UNIVERSIDADE FEDERAL DO MARANHÃO

CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

ÁREA: CIÊNCIA DA COMPUTAÇÃO

UM FRAMEWORK MULTIAGENTE PARA A PERSONALIZAÇÃO DA WEB BASEADO NA MODELAGEM DE USUÁRIOS E NA MINERAÇÃO DE USO

Leandro Balby Marinho

São Luís, MA

UNIVERSIDADE FEDERAL DO MARANHÃO

CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

ÁREA: CIÊNCIA DA COMPUTAÇÃO

UM FRAMEWORK MULTIAGENTE PARA A PERSONALIZAÇÃO DA WEB BASEADO NA MODELAGEM DE USUÁRIOS E NA MINERAÇÃO DE USO

Leandro Balby Marinho

Dissertação apresentada ao curso de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica na área de Ciência da Computação.

Orientadora*: Profa. Dra. Rosario Girardi*

São Luís, MA

UM FRAMEWORK MULTIAGENTE PARA A PERSONALIZAÇÃO DA WEB BASEADO NA MODELAGEM DE USUÁRIOS E NA MINERAÇÃO DE USO

Leandro Balby Marinho

Dissertação aprovada em / /

_____

Profª. Drª. Maria del Rosario Girardi
Universidade Federal do Maranhão
(Orientadora)

_____

Prof. Dr. Francisco José da Silva e Silva
Universidade Federal do Maranhão

_____

Prof. Dr. Guilherme Bittencourt
Universidade Federal de Santa Catarina

Aos meus pais.

"Quem chegou à liberdade da razão, ainda que apenas em certa medida, não pode sentir-se sobre a terra senão como andarilho."

Friedrich Nietzche

AGRADECIMENTOS

Agradeço a todas as pessoas que de alguma forma me ajudaram a fazer dessa jornada uma fonte inestimável de aprendizado e experiências, as quais levarei por toda vida:

A meus pais, Humberto e Terezinha, não só pelo carinho e exemplo de caráter, mas como também pelo incansável apoio e suporte;

A Milena, pelo companheirismo e compreensão, principalmente nas horas ausentes e difíceis;

A meus irmãos, Eduardo e Thales, que sempre foram valiosos companheiros;

A toda a minha família, por sempre acreditar e torcer pelo meu sucesso;

À Professora Rosário, por sempre ter mostrado com inabalável paciência a direção certa para continuar seguindo;

A Alisson, por ter sido um grande amigo e companheiro em todas as horas, cujas discussões e reflexões sempre renderam idéias valiosas a este trabalho;

Aos meus companheiros de pesquisa, em especial, Carla, Ivo, Steferson, Alisson, Ismênia, Antônio Carlos e Erich, pela amizade no decorrer do curso;

A toda Coordenação do Mestrado, funcionários e professores, pelos bons serviços oferecidos e que foram fundamentais para a conclusão do mestrado.

A Capes pelo suporte financeiro ao desenvolvimento da pesquisa;

A meus amigos e amigas por compartilharem comigo da alegria desta conquista.

RESUMO

Com a incessante migração das mais diversas categorias de serviços ao ambiente Web, a necessidade de caracterizar os usuários nesse ambiente nunca foi tão presente. Para isso, são necessários componentes que tenham a habilidade de continuamente perceber o ambiente e rapidamente se adaptarem a ele, refletindo assim as próprias mudanças no comportamento do usuário.

Dentre as abordagens existentes para a modelagem de usuários da Web, a mineração de uso da Web figura entre as mais interessantes, pois através dessa abordