantido pelo *framework* de agentes JADE e tem como função principal gerenciar a entrada e saída dos agentes e fazer a ligação entre a interface da plataforma com o usuário final (pesquisador). Temos para isto três agentes especializados: *AMSAgent*, *DFAgent* e *GatewayAgent*. O GatewayAgent foi detalhado anteriormente (figura 5.11). O *AMSAgent* é o responsável pelo registro dos agentes que entram e saem do framework, semelhante ao serviço de páginas brancas das listas telefônicas. Por último, o *DFAgent* é responsável pelo serviço de páginas amarelas do framework controlando as descrições sobre as funcionalidades disponíveis de cada agente que está sendo executado no framework.

O objetivo de realizar busca por artefatos na arquitetura proposta é permitir que o pesquisador faça uma busca por artefatos científicos que atendam a seus

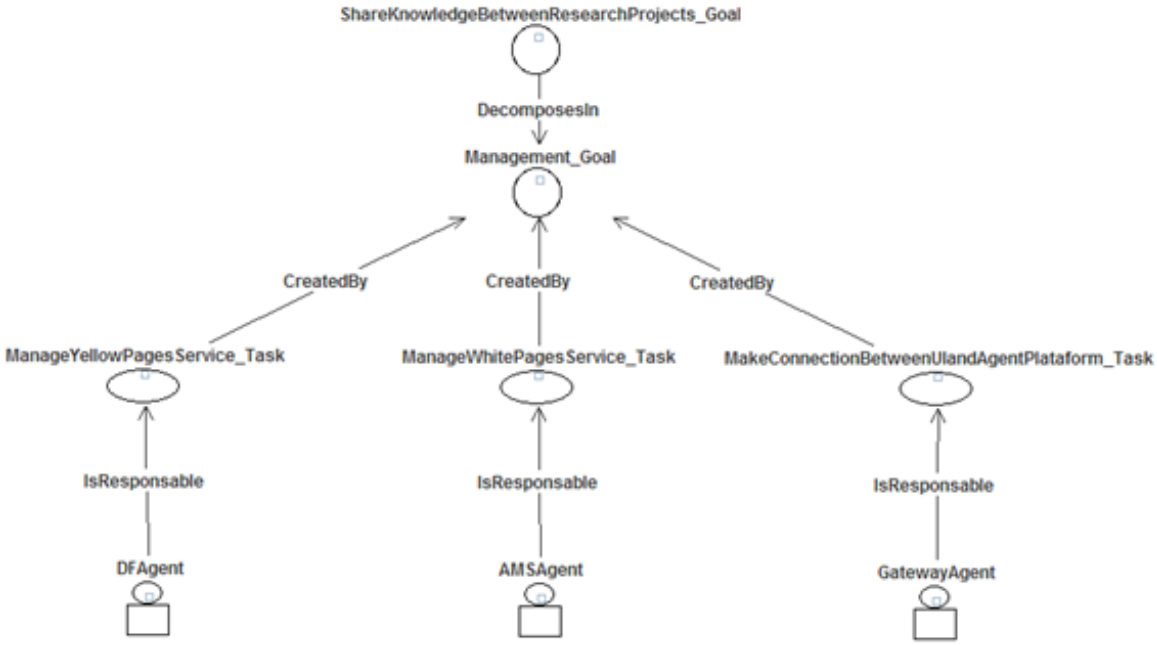

Figura 5.13: Modelo de Tarefas e Objetivos de Gerenciamento

critérios. Estes critérios serão utilizados pelo agente *QueryServiceAgent* no intuito de selecionar os artefatos que atendem às condições iniciais de pesquisa. A busca poderá ser feita de três maneiras diferentes: i) através do perfil de usuário (profile) do pesquisador, com instituições, pesquisadores, áreas de interesse e outras informações que são de sua preferência; ii) através do processo de inferência lógica com o uso da ontologia ou ainda iii) através de uma pesquisa livre e simples por palavras-chave (ver figura 5.14). É importante salientar, que quando os artefatos científicos forem serviços Web, o InvokerAgent entra em ação para permitir a interação do pesquisador com o artefato, de maneira transparente e direta alcançando os resultados desejados.

Como último objetivo da arquitetura, temos o modelo de tarefas e objetivos que provê os artefatos científicos para serem publicados na plataforma (figura 5.15). São duas operações principais: a inserção e sua conseqüente publicação na plataforma e a exclusão do artefato científico. A exclusão é um processo mais simples bastando o pesquisador acessar o artefato científico que seja de sua autoria para a exclusão da base de dados semântica onde o mesmo se encontra. A inserção, que também é realizada pelo próprio pesquisador, representado pelo *UserAgent*, é dividida em três etapas: fornecer os parâmetros de entrada do artefato (por exemplo, de um artigo deve ser fornecido os autores, palavras-chave, resumo e área afim), inseri-lo na base de dados semântica (correlacionando o mesmo com os termos da ontologia)

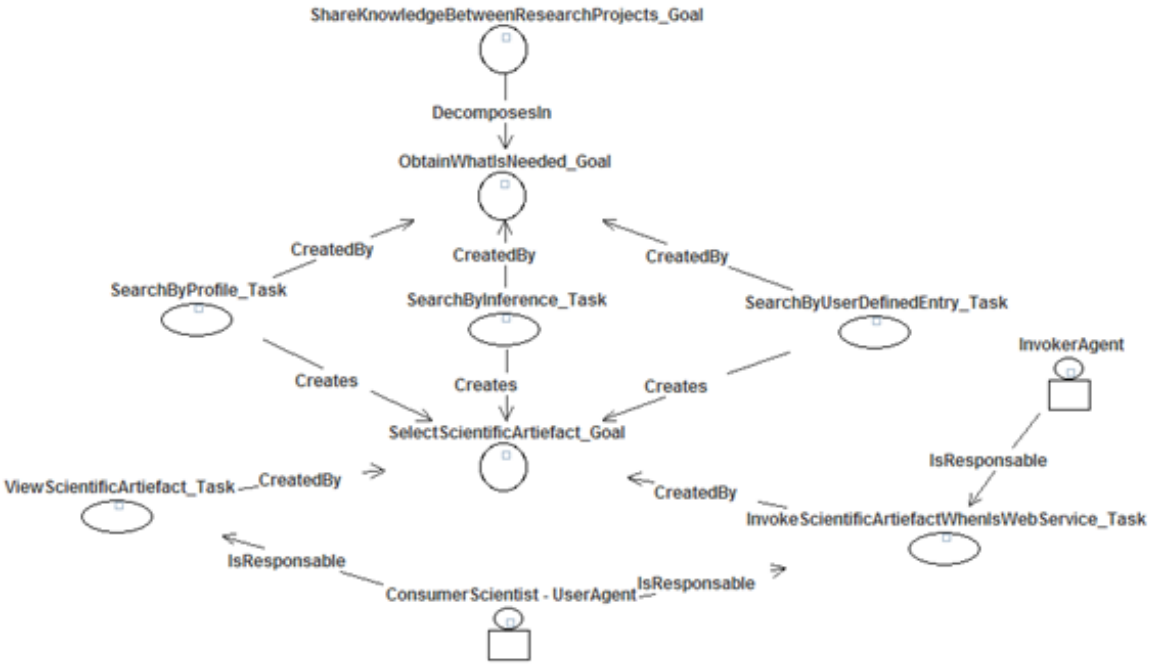

Figura 5.14: Modelo de Tarefas e Objetivos - Atender às necessidades

e torná-lo disponível para consulta, esta última, obviamente é automática a partir da inclusão na base de dados.

5.2.5 Modelo de Interação

O modelo de interação descreve como se dão as interações entre os agentes dentro do *SASAgent*. A descrição do modelo envolve atores e os objetivos que precisam ser alcançados através daquela interação. Com isto, é fácil observar que o modelo de interação possui um relacionamento estrito com o modelo de tarefas e objetivos.

A partir dos modelos de tarefas e objetivos, os modelos de interação foram criados. Os primeiros a serem apresentados são relacionados aos objetivos e tarefas dos agentes que compõem a base do framework JADE: *DFAgent* (responsável pelo serviço de "páginas amarelas"), *AMSAgent* (responsável pelo serviço de "páginas brancas") e o *GatewayAgent* (responsável pela interação do usuário através de uma interface Web com toda a plataforma de agentes e semântica). Na figura 5.16 a interação entre o *DFAgent* e o *UserAgent* é mostrada. Representa a entrada do usuário no sistema de busca dos artefatos científicos que por conseqüência o agente que o representa sempre é registrado na plataforma pelo *DFAgent*. Na figura 5.17 a

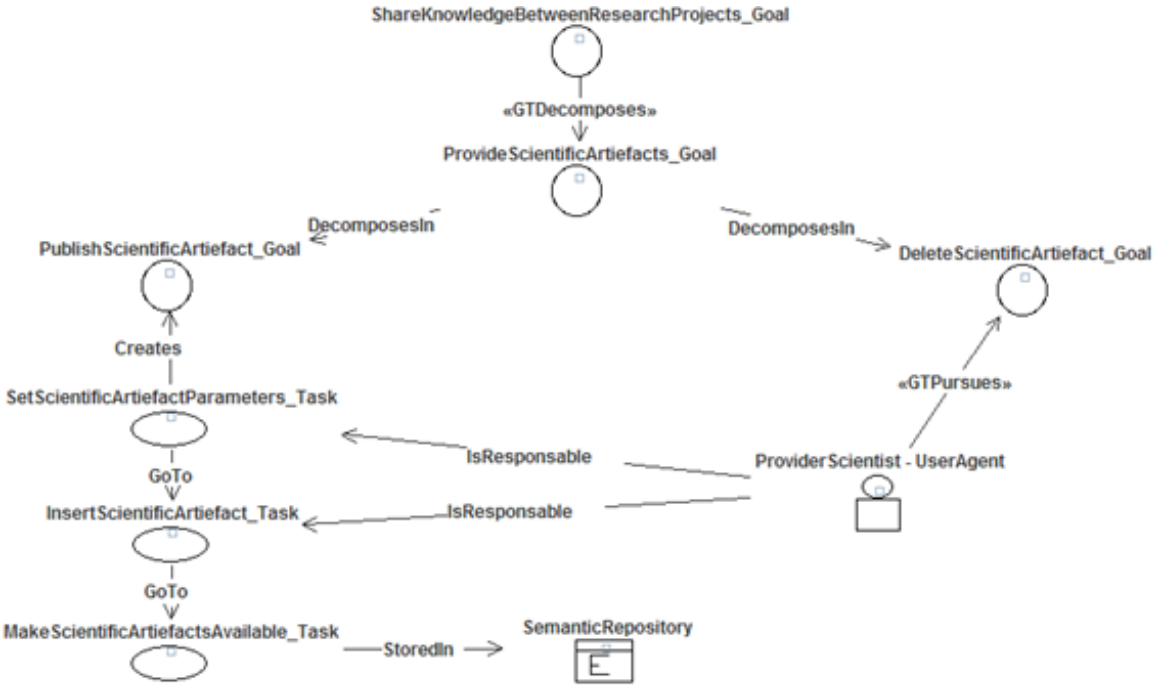

Figura 5.15: Modelo de Tarefas e Objetivos Prover Artefatos Científicos

troca de mensagens entre os dois agentes é representada. A troca é composta de um pedido para registrar no serviço de "páginas amarelas"e a resposta com a confirmação do registro. No caso da mensagem conter algum erro, a resposta do *DFAgent* é uma mensagem na qual ele diz "não entender"a requisição do *UserAgent*.

As figuras 5.18 e 5.19 representam a interação do *AMSAgent* e do *UserAgent* que modela o serviço de registro dos agentes nas "páginas brancas"da plataforma. Exatamente como no modelo anterior, todo agente (representante de um usuário no sistema) registra-se na plataforma também no serviço de "páginas brancas". A troca de mensagens também é semelhante: o *UserAgent* tenta se registrar na plataforma no serviço provido pelo *AMSAgent*, em caso de sucesso ele recebe uma confirmação positiva e em caso da mensagem conter algum erro, o retorno do *AMSAgent* é uma mensagem de não entendimento do que foi requisitado.

Por último, no que diz respeito ao gerenciamento da plataforma, temos as interações do *GatewayAgent*. As figuras 5.20 e 5.21 representam a interação dele com o *UserAgent*, e depois, após a receber a requisição de pesquisa, interage com o *QueryServiceAgent* para que o mesmo possa realizar a busca pelos artefatos científicos, de acordo com os critérios passados pelo *UserAgent*.

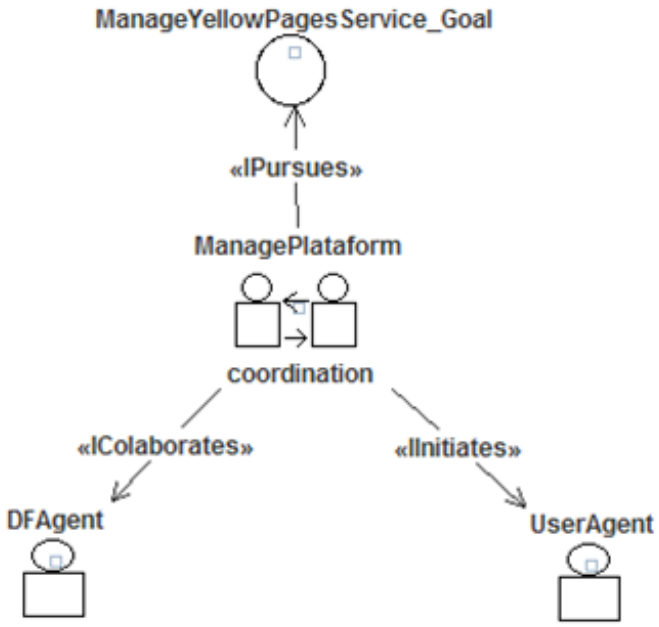

Figura 5.16: Interação para o registro nas "páginas amarelas"

Os outros modelos de interação remetem ao objetivo de atender às necessidades de busca dos usuários (representados pelo *UserAgent*) e a invocação de serviços Web semânticos, quando for o caso. A figura 5.22 e 5.23 representam a interação *UserAgent-QueryServiceAgent* (que possui o *GatewayAgent* como mediador). O usuário pode fazer três tipos de busca por artefatos científicos: usar seu perfil construído na aplicação, utilizar inferência sobre a ontologia que compõe a semântica dos dados armazenados sobre os artefatos científicos e por último, uma pesquisa livre, baseada em palavra-chave, como uma pesquisa em um portal de busca tradicional.

Quando o usuário, dentre os artefatos científicos retornados na sua busca, seleciona um serviço Web semântico que deseja executar, entra em cena uma nova interação, necessária para que a invocação ao serviço aconteça. Para isto, as figuras 5.24 e 5.25 modelam a interação entre os agentes: *UserAgent* e *InvokerAgent* para um determinado serviço Web.

A troca de mensagens entre os dois agentes representa de maneira bem clara e objetiva a interação entre o usuário (*UserAgent*) e o respectivo serviço Web através do *InvokerAgent*. Quando o usuário seleciona um serviço, o que a plataforma tem disponível, inicialmente, é o WSDL, o arquivo que representa a estrutura do serviço e fornece seu endereço de acesso. A plataforma, através do WSDL, lista todas as operações disponíveis para o usuário. O usuário seleciona uma operação, e mais

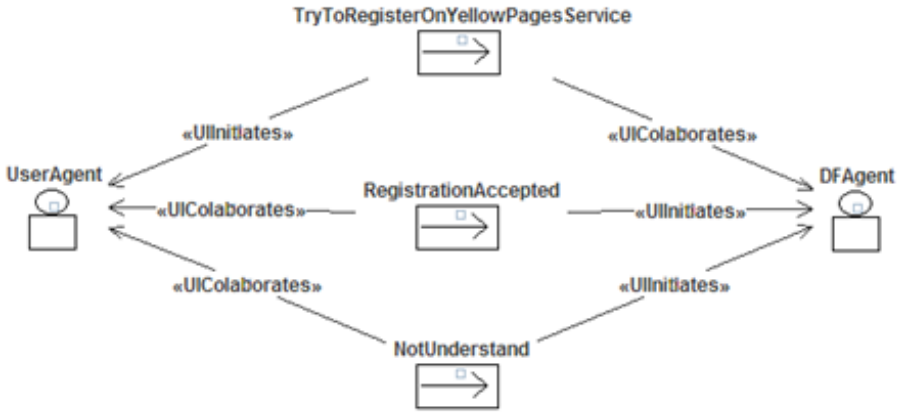

Figura 5.17: Troca de mensagens entre o DFAgent e UserAgent

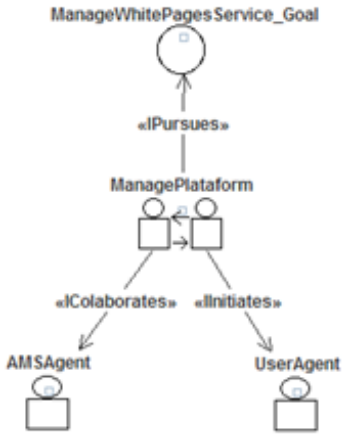

Figura 5.18: Interação para o registro das "páginas brancas"

uma vez a plataforma entra em operação (trocando estas mensagens) pesquisando pelos parâmetros necessários para que a operação seja executada. Por último, em posse da operação escolhida, os parâmetros para a execução, o usuário entra com os valores de cada parâmetro e a plataforma termina a invocação ao serviço exibindo o resultado da operação selecionada e os parâmetros que foram passados.

5.2.6 Modelo de Ambiente

O modelo de ambiente define a percepção do agente quanto a componentes existentes no sistema que não necessariamente sejam outros agentes. Uma base de dados, uma ontologia, uma aplicação externa, ou interna, até mesmo relação com o hardware como um disco rígido e os recursos que eles oferecem, podem e devem ser

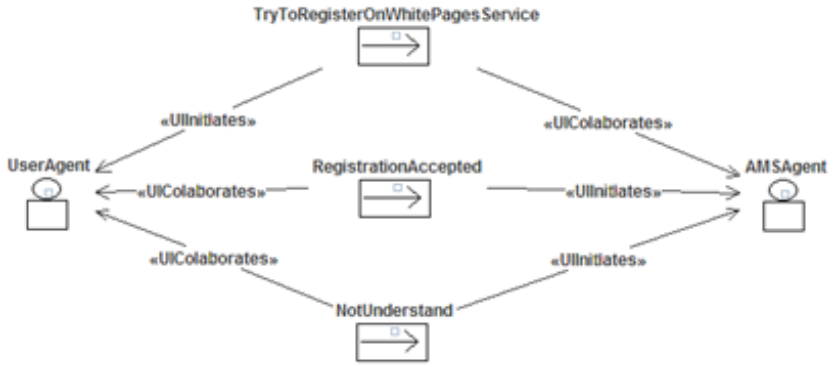

Figura 5.19: Troca de mensagens entre o AMSAgent e o UserAgent

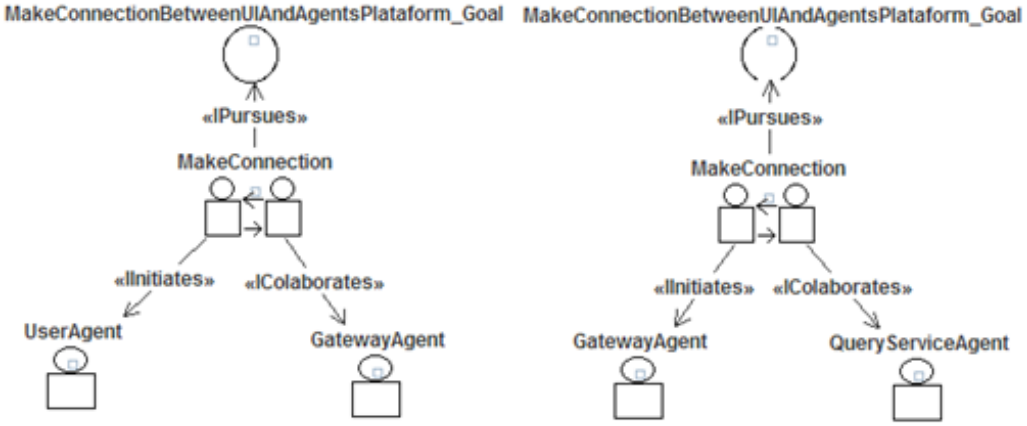

Figura 5.20: Interações para a busca dos artefatos científicos

modelados nesta seção.

Para esta plataforma foram identificados três componentes presentes no ambiente dos agentes: o SOR (*Scalable Ontology Repository*), responsável pelo armazenamento das informações sobre os usuários e os artefatos científicos desde artigos até serviços Web; a aplicação Web, escrita em JSP, na qual os usuários tem acesso a plataforma e, finalmente, a ontologia de domínio, utilizada como principal referência no gerenciamento dos artefatos científicos e que será detalhada posteriormente. A figura 5.26 ilustra o modelo de ambiente da plataforma.

Os três principais agentes (não nativos da plataforma JADE) se relacionam com todos os componentes. O UserAgent utiliza o SOR em conjunto com a ontologia (*SASOntology*) para armazenar suas preferências e os seus artefatos científicos. Já o QueryServiceAgent, utiliza a base de dados SOR a partir da ontologia para realizar a pesquisa pelos artefatos científicos requerida pelo UserAgent. Obviamente, tudo

Figura 5.21: Troca de mensagens entre UAgent, GAgent e QSAgent

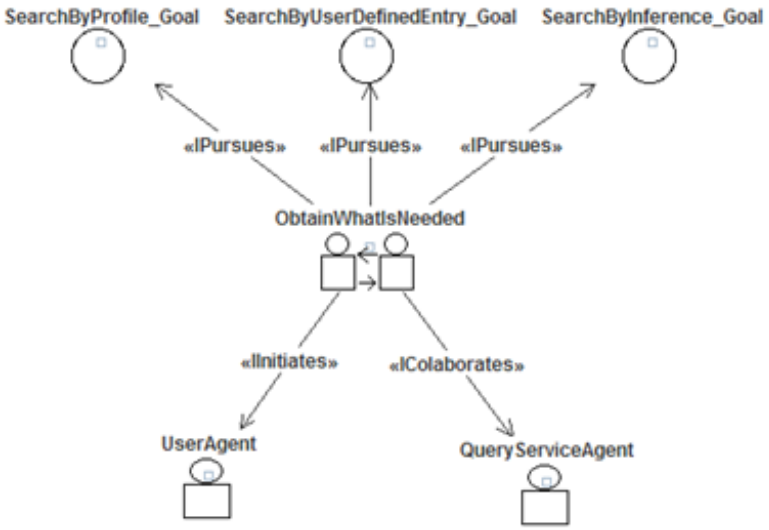

Figura 5.22: Interação entre o UAgent e QSAgent para a busca dos artefatos

isto é possível graças à interface Web escrita em JSP que permitirá a interação destes agentes com todos os componentes da plataforma.

5.3 Arquitetura

A arquitetura é composta por três arcabouços tecnológicos. O primeiro deles é a plataforma onde os agentes existirão. Esta plataforma deve prover escalabilidade, ser distribuída, ser genérica para permitir o desenvolvimento de qualquer tipo de aplicação e ainda prover recursos para a comunicação entre os agentes e dos agentes com aplicações externas de maneira transparente. O segundo é responsável pela representação do conhecimento: como alcançar um modelo que seja tão próximo da representação real do domínio trabalhado e, ao mesmo tempo, ser capaz de usar componentes de software (como agentes) para entendê-lo, até mesmo interpretá-lo

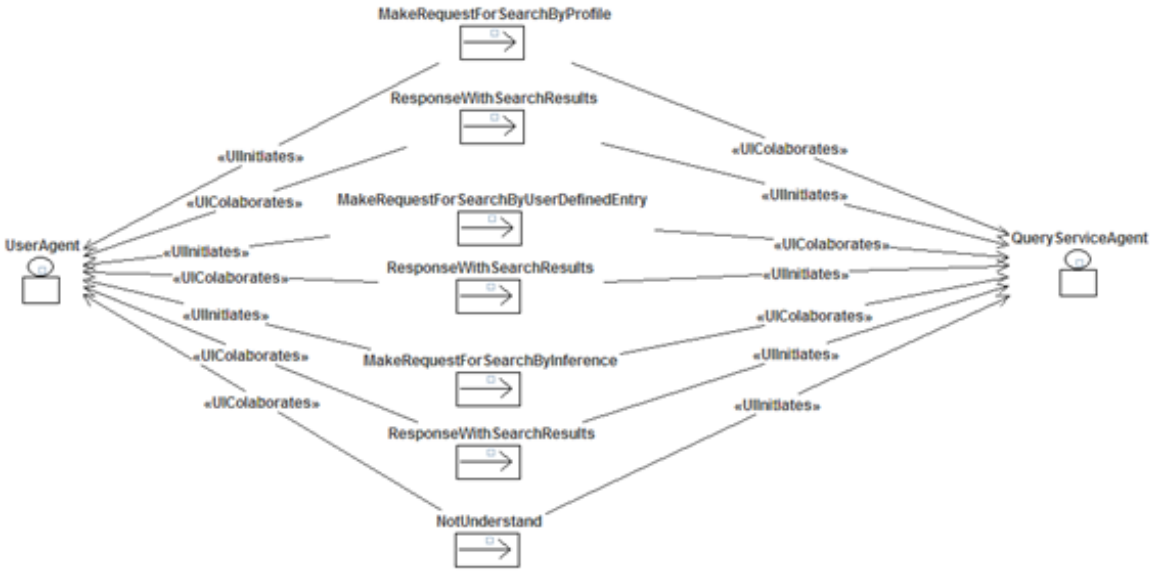

Figura 5.23: Troca de mensagens entre o UAgent e QSAgent

para a aplicação em questão. O último, e talvez o mais importante é a integração dos dois anteriores e a interação com o usuário final da plataforma.

Considerando as questões anteriores, a arquitetura deve ser baseada na Web, que hoje é o principal meio de atingir o maior número possível de grupos de pesquisa que estejam geograficamente dispersos ou até mesmo dentro da mesma instituição.

5.3.1 O Framework para Agentes

Após a fase de projeto e modelagem, detalhada na seção anterior, é necessário ter o foco voltado para a implementação dos agentes. O objetivo é escolher a plataforma para o desenvolvimento dos agentes. Uma característica importante é o modelo de comunicação e a linguagem para a comunicação entre os agentes.

Podemos dividir em dois os tipos de modelos de comunicação entre os agentes (LUCENA, 2003). A primeira delas é a arquitetura do quadro negro onde os agentes se comunicam entre si de maneira indireta através de um quadro negro (BUSCHMANN *et al.*, 1996). O quadro negro é uma estrutura de dados persistente, onde existe uma divisão em regiões ou níveis para facilitar a busca pela informação desejada. Ele funciona exatamente como o meio de interação entre os agentes, funcionando como uma espécie de repositório de mensagens. Os agentes escrevem

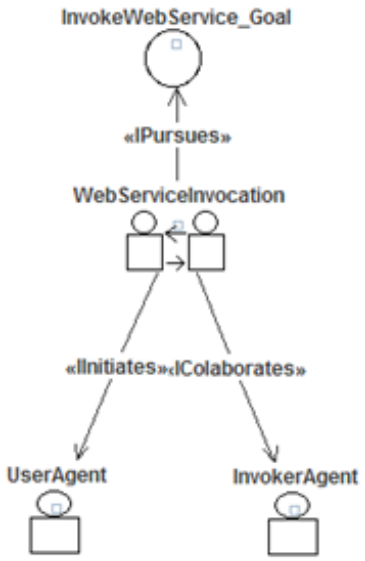

Figura 5.24: Interação entre os agentes: UserAgent e InvokerAgent

as mensagens no quadro e ao mesmo tempo lêem mensagens escritas por outros agentes.

O segundo tipo de comunicação entre agentes é feita utilizando troca de mensagens diretas entre eles. Para que este paradigma seja colocado em prática baseada no padrão FIPA que hoje é o mais utilizado para desenvolvimento da comunicação entre agentes (FIPA, 2004) alguns detalhes precisam existir na plataforma. Primeiro a funcionalidade de *whitepages*, que basicamente funciona como um serviço de encontrar os agentes pela sua identificação, como uma lista telefônica normal onde buscamos o telefone de alguém através do nome da pessoa. E outra funcionalidade importante é o serviço de *yellowpages*, onde os agentes localizam outros agentes não pela identificação, mas pelos serviços que eles podem prover dentro da plataforma, exatamente igual a uma lista telefônica onde buscamos os telefones de empresas pelos serviços que elas oferecem. Todo este paradigma é sustentado pelos chamados agentes facilitadores, que são os responsáveis pelos endereços (identificação) dos agentes e seus respectivos serviços ou em nomes mais específicos no padrão FIPA temos um *Directory Facilitator* (DF), responsável pelo serviço de *yellowpages*, e um *Agent Management System* (AMS), responsável pelas *whitepages*.

A plataforma, além de prover todos estes serviços, deve ainda trabalhar com uma linguagem específica de comunicação entre os agentes (ACL: *Agent Communication Language*). Na literatura temos duas linguagens principais que se destacam: KQML

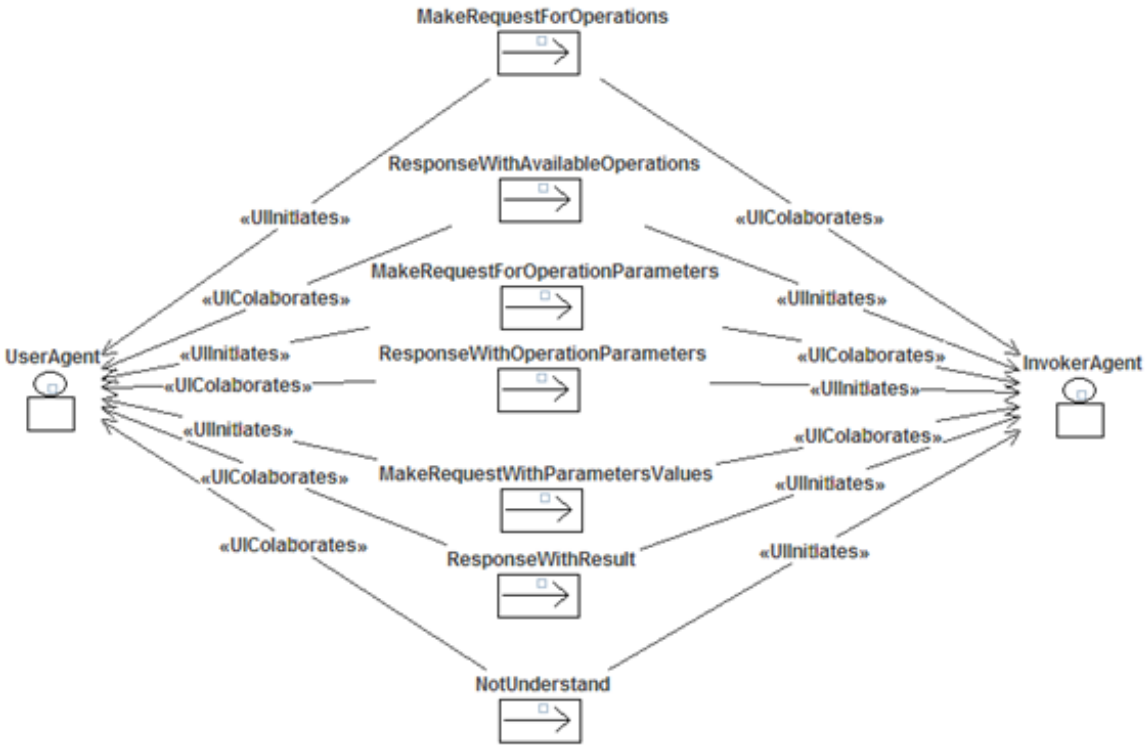

Figura 5.25: Troca de mensagens entre os agentes: UserAgent e InvokerAgent

(*Knowledge Query Manipulation Language*) (FININ *et al.*, 1994) e que já não é tão utilizada atualmente e FIPA ACL (FIPA, 2004), que tem o objetivo principal de estabelecer padrões para o desenvolvimento de agentes inteligentes. Hoje, a FIPA ACL é o padrão estabelecido na comunidade e por isso teve peso fundamental na escolha da plataforma de desenvolvimento dos agentes da arquitetura *SASAgent*.

Com todos estes requisitos necessários analisados, optamos pela plataforma de desenvolvimento de agentes JADE (BELLIFEMINE, 2003), desenvolvida em Java e que oferece todos os recursos que já foram mencionados. Fornece interoperabilidade, já que é implementada de acordo com o padrão FIPA, sendo assim, é possível até mesmo se comunicar com outros agentes de outras plataformas se necessário. Oferece portabilidade, pois possui um conjunto de APIs para vários tipos de rede, dispositivos móveis como celulares, PDAs, etc, aumentando assim a gama de aplicações que podem ser desenvolvidas. A versão utilizada no desenvolvimento do protótipo foi a JADE 3.5.

O modelo arquitetural da plataforma JADE pode ser visto na figura 5.27. São oferecidas tanto as bibliotecas necessárias para desenvolver os agentes da aplicação quanto o ambiente run-time que provê os serviços como *yellowpages* e *whitepages*,

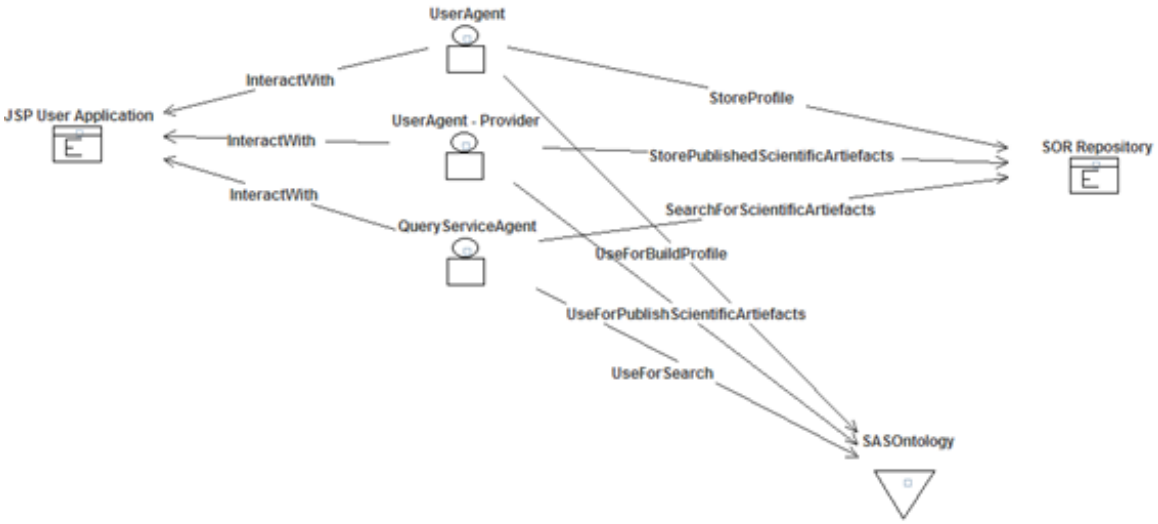

Figura 5.26: Modelo de ambiente da plataforma

por exemplo, e que precisam estar ativos no dispositivo onde os agentes irão ser executados. Cada instância da plataforma JADE de run-time é chamada container onde os agentes são "hospedados". Cada agente dentro da plataforma é identificado por um nome único e pode prover um conjunto de serviços. Ele pode modificar ou registrar seus serviços e ainda procurar por outros agentes que provêem os mesmos serviços. Ele mesmo controla seu ciclo de vida, e em particular, se comunica com outros agentes numa arquitetura ponto a ponto (BELLIFEMINE, 2003).

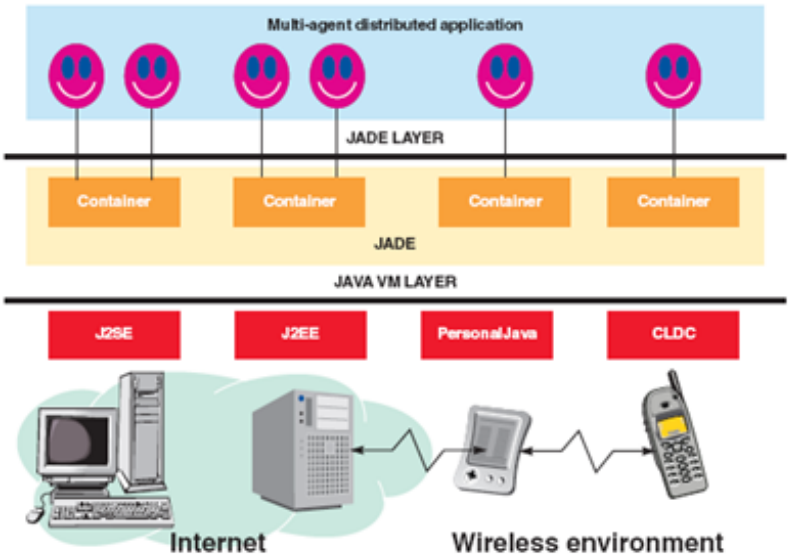

Figura 5.27: Arquitetura da plataforma JADE (BELLIFEMINE, 2003)

A arquitetura da plataforma de busca é apresentada na figura 5.28, sendo uma

arquitetura em camadas, composta pelos seguintes módulos:

- Módulo Agentes: responsável por gerenciar e abrigar todos os agentes do sistema: *UserAgent*, *DFAgent*, *AMSAgent*, *QueryServiceAgent*, *GatewayAgent* e *InvokerAgent*;
- Módulo Web: aplicação desenvolvida em JSP para a Web, que serve como interface do usuário (pesquisador), permitindo o gerenciamento dos artefatos científicos, desde publicação até pesquisa e invocação, no caso de serviços Web;
- Módulo Semântico: seus componentes são: SOR e *SASOntology*, diretamente responsáveis por permitir a pesquisa dos artefatos científicos, seja através do perfil do usuário, inferência lógica ou busca por palavras-chave.

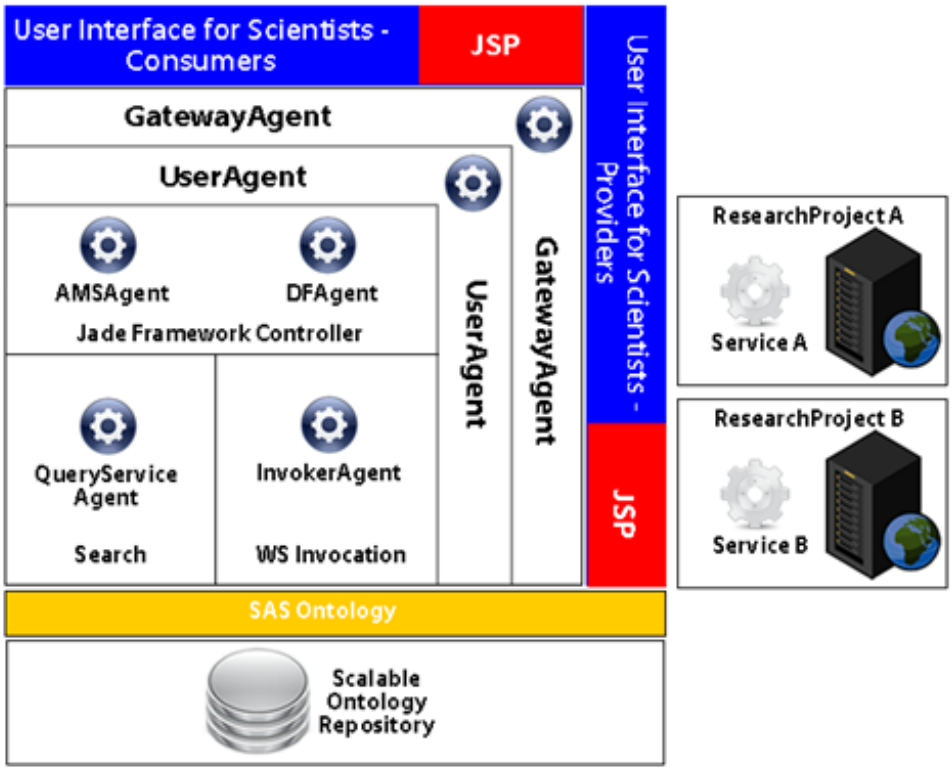

Figura 5.28: Arquitetura geral da plataforma *SASAgent*

O módulo de agentes possui seis agentes que contemplam as funcionalidades necessárias:

- QueryServiceAgent (QSA): Este agente é responsável pela busca dos artefatos científicos existentes na aplicação através de uma consulta SPARQL

(SPARQL, 2008) sobre a ontologia que como resultado é retornado as URIs (Uniform Resource Identifier) dos artefatos científicos encontrados;

- InvokerAgent (IA): Quando serviços Web são encontrados pelo QSA, este agente tem a capacidade de invocar estes serviços para que possam ser utilizados pelo usuário dentro da infraestrutura de maneira transparente;

- UserAgent (UA): Sempre que o usuário se logar na arquitetura criada, dentro da plataforma JADE, automaticamente um novo agente é criado e seu perfil é carregado com informações baseadas na ontologia criada como, por exemplo, as instituições de pesquisa que mais lhe interessam, quais são as áreas de interesse de preferência, entre outros;

- GatewayAgent (GA): O GA tem a função de integrar a aplicação Web desenvolvida em JSP com a plataforma JADE que está rodando no servidor. Além disto, toda e qualquer requisição que é feita aos outros agentes existentes, passa pelo GA através do modelo de comunicação de quadro negro, onde a mensagem é "escrita"e endereçada a algum agente específico. O destinatário "lê"esta mensagem e a responde, sendo que esta resposta é recebida pelo GA e exibida dentro da aplicação Web;

- DFAgent (DFA): O DFA é um agente nativo da plataforma JADE que é criado sempre que a plataforma é inicializada. O seu papel é prover os serviços de yellowpages. Portanto, quando um novo agente é criado na plataforma, se ele provê serviços (não no sentido de serviços Web, mas sim uma descrição do que ele é capaz) ele pode se comunicar com o DFA e se registrar informando que serviços ele provê. Este registro no DFA é opcional para os agentes recém-criados, a plataforma não obriga a registrá-los;

- AMSAgent (AMSA): O AMSA também é um agente nativo da plataforma JADE e criado sempre que a plataforma é inicializada. O serviço de white-pages é provido por ele. Portanto, sempre que um novo agente é criado, ele é registrado pelo AMSA e este registro é obrigatório na plataforma, mas feito de maneira automática quando se cria um agente.

5.3.2 Base de Dados Semântica

Uma das características importantes do *SASAgent* é o uso de recursos semânticos na busca por artefatos científicos. Para isto, ela faz uso de dois componentes: uma ontologia, que modela o domínio da aplicação, e um repositório de dados semântico, que permite a gerência das ontologias e dos dados a serem armazenados.

A ontologia é escrita em OWL e visa modelar o domínio de busca e recuperação de artefatos científicos. Mais que isso, a ontologia deve permitir que inferências sejam realizadas sobre os dados armazenados para encontrar relações que estejam implícitas e artefatos científicos importantes que não seriam encontrados em uma pesquisa comum por palavras-chave, por exemplo. A ontologia, em sua primeira versão, será apresentada no Capítulo 6. A interação entre os agentes e a base semântica é mostrada pela figura 5.29.

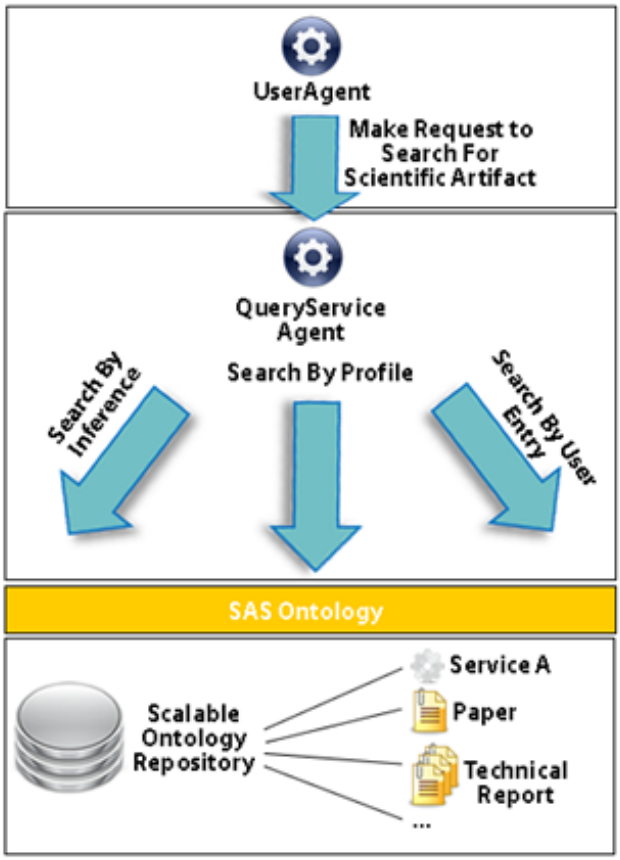

Figura 5.29: Interação dos agentes com a base semântica

5.3.3 Uso de Serviços Web Semânticos no *SASAgent*

O artefato científico que exige a maior complexidade, tanto para publicação na plataforma, quanto para sua busca, é um serviço Web. É necessário que ele esteja registrado em algum repositório, com algum mecanismo de busca, para que possamos conseguir localizá-lo. Mesmo assim, há ainda outras barreiras como: política de segurança, não é livre para uso para qualquer usuário; ou ainda pode exigir uma cobrança, um custo para a sua utilização. Estes são alguns fatores que tornam complexa a busca e utilização de serviços Web.

A estrutura de um serviço Web é descrita em um arquivo WSDL que oferece uma descrição completa das operações disponíveis e seus respectivos parâmetros. No entanto, esta estrutura é válida somente na hora de construir as aplicações que irão consumi-los. Naturalmente, não há ligação dos parâmetros a nenhum conceito do domínio que o serviço Web está inserido, nem as operações, fazendo com que o arquivo WSDL seja uma descrição sintática do funcionamento do serviço.

Com o objetivo de adicionar semântica à descrição dos serviços, para que a busca por um serviço Web seja mais precisa, o *SASAgent* usa as descrições dos serviços em OWL-S, uma ontologia para a realização de anotações semânticas sobre os serviços. A OWL-S basicamente provê três tipos de conhecimento sobre um serviço (figura 5.30):

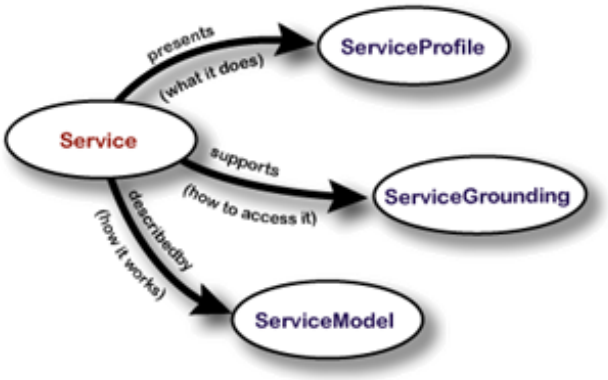

Figura 5.30: OWL-S no seu mais alto nível de abstração (OWLS, 2004)

- O que os serviços provêem para os clientes que estão consumindo-os: cada serviço, que é uma instância da classe Service, apresenta um ServiceProfile;

- Como ele é usado: relacionado à modelagem do processo podendo ser um serviço atômico ou formar um grupo de serviços relacionados através de uma composição. Estas anotações estão descritas na classe ServiceModel;
- Como interagir com o serviço: são os detalhes relacionados aos protocolos de transporte, como é feita a invocação destes serviços. As anotações para tal, estão na classe ServiceGrounding.

Quando o usuário busca por artefatos científicos, o QueryServiceAgent irá realizar a pesquisa sobre a ontologia de domínio e ainda sobre as anotações semânticas de cada serviço publicado no contexto do *SASAgent* através da OWL-S. Cada serviço Web, disponível na plataforma, terá um arquivo OWL-S associado. Nesta proposta não será focado em como fazer as anotações semânticas em OWL-S para um serviço, uma vez que existem ferramentas simples disponíveis que permitem a automatização desta anotação semântica, mas sim à busca do mesmo posteriormente. Estas anotações semânticas ficam armazenadas no repositório juntamente com a ontologia de domínio.

Quando o resultado da pesquisa retornar um serviço Web, o que será retornado é a URI do serviço para permitir a sua invocação. O *SASAgent* irá invocá-lo através da interação do UserAgent (representante do usuário) e o InvokerAgent, responsável pela invocação do serviço. Ao selecionar o serviço desejado, o *InvokerAgent* retorna ao usuário todas as operações disponíveis do serviço, permitindo ao usuário selecionar aquela que lhe satisfaz. Operação selecionada, o segundo passo do *InvokerAgent* é trazer os parâmetros exigidos para a execução da operação. Para isto, o usuário terá como retorno, dinamicamente, a possibilidade de entrar com o valor de cada parâmetro e o último passo é o *InvokerAgent* realizar a chamada final ao serviço através da operação e os respectivos parâmetros com seus valores digitados pelo usuário. Na figura 5.31 está representada uma interação entre os agentes, bases de dados e cientista e no capítulo 6 será apresentado um exemplo desta invocação.

O repositório para as anotações semânticas que são realizadas pelo usuário para posterior recuperação faz parte de um conjunto de ferramentas denominado IBM Integrated Ontology Development Toolkit (LU *et al.*, 2007), que fornece um ferramental completo para armazenar, manipular, pesquisar e inferir sobre as ontologias e suas respectivas instâncias. O banco de dados de ontologias, que é somente uma

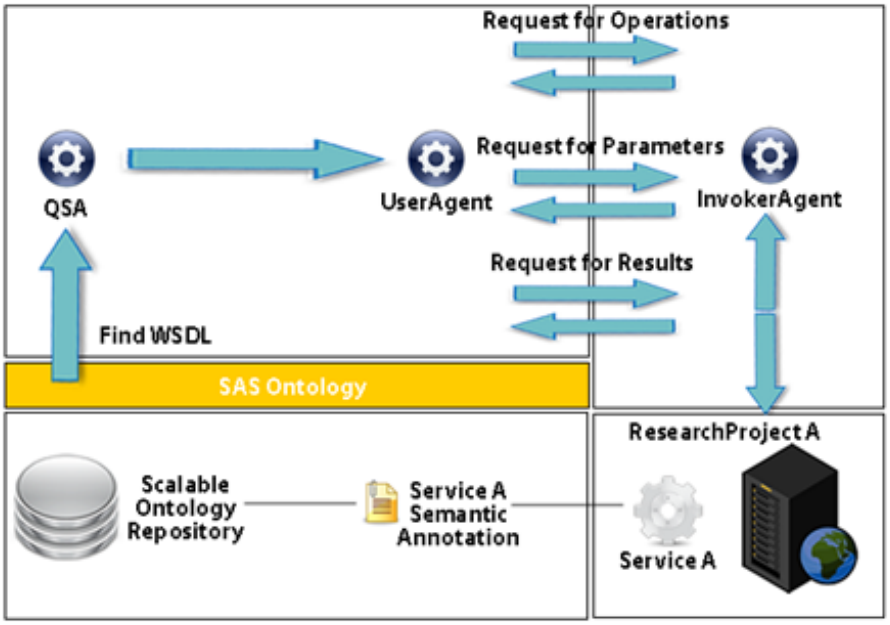

Figura 5.31: Uso de serviços web semânticos na plataforma

parte deste ferramental, chama-se SOR (*Scalable Ontology Repository*) e é baseado em um banco de dados relacional como o DB2 da própria IBM.

5.3.4 Integração Agentes-Semântica-Usuários

Podemos considerar que o diferencial do *SASAgent* e conseguir unir duas tecnologias que na grande maioria dos trabalhos não estão juntas: agentes e semântica. Existem alguns trabalhos sobre agentes inteligentes, porém sem semântica, onde a inteligência é toda baseada em técnicas puras de IA e também outros trabalhos que buscam a semântica sem o uso de agentes.

A proposta aqui é integrar estas duas vertentes através da Web e oferecer uma estrutura funcional aos grupos de pesquisa para que possam compartilhar seus diversos artefatos científicos, permitindo a busca e publicação de artefatos científicos relacionados a um dado domínio de aplicação.

As tecnologias utilizadas no *SASAgent* são tecnologias bastante utilizadas nas comunidades de agentes e web semântica. Os agentes são gerenciados pela plataforma JADE, hoje a mais utilizada no meio acadêmico e industrial. A semântica é construída sobre ontologias OWL, o principal conceito para formalização do conhe-

cimento e torná-lo entendível por máquina, e por último, os usuários usam a Web como a porta de entrada para a busca, publicação e recuperação dos artefatos científicos existentes, pois através dela a integração entre os diversos grupos de pesquisa separados geograficamente torna-se mais simples.

O uso de Agentes-Semântica-Usuários é a integração entre os componentes detalhados nas seções anteriores e o estudo de caso apresentado no próximo capítulo desmostrará esta integração, com o objetivo de validar o processo como um todo.

Capítulo 6

Estudo de Caso: Busca por Artefatos no Domínio de e-Science

Com o objetivo de verificar a aplicabilidade da arquitetura proposta no capítulo anterior, o desenvolvimento de um protótipo torna-se necessário. O principal objetivo desse protótipo é mostrar a integração das tecnologias envolvidas agentes, semântica e serviços Web aplicados ao domínio da modelagem computacional.

O estudo de caso é composto de dois cenários: a busca por serviços Web semânticos e a busca por outros artefatos científicos como publicações do tipo artigos, relatórios técnicos, etc. Podemos destacar como um componente importante deste estudo de caso, além dos agentes nas suas representações computacionais, a ontologia que fornece o background para a realização das buscas, modelando o ambiente de e-Science aplicado à modelagem computacional. Representações semânticas de **Pesquisador**, **Projeto de Pesquisa**, **Área de Interesse**, **Artefato Científico**, dentre outros conceitos e relacionamentos estarão representadas pela ontologia *SASOntology*.

Portanto, os projetos de pesquisa que disponibilizam os artefatos científicos, desde publicações, ferramentas computacionais na forma de bibliotecas (API), aplicações stand-alone, aplicações Web e serviços Web podem ser atendidos pela arquitetura proposta. A modelagem computacional, e respectivos grupos de pesquisa se encaixam perfeitamente neste contexto uma vez que geram, além de publicações científicas, modelos que representam seus problemas, ferramentas para o processa-

mento dos dados, geração dos gráficos, dentre outros.

6.1 A Ontologia *SASOntology*

Apresentamos, a seguir, uma versão inicial da ontologia utilizada para a modelagem de conhecimento no contexto do protótipo desenvolvido a partir da arquitetura SASAgent. A ontologia SASOntology (Scientific Artifact Search Ontology) é modelada utilizando a linguagem OWL (OWL, 2004), padrão recomendado pelo W3C. Dentre os principais motivos para a escolha da linguagem podemos citar: uso de um padrão recomendado, permitir a interoperabilidade da ontologia com outros sistemas baseados em conhecimento, número crescente de ferramentas para manipulação como Protégé-OWL (PROTEGE, 2009), Jena (JENA, 2009), a existência de APIs como Protégé-OWL API OWLAPI (OWLAPI, 2009) e a implementação de mecanismos de inferência como FACT++ (FACT, 2009), Racer (RACER, ????)e Pellet (PELLET, ????).

6.1.1 Objetivos da *SASOntology*

O objetivo principal da ontologia proposta é modelar o contexto de e-Science em geral com ênfase em modelagem computacional, permitindo que grupos de pesquisa de domínio correlatos como engenharias, biologia, matemática, física, química, etc, possam compartilhar os artefatos científicos gerados pela pesquisa realizada nos seus respectivos projetos, desde publicações até ferramentas computacionais desenvolvidas dentro do projeto, como APIs, códigos-fonte de aplicações stand-alone e serviços Web.

Para que o objetivo principal seja alcançado, alguns objetivos secundários precisam ser contemplados. Dentre eles podemos citar: i) criação de um repositório de dados semânticos, responsável por armazenar a ontologia em si e as anotações semânticas, baseadas na ontologia SASOntology, de todos os artefatos científicos; ii) promover o reuso de conhecimento através da publicação, busca e recuperação destes artefatos científicos e iii) representar o domínio através de uma linguagem lógica, com capacidade de inferência e utilização de regras.

6.1.2 Visão Geral da *SASOntology*

A estrutura proposta para a ontologia SASOntology é motivada pela necessidade de prover conhecimento sobre os principais artefatos científicos gerados por um grupo de pesquisa e como estes artefatos podem ser compartilhados com outros pesquisadores de outros grupos. Para tal, duas questões precisam ser respondidas pela modelagem:

- Quais são os artefatos científicos mais importantes que são gerados pelos grupos de pesquisa?
- Como os conceitos relacionados ao contexto de e-Science em geral e modelagem computacional em específico se relacionam de maneira a tornar a arquitetura genérica o suficiente para atender aos sub-domínios correlatos (engenharias, biologia, matemática, física, química, etc)?

Estas questões levaram a definição de duas superclasses genéricas que representam os artefatos científicos gerados por projetos de pesquisa e os conceitos relacionados. Esta organização é apresentada na figura 6.1.

Figura 6.1: Nível topo da ontologia SASOntology

6.1.2.1 Classe *ScientificArtifact*

Esta classe representa, através de suas sub-classes, os principais artefatos científicos que podem ser gerados no contexto de um grupo de pesquisa. A busca pelos artefatos está diretamente ligada aos conceitos modelados representados pelas classes. A figura 6.2 representa a hierarquia destas classes.

As classes da figura 6.2 representam os artefatos científicos gerados, a saber, teses de doutorado, dissertações de mestrado e eventos relacionados ao domínio do grupo

Figura 6.2: Classe ScientificArtifact e subclasses da ontologia SASOntology

de pesquisa que são respectivamente representados pelas classes: **Thesis**, **Dissertation** e **Event**. Além deles, temos ainda ferramentas computacionais, modelos conceituais e publicações que possuem subclasses que representam seus respectivos artefatos. Na figura 6.3 temos a hierarquia da classe **Publication**. Os artefatos científicos que são considerados publicações são: periódico, artigo, livro e relatório técnico, representados respectivamente pelas classes **Journal**, **Paper**, **Book** e **TechnicalReport**.

Na figura 6.4 os modelos conceituais como artefatos científicos são representados. Considerou-se três modelos principais: modelos entidade-relacionamento, diagramas da UML e ontologias do domínio de aplicação do projeto que também podem ser inseridas. No entanto, novas classes podem ser acrescentadas ao modelo. As classes que representam estes modelos são ERM, UML e Ontology.

A figura 6.5 representa as ferramentas computacionais geradas como artefatos científicos. Talvez, dentre todos os conceitos, sejam os principais elementos da ontologia e o principal alvo de pesquisa por parte dos pesquisadores quando pensamos em trabalho cooperativo entre grupos de pesquisa e conseqüentemente a modelagem de workflows científicos. Podemos destacar, dentre estas ferramentas, os serviços Web semânticos, que, a partir da publicação das suas anotações semânticas em OWL-S, pode-se proceder a invocação dinâmica dos serviços escolhidos pelos pesquisadores.

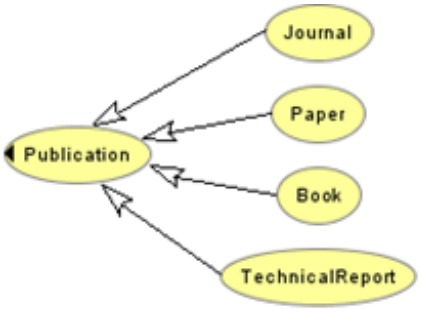

Figura 6.3: Classe Publication da ontologia SASOntology

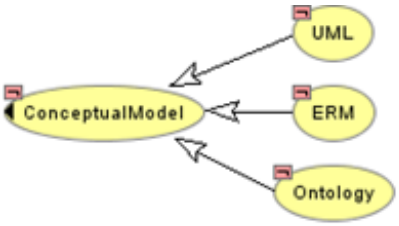

Figura 6.4: Classe ConceptualModel da ontologia SASOntology

6.1.2.2 Classe *ContextConcepts*

Os conceitos que compõe a hierarquia a partir da classe **ContextConcepts** compõem o que podemos chamar de apoio aos artefatos científicos. Estes conceitos representam os projetos de pesquisa, os pesquisadores, as instituições dentre outros, e qual o papel de cada um dentro do domínio e seus relacionamentos.

A figura 6.6 representa as subclasses da classe ***ContextConcepts***. Três delas, à saber: ***ScientistRole***, ***Database*** e ***TitleDegree*** são chamadas classes anônimas pois são classes que são compostas somente de indivíduos e vão servir como domínio de propriedades de outras classes. Por exemplo, a classe ScientistRole contém os indivíduos: ***Collaborator***, ***Coordinator***, ***Student*** que representam os papéis que um pesquisador pode assumir dentro de um projeto de pesquisa, limitando assim os valores desta propriedade a somente uma destas três opções. Já a classe ***Database*** está intimamente ligada à classe ERM que representa um modelo entidade-relacionamento, um artefato científico possível de ser gerado. Ela possui os indivíduos: DB2, MySQL, Oracle, PostgreSQL e SQLServer limitando também o banco de dados que será o destino de um modelo entidade-relacionamento. Estas restrições, associações das classes às classes anônimas, são uma maneira de manter a integridade semântica das informações neste contexto.

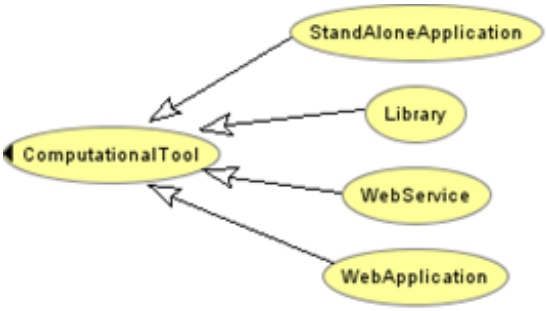

Figura 6.5: Classe ComputationalTool da ontologia SASOntology

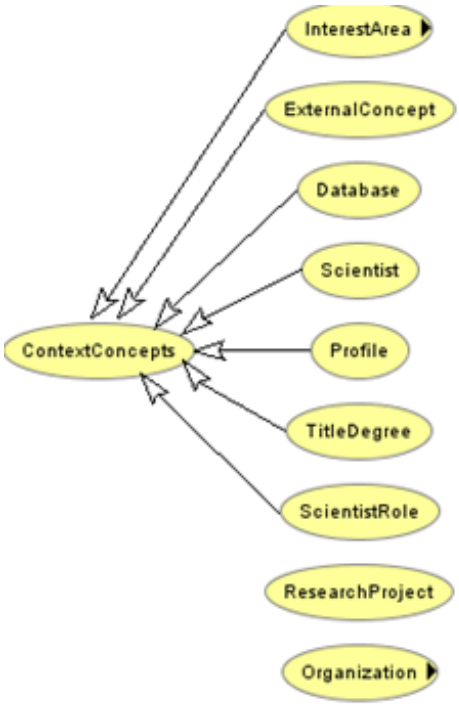

Figura 6.6: Classe ContextConcepts da ontologia SASOntology

Além das classes anônimas temos: a classe **Scientist**, que representa o pesquisador e modela todas as suas informações como titulação, artefatos científicos publicados, projetos de pesquisa que trabalha, departamento de instituição que esteja alocado, dentre outras informações; a classe **ExternalConcept** que é responsável por representar conceitos vindos de outras ontologias para uma integração inicial e torná-los disponíveis na plataforma SASAgent para as anotações semânticas de qualquer artefato científico; a classe **ResearchProject**, representante do projeto de pesquisa com propriedades que modelam as áreas de interesse do projeto, os pesquisadores que participam, os artefatos que já foram gerados, dentre outras informações; a classe **InterestArea** modela as áreas de interesse de um determinado

domínio, por exemplo, computação, matemática, engenharias, biologia, etc; a classe **Organization** representa as organizações envolvidas nos projetos de pesquisa como as universidades que sediam os projetos e as agências de fomento também e por último, podemos citar a classe **Profile**. Esta classe modela o conceito de perfil de preferências dos pesquisadores para que possam nortear as buscas por artefatos científicos. Este perfil pode ser construído através da escolha das instituições, projetos de pesquisa, áreas de interesse e pesquisadores que sejam de preferência do usuário dentro da infra-estrutura. A figura 6.7 mostra a hierarquia das classes **Organization** e **InterestArea**.

Figura 6.7: Classes InterestArea e Organization da ontologia SASOntology

No Anexo 1 a ontologia SASOntology é apresentada na sua íntegra em código OWL. No código estão ressaltadas as restrições, regras e assertivas pertinentes a esta ontologia.

6.2 Cenário Um - Busca por Serviços Web Semânticos

A importância deste cenário que engloba a busca, recuperação e invocação de serviços Web Semânticos aplicados ao domínio científico da Bioinformática através da ontologia SASOntology e do framework SASAgent é justificada pelo grande interesse tanto da academia quanto da indústria em torno da tecnologia dos serviços Web e todos os benefícios que seu uso traz. A independência de tecnologia e a facilidade na interoperabilidade de dados que os serviços podem prover, e o fato da busca por estes serviços não ser trivial motiva a criação de um cenário específico para análise do estudo de caso.

6.2.1 Anotações Semânticas dos Serviços Web em OWL-S

Os principais editores encontrados na literatura relacionados à edição das anotações semânticas em OWL-S para serviços Web são: OWL-S Editor (SCICLUNA *et al.*, 2004) e o plug-in para o editor de ontologias Protégé, OWL-Editor Plug-In (ELENIUS *et al.*, 2005).

A opção foi pelo plug-in do Protégé por já estar no mesmo ambiente que a ontologia e facilitar a integração com a mesma para a edição das anotações semânticas. O plug-in pode ser baixado em: http://owlseditor.semwebcentral.org/projects/owlseditor/ e a versão utilizada é a Build23 aplicada na versão 3.2.1 do editor Protégé.

O primeiro serviço Web escolhido é um serviço web disponibilizado pelo projeto OLS (Ontology Lookup Service) (COTE *et al.*, 2006), um produto de outro projeto relacionado à Bioinformática chamado PRIDE (The PRoteomics IDentifications database) (JONES e COTE, 2008). O projeto PRIDE é uma base de dados centralizada, projetada de acordo com padrões, de acesso público, para dados relacionados a proteínas e peptídeos. Esta base de dados lida com múltiplas ontologias e o projeto OLS faz com que o acesso a todas estas ontologias seja feito por uma única via de consulta através do serviço Web desenvolvido e utilizado neste estudo de caso. Algumas das operações disponíveis e suas respectivas funcionalidades do serviço podem ser vistas na tabela 6.1.

Tabela 6.1: Algumas operações disponíveis do serviço Web OLS

Operação	Descrição
getTermById(termId, ontologyName)	Retorna o nome do termo associado ao ID
getAllTermsFromOntology(ontologyName)	Retorna todos os termos (ID-Nome) de uma dada ontologia
getTermsByName(partialName, ontologyName, reverseKeyOrder)	Retorna todos os termos que coincidem com a palavra de entrada da pesquisa para uma dada ontologia

Os passos para a realização das anotações semânticas levam em consideração a configuração dos três tipos de conhecimento essenciais sobre um serviço que a ontologia OWL-S define: ***ServiceProfile***, ***ServiceGrounding*** e ***ServiceModel***. O primeiro serve como uma apresentação do serviço modelando o que ele faz. O segundo é a descrição do serviço que especifica como acessá-lo através de seus parâmetros, mensagens e tipos. E o último modelo descreve como o serviço funciona, entradas, saídas, condições, conseqüências que tem papel fundamental na composi-

ção. O plug-in utilizado provê recursos para a configuração de todos eles e ainda a associação com conceitos da ontologia SASOntology. Para o editor, o *ServiceModel* é sinônimo de *Process*. Para ganhar produtividade é possível criar um esqueleto do serviço em OWL-S a partir do arquivo WSDL disponível e foi este o caminho seguido por este estudo de caso para ambos os serviços. A figura 6.8 mostra o editor onde se requisita a URL do arquivo WSDL ou o próprio arquivo WSDL, caso esteja armazenado localmente. O arquivo é automaticamente lido pelo editor e traz todas as operações disponíveis para o serviço selecionado, bem como seus respectivos parâmetros de entrada e saída. Uma observação importante é que o plug-in do Protégé, especificamente, gera um arquivo OWL para cada OPERAÇÃO do serviço Web.

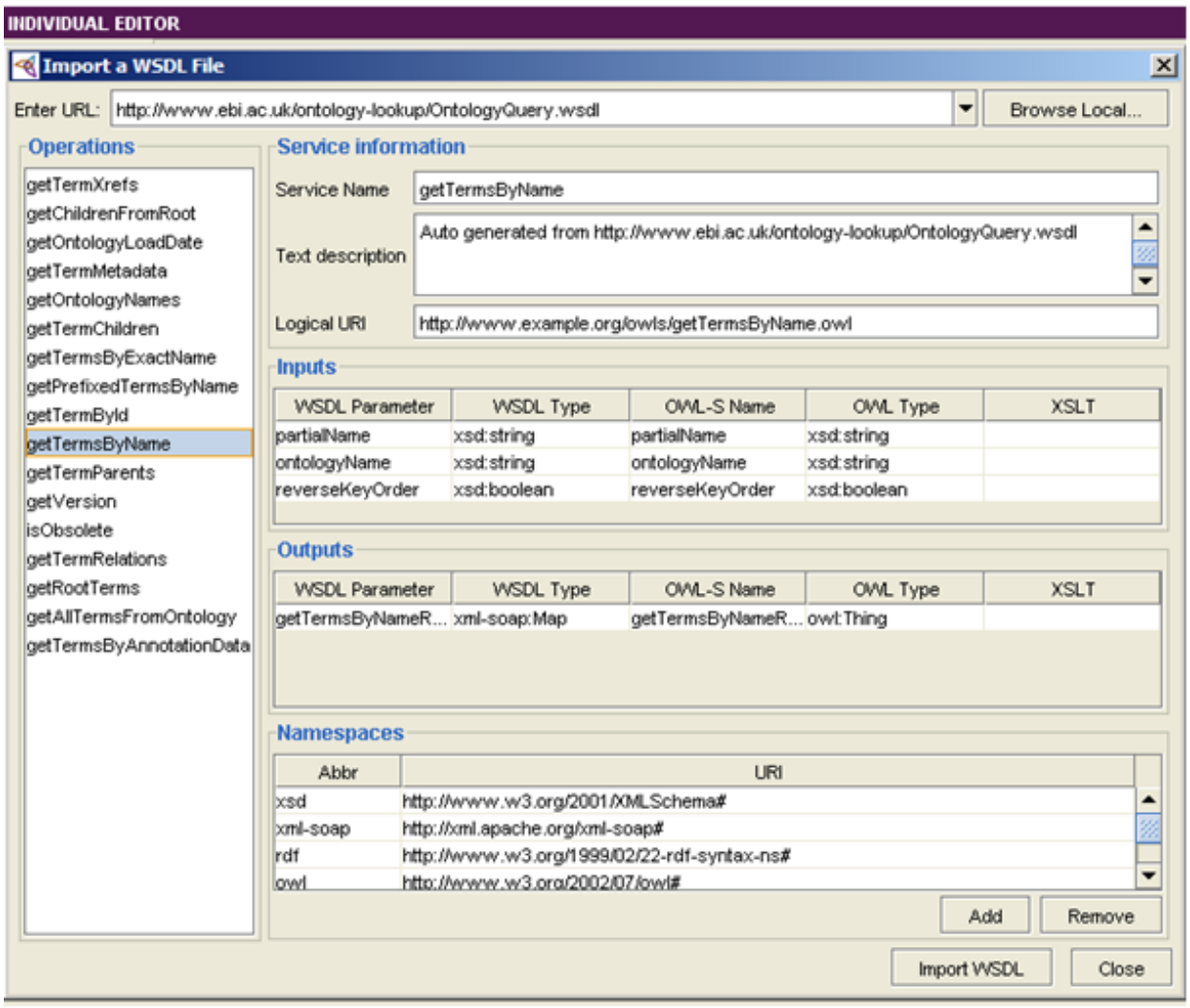

Figura 6.8: Importação do arquivo WSDL para a geração do esqueleto em OWL-S

A operação ***getTermsByName*** foi a selecionada como exemplo ilustrativo para apresentar as anotações semânticas ligando o serviço Web à ontologia *SASOntology*.

Possui dois parâmetros de entrada principais: o nome a ser procurado e qual a ontologia que deve ser pesquisada pelos termos. Na figura 6.9 o editor exibe o esqueleto montado a partir do WSDL do serviço Web e a operação escolhida.

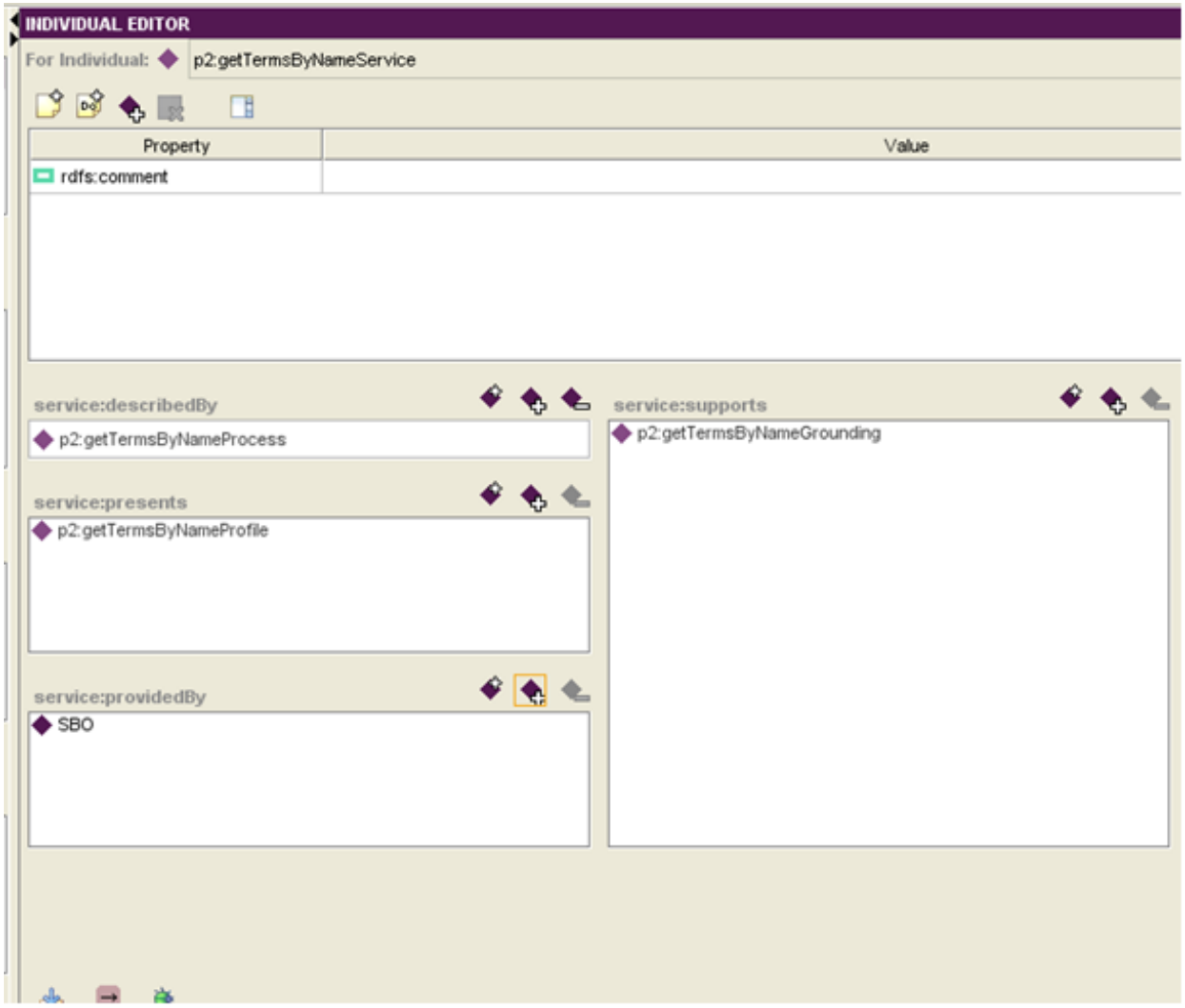

Figura 6.9: Esqueleto em OWL-S do serviço Web OLS após leitura do WSDL

A associação com a ontologia *SASOntology* deu-se de duas maneiras. Primeiramente, cada serviço pertence à classe ***Service*** da ontologia OWL-S que possui quatro propriedades: ***describedBy*** (que liga ao ***ServiceModel*** ou, no caso do editor, ***ServiceProcess***), supports (ligada ao ServiceGrounding), presents (conectada a um ServiceProfile) e por último, quem é o provedor do serviço, ***providedBy***, onde têm-se a possibilidade de associar a conceitos externos à OWL-S. Neste exemplo foi preenchida com o projeto de pesquisa SBO (*Systems Biology Ontology*), instância da classe ***ResearchProject*** da ontologia *SASOntology*. A segunda ligação com a ontologia proposta é feita através do ***ServiceProfile*** que permite adicionar informações

de contato. Para o exemplo, foram adicionadas instâncias da classe ***Scientist*** como pode ser visto na figura 6.11.

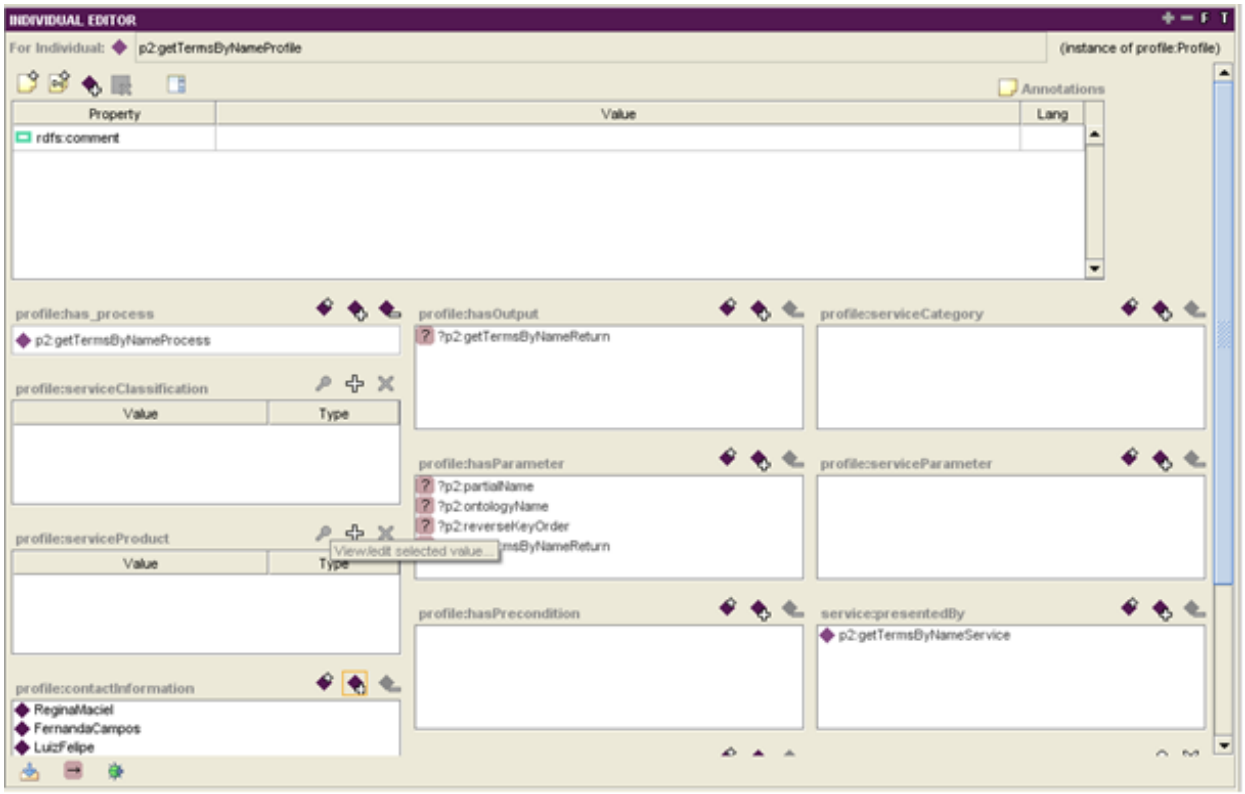

Figura 6.10: Associação de instâncias de pesquisadores da ontologia *SASOntology* ao *ServiceProfile*

O segundo serviço Web, também relacionado à Bioinformática, chama-se PICR (*Protein Identifier Cross-Reference Service*) (COTE *et al.*, 2007). Seu principal objetivo é a habilidade de mapear sequências de proteínas e seus respectivos identificadores através de múltiplas bases de dados, provendo ainda o set up de diversos parâmetros relacionados às espécies envolvidas e taxonomias específicas. Ele utiliza a base de dados *UniProt Archive* (UniParc) como um *datawarehouse* para oferecer este mapeamento de maneira transparente ao usuário consultando mais de 70 bases de dados diferentes. Os mapeamentos podem ser feitos configurando parâmetros como: seleção das bases de dados envolvidas, identificadores taxonômicos e status de atividade das bases de dados. As três operações disponíveis no serviço estão descritas na tabela 6.2. As anotações semânticas foram feitas da mesma maneira que para o primeiro serviço Web.

Tabela 6.2: Operações disponíveis do serviço Web PICR

Operação	Descrição
getUPIForSequence(sequence, searchDatabases, taxonId, onlyActive)	Retorna os identificadores da sequência de proteínas
getUPIForAcession(acession, acVersion, searchDatabases, taxonId, onlyActive)	Retorna os identificadores da sequência levando em consideração status de atividade das bases de dados existentes
getMappedDatabaseNames()	Retorna as bases de dados que compõem o *datawarehouse* UniParc

6.2.2 Publicação dos Serviços Web

Para que os artefatos científicos sejam publicados e tornem-se disponíveis para a busca por pesquisadores, em especial os serviços Web, um repositório de dados torna-se necessário. Com o intuito de atingir este objetivo, o repositório selecionado foi o SOR (Scalable Ontology Repository) (LU *et al.*, 2007), específico para o armazenamento de ontologias OWL-DL em um banco de dados relacional. Ele suporta ainda mecanismos de inferência, armazenando não somente o conhecimento explícito na ontologia, mas também o conhecimento inferido. Finalmente, para consultar os dados armazenados, a linguagem de consulta utilizada é a SPARQL.

Segundo (LU *et al.*, 2007), da perspectiva do gerenciamento de dados, ontologias poderiam ser consideradas como um modelo de dados (semelhante a um esquema relacional ou esquema XML) e os dados podem ser vistos como instâncias da ontologia. A Figura 6.11 apresenta os principais componentes da arquitetura SOR.

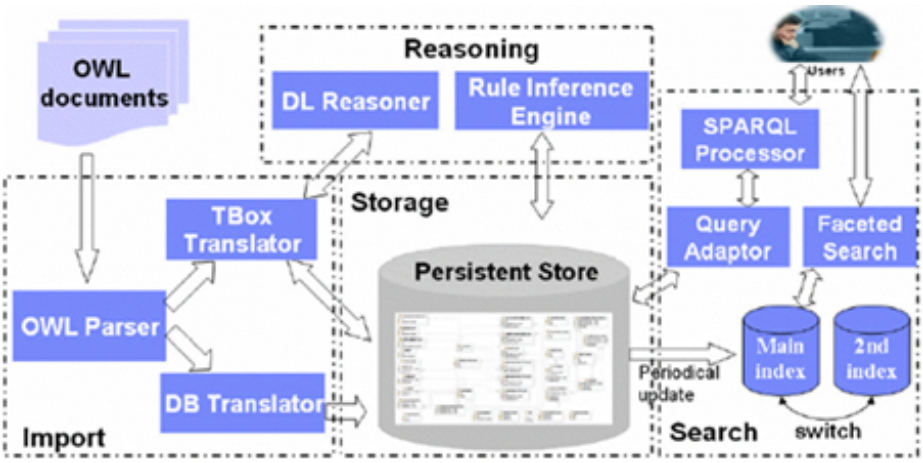

Figura 6.11: Componentes do SOR (LU *et al.*, 2007)

A figura 6.12 exibe um diagrama de sequência referente à realização das anotações

semânticas e à posterior publicação no SASAgent através da ontologia SASOntology. Na figura 6.13 temos a interface para configurar os parâmetros necessários para a publicação do serviço Web. Informações básicas como título, palavras-chave, linguagem na qual foi desenvolvido, breve descrição e autor/desenvolvedor responsável compõem os parâmetros do serviço, assim como informações sobre as áreas de estudo que o serviço está envolvido como Modelagem Computacional, Bioinformática, etc e o arquivo OWL correspondente às anotações semânticas do serviço.

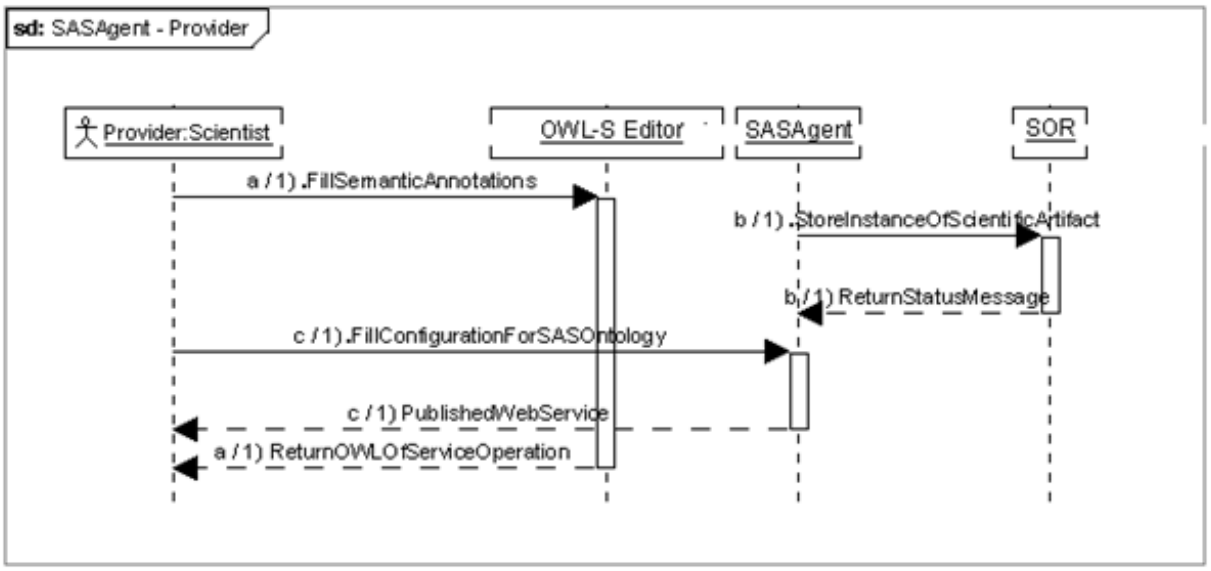

Figura 6.12: Diagrama de Sequência para a publicação dos serviços Web

6.2.3 Busca e Invocação dos Serviços

A pesquisa destes serviços será baseada em um perfil de preferências fictício pertencente a uma pesquisadora chamada Maria. Este perfil é construído com base na área de interesse de Bioinformática e conceitos externos como ***Mapping***, ***Protein***, ***Cross-Reference*** e ***Identifiers*** que também compõem o profile. Estes conceitos isoladamente representam a possibilidade da utilização de ontologias externas à *SASOntology* no papel de ontologias de domínio dos respectivos grupos de pesquisa integradas ao SASAgent, integração esta, feita através da adição destes conceitos externos à *SASOntology* através da classe ***ExternalConcepts***.

A figura 6.14 representa a consulta em SPARQL que retorna todos os artefatos científicos que sejam serviços Web (***WebServices***) que atendem ao perfil da pesqui-

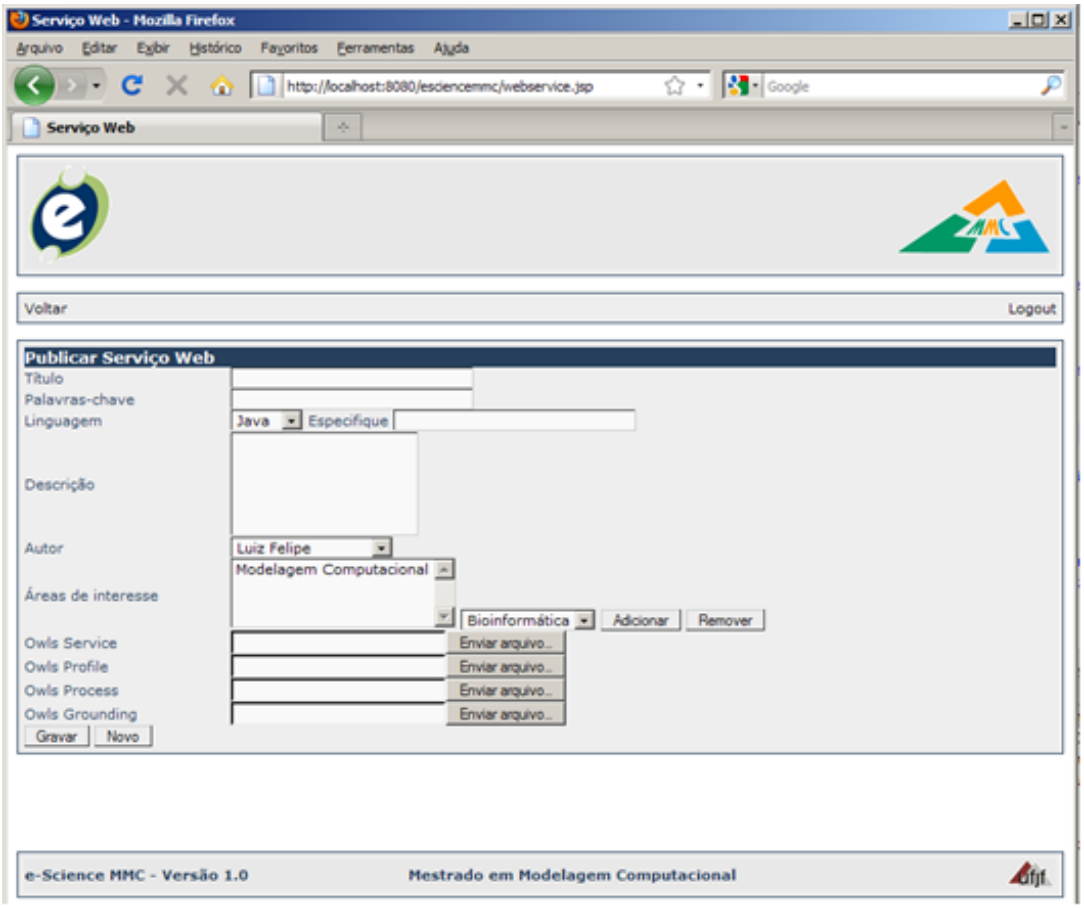

Figura 6.13: Formulário de publicação de serviços Web

sadora fictícia Maria. Ela pode ser dividida em três partes assinaladas em azul, preto e vermelho. O trecho compreendido pelo retângulo 1, é a parte da consulta responsável em retornar todos os artefatos científicos que sejam serviços Web (***WebServices***) cujos pesquisadores pertencentes ao profile da Maria publicaram. Unindo-se a este conjunto de resultados, temos o trecho compreendido pelo retângulo 2 responsável por retornar todos os artefatos científicos que sejam artigos cujas áreas de interesse pertencentes ao profile da Maria publicaram. Finalmente, o trecho compreendido pelo retângulo 3 é o responsável por retornar todos os artefatos científicos, sempre serviços Web, cuja publicação está diretamente relacionada aos conceitos externos selecionados pela pesquisadora para compor seu profile como ***Mapping***, ***Protein***, ***Cross-Reference*** e ***Identifiers***.

Já na figura 6.15 obtemos a tela do resultado da pesquisa pelos serviços Web. No momento que o pesquisador selecionar o serviço desejado, o *InvokerAgent* entra em ação, pois a partir do arquivo WSDL do serviço ele começa a interagir com suas

```
1  PREFIX ufjf:<http://www.ufjf.br/escience.owl#>
2  SELECT DISTINCT ?titulo
3  WHERE
4  {
5    {
6      ?p ufjf:belongsToScientist ufjf:Maria .
7      ?p ufjf:hasScientistAssociated ?pesq .        1
8      ?pesq ufjf:hasPublishedScientificArtifact ?ws .
9      ?ws a ufjf:WebService .
10     ?ws ufjf:hasTitle ?titulo
11   }
12   UNION
13   {
14     ?p ufjf:belongsToScientist ufjf:Maria .
15     ?p ufjf:belongsToSubArea ?area .              2
16     ?area ufjf:hasPublishedScientificArtifact ?ws .
17     ?ws a ufjf:WebService .
18     ?ws ufjf:hasTitle ?titulo
19   }
20   UNION
21   {
22     ?p ufjf:belongsToScientist ufjf:Maria .
23     ?p ufjf:hasRelatedWith ?conc .
24     ?ws ufjf:hasRelatedWith ?conc .
25     ?ws a ufjf:WebService .
26     ?ws ufjf:hasTitle ?titulo                     3
27   }
28 }
```

Figura 6.14: Consulta em SPARQL para localizar os serviços Web

operações, parâmetros e o pesquisador. A figura 6.16 mostra a tela de seleção de operações após o *InvokerAgent* retornar quais são elas para o serviço Web OLS. A figura 6.17 exibe os parâmetros para a operação escolhida, no caso a operação ***isObsolete*** e a figura 6.18 exibe o diagrama de sequência referente à busca por serviços Web por parte do pesquisador.

6.3 Cenário Dois - Busca por Publicações Científicas Relacionadas ao Domínio de *e-Science*

Não menos importante, os artefatos científicos mais simples de visualização como artigos também são importantes e são envolvidos em um invólucro semântico semelhante a que os serviços Web são envoltos. Neste cenário, mostraremos a publicação dos artigos, através da inserção das anotações semânticas e a sua posterior recupe-

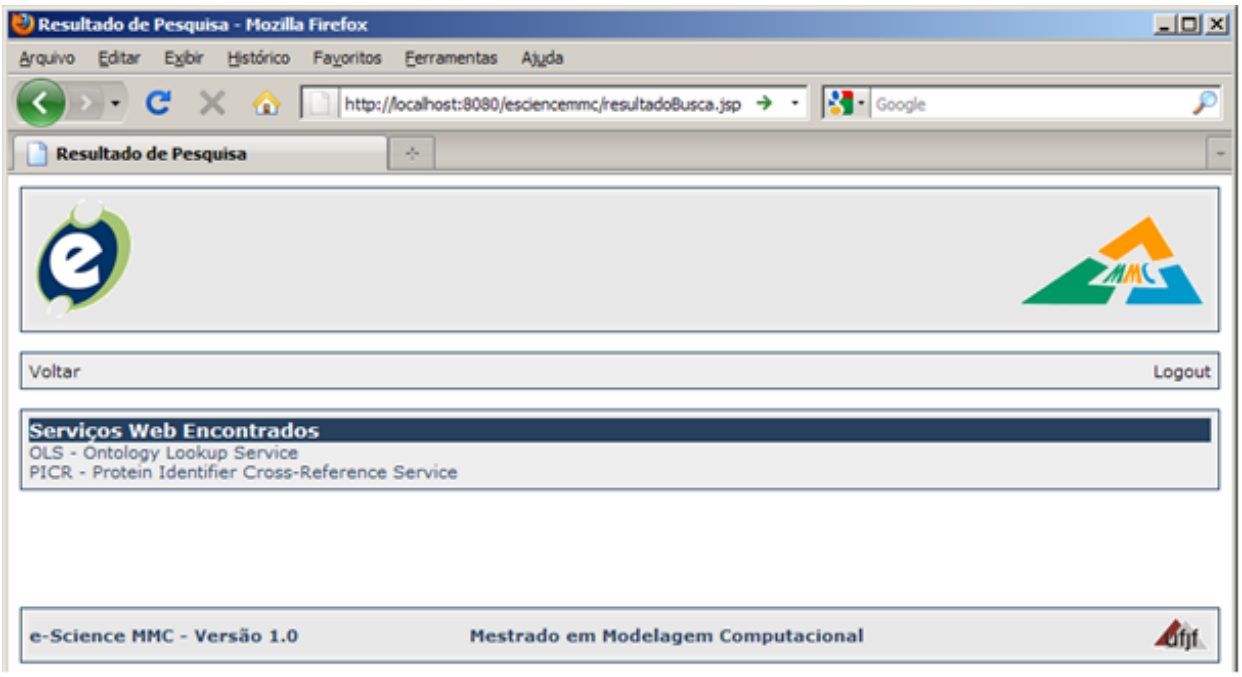

Figura 6.15: Tela de resultados da busca por serviços Web

ração através de consultas SPARQL baseando-se no *profile* do pesquisador.

6.3.1 Publicação dos Artigos

O formulário de inserção e publicação dos artigos pode ser visto na figura 6.19. Os principais conceitos da ontologia *SASOntology*, que serão posteriormente usados para a consulta pelos artigos e são utilizadas no formulário para criar os relacionamentos ontológicos são: Autores (pesquisadores), Conceitos (classes advindas de ontologias de domínio do artigo), Eventos, Áreas de Interesse e Projetos de Pesquisa relacionados ao artigo.

6.3.2 Busca e Visualização dos Artigos

A busca de qualquer artefato científico na proposta deste trabalho pode ser feita de três maneiras distintas: inferência lógica, perfil de preferências do pesquisador (profile) e pesquisa livre por termos definidos pelo pesquisador. Para este estudo de caso desenvolvido, a busca por estes artefatos científicos será realizada através da pesquisa pelo perfil de preferência do pesquisador.

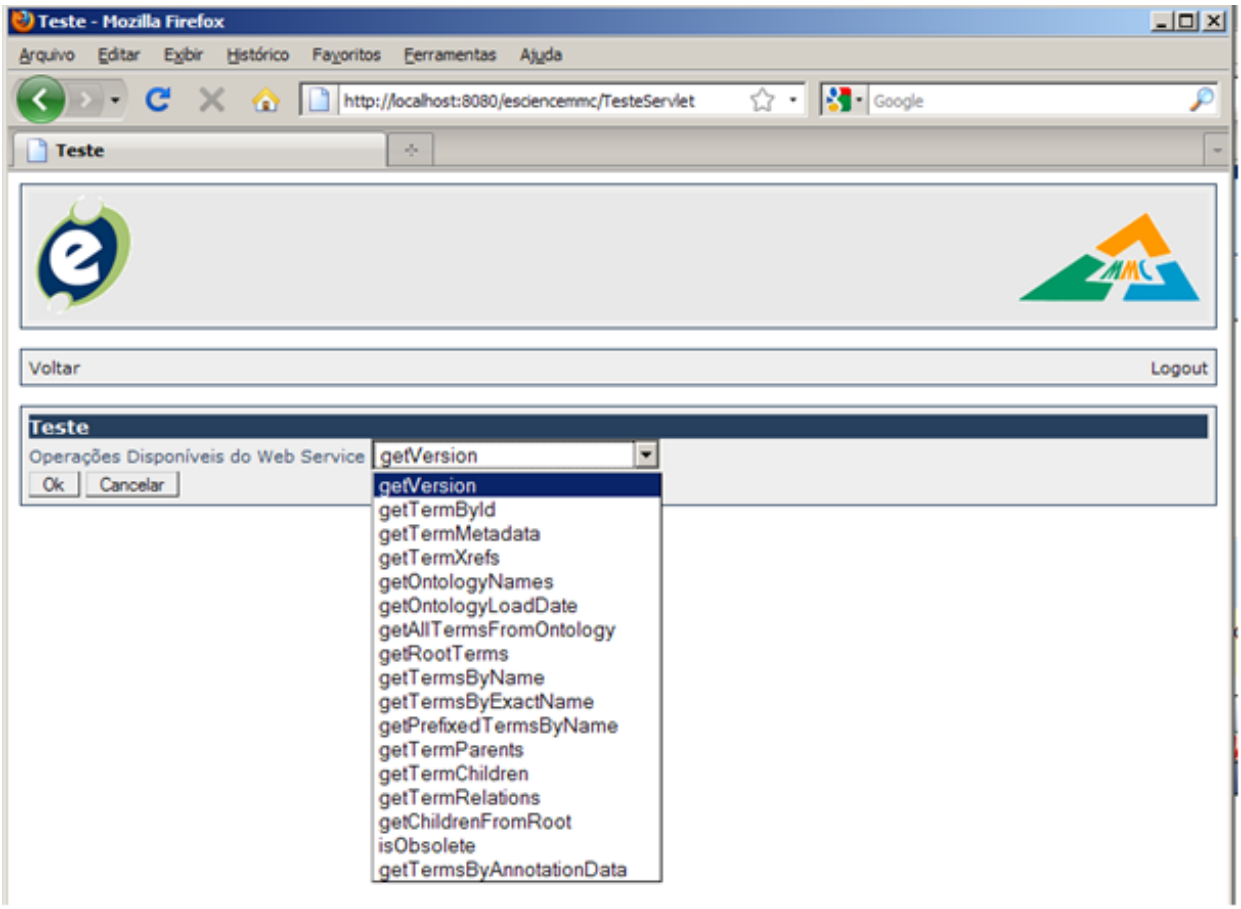

Figura 6.16: Atuação do InvokerAgent exibindo as operações disponíveis do serviço

Portanto, para que a busca via profile seja possível, é necessário que o pesquisador tenha condição de construir o seu próprio profile. A figura 6.20 mostra o formulário de inserção do profile do pesquisador.

O formulário de busca de artigos é apresentado na figura 6.21. O pesquisador seleciona o tipo de pesquisa que deseja realizar, neste caso é a busca via profile, e o *framework* segue em frente. A pesquisa por profile é totalmente automatizada. O pesquisador que é representado por um agente no framework (***UserAgent***) carrega consigo o profile, utilizado no momento da pesquisa no repositório SOR sobre a ontologia *SASOntology*.

Como dito anteriormente, a consulta realizada sobre o perfil de preferências do pesquisador é construída na linguagem SPARQL. A figura 6.22 representa a consulta em SPARQL que retorna todos os artefatos científicos que sejam Artigos (***Paper***) que atendem ao perfil do pesquisador fictício João. Ela pode ser dividida em três partes compreendidas pelos retângulos numerados de um a três. No trecho

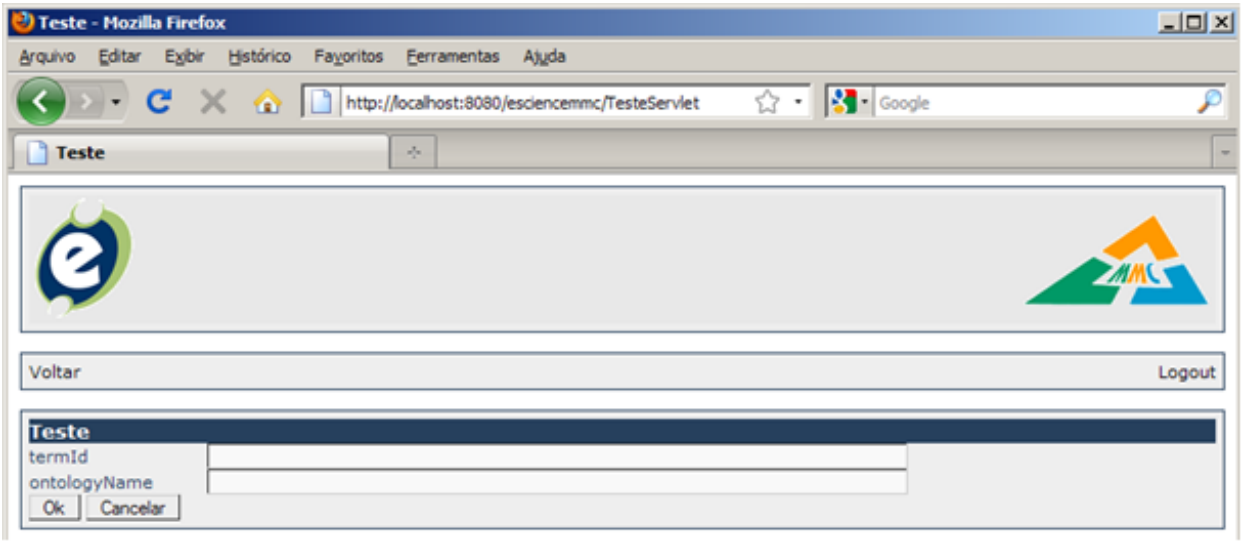

Figura 6.17: Atuação do InvokerAgent exibindo os parâmetros da operação selecionada

compreendido pelo retângulo 1, é a parte da consulta responsável em retornar todos os artefatos científicos que sejam artigos (***Papers***) cujos pesquisadores pertencentes ao profile do pesquisador publicaram. Unindo-se a este conjunto de resultados, temos o trecho compreendido pelo retângulo 2 responsável por retornar todos os artefatos científicos que sejam artigos cujas áreas de interesse pertencentes ao profile do João publicaram. Finalmente, o trecho compreendido pelo retângulo 3 é o responsável por retornar todos os artefatos científicos, sempre artigos, cuja publicação pertence aos projetos de pesquisa associados ao profile selecionado. A figura 6.23 mostra a tela do resultado da pesquisa pelos artefatos científicos.

6.4 Discussões

O estudo de caso tem o objetivo principal de verificar a viabilidade da arquitetura *SASAgent* e modelada pela ontologia *SASOntology*. Implementa os principais atores da arquitetura: plataforma JADE, ontologias em OWL *SASOntology* e OWL-S além de um banco de dados específico para armazenar ontologias.

O que é possível verificar é que a plataforma JADE é robusta e escalável, além da facilidade na criação dos agentes e a definição dos seus objetivos pelo fato de ser uma classe Java comum. A ontologia *SASOntology*, em sua primeira versão, atende

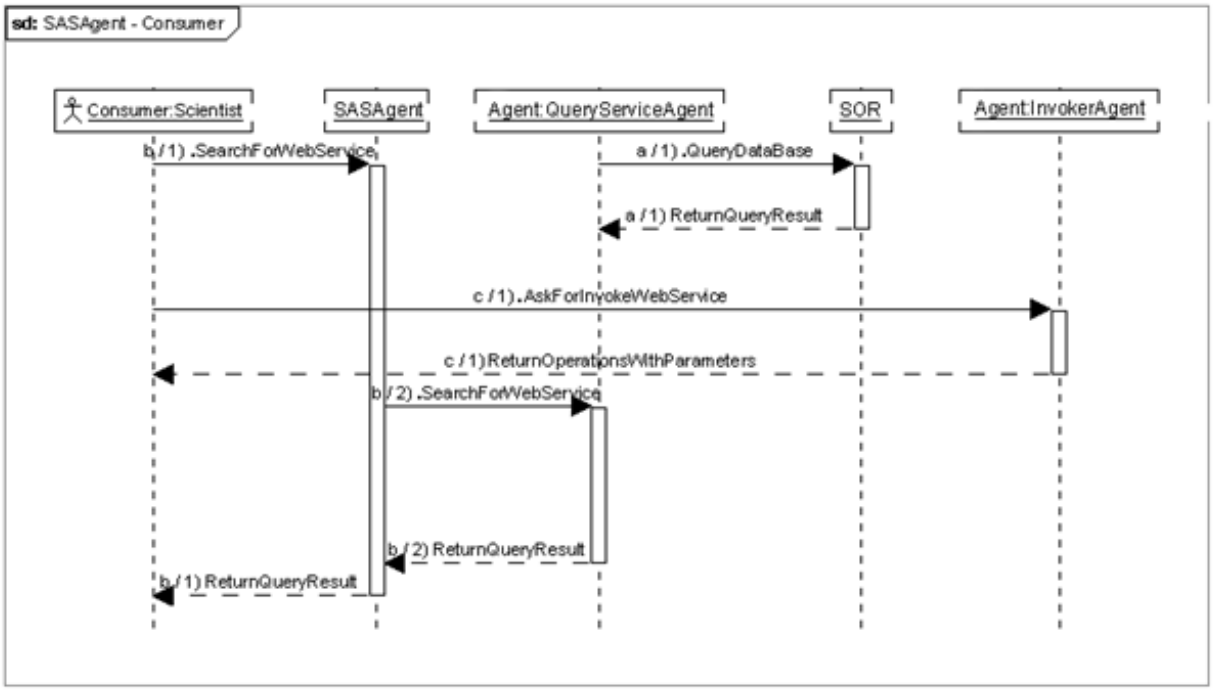

Figura 6.18: Diagrama de sequência para a busca e invocação dos serviços Web

aos requisitos da aplicação pois engloba conceitos relacionados a grupos de pesquisa diversos. A integração com OWL-S é transparente e facilitada pelo uso do editor de OWL-S para o *Protégé*. O SOR atendeu às expectativas, pois une as facilidades de uma base de dados relacional às necessidades específicas de armazenamento das ontologias, e consequentemente, dados ricos semanticamente.

Por outro lado, apesar da invocação de serviços Web em Java gerados pelos plug-ins do Eclipse ter funcionado, testes com serviços desenvolvidos em outras tecnologias como .NET não foram satisfatórios. A ontologia *SASOntology* atende ao domínio de e-Science de maneira geral porém é necessária uma evolução no sentido de integração com ontologias de domínio mais específicas dos grupos de pesquisa. A plataforma *SASAgent* apesar de conseguir publicar, buscar e recuperar artefatos científicos complexos como serviços Web necessita de novos recursos que permitam também a execução e composição de *workflows* científicos.

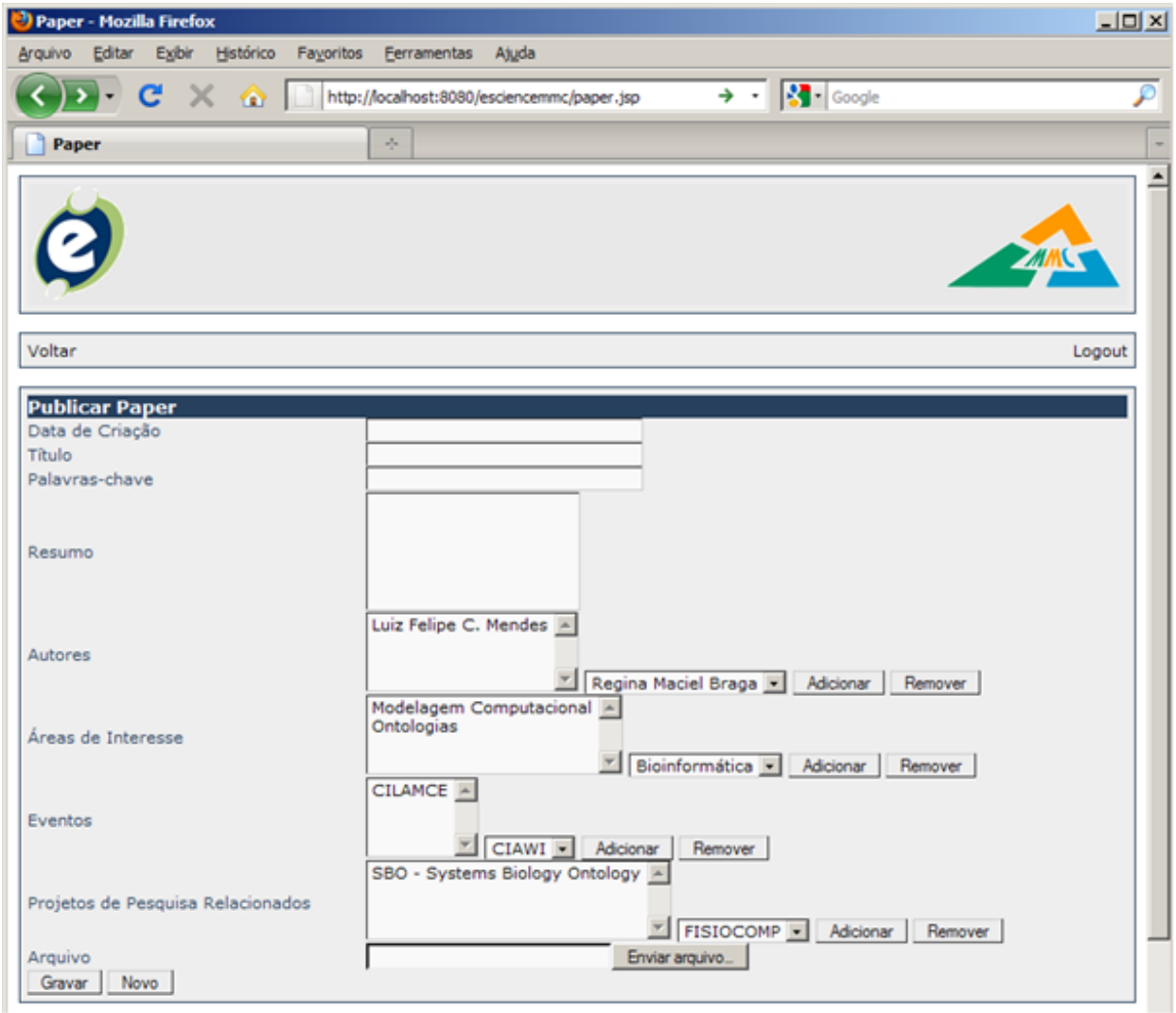

Figura 6.19: Formulário de inserção de artigos utilizando anotações semânticas

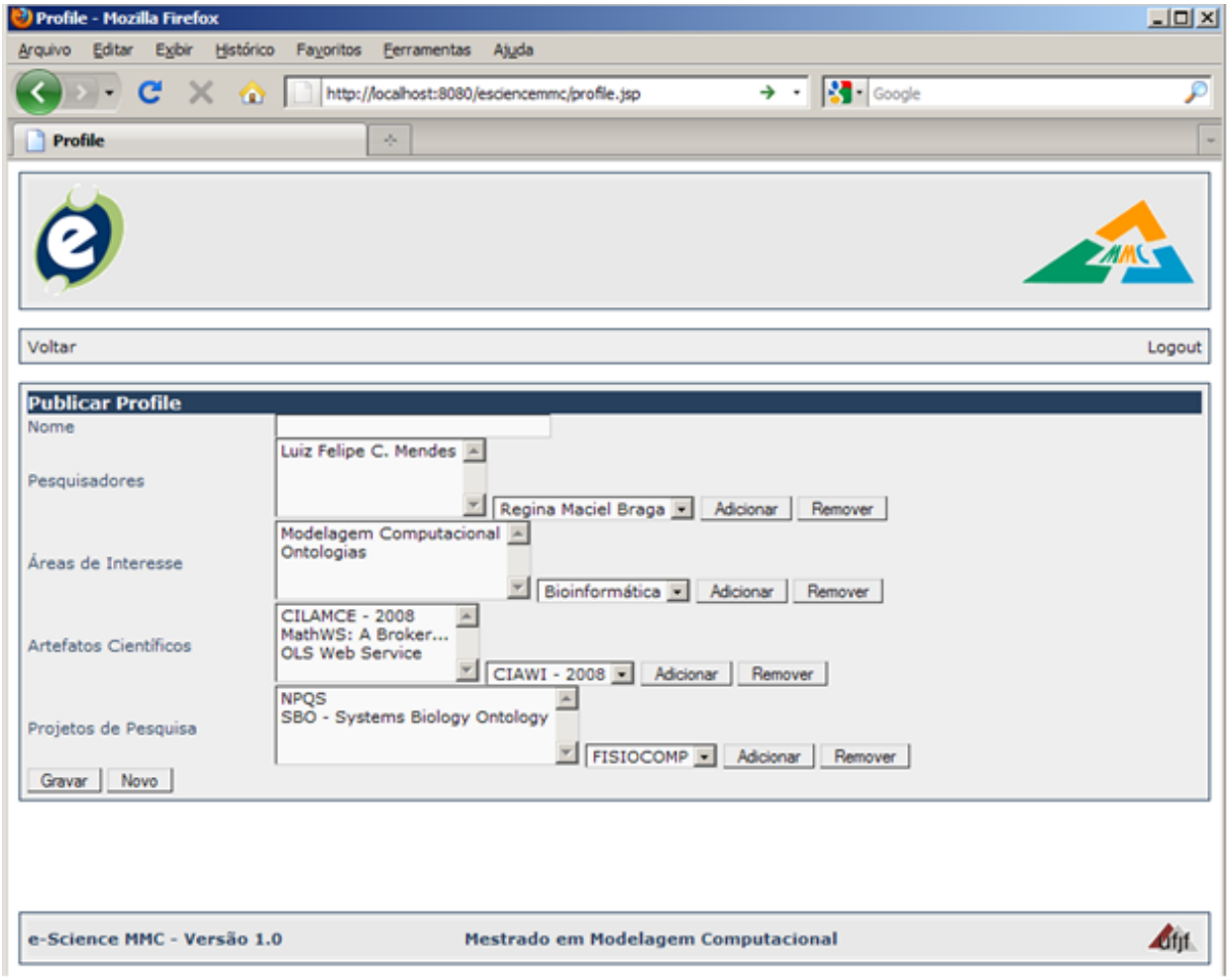

Figura 6.20: Formulário de inserção de *profiles*

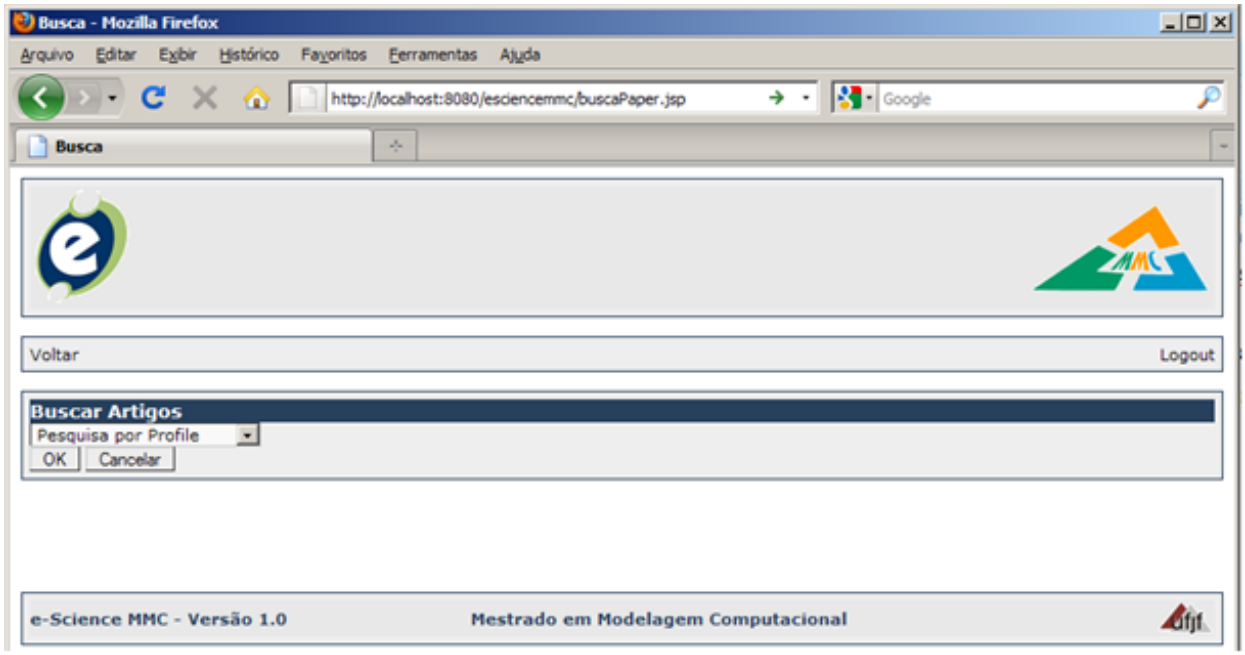

Figura 6.21: Formulário de inserção de *profiles*

```
1  PREFIX ufjf:<http://www.ufjf.br/escience.owl#>
2  SELECT DISTINCT ?titulo
3  WHERE
4  {
5    {
6      ?p ufjf:belongsToScientist ufjf:Joao .          1
7      ?p ufjf:hasScientistAssociated ?pesq .
8      ?pesq ufjf:hasPublishedScientificArtifact ?art .
9      ?art a ufjf:Paper .
10     ?art ufjf:hasTitle ?titulo
11   }
12   UNION
13   {
14     ?p ufjf:belongsToScientist ufjf:Joao .          2
15     ?p ufjf:belongsToSubArea ?area .
16     ?area ufjf:hasPublishedScientificArtifact ?art .
17     ?art a ufjf:Paper .
18     ?art ufjf:hasTitle ?titulo
19   }
20   UNION
21   {
22     ?p ufjf:belongsToScientist ufjf:Joao .
23     ?p ufjf:hasResearchProjectsAllocated ?proj .
24     ?proj ufjf:hasPublishedScientificArtifact ?art .
25     ?art a ufjf:Paper .                             3
26     ?art ufjf:hasTitle ?titulo
27   }
28 }
```

Figura 6.22: Consulta em SPARQL para o profile do João

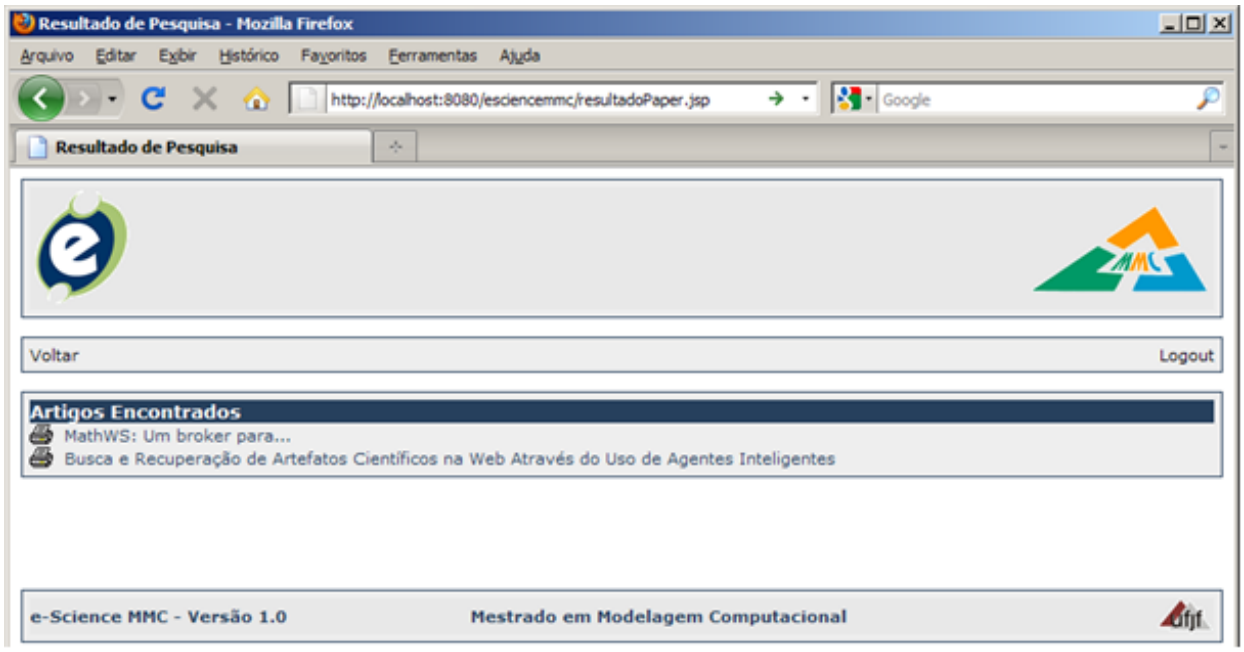

Figura 6.23: Tela de resultado da pesquisa por artigos

Capítulo 7

Considerações Finais

É de conhecimento geral que os projetos de pesquisa que fazem uso da modelagem computacional e recursos computacionais são projetos que geram grande quantidade de produtos, ou, melhor dizendo, artefatos científicos. Grandes bases de dados, algoritmos, modelos, publicações, ferramentas e serviços Web são exemplos deste tipo de artefato. A distância geográfica também não é mais um empecilho no desenvolvimento de trabalhos cooperativos entre projetos de diferentes instituições e países que através da Web tentam disseminar o conhecimento e os resultados gerados.

Assim, é necessário perceber que novos mecanismos são necessários para a organização de todo esse conhecimento e artefatos gerados para prover uma busca e posterior recuperação de maneira inteligente e otimizada. Este ambiente heterogêneo, no qual se constitui a e-Science, é propício à aplicação de conceitos estabelecidos como o uso de agentes com características de reatividade, mobilidade e racionalização, assim como o uso de ontologias para a representação semântica das informações. Este trabalho propõe a integração destes conceitos para a organização e acessibilidade deste conhecimento científico.

Esta dissertação apresentou uma revisão bibliográfica detalhada do conceito de e-Science, o contexto de aplicação, as tendências e limitações do mesmo, justificando a importância do tema. Além disto, revisou o conceito de agentes, desde associações clássicas vindas da IA, metodologias de desenvolvimento de software baseadas em agentes, até as limitações e tendências, tudo com o intuito de justificar a abordagem

escolhida.

Além da revisão bibliográfica sobre os fundamentos teóricos, foi apresentada a arquitetura SASAgent que tem como principal objetivo atender o domínio da modelagem computacional no contexto da e-Science através do uso de agentes, ontologias e artefatos científicos complexos, como serviços Web, permitindo a pesquisadores de diversos grupos compartilharem suas pesquisas. Com o intuito de verificar a viabilidade da mesma, foi desenvolvido uma arquitetura e um estudo de caso testando os componentes da arquitetura.

De acordo com os objetivos ressaltados na introdução, podemos destacar:

- A arquitetura SASAgent foi especificada e detalhada no capítulo 5 e um protótipo foi desenvolvido com o objetivo de verificar a viabilidade da proposta;
- Uma ontologia foi modelada SASOntology e armazenada em um repositório semântico SOR, com o objetivo de dar suporte ao gerenciamento dos artefatos científicos relacionados ao domínio;
- A independência de domínio de aplicação é alcançada através da utilização de ontologias diversas para realizar a busca. Nesta primeira versão, utilizamos apenas a SASOntology. No entanto, a arquitetura SASAgent pode utilizar qualquer ontologia especificada em OWL. O suporte a diferentes papéis também foi delianeado, através da configuração do "profile"do usuário.

A partir dos objetivos definidos para a dissertação e da implementação do estudo de caso algumas considerações são relevantes:

- A plataforma de agentes JADE é se mostrou estável e bem documentada. Além, ainda, de atender aos padrões FIPA de comunicação de agentes tornando-a compatível com qualquer outra plataforma que faça uso do mesmo padrão recomendado. Com isto, qualquer software que seja baseado no uso de agentes, tem no framework JADE um grande aliado. O InvokerAgent, UserAgent, e o QueryServiceAgent são os principais atores da plataforma já que o primeiro é responsável em invocar serviços Web dinamicamente, o segundo representa computacionalmente o pesquisador dentro da plataforma e o último realiza as buscas semânticas na base de dados. O GatewayAgent é o responsável na

interação Pesquisador-Sistema. Vale resslatar que no estudo de caso só não foi implementado um agente que monitorasse os artefatos;

- Em relação às metodologias de desenvolvimento de software baseado em agentes, por outro lado, não há unanimidade no uso, principalmente no que diz respeito ao comportamento e interação dos mesmos. A metodologia INGENIAS foi escolhida principalmente pelo nível de detalhamento encontrado na literatura e por oferecer um editor próprio para a modelagem dos agentes;
- As anotações semânticas são modeladas pela ontologia SASOntology desenvolvida em OWL, padrão recomendado e seguido por qualquer grupo que deseja utilizar de ontologias. A ferramenta Protégé é eficaz neste processo. No entanto, para a realização das anotações semânticas para serviços Web, o plug-in respectivo ainda necessita ser aprimorado;
- A ontologia SASOntology é uma contribuição importante pois foi possível a integração com a OWL-S permitindo a publicação dos serviços Web, , e ainda atende de maneira genérica a qualquer projeto de pesquisa pertencente a qualquer domínio.

7.1 Trabalhos Futuros

Podemos considerar a proposta apresentada nesta dissertação como uma proposta integradora, que faz uso de componentes já conhecidos (e-Science, agentes e ontologias), mas que no entanto não são comumente utilizados de maneira integrada. Desta forma, é possível destacar ainda alguns trabalhos futuros a partir da base da proposta.

O primeiro trabalho seria a evolução da ontologia SASOntology com o intuito de haver maior abrangência na cobertura do domínio da área de pesquisa e mantê-la genérica o suficiente para atender a projetos relacionados. Além disto, mais regras lógicas devem ser consideradas para que o conhecimento implícito seja possível de ser alcançado através de inferência pelos agentes. Estas regras poderiam ser escritas em SWRL, por exemplo. Outra necessidade seria a evolução do protótipo para um sistema completo possibilitando a operacionalização de todas as funcionalidades propostas.

Além disto, a integração com outras ontologias de domínio (Biologia, Química, etc) seja possível com a ontologia SASOntology principalmente para as anotações semânticas e posterior busca dos artefatos através destes conceitos integrados.

O fato deste trabalho ser bastante abrangente outros dois pontos merecem destaque nas suas elucidações e que constituem uma evolução da proposta inicial. O primeiro item é a questão de tolerância a falhas, como, por exemplo, a indisponibilidade de um serviço Web no momento que ele é selecionado. Existem diversos trabalhos que lidam claramente com a disponibilidade de serviços Web no ambiente de grid e talvez, técnicas já existentes possam ser incorporadas a esta arquitetura. O segundo ponto que merece atenção na sua evolução é a Autorização para o acesso a plataforma de gerenciamento dos artefatos científicos. Atualmente, é baseada somente em um login e senha mas que futuramente novas necessidades serão identificadas e uma evolução torna-se necessária.

Apêndice A

Ontologia *SASOntology*

Este apêndice apresenta o código completo na linguagem OWL referente a ontologia *SASOntology*

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:service="http://www.daml.org/services/owl-s/1.2/Service.owl#"
xmlns="http://www.ufjf.br/escience.owl#"
xmlns:swrl="http://www.w3.org/2003/11/swrl#"
xmlns:time="http://www.isi.edu/~pan/damltime/time-entry.owl#"
xmlns:list="http://www.daml.org/services/owl-s/1.2/generic/ObjectList.
    owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:wi1="http://www.example.org/owls/getTermsByName.owl"
xmlns:expr="http://www.daml.org/services/owl-s/1.2/generic/Expression.
    owl#"
xmlns:p1="http://www.owl-ontologies.com/assert.owl#"
xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
xmlns:profile="http://www.daml.org/services/owl-s/1.2/Profile.owl#"
xmlns:process="http://www.daml.org/services/owl-s/1.2/Process.owl#"
xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
xmlns:grounding="http://www.daml.org/services/owl-s/1.2/Grounding.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
```

```
21 xml:base="http://www.ufjf.br/escience.owl">
22 <owl:Ontology rdf:about="">
23 <owl:imports rdf:resource="http://www.example.org/owls/getTermsByName.
       owl"/>
24 <owl:imports rdf:resource="http://www.w3.org/2003/11/swrlb"/>
25 <owl:imports rdf:resource="http://www.w3.org/2003/11/swrl"/>
26 <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/
       Grounding.owl"/>
27 <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/
       Profile.owl"/>
28 </owl:Ontology>
29 <owl:Class rdf:ID="EScienceConcepts"/>
30 <owl:Class rdf:ID="ConceptualModel">
31 <rdfs:subClassOf>
32 <owl:Class rdf:ID="ScientificArtifact"/>
33 </rdfs:subClassOf>
34 <owl:disjointWith>
35 <owl:Class rdf:ID="Dissertation"/>
36 </owl:disjointWith>
37 <owl:disjointWith>
38 <owl:Class rdf:ID="Event"/>
39 </owl:disjointWith>
40 <owl:disjointWith>
41 <owl:Class rdf:ID="ComputationalTool"/>
42 </owl:disjointWith>
43 <owl:disjointWith>
44 <owl:Class rdf:ID="Thesis"/>
45 </owl:disjointWith>
46 <owl:disjointWith>
47 <owl:Class rdf:ID="Publication"/>
48 </owl:disjointWith>
49 </owl:Class>
50 <owl:Class rdf:ID="InterestArea">
51 <rdfs:subClassOf>
52 <owl:Class rdf:ID="ContextConcepts"/>
53 </rdfs:subClassOf>
54 </owl:Class>
55 <owl:Class rdf:ID="Journal">
```

```
<rdfs:subClassOf>
<owl:Class rdf:about="#Publication"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ResearchProject">
<rdfs:subClassOf>
<owl:Class rdf:about="#ContextConcepts"/>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:about="#ScientificArtifact"/>
</owl:allValuesFrom>
<owl:onProperty>
<owl:InverseFunctionalProperty rdf:ID="hasPublishedScientificArtifact"
    />
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:ID="Scientist"/>
</owl:allValuesFrom>
<owl:onProperty>
<owl:InverseFunctionalProperty rdf:ID="hasScientistAssociated"/>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Book">
<rdfs:subClassOf>
<owl:Class rdf:about="#Publication"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ExternalConcept">
<rdfs:subClassOf>
<owl:Class rdf:about="#ContextConcepts"/>
```

```
93  </rdfs:subClassOf>
94  </owl:Class>
95  <owl:Class rdf:ID="Paper">
96  <rdfs:subClassOf>
97  <owl:Class rdf:about="#Publication"/>
98  </rdfs:subClassOf>
99  </owl:Class>
100 <owl:Class rdf:ID="Departament">
101 <rdfs:subClassOf>
102 <owl:Class rdf:ID="Institute"/>
103 </rdfs:subClassOf>
104 </owl:Class>
105 <owl:Class rdf:about="#Dissertation">
106 <owl:disjointWith>
107 <owl:Class rdf:about="#Event"/>
108 </owl:disjointWith>
109 <owl:disjointWith>
110 <owl:Class rdf:about="#ComputationalTool"/>
111 </owl:disjointWith>
112 <owl:disjointWith rdf:resource="#ConceptualModel"/>
113 <owl:disjointWith>
114 <owl:Class rdf:about="#Thesis"/>
115 </owl:disjointWith>
116 <owl:disjointWith>
117 <owl:Class rdf:about="#Publication"/>
118 </owl:disjointWith>
119 <rdfs:subClassOf>
120 <owl:Class rdf:about="#ScientificArtifact"/>
121 </rdfs:subClassOf>
122 </owl:Class>
123 <owl:Class rdf:ID="WebApplication">
124 <rdfs:subClassOf>
125 <owl:Class rdf:about="#ComputationalTool"/>
126 </rdfs:subClassOf>
127 </owl:Class>
128 <owl:Class rdf:ID="University">
129 <rdfs:subClassOf>
130 <owl:Class rdf:ID="Organization"/>
```

```
131 </rdfs:subClassOf>
132 </owl:Class>
133 <owl:Class rdf:ID="TechnicalReport">
134 <rdfs:subClassOf>
135 <owl:Class rdf:about="#Publication"/>
136 </rdfs:subClassOf>
137 </owl:Class>
138 <owl:Class rdf:ID="ERM">
139 <rdfs:subClassOf rdf:resource="#ConceptualModel"/>
140 <rdfs:subClassOf>
141 <owl:Restriction>
142 <owl:onProperty>
143 <owl:FunctionalProperty rdf:ID="hasDataBaseDestination"/>
144 </owl:onProperty>
145 <owl:allValuesFrom>
146 <owl:Class>
147 <owl:oneOf rdf:parseType="Collection">
148 <Database rdf:ID="SQLServer"/>
149 <Database rdf:ID="DB2"/>
150 <Database rdf:ID="Oracle"/>
151 <Database rdf:ID="MySQL"/>
152 <Database rdf:ID="PostgreSQL"/>
153 </owl:oneOf>
154 </owl:Class>
155 </owl:allValuesFrom>
156 </owl:Restriction>
157 </rdfs:subClassOf>
158 </owl:Class>
159 <owl:Class rdf:ID="SubInterestArea">
160 <rdfs:subClassOf rdf:resource="#InterestArea"/>
161 </owl:Class>
162 <owl:Class rdf:about="#Institute">
163 <rdfs:subClassOf rdf:resource="#University"/>
164 </owl:Class>
165 <owl:Class rdf:ID="ScientistRole">
166 <rdfs:subClassOf>
167 <owl:Class rdf:about="#ContextConcepts"/>
168 </rdfs:subClassOf>
```

```
169 </owl:Class>
170 <owl:Class rdf:ID="Agency">
171 <rdfs:subClassOf>
172 <owl:Class rdf:about="#Organization"/>
173 </rdfs:subClassOf>
174 </owl:Class>
175 <owl:Class rdf:about="#Publication">
176 <owl:disjointWith rdf:resource="#Dissertation"/>
177 <owl:disjointWith>
178 <owl:Class rdf:about="#Event"/>
179 </owl:disjointWith>
180 <owl:disjointWith>
181 <owl:Class rdf:about="#ComputationalTool"/>
182 </owl:disjointWith>
183 <owl:disjointWith rdf:resource="#ConceptualModel"/>
184 <owl:disjointWith>
185 <owl:Class rdf:about="#Thesis"/>
186 </owl:disjointWith>
187 <rdfs:subClassOf>
188 <owl:Class rdf:about="#ScientificArtifact"/>
189 </rdfs:subClassOf>
190 </owl:Class>
191 <owl:Class rdf:ID="Library">
192 <rdfs:subClassOf>
193 <owl:Class rdf:about="#ComputationalTool"/>
194 </rdfs:subClassOf>
195 </owl:Class>
196 <owl:Class rdf:about="#Scientist">
197 <rdfs:subClassOf>
198 <owl:Restriction>
199 <owl:onProperty>
200 <owl:FunctionalProperty rdf:ID="hasTitleDegree"/>
201 </owl:onProperty>
202 <owl:allValuesFrom>
203 <owl:Class>
204 <owl:oneOf rdf:parseType="Collection">
205 <TitleDegree rdf:ID="Graduated"/>
206 <TitleDegree rdf:ID="MasterDegree"/>
```

```
<TitleDegree rdf:ID="DoctorDegree"/>
<TitleDegree rdf:ID="PosDoctorDegree"/>
</owl:oneOf>
</owl:Class>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:FunctionalProperty rdf:ID="hasRole"/>
</owl:onProperty>
<owl:allValuesFrom>
<owl:Class>
<owl:oneOf rdf:parseType="Collection">
<ScientistRole rdf:ID="Student"/>
<ScientistRole rdf:ID="Collaborator"/>
<ScientistRole rdf:ID="Coordinator"/>
</owl:oneOf>
</owl:Class>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="#ContextConcepts"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Profile">
<rdfs:subClassOf>
<owl:Class rdf:about="#ContextConcepts"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Database">
<rdfs:subClassOf>
<owl:Class rdf:about="#ContextConcepts"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Ontology">
```

```
245 <rdfs:subClassOf rdf:resource="#ConceptualModel"/>
246 </owl:Class>
247 <owl:Class rdf:about="#Thesis">
248 <rdfs:subClassOf>
249 <owl:Class rdf:about="#ScientificArtifact"/>
250 </rdfs:subClassOf>
251 <owl:disjointWith rdf:resource="#Dissertation"/>
252 <owl:disjointWith>
253 <owl:Class rdf:about="#Event"/>
254 </owl:disjointWith>
255 <owl:disjointWith>
256 <owl:Class rdf:about="#ComputationalTool"/>
257 </owl:disjointWith>
258 <owl:disjointWith rdf:resource="#ConceptualModel"/>
259 <owl:disjointWith rdf:resource="#Publication"/>
260 </owl:Class>
261 <owl:Class rdf:about="#Organization">
262 <rdfs:subClassOf>
263 <owl:Class rdf:about="#ContextConcepts"/>
264 </rdfs:subClassOf>
265 </owl:Class>
266 <owl:Class rdf:about="#Event">
267 <rdfs:subClassOf>
268 <owl:Class rdf:about="#ScientificArtifact"/>
269 </rdfs:subClassOf>
270 <owl:disjointWith rdf:resource="#Dissertation"/>
271 <owl:disjointWith>
272 <owl:Class rdf:about="#ComputationalTool"/>
273 </owl:disjointWith>
274 <owl:disjointWith rdf:resource="#ConceptualModel"/>
275 <owl:disjointWith rdf:resource="#Thesis"/>
276 <owl:disjointWith rdf:resource="#Publication"/>
277 </owl:Class>
278 <owl:Class rdf:ID="TitleDegree">
279 <rdfs:subClassOf>
280 <owl:Class rdf:about="#ContextConcepts"/>
281 </rdfs:subClassOf>
282 </owl:Class>
```

```
283 <owl:Class rdf:ID="StandAloneApplication">
284 <rdfs:subClassOf>
285 <owl:Class rdf:about="#ComputationalTool"/>
286 </rdfs:subClassOf>
287 </owl:Class>
288 <owl:Class rdf:about="#ComputationalTool">
289 <rdfs:subClassOf>
290 <owl:Class rdf:about="#ScientificArtifact"/>
291 </rdfs:subClassOf>
292 <owl:disjointWith rdf:resource="#Dissertation"/>
293 <owl:disjointWith rdf:resource="#Event"/>
294 <owl:disjointWith rdf:resource="#ConceptualModel"/>
295 <owl:disjointWith rdf:resource="#Thesis"/>
296 <owl:disjointWith rdf:resource="#Publication"/>
297 </owl:Class>
298 <owl:Class rdf:ID="UML">
299 <rdfs:subClassOf rdf:resource="#ConceptualModel"/>
300 </owl:Class>
301 <owl:Class rdf:about="#ContextConcepts">
302 <rdfs:subClassOf rdf:resource="#EScienceConcepts"/>
303 </owl:Class>
304 <owl:Class rdf:ID="WebService">
305 <rdfs:subClassOf rdf:resource="#ComputationalTool"/>
306 </owl:Class>
307 <owl:Class rdf:about="#ScientificArtifact">
308 <rdfs:subClassOf rdf:resource="#EScienceConcepts"/>
309 </owl:Class>
310 <owl:ObjectProperty rdf:ID="hasUniversityAssociated">
311 <rdfs:range rdf:resource="#University"/>
312 </owl:ObjectProperty>
313 <owl:ObjectProperty rdf:ID="scientificArtifactPublishedBy">
314 <rdfs:range>
315 <owl:Class>
316 <owl:unionOf rdf:parseType="Collection">
317 <owl:Class rdf:about="#Profile"/>
318 <owl:Class rdf:about="#SubInterestArea"/>
319 <owl:Class rdf:about="#ResearchProject"/>
320 <owl:Class rdf:about="#Scientist"/>
```

```
321 </owl:unionOf>
322 </owl:Class>
323 </rdfs:range>
324 <rdfs:domain rdf:resource="#ScientificArtifact"/>
325 <owl:inverseOf>
326 <owl:InverseFunctionalProperty rdf:about="#
        hasPublishedScientificArtifact"/>
327 </owl:inverseOf>
328 </owl:ObjectProperty>
329 <owl:ObjectProperty rdf:ID="hasResearchProjectsAllocated">
330 <rdfs:domain>
331 <owl:Class>
332 <owl:unionOf rdf:parseType="Collection">
333 <owl:Class rdf:about="#Departament"/>
334 <owl:Class rdf:about="#Profile"/>
335 <owl:Class rdf:about="#SubInterestArea"/>
336 <owl:Class rdf:about="#Paper"/>
337 </owl:unionOf>
338 </owl:Class>
339 </rdfs:domain>
340 <rdfs:range rdf:resource="#ResearchProject"/>
341 </owl:ObjectProperty>
342 <owl:ObjectProperty rdf:ID="hasProfileAssociated">
343 <rdfs:range rdf:resource="#Profile"/>
344 <rdfs:domain rdf:resource="#SubInterestArea"/>
345 </owl:ObjectProperty>
346 <owl:ObjectProperty rdf:ID="belongsToAreaInteresse">
347 <rdfs:domain rdf:resource="#SubInterestArea"/>
348 <rdfs:range rdf:resource="#InterestArea"/>
349 </owl:ObjectProperty>
350 <owl:ObjectProperty rdf:ID="hasRelatedWith">
351 <rdfs:range rdf:resource="#ExternalConcept"/>
352 <rdfs:domain>
353 <owl:Class>
354 <owl:unionOf rdf:parseType="Collection">
355 <owl:Class rdf:about="#ScientificArtifact"/>
356 <owl:Class rdf:about="#Profile"/>
357 </owl:unionOf>
```

```
358 </owl:Class>
359 </rdfs:domain>
360 </owl:ObjectProperty>
361 <owl:DatatypeProperty rdf:ID="hasISBN">
362 <rdfs:domain rdf:resource="#Book"/>
363 </owl:DatatypeProperty>
364 <owl:FunctionalProperty rdf:ID="hasLocation">
365 <rdfs:domain rdf:resource="#Organization"/>
366 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
      />
367 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
368 </owl:FunctionalProperty>
369 <owl:FunctionalProperty rdf:ID="hasHTTPAdress">
370 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
371 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
      />
372 <rdfs:domain>
373 <owl:Class>
374 <owl:unionOf rdf:parseType="Collection">
375 <owl:Class rdf:about="#WebApplication"/>
376 <owl:Class rdf:about="#ResearchProject"/>
377 </owl:unionOf>
378 </owl:Class>
379 </rdfs:domain>
380 </owl:FunctionalProperty>
381 <owl:FunctionalProperty rdf:ID="hasStartDate">
382 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
      />
383 <rdfs:domain>
384 <owl:Class>
385 <owl:unionOf rdf:parseType="Collection">
386 <owl:Class rdf:about="#Event"/>
387 <owl:Class rdf:about="#ResearchProject"/>
388 </owl:unionOf>
389 </owl:Class>
390 </rdfs:domain>
391 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
392 </owl:FunctionalProperty>
```

```
<owl:FunctionalProperty rdf:ID="guideResearhProject">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
<owl:inverseOf>
<owl:InverseFunctionalProperty rdf:ID="hasGuide"/>
</owl:inverseOf>
<rdfs:domain rdf:resource="#Scientist"/>
<rdfs:range>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#ResearchProject"/>
<owl:Class rdf:about="#Thesis"/>
</owl:unionOf>
</owl:Class>
</rdfs:range>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="belongsToScientist">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
<rdfs:range rdf:resource="#Scientist"/>
<rdfs:domain rdf:resource="#Profile"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#hasDataBaseDestination">
<rdfs:domain rdf:resource="#ERM"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#hasTitleDegree">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
<rdfs:domain rdf:resource="#Scientist"/>
<rdfs:range rdf:resource="#TitleDegree"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasCreationDate">
<rdfs:domain rdf:resource="#ScientificArtifact"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
    />
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasModellingTool">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
```

```
    />
<rdfs:domain>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#UML"/>
<owl:Class rdf:about="#ERM"/>
</owl:unionOf>
</owl:Class>
</rdfs:domain>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasISRN">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
    />
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdfs:domain rdf:resource="#TechnicalReport"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasName">
<rdfs:domain>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#InterestArea"/>
<owl:Class rdf:about="#Scientist"/>
<owl:Class rdf:about="#ResearchProject"/>
<owl:Class rdf:about="#Organization"/>
<owl:Class rdf:about="#Profile"/>
<owl:Class rdf:about="#ExternalConcept"/>
</owl:unionOf>
</owl:Class>
</rdfs:domain>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
    />
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasDiagramName">
<rdfs:domain rdf:resource="#UML"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
    />
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
464 </owl:FunctionalProperty>
465 <owl:FunctionalProperty rdf:ID="hasTitle">
466 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
      />
467 <rdfs:domain>
468 <owl:Class>
469 <owl:unionOf rdf:parseType="Collection">
470 <owl:Class rdf:about="#ScientificArtifact"/>
471 <owl:Class rdf:about="#ResearchProject"/>
472 </owl:unionOf>
473 </owl:Class>
474 </rdfs:domain>
475 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
476 </owl:FunctionalProperty>
477 <owl:FunctionalProperty rdf:ID="hasEdition">
478 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
      />
479 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
480 <rdfs:domain rdf:resource="#Book"/>
481 </owl:FunctionalProperty>
482 <owl:FunctionalProperty rdf:about="#hasRole">
483 <rdfs:range rdf:resource="#ScientistRole"/>
484 <rdfs:domain rdf:resource="#Scientist"/>
485 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
486 </owl:FunctionalProperty>
487 <owl:FunctionalProperty rdf:ID="hasOWLSServiceFile">
488 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
      />
489 <rdfs:domain rdf:resource="#WebService"/>
490 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
491 </owl:FunctionalProperty>
492 <owl:FunctionalProperty rdf:ID="hasVersion">
493 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
      />
494 <rdfs:domain rdf:resource="#TechnicalReport"/>
495 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
496 </owl:FunctionalProperty>
497 <owl:FunctionalProperty rdf:ID="hasKeyWords">
```

```
498 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
499 <rdfs:domain>
500 <owl:Class>
501 <owl:unionOf rdf:parseType="Collection">
502 <owl:Class rdf:about="#ScientificArtifact"/>
503 <owl:Class rdf:about="#ResearchProject"/>
504 </owl:unionOf>
505 </owl:Class>
506 </rdfs:domain>
507 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
      />
508 </owl:FunctionalProperty>
509 <owl:FunctionalProperty rdf:ID="hasFile">
510 <rdfs:domain>
511 <owl:Class>
512 <owl:unionOf rdf:parseType="Collection">
513 <owl:Class rdf:about="#Publication"/>
514 <owl:Class rdf:about="#ConceptualModel"/>
515 <owl:Class rdf:about="#Library"/>
516 <owl:Class rdf:about="#StandAloneApplication"/>
517 </owl:unionOf>
518 </owl:Class>
519 </rdfs:domain>
520 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
      />
521 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
522 </owl:FunctionalProperty>
523 <owl:FunctionalProperty rdf:ID="hasNumber">
524 <rdfs:domain rdf:resource="#Journal"/>
525 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
526 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
      />
527 </owl:FunctionalProperty>
528 <owl:FunctionalProperty rdf:ID="isDraft">
529 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
530 <rdfs:domain rdf:resource="#TechnicalReport"/>
531 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
      />
```

```
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasProfile">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
<rdfs:range rdf:resource="#Profile"/>
<rdfs:domain rdf:resource="#Scientist"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasOWLSProfileFile">
<rdfs:domain rdf:resource="#WebService"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
    />
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasModellingLanguage">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
    />
<rdfs:domain rdf:resource="#Ontology"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasEndDate">
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
    />
<rdfs:domain>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#Event"/>
<owl:Class rdf:about="#ResearchProject"/>
</owl:unionOf>
</owl:Class>
</rdfs:domain>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasOWLProcessFile">
<rdfs:domain rdf:resource="#WebService"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
    />
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasOWLSGroundingFile">
```

```
566 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
        />
567 <rdfs:domain rdf:resource="#WebService"/>
568 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
569 </owl:FunctionalProperty>
570 <owl:FunctionalProperty rdf:ID="hasDevelopmentLanguage">
571 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
        />
572 <rdfs:domain rdf:resource="#ComputationalTool"/>
573 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
574 </owl:FunctionalProperty>
575 <owl:FunctionalProperty rdf:ID="hasAbstract">
576 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
        />
577 <rdfs:domain>
578 <owl:Class>
579 <owl:unionOf rdf:parseType="Collection">
580 <owl:Class rdf:about="#ScientificArtifact"/>
581 <owl:Class rdf:about="#ResearchProject"/>
582 </owl:unionOf>
583 </owl:Class>
584 </rdfs:domain>
585 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
586 </owl:FunctionalProperty>
587 <owl:FunctionalProperty rdf:ID="hasID">
588 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
        />
589 <rdfs:domain>
590 <owl:Class>
591 <owl:unionOf rdf:parseType="Collection">
592 <owl:Class rdf:about="#InterestArea"/>
593 <owl:Class rdf:about="#ScientificArtifact"/>
594 <owl:Class rdf:about="#Scientist"/>
595 <owl:Class rdf:about="#ResearchProject"/>
596 <owl:Class rdf:about="#Organization"/>
597 <owl:Class rdf:about="#Profile"/>
598 </owl:unionOf>
599 </owl:Class>
```

```
600 </rdfs:domain>
601 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
602 </owl:FunctionalProperty>
603 <owl:FunctionalProperty rdf:ID="hasAcronym">
604 <rdfs:domain rdf:resource="#Organization"/>
605 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
      />
606 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
607 </owl:FunctionalProperty>
608 <owl:InverseFunctionalProperty rdf:ID="belongsToDepartament">
609 <rdfs:domain>
610 <owl:Class>
611 <owl:unionOf rdf:parseType="Collection">
612 <owl:Class rdf:about="#Scientist"/>
613 <owl:Class rdf:about="#ResearchProject"/>
614 </owl:unionOf>
615 </owl:Class>
616 </rdfs:domain>
617 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
      FunctionalProperty"/>
618 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
619 <rdfs:range>
620 <owl:Class>
621 <owl:unionOf rdf:parseType="Collection">
622 <owl:Class rdf:about="#Profile"/>
623 <owl:Class rdf:about="#SubInterestArea"/>
624 <owl:Class rdf:about="#ResearchProject"/>
625 </owl:unionOf>
626 </owl:Class>
627 </rdfs:range>
628 </owl:InverseFunctionalProperty>
629 <owl:InverseFunctionalProperty rdf:ID="approvedPapers">
630 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
631 <owl:inverseOf>
632 <owl:InverseFunctionalProperty rdf:ID="publishedIn"/>
633 </owl:inverseOf>
634 <rdfs:range rdf:resource="#Paper"/>
635 <rdfs:domain rdf:resource="#Event"/>
```

```
636 </owl:InverseFunctionalProperty>
637 <owl:InverseFunctionalProperty rdf:about="#
        hasPublishedScientificArtifact">
638 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
639 <owl:inverseOf rdf:resource="#scientificArtifactPublishedBy"/>
640 <rdfs:domain>
641 <owl:Class>
642 <owl:unionOf rdf:parseType="Collection">
643 <owl:Class rdf:about="#ResearchProject"/>
644 <owl:Class rdf:about="#Scientist"/>
645 <owl:Class rdf:about="#Profile"/>
646 <owl:Class rdf:about="#SubInterestArea"/>
647 </owl:unionOf>
648 </owl:Class>
649 </rdfs:domain>
650 <rdfs:range rdf:resource="#ScientificArtifact"/>
651 </owl:InverseFunctionalProperty>
652 <owl:InverseFunctionalProperty rdf:ID="belongsToSubArea">
653 <rdfs:range>
654 <owl:Class>
655 <owl:unionOf rdf:parseType="Collection">
656 <owl:Class rdf:about="#ResearchProject"/>
657 <owl:Class rdf:about="#SubInterestArea"/>
658 </owl:unionOf>
659 </owl:Class>
660 </rdfs:range>
661 <rdfs:domain>
662 <owl:Class>
663 <owl:unionOf rdf:parseType="Collection">
664 <owl:Class rdf:about="#ScientificArtifact"/>
665 <owl:Class rdf:about="#Scientist"/>
666 <owl:Class rdf:about="#ResearchProject"/>
667 <owl:Class rdf:about="#Profile"/>
668 </owl:unionOf>
669 </owl:Class>
670 </rdfs:domain>
671 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
672 </owl:InverseFunctionalProperty>
```

```
673 <owl:InverseFunctionalProperty rdf:ID="hasFinanced">
674 <owl:inverseOf>
675 <owl:InverseFunctionalProperty rdf:ID="financedBy"/>
676 </owl:inverseOf>
677 <rdfs:domain rdf:resource="#Agency"/>
678 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
      FunctionalProperty"/>
679 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
680 <rdfs:range rdf:resource="#ResearchProject"/>
681 </owl:InverseFunctionalProperty>
682 <owl:InverseFunctionalProperty rdf:about="#financedBy">
683 <rdfs:range rdf:resource="#Agency"/>
684 <owl:inverseOf rdf:resource="#hasFinanced"/>
685 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
686 <rdfs:domain rdf:resource="#ResearchProject"/>
687 </owl:InverseFunctionalProperty>
688 <owl:InverseFunctionalProperty rdf:about="#publishedIn">
689 <rdfs:domain rdf:resource="#Paper"/>
690 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
      FunctionalProperty"/>
691 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
692 <rdfs:range rdf:resource="#Event"/>
693 <owl:inverseOf rdf:resource="#approvedPapers"/>
694 </owl:InverseFunctionalProperty>
695 <owl:InverseFunctionalProperty rdf:about="#hasScientistAssociated">
696 <rdfs:range>
697 <owl:Class>
698 <owl:unionOf rdf:parseType="Collection">
699 <owl:Class rdf:about="#Profile"/>
700 <owl:Class rdf:about="#ResearchProject"/>
701 <owl:Class rdf:about="#Scientist"/>
702 </owl:unionOf>
703 </owl:Class>
704 </rdfs:range>
705 <rdfs:domain>
706 <owl:Class>
707 <owl:unionOf rdf:parseType="Collection">
708 <owl:Class rdf:about="#ResearchProject"/>
```

```
709 <owl:Class rdf:about="#Profile"/>
710 <owl:Class rdf:about="#SubInterestArea"/>
711 </owl:unionOf>
712 </owl:Class>
713 </rdfs:domain>
714 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
715 </owl:InverseFunctionalProperty>
716 <owl:InverseFunctionalProperty rdf:about="#hasGuide">
717 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
718 <rdfs:domain>
719 <owl:Class>
720 <owl:unionOf rdf:parseType="Collection">
721 <owl:Class rdf:about="#ResearchProject"/>
722 <owl:Class rdf:about="#Thesis"/>
723 </owl:unionOf>
724 </owl:Class>
725 </rdfs:domain>
726 <rdfs:range rdf:resource="#Scientist"/>
727 <owl:inverseOf rdf:resource="#guideResearhProject"/>
728 </owl:InverseFunctionalProperty>
729 <SubInterestArea rdf:ID="AnaliseAlgoritmosComplexidade">
730 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
731 >AnÃ¡lise de Algoritmos e Complexidade</hasName>
732 <belongsToAreaInteresse>
733 <InterestArea rdf:ID="TeoriadaComputacao">
734 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
735 >4</hasID>
736 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
737 >Teoria da ComputaÃ§Ã£o</hasName>
738 </InterestArea>
739 </belongsToAreaInteresse>
740 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
741 >12</hasID>
742 </SubInterestArea>
743 <Scientist rdf:ID="LuizFelipe">
744 <hasPublishedScientificArtifact>
745 <Paper rdf:ID="BuscaRecuperacaoArtefatosCientificosWeb">
746 <hasFile rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
747 >CILAMCE-PAP0156.pdf</hasFile>
748 <hasTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
749 >Busca e RecuperaÃ§Ã£o de Artefatos CientÃ?ficos na Web com o Uso de
        Agentes Inteligentes</hasTitle>
750 <scientificArtifactPublishedBy>
751 <Scientist rdf:ID="FernandaCampos">
752 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
753 >Fernanda Campos</hasName>
754 <hasRole rdf:resource="#Collaborator"/>
755 <hasPublishedScientificArtifact rdf:resource="#
        BuscaRecuperacaoArtefatosCientificosWeb"/>
756 <hasPublishedScientificArtifact>
757 <Paper rdf:ID="MathWS">
758 <hasAbstract rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
759 >MathWS: Um broker para....</hasAbstract>
760 <publishedIn>
761 <Event rdf:ID="CILAMCE2008">
762 <scientificArtifactPublishedBy rdf:resource="#LuizFelipe"/>
763 <hasKeyWords rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
764 >engenharia metodos computacionais</hasKeyWords>
765 <hasAbstract rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
766 >CILAMCE 2008</hasAbstract>
767 <hasTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
768 >CILAMCE 2008</hasTitle>
769 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
770 >1</hasID>
771 <approvedPapers rdf:resource="#MathWS"/>
772 <approvedPapers rdf:resource="#BuscaRecuperacaoArtefatosCientificosWeb"
        />
773 </Event>
774 </publishedIn>
775 <hasTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
776 >MathWS: Um broker para....</hasTitle>
777 <hasResearchProjectsAllocated>
778 <ResearchProject rdf:ID="MIOLO">
779 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
780 >2</hasID>
781 <belongsToDepartament>
```

```
<Departament rdf:ID="CGCO">
<hasResearchProjectsAllocated rdf:resource="#MIOLO"/>
<hasAcronym rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>CGCO</hasAcronym>
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Centro de GestÃ£o de Conhecimento Organizacional</hasName>
<hasLocation rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Juiz de Fora/MG/Brasil</hasLocation>
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>2</hasID>
</Departament>
</belongsToDepartament>
<hasKeyWords rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>php framework software livre</hasKeyWords>
<hasScientistAssociated>
<Scientist rdf:ID="ElyMatos">
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Ely Matos</hasName>
<hasTitleDegree rdf:resource="#MasterDegree"/>
<guideResearhProject rdf:resource="#MIOLO"/>
<hasRole rdf:resource="#Student"/>
<belongsToDepartament>
<Profile rdf:ID="profileJoao">
<hasResearchProjectsAllocated rdf:resource="#MIOLO"/>
<belongsToScientist>
<Scientist rdf:ID="Joao">
<hasProfile rdf:resource="#profileJoao"/>
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>5</hasID>
<hasTitleDegree rdf:resource="#MasterDegree"/>
<hasRole rdf:resource="#Student"/>
<belongsToDepartament rdf:resource="#CGCO"/>
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>JoÃ£o</hasName>
</Scientist>
</belongsToScientist>
<belongsToSubArea>
<SubInterestArea rdf:ID="EngenhariadeSoftware">
```

```
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Engenharia de Software</hasName>
<belongsToAreaInteresse>
<InterestArea rdf:ID="MetodologiasTecnicasComputacao">
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Metodologia e TÃ©cnicas da ComputaÃ§Ã£o</hasName>
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>2</hasID>
</InterestArea>
</belongsToAreaInteresse>
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>4</hasID>
</SubInterestArea>
</belongsToSubArea>
<hasScientistAssociated rdf:resource="#FernandaCampos"/>
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Profile do JoÃ£o</hasName>
<belongsToSubArea>
<SubInterestArea rdf:ID="BancodeDados">
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>3</hasID>
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Banco de Dados</hasName>
<belongsToAreaInteresse rdf:resource="#MetodologiasTecnicasComputacao"
    />
</SubInterestArea>
</belongsToSubArea>
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>1</hasID>
<hasScientistAssociated rdf:resource="#LuizFelipe"/>
<hasResearchProjectsAllocated>
<ResearchProject rdf:ID="NPQS">
<hasAbstract rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Abstract</hasAbstract>
<hasScientistAssociated rdf:resource="#LuizFelipe"/>
<hasScientistAssociated>
<Scientist rdf:ID="ReginaMaciel">
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
```

```
857 >2</hasID>
858 <guideResearhProject rdf:resource="#NPQS"/>
859 <hasPublishedScientificArtifact rdf:resource="#MathWS"/>
860 <belongsToDepartament rdf:resource="#NPQS"/>
861 <hasPublishedScientificArtifact rdf:resource="#
        BuscaRecuperacaoArtefatosCientificosWeb"/>
862 <hasRole rdf:resource="#Coordinator"/>
863 <belongsToDepartament>
864 <Departament rdf:ID="DCC">
865 <hasResearchProjectsAllocated rdf:resource="#NPQS"/>
866 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
867 >1</hasID>
868 <hasLocation rdf:datatype=
869 "http://www.w3.org/2001/XMLSchema#string"
870 >Juiz de Fora/MG/Brasil</hasLocation>
871 <hasName rdf:datatype=
872 "http://www.w3.org/2001/XMLSchema#string"
873 >Departamento de CiÃ?ncia da ComputaÃ§Ã£o</hasName>
874 <hasAcronym rdf:datatype=
875 "http://www.w3.org/2001/XMLSchema#string"
876 >DCC</hasAcronym>
877 </Departament>
878 </belongsToDepartament>
879 <hasTitleDegree rdf:resource="#DoctorDegree"/>
880 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
881 >Regina Maciel</hasName>
882 </Scientist>
883 </hasScientistAssociated>
884 <financedBy>
885 <Agency rdf:ID="CNPQ">
886 <hasLocation rdf:datatype=
887 "http://www.w3.org/2001/XMLSchema#string"
888 >BrasÃ?lia/DF/Brasil</hasLocation>
889 <hasAcronym rdf:datatype=
890 "http://www.w3.org/2001/XMLSchema#string"
891 >CNPQ</hasAcronym>
892 <hasFinanced>
893 <ResearchProject rdf:ID="FISIOCOMP">
```

```
894 <hasHTTPAdress rdf:datatype=
895 "http://www.w3.org/2001/XMLSchema#string"
896 >www.fisiocomp.ufjf.br</hasHTTPAdress>
897 <belongsToSubArea>
898 <SubInterestArea rdf:ID="ModelosAnaliticosSimulacao">
899 <belongsToAreaInteresse>
900 <InterestArea rdf:ID="MatematicadaComputacao">
901 <hasName rdf:datatype=
902 "http://www.w3.org/2001/XMLSchema#string"
903 >MatemÃ¡tica da ComputaÃ§Ã£o</hasName>
904 <hasID rdf:datatype=
905 "http://www.w3.org/2001/XMLSchema#int"
906 >1</hasID>
907 </InterestArea>
908 </belongsToAreaInteresse>
909 <hasID rdf:datatype=
910 "http://www.w3.org/2001/XMLSchema#int"
911 >2</hasID>
912 <hasName rdf:datatype=
913 "http://www.w3.org/2001/XMLSchema#string"
914 >Modelos AnalÃ?ticos e de SimulaÃ§Ã£o</hasName>
915 </SubInterestArea>
916 </belongsToSubArea>
917 <belongsToSubArea>
918 <SubInterestArea rdf:ID="ArquiteturaSistemasComputacao">
919 <belongsToAreaInteresse>
920 <InterestArea rdf:ID="SistemasdeComputacao">
921 <hasName rdf:datatype=
922 "http://www.w3.org/2001/XMLSchema#string"
923 >Sistemas de ComputaÃ§Ã£o</hasName>
924 <hasID rdf:datatype=
925 "http://www.w3.org/2001/XMLSchema#int"
926 >3</hasID>
927 </InterestArea>
928 </belongsToAreaInteresse>
929 <hasName rdf:datatype=
930 "http://www.w3.org/2001/XMLSchema#string"
931 >Arquitetura e Sistemas de ComputaÃ§Ã£o</hasName>
```

```
<hasID rdf:datatype=
"http://www.w3.org/2001/XMLSchema#int"
>8</hasID>
</SubInterestArea>
</belongsToSubArea>
<hasName rdf:datatype=
"http://www.w3.org/2001/XMLSchema#string"
>LaboratÃ³rio de Fisiologia Computacional</hasName>
<hasAbstract rdf:datatype=
"http://www.w3.org/2001/XMLSchema#string"
>FISIOCOMP</hasAbstract>
<hasTitle rdf:datatype=
"http://www.w3.org/2001/XMLSchema#string"
>LaboratÃ³rio de Fisiologia Computacional</hasTitle>
<financedBy rdf:resource="#CNPQ"/>
<belongsToSubArea>
<SubInterestArea rdf:ID="MatematicaSimbolica">
<hasID rdf:datatype=
"http://www.w3.org/2001/XMLSchema#int"
>1</hasID>
<belongsToAreaInteresse rdf:resource="#MatematicadaComputacao"/>
<hasName rdf:datatype=
"http://www.w3.org/2001/XMLSchema#string"
>MatemÃ¡tica SimbÃ³lica</hasName>
</SubInterestArea>
</belongsToSubArea>
<belongsToDepartament rdf:resource="#DCC"/>
<hasGuide>
<Scientist rdf:ID="RodrigoWeber">
<guideResearhProject rdf:resource="#FISIOCOMP"/>
<hasID rdf:datatype=
"http://www.w3.org/2001/XMLSchema#int"
>6</hasID>
<hasRole rdf:resource="#Coordinator"/>
<hasTitleDegree rdf:resource="#DoctorDegree"/>
<belongsToSubArea rdf:resource="#ArquiteturaSistemasComputacao"/>
<belongsToDepartament rdf:resource="#DCC"/>
<belongsToSubArea rdf:resource="#ModelosAnaliticosSimulacao"/>
```

```
970 <belongsToSubArea rdf:resource="#MatematicaSimbolica"/>
971 <hasName rdf:datatype=
972 "http://www.w3.org/2001/XMLSchema#string"
973 >Rodrigo Weber</hasName>
974 <hasPublishedScientificArtifact>
975 <Paper rdf:ID="Agos">
976 <hasTitle rdf:datatype=
977 "http://www.w3.org/2001/XMLSchema#string"
978 >AGOS: Um framework...</hasTitle>
979 <hasAbstract rdf:datatype=
980 "http://www.w3.org/2001/XMLSchema#string"
981 >AGOS: Um framework...</hasAbstract>
982 <scientificArtifactPublishedBy rdf:resource="#RodrigoWeber"/>
983 <belongsToSubArea rdf:resource="#ArquiteturaSistemasComputacao"/>
984 <belongsToSubArea rdf:resource="#ModelosAnaliticosSimulacao"/>
985 <belongsToSubArea rdf:resource="#MatematicaSimbolica"/>
986 <hasID rdf:datatype=
987 "http://www.w3.org/2001/XMLSchema#int"
988 >3</hasID>
989 <hasKeyWords rdf:datatype=
990 "http://www.w3.org/2001/XMLSchema#string"
991 >modelagem eletrofisiologia cardiaca computacao</hasKeyWords>
992 </Paper>
993 </hasPublishedScientificArtifact>
994 </Scientist>
995 </hasGuide>
996 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
997 >3</hasID>
998 <hasKeyWords rdf:datatype=
999 "http://www.w3.org/2001/XMLSchema#string"
1000 >fisiologia computacional ufjf</hasKeyWords>
1001 </ResearchProject>
1002 </hasFinanced>
1003 <hasFinanced rdf:resource="#NPQS"/>
1004 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1005 >Conselho Nacional de Pesquisa e Qualidade</hasName>
1006 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
1007 >1</hasID>
```

```
1008 </Agency>
1009 </financedBy>
1010 <hasKeyWords rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1011 >engenharia software qualidade</hasKeyWords>
1012 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1013 >NÚCLEO DE PESQUISA EM QUALIDADE DE SOFTWARE</hasName>
1014 <hasGuide rdf:resource="#ReginaMaciel"/>
1015 <hasPublishedScientificArtifact rdf:resource="#
        BuscaRecuperacaoArtefatosCientificosWeb"/>
1016 <belongsToDepartament rdf:resource="#DCC"/>
1017 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
1018 >1</hasID>
1019 <hasTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1020 >NPQS</hasTitle>
1021 <hasScientistAssociated rdf:resource="#FernandaCampos"/>
1022 <hasHTTPAdress rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1023 >www.npqs.ufjf.br</hasHTTPAdress>
1024 </ResearchProject>
1025 </hasResearchProjectsAllocated>
1026 <hasScientistAssociated rdf:resource="#ElyMatos"/>
1027 </Profile>
1028 </belongsToDepartament>
1029 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
1030 >4</hasID>
1031 <hasPublishedScientificArtifact rdf:resource="#MathWS"/>
1032 </Scientist>
1033 </hasScientistAssociated>
1034 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1035 >MIOLO Framework</hasName>
1036 <hasHTTPAdress rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1037 >www.miolo.org.br</hasHTTPAdress>
1038 <hasTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1039 >MIOLO</hasTitle>
1040 <hasGuide rdf:resource="#ElyMatos"/>
1041 <hasAbstract rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1042 >Framework para desenvolvimento em PHP</hasAbstract>
1043 </ResearchProject>
1044 </hasResearchProjectsAllocated>
```

```
<hasFile rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>mathws.pdf</hasFile>
<belongsToSubArea>
<SubInterestArea rdf:ID="SistemasInformacao">
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>7</hasID>
<belongsToAreaInteresse rdf:resource="#MetodologiasTecnicasComputacao"
    />
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Sistemas de InformaÃ§Ã£o</hasName>
</SubInterestArea>
</belongsToSubArea>
<belongsToSubArea rdf:resource="#EngenhariadeSoftware"/>
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>2</hasID>
<scientificArtifactPublishedBy rdf:resource="#ReginaMaciel"/>
<scientificArtifactPublishedBy rdf:resource="#ElyMatos"/>
<belongsToSubArea rdf:resource="#BancodeDados"/>
<hasKeyWords rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>math web services semantica ontologia</hasKeyWords>
<scientificArtifactPublishedBy rdf:resource="#FernandaCampos"/>
</Paper>
</hasPublishedScientificArtifact>
<belongsToDepartament rdf:resource="#profileJoao"/>
<belongsToDepartament rdf:resource="#NPQS"/>
<belongsToDepartament rdf:resource="#DCC"/>
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>3</hasID>
<hasTitleDegree rdf:resource="#DoctorDegree"/>
</Scientist>
</scientificArtifactPublishedBy>
<hasResearchProjectsAllocated rdf:resource="#NPQS"/>
<publishedIn rdf:resource="#CILAMCE2008"/>
<hasKeyWords rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>web agentes ontologia e-science</hasKeyWords>
<belongsToSubArea rdf:resource="#EngenhariadeSoftware"/>
<belongsToSubArea rdf:resource="#SistemasInformacao"/>
<scientificArtifactPublishedBy rdf:resource="#NPQS"/>
```

```
<scientificArtifactPublishedBy rdf:resource="#ReginaMaciel"/>
<hasAbstract rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Busca e RecuperaÃ§Ã£o de Artefatos CientÃ?ficos na Web com o Uso de
    Agentes Inteligentes</hasAbstract>
<belongsToSubArea rdf:resource="#BancodeDados"/>
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>1</hasID>
<scientificArtifactPublishedBy rdf:resource="#LuizFelipe"/>
</Paper>
</hasPublishedScientificArtifact>
<belongsToDepartament rdf:resource="#DCC"/>
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>1</hasID>
<hasPublishedScientificArtifact rdf:resource="#CILAMCE2008"/>
<hasRole rdf:resource="#Collaborator"/>
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Luiz Felipe</hasName>
<belongsToDepartament rdf:resource="#profileJoao"/>
<belongsToDepartament rdf:resource="#NPQS"/>
<hasTitleDegree rdf:resource="#Graduated"/>
</Scientist>
<InterestArea rdf:ID="Biologia">
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Biologia</hasName>
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>5</hasID>
</InterestArea>
<ExternalConcept rdf:ID="Mapping">
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Mapping</hasName>
</ExternalConcept>
<WebService rdf:ID="OLS">
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>1</hasID>
<scientificArtifactPublishedBy>
<SubInterestArea rdf:ID="BioInformatica">
<hasScientistAssociated>
<Scientist rdf:ID="Maria">
```

```
1119 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1120 >Maria</hasName>
1121 <hasProfile>
1122 <Profile rdf:ID="profileMaria">
1123 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
1124 >2</hasID>
1125 <belongsToSubArea rdf:resource="#BioInformatica"/>
1126 <belongsToScientist rdf:resource="#Maria"/>
1127 <hasRelatedWith rdf:resource="#Mapping"/>
1128 <hasRelatedWith>
1129 <ExternalConcept rdf:ID="Proteins">
1130 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1131 >Proteins</hasName>
1132 </ExternalConcept>
1133 </hasRelatedWith>
1134 <hasRelatedWith>
1135 <ExternalConcept rdf:ID="Identifiers">
1136 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1137 >Identifiers</hasName>
1138 </ExternalConcept>
1139 </hasRelatedWith>
1140 <hasRelatedWith>
1141 <ExternalConcept rdf:ID="CrossReference">
1142 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1143 >CrossReference</hasName>
1144 </ExternalConcept>
1145 </hasRelatedWith>
1146 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1147 >Profile da Maria</hasName>
1148 </Profile>
1149 </hasProfile>
1150 <hasTitleDegree rdf:resource="#MasterDegree"/>
1151 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
1152 >8</hasID>
1153 <hasRole rdf:resource="#Student"/>
1154 <belongsToSubArea rdf:resource="#BioInformatica"/>
1155 </Scientist>
1156 </hasScientistAssociated>
```

```
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>45</hasID>
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>BioInformÃ¡tica</hasName>
<hasProfileAssociated rdf:resource="#profileMaria"/>
<hasPublishedScientificArtifact rdf:resource="#OLS"/>
<belongsToAreaInteresse rdf:resource="#Biologia"/>
</SubInterestArea>
</scientificArtifactPublishedBy>
<hasTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Ontology Lookup Service</hasTitle>
<hasAbstract rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Ontology Lookup Service</hasAbstract>
<belongsToSubArea rdf:resource="#BioInformatica"/>
</WebService>
<SubInterestArea rdf:ID="ProcessamentoGrafico">
<belongsToAreaInteresse rdf:resource="#MetodologiasTecnicasComputacao"
    />
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Processamento GrÃ¡fico</hasName>
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>6</hasID>
</SubInterestArea>
<SubInterestArea rdf:ID="TeleInformatica">
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>11</hasID>
<belongsToAreaInteresse rdf:resource="#SistemasdeComputacao"/>
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>TeleInformatica</hasName>
</SubInterestArea>
<SubInterestArea rdf:ID="LinguagensFormaisAutomatos">
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Linguagens Formais e AutÃ´matos</hasName>
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>14</hasID>
<belongsToAreaInteresse rdf:resource="#TeoriadaComputacao"/>
</SubInterestArea>
<SubInterestArea rdf:ID="ComputabilidadeModelosComputacao">
```

```
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>13</hasID>
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Computabilidade e Modelos de ComputaÃ§Ã£o</hasName>
<belongsToAreaInteresse rdf:resource="#TeoriadaComputacao"/>
</SubInterestArea>
<SubInterestArea rdf:ID="LinguagensProgramacao">
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Linguagens de ProgramaÃ§Ã£o</hasName>
<belongsToAreaInteresse rdf:resource="#MetodologiasTecnicasComputacao"
    />
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>5</hasID>
</SubInterestArea>
<rdf:Description rdf:about="http://www.example.org/owls/isObsolete.owl#
    isObsoleteProfile">
<profile:contactInformation rdf:resource="#ReginaMaciel"/>
<profile:contactInformation rdf:resource="#FernandaCampos"/>
<profile:contactInformation rdf:resource="#LuizFelipe"/>
<profile:has_process rdf:resource="http://www.example.org/owls/
    isObsolete.owl#isObsoleteProcess"/>
</rdf:Description>
<rdf:Description rdf:about="http://www.example.org/owls/getTermsByName.
    owl#getTermsByNameService">
<service:providedBy>
<ResearchProject rdf:ID="SBO">
<service:provides rdf:resource="http://www.example.org/owls/
    getTermsByName.owl#getTermsByNameService"/>
<hasEndDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date"
>2009-07-29</hasEndDate>
</ResearchProject>
</service:providedBy>
</rdf:Description>
<SubInterestArea rdf:ID="Hardware">
<hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>9</hasID>
<belongsToAreaInteresse rdf:resource="#SistemasdeComputacao"/>
<hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
1227 >Hardware</hasName>
1228 </SubInterestArea>
1229 <process:Local rdf:ID="Local_2"/>
1230 <SubInterestArea rdf:ID="LogicaSemanticaProgramas">
1231 <belongsToAreaInteresse rdf:resource="#TeoriadaComputacao"/>
1232 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
1233 >15</hasID>
1234 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1235 >LogicaSemanticaProgramas</hasName>
1236 </SubInterestArea>
1237 <Agency rdf:ID="Agency_16"/>
1238 <grounding:WsdlAtomicProcessGrounding rdf:ID="
        WsdlAtomicProcessGrounding_3"/>
1239 <SubInterestArea rdf:ID="SoftwareBasico">
1240 <belongsToAreaInteresse rdf:resource="#SistemasdeComputacao"/>
1241 <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1242 >Software BÃ¡sico</hasName>
1243 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
1244 >10</hasID>
1245 </SubInterestArea>
1246 <WebService rdf:ID="PICR">
1247 <hasRelatedWith rdf:resource="#Mapping"/>
1248 <hasRelatedWith rdf:resource="#Proteins"/>
1249 <hasID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
1250 >2</hasID>
1251 <hasTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1252 >Protein Identifier Cross-Reference Service</hasTitle>
1253 <hasAbstract rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1254 >Protein Identifier Cross-Reference Service</hasAbstract>
1255 </WebService>
1256 <rdf:Description rdf:about="http://www.example.org/owls/getTermsByName.
        owl#getTermsByNameProfile">
1257 <profile:contactInformation rdf:resource="#LuizFelipe"/>
1258 <profile:contactInformation rdf:resource="#FernandaCampos"/>
1259 <profile:contactInformation rdf:resource="#ReginaMaciel"/>
1260 <profile:has_process rdf:resource="http://www.example.org/owls/
        getTermsByName.owl#getTermsByNameProcess"/>
1261 </rdf:Description>
```

```
1262 </rdf:RDF>
1263
1264 <!-- Created with Protege (with OWL Plugin 3.2.1, Build 365)  http://
         protege.stanford.edu -->
```

Referências Bibliográficas

AVILA-ROSAS, A., MOREAU, L., DIALANI, V., MILES, S. e LIU, X. 2002, Agents for the grid: A comparison with web services (part ii: Service discovery), In: *In Proceedings of Workshop on Challenges in Open Agent Systems (AAMAS02*, pp. 52–56.

BASTOS, R. M. e RIBEIRO, M. B. 2005, Masup: An agent-oriented modeling process for information systems, In: R. Choren, A. Garcia, C. Lucena e A. Romanovsky, eds., *Software Engineering for Multi-agent Systems III*, vol. 3390, pp. 19–35, Berlim, Springer-Verlag.

BELLIFEMINE, F. 2003, *JADE A White Paper*, Rel. Téc., Telecom Italia.

BERNERS-LEE, T., HENDLER, J. e LASSILA, O. 2001, "The semantic web", *Scientific American*, Publicação Eletrônica: http://www.scientificamerican.com/article.cfm?id=the-semantic-web.

BERSTIS, V. 2002, *Fundamentals of Grid Computing*, Rel. Téc., IBM Corporation International Technical Support Organization, Austin, Texas, Redbooks Paper.

BEVILACQUA, F. e SARDINHA, J. A. R. P. 2001, *Estruturas dinâmicas de incentivos para grupos de consumo*, cap. 5, pp. 145–159, Papel Virtual Editora.

BLOIS, M. e LUCENA, C. 2004, Multi-agent systems and the semantic web: The semanticcore agent-based abstraction layer, In: INSTICC, ed., *Proceedings of Sixth International Conference on Enterprise Information Systems ICEIS*, vol. 4, pp. 263–170.

BRESCIANI, P. e *et al.* 2004, "Tropos: An agent-oriented software development methodology", *Autonomous Agents and Multi-Agent Systems*, v. 8, n. 3, pp. 203–236.

BURKE, S., CAMPANA, S., LANCIOTTI, E., LORENZO, P. M., MICCIO, V., NATER, C., SANTINELLI, R. e SCIABÀ, A., 2009. glite 3.1 user guide - manual series.

BUSCHMANN, F., MEUNIER, R., ROHNERT, H., SOMMERLAD, P. e STAL, M. 1996, *Pattern-Oriented Software Architecture Volume 1: A System of Patterns (Hardcover)*, 1º edic., Wiley, ISBN 0471958697.

CHRIS, C. G., WROE, C. e STEVENS, R. 2003, Grid project: Services, architecture and demonstrator, In: S. J. Cox, ed., *Proceedings of UK e-Science All Hands Meeting*, pp. 595–601, Nottingham, UK, UK e-Science Core Programme, EPSRC.

COTE, R., JONES, P., APWEILER, R. e HERMJAKOB, H. 2006, "The ontology lookup service, a lightweight cross-platform tool for controlled vocabulary queries", *BMC Bioinformatics*, v. 7, n. 1, pp. 97.

COTE, R., JONES, P., MARTENS, L., KERRIEN, S., REISINGER, F., LIN, Q., LEINONEN, R., APWEILER, R. e HERMJAKOB, H. 2007, "The protein identifier cross-referencing (picr) service: reconciling protein identifiers across multiple source databases", *BMC Bioinformatics*, v. 8, n. 1, pp. 401.

DAM, K. H. e WINIKOFF, M. 2003, Comparing agent-oriented methodologies, In: *Proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems held in Melbourne*, pp. 78–93.

DE ROURE, D., JENNINGS, N. R. e SHADBOLT, N. R. 2002, *Research Agenda for the Semantic Grid: A Future e-Science Infrastructure*, Rel. Téc. 1.9, National e-Science Centre, Report for EPSRC/DTI e-Science Core Programme.

ELAMY, A. H. 2005, "Perspectives in agent-based technology", *AgentLink News*, n. 18, pp. 19–22.

ELENIUS, D., DENKER, G., MARTIN, D., GILHAM, F., KHOURI, J., SADAATI, S. e SENANAYAKE, R. 2005, The owl-s editor - a development tool for semantic web services, In: *ESWC*, pp. 78–92.

ESCOBAR, E. e LEMKE, A. P e BLOIS, M. R. 2006, Semanticore 2006: Permitindo o desenvolvimento de aplicações baseadas em agentes na web semântica, In:

Second Workshop on Software Engineering for Agent-oriented Systems (SEAS), Florianópolis, SC, Brasil.

EVANS, R., KEARNEY, P., CAIRE, G., GARIJO, F. J., GOMES-SANZ, J. J., PAVON, J., LEAL, F., CHAINHO, P. e MASSONET, P. 2001, *MESSAGE: Methodology for Engineering Systems of Software Agentes*, Rel. Téc., European Institute for Research and Strategic Studies in Telecommunications.

FACT, Fact++, Web, URL: http://owl.man.ac.uk/factplusplus/, Acesso em 07 de Agosto de 2009.

FININ, T., WEBER, J., WIEDERHOLD, G., GENESERETH, M., FRITZSON, R., MCKAY, D., SHAPIRO, S., MCGUIRE, J., PELAVIN, R. e BECK, C., Specification of the kqml agent-communication language.

FIPA, Foundation for intelligent physical agents (fipa): Agent management specification, Web, URL: http://www.fipa.org/, Acesso em 07 de Agosto de 2009.

FOSTER, I. 2002, "What is the grid? a three point checklist", *GRIDToday*, v. 1, n. 9, .

FOSTER, I. 2005, Globus toolkit version 4: Software for service-oriented systems, In: *IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779*, pp. 2–13.

FOSTER, I., CZAJKOWSKI, K., FERGUSON, D. E., FREY, J., GRAHAM, S., MAGUIRE, T., SNELLING, D. e TUECKE, S. 2005, "Modeling and managing state in distributed systems: the role of ogsi and wsrf", *Proceedings of the IEEE*, v. 93, n. 3, pp. 604–612.

FOX, G. e WALKER, D. 2003, *e-Science Gap Analysis*, Rel. Téc., Indiana University e Cardiff University, USA e UK.

GAMS, Guide to available mathematical software, Web, URL: http://gams.nist.gov/, Último acesso em: 07 de Agosto de 2009.

GARCÍA, F. S. 2007, *Sistema Basado en Tecnologías del Conocimento para Entornos de Servicios Web Semánticos*, Tese de Doutorado, Departamento de Ingeniería de la Información y las Comunicaciones, Universidad de Murcia, Spain.

GARCÍA-SANCHÉZ, F., VALENCIA-GARCÍA, R., MARTÍNEZ-BÉJAR, R. e FERNÁNDEZ-BREIS, J. T. 2009, "An ontology, intelligent agent-based framework for the provision of semantic web services", *Expert Systems With Applications*, v. 36, n. 2, pp. 3167–3187.

GATTI, M. A., CARVALHO, G., PAES, R., VON STAA, A., LUCENA, C. e BRIOT, J. 2006, O rationale da fidedignidade em sistemas multiagentes abertos governados por leis, In: *Second Workshop on Software Engineering for Agent-oriented Systems (SEAS)*, Florianópolis, SC, Brasil.

GOSLING, J. 2005, *The Java Language Specification*, 3º edic., Addison-Wesley.

GROUP, T. O. G. S. W. 2007, *Service-Oriented Architecture (SOA)*, Rel. Téc. W074, The Open Group, San Francisco, CA, USA, White Paper.

GRUBER, T. R. 1993, A translation approach to portable ontology specifications, In: *Proceedings of Knowledge Acquisition*, pp. 199–220.

GUARINO, N. e GIARETTA, P. 1995, Ontologies and knowledge bases: Towards a terminological clarification, In: *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pp. 25–32, Amsterdam, NL, IOS Press.

HARDING, C. 2007, *SOA Reference Architecture Project Description*, Rel. Téc. 1.0, The Open Group, San Francisco, CA, USA.

HEINE, F., HOVESTADT, M. e KAO, O. 2004, Towards ontology-driven p2p grid resource discovery, In: *Fifth IEEE/ACM International Workshop on Grid Computing*, pp. 76–83.

HEY, T. e TREFETHEN, A. E. 2005, "Cyberinfrastructure for e-science", *Science*, v. 308, n. 5723, pp. 817–821.

IGLESIAS, C. . 1998, *Definición de una Metodología para el Desarrollo de Sistemas Multiagente*, Tese de Doutorado, Departamento de Engenharia de Sistemas de Telecomunicação, Universidade Politécnica de Madri.

IGLESIAS, C. A. e GARIJO, M. 1999, Uer technique: Conceptualisation for agent-oriented development, In: *5th International Conference on Information Systems Analysis and Synthesis (ISAS'99)*, vol. 5, pp. 535–540.

IGLESIAS, C. A. e GARIJO, M. 2005, *The Agent Oriented Methodology MASCommonKADS*, cap. 3, pp. 46–78, Agented-Oriented Methodologies, IDEA Group Publishing.

J3, Fortran standards technical committee, URL: http://www.j3-fortran.org, Acesso em 07 de Agosto de 2009.

JACKSON, T., AUSTIN, J., FLETCHER, M., JESSOP, M., PASLEY, A. B. L., ONG, M., REN, X., ALLAN, G., KADIRKAMANATHAN, V., THOMPSON, H. e FLEMING, P. J. 2005, Distributed health monitoring for aero-engines on the grid: Dame, In: *Proceedings of IEEE Aerospace Conference*, pp. 3738–3747, Big Sky, MT.

JENA, Jena - a semantic web framework for java, Web, URL: http://jena.sourceforge.net/index.html, Acesso em 07 de Agosto de 2009.

JONES, P. e COTE, R. 2008, The pride proteomics identifications database: Data submission, query, and dataset comparison, In: *Functional Proteomics*, vol. 484 de *Methods in Molecular Biology*, pp. 287–303, Humana Press.

KERNIGHAN, B. W. e RITCHIE, D. M. 1988, *C Programming Language*, Prentice-Hall, ISBN 0-13-110362-8.

KLUSCH, M., FRIES, B. e SYCARA, K. 2006, Automated semantic web service discovery with owls-mx, In: *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 915–922, New York, NY, USA, ACM, ISBN 1-59593-303-4.

KOPECKY, J., VITVAR, T., BOURNEZ, C. e FARRELL, J. 2007, Sawsdl: Semantic annotations for wsdl and xml schema, In: *Internet Computing, IEEE*, vol. 11, pp. 60–67.

KOURTESIS, D. e PARASKAKIS, I. 2008, Combining sawsdl, owl-dl and uddi for semantically enhanced web service discovery., In: *ESWC*, vol. 5021 de *Lecture Notes in Computer Science*, pp. 614–628, Springer, ISBN 978-3-540-68233-2.

LANGEGGER, A., Wöß, W. e Blöchl, M. 2008, A semantic web middleware for virtual data integration on the web, pp. 493–507.

LOUIS, V. e MARTINEZ, T. 2005, "The jade semantic agent: Towards agente communication oriented middleware", *AgentLink News*, n. 18, pp. 16–18.

LU, J., MA, L., ZHANG, L., BRUNNER, J., WANG, C., PAN, Y. e YU, Y. 2007, Sor: a practical system for ontology storage, reasoning and search, In: *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pp. 1402–1405, VLDB Endowment, ISBN 978-1-59593-649-3.

LUCENA, P. 2003, *SemanticAgent, uma plataforma para desenvolvimento de agentes inteligentes*, Dissert. de Mestrado, Instituto de Ciências Matemáticas e de Computação, USP.

LYYTINEN, K. S. e ROSSI, M. 1996, Metaedit+: A fully configurable multi-user and multi-tool case and came environment, In: *CAiSE'96: Proceedings of the 8th International Conference on Advances Information System Engineering*, pp. 1–21, London, UK, Springer-Verlag, ISBN 3-540-61292-0.

MARINHO, A., MURTA, L., WERNER, C. M. L., BRAGANHOLO, V. P., CRUZ, S. M. S. e MATTOSO, M. 2009, A strategy for provenance gathering in distributed scientific workflows, In: *IEEE 2009 Third International Workshop on Scientific Workflows (SWF). 7th IEEE International Conference on Web Services (ICWS)*, Los Angeles, USA.

MAS, A. 2005, *Agentes Software y Sistemas Multi-Agente: Conceptos, Arquitecturas y Aplicaciones*, Pearson Education, ISBN 8420543675.

MATOS, E. e *et al.* 2007, Mathws: Broker de serviços web para e-science, In: *1st Brazilian e-Science WorkShop. in conjunction with 22nd Brazilian Symposium on Data Base*, João Pessoa, Brazil.

MATOS, E. E., MENDES, L. F., CAMPOS, F. e BRAGA, R. 2009, Asow-science:a service oriented framework to support e-science applications, In: *Proceedings of 10th IEEE International Conference on Information Reuse and Integration*, Las Vegas, USA.

M.GASPARI e MOTTA, E. 1994, Symbol-level requirements for agent-level programming, In: *ECAI94 the 11th European Conference on Artificial Intelligence*, pp. 364–368, John Wiley and SonsLtd.

MILIDIU, R. L., LUCENA, C. J. P. e SARDINHA, J. A. R. P. 2001, An object-oriented framework for creating offerings, In: *Proceedings of the International Conference on Internet Computing (IC'2001)*, vol. 1, pp. 119–123, CSREA Press.

MONET, Monet consortium, Web, URL: http://monet.nag.co.uk, Acesso em 07 de Agosto de 2009.

MOREAU, L., MILES, S., GOBLE, C., GREENWOOD, M., DIALANI, V., ADDIS, M., ALPDEMIR, N., CAWLEY, R., ROURE, D. D., FERRIS, J., GAIZAUSKAS, R., GLOVER, K., GREENHALGH, C., LI, P., LIU, X., LORD, P., LUCK, M., MARVIN, D., OINN, T., PATON, N., PETTIFER, S., RADENKOVIC, V., ROBERTS, A., ROBINSON, A., RODDEN, T., SENGER, M., SHARMAN, N., STEVENS, R., WARBOYS, B., WIPAT, A. e WROE, C. 2003, On the use of agents in a bioinformatics grid, In: *Proc. of the 3rd IEEE/ACM CCGRID'2003 Workshop on Agent Based Cluster and Grid Computing*, pp. 653–661.

NSF 2006, *NSF's Cyberinfrastructure Vision for 21st Century Discovery*, Rel. Téc. 7.1, National Science Foundation, National Science Foundation Report.

NWANA, H. S. 1996, "Software agents: An overview", *Knowledge Engineering Review*, v. 11, n. 3, pp. 1–40.

ODELL, J., P., H. V. D. e BAUER, B. 2000, Extending uml for agents, In: *Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence*, pp. 3–17, Austin, TX, USA.

OINN, T., ADDIS, M., FERRIS, J., MARVIN, D., CARVER, T., POCOCK, M. R. e WIPAT, A. 2004, "Taverna: A tool for the composition and enactment of bioinformatics workflows", *Bioinformatics*, v. 20, pp. 3045–3054.

OWL, Ontology web language, Web, http://www.w3.org/TR/owl-features/. Acesso em 07 de Agosto de 2009.

OWL-S, Owl-s: Semantic markup for web services, Web, URL: http://www.w3.org/Submission/OWL-S/, Acesso em 07 de Agosto de 2009.

OWLAPI, The owl api, Web, URL: http://owlapi.sourceforge.net, Acesso em 07 de Agosto de 2009.

PADGHAM, L. e WINIKOFF, M. 2003, Prometheus: A methodology for developing intelligent agents, In: *Agent-Oriented Software Engineering III*, vol. 2585 de *Lecture Notes in Computer Science*, pp. 174–185, Springer Berlin / Heidelberg.

PAVÓN, J., GÓMEZ-SANZ, J. J. e FUENTES, R. 2005, *The INGENIAS Methodology and Tools*, cap. 9, pp. 239–276, Agented-Oriented Methodologies, IDEA Group Publishing.

PEARCE, S. 2007, *Grids and e-Science*, Post Note 286, PARLAMENTARY OFFICE OF SCIENCE AND TECHNOLOGY, UK.

PELLET, Pellet: The open source owl dl reasoner, Web, URL: http://pellet.owldl.com, Acesso em 07 de Agosto de 2009.

PROTEGE, Protege owl - ontology editor for semantic web, Web, URL: http://protege.stanford.edu/plugins/owl/, Acesso em 07 de Agosto de 2009.

RACER, Racer manager, Web, URL: http://www.racer-systems.com /index.phtml, Acesso em 07 de Agosto de 2009.

RAO, A., RAO, S. e GEORGEFF, M. P. 1995, Bdi agents: From theory to practice, In: *In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95*, pp. 312–319.

RCUK 2007, *Highlights from the UK e-science Programme*, Rel. Téc., RESEARCH COUNCILS UK, UK.

RIBEIRO, M. B. e SILVA, M. E., Agentes e ambientes de programação para web: Uma visão da Área, Web.

RIBEIRO, P. C. 2001, *Modelagem e Implementação OO de Sistemas Multi-Agentes*, Dissert. de Mestrado, Departamento de Informática, PUC-Rio.

RODRIGUES, J. G. C. L. 2005, *Matching Semântico de Recursos Computacionais em Ambientes de Grade com Múltiplas Ontologias*, Dissert. de Mestrado, Programa de Pós-Graduação em Informática, Departamento de Ciência da Computação, Instituto de Ciências Exatas, Universidade de Brasília.

ROURE, D. D., JENNINGS, N. R. e SHADBOLT, N., The semantic grid: A future e-science infrastructure, URL: http://eprints.ecs.soton.ac.uk/6869/.

RUMBAUGH, J., JACOBSON, I. e BOOCH, G. 1999, *The Unified Modelling Language Reference Manual*, Addison-Wesley.

RUSSEL, S. e NORVIG, P. 1995, *Artificial Intelligence: A Modern Approach*, cap. 2, pp. 31–52, Prentice-Hall Inc.

SANTOS, D. R. 2006, A comparative study of multi-agent systems development methodologies, In: *Second Workshop on Software Engineering for Agent-oriented Systems (SEAS)*, Florianópolis, SC, Brasil.

SANZ, J. J. G. 2002, *Modelado de Sistemas Multi-agente*, Tese de Doutorado, Departamento de Sistemas Informáticos Y Programación, Facultad de Informática, Universidad Complutense de Madrid, Espanha.

SARDINHA, J., RIBEIRO, P., LUCENA, C. e MILIDIÚ, R. 2003, "An object-oriented framework for building software agents", *Journal of Object Technology*, v. 2, n. 1, pp. 85–97.

SARDINHA, J. A. R. P. 2001, *VGroups: Um Framework para Grupos Virtuais de Consumo*, Dissert. de Mestrado, Departamento de Informática, PUC-Rio.

SARDINHA, J. A. R. P. 2005, *MAS-School e ASYNC: Um Método e um Framework para Construção de Agentes Inteligentes*, Tese de Doutorado, Departamento de Informática, PUC-Rio.

SBC 2006, *Grandes Desafios de Pesquisa em Computação no Brasil: 2006 a 2016*, Rel. Téc., Sociedade Brasileira de Computação, Brasil, Relatório Final.

SCHREIBER, G. e et al. 1999, *Knowledge engineering and management: the CommonKADS methodology*, The MIT Press.

SCICLUNA, J., ABELA, C. e MONTEBELLO, M. 2004, Visual modelling of OWL-S services, In: *IADIS International Conference WWW/Internet*.

SHARON, T. e *et al.* 2002, Searching the web with a little help from your friends, In: *ACM Conference on Computer-Supported Cooperative Work*, New Orleans, USA.

SILVA, A., OLIVEIRA, R., BITTENCOURT, I., SILVA, M., NETO, J., COSTA, E. e ALMEIDA, H. 2006, Um ambiente baseado em agentes para suporte a negociações eletrônicas automatizadas utilizando ontologias e regras de produção, In:

Second Workshop on Software Engineering for Agent-oriented Systems (SEAS), Florianópolis, SC, Brasil.

STEVENS, R., GLOVER, K., GREENHALGH, C., JENNINGS, C., PEARCE, S., LI, P., RADENKOVIC, M., WIPAT, A., TOWER, C., RU, N., BB, N. e B, N., Performing in silico experiments on the grid: A users perspective.

SURE, Y., GOBLE, C., KESSELMAN, C. e KARLSRUHE, U., Semantic grid - convergence of technologies.

TAVARES, J. C. S. 2004, *Um modelo para avaliação de aprendizagem no uso de ferramentas síncronas em ensino mediado pela Web*, Tese de Doutorado, Programa de Pós-Graduação em Informática da PUC-Rio.

THAIN, D., TANNEMBAUM, T. e LIVNY, M. 2005, "Distributed computing in practice: The condor experience", *Concurrency and Computation: Practice and Experience*, v. 17, pp. 2–4.

VAN AART, C. 2005, *Organizational Principles for Multi-Agent Systems*, Whitestein Series in Software Agent Technologies, Birkhauser.

WALTON, C. D. e BARKER, A. 2004, An agent-based e-science experiment builder, In: *Proceedings of the 1st International Workshop on Semantic Intelligent Middleware for the Web and the Grid.*

WELCH, V. 2004, *Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective*, Rel. Téc. 4, Globus Project.

WERNECK, V. M. B. e et al. 2006, Uma avaliação da metodologia mas-commonkads, In: *Second Workshop on Software Engineering for Agent-oriented Systems (SEAS)*, Florianópolis, SC, Brasil.

WINIKOFF, M. e *et al.* 2001, Simplifying the development of intelligent agents, In: *Advances in Artificial Intelligence. 14th Australian Joint Conference on Artificial Intelligence*, pp. 557–568, Adelaide, Australia.

WOOD, M. e DELOACH, S. A. 2000, An overview of the multiagent systems engineering methodology, In: *The First International Workshop on Agent-Oriented software Engineering (AOSE-2000*, pp. 207–222.

WOOLDRIDGE, M. 2009, *An Introduction to MultiAgent Systems - Second Edition*, 2º edic., John Wiley and Sons, ISBN 0470519460.

WOOLDRIDGE, M., JENNINGS, N. R. e KINNY, D. 2000, "The gaia methodology for agent-oriented analysis and design", *Journal of Autonomous Agents and Multi-Agent Systems*, v. 3, pp. 285–312.

YU, E. 1995, *Modelling Strategic Relationships for Business Process Reengineering*, Tese de Doutorado, Dept. of Computer Science, University of Toronto.

ZHUGE, H. 2005, "Semantic grid: Scientific issues, infrastructure, and methodology", *COMMUNICATIONS OF THE ACM*, v. 48, n. 4, pp. 117–119.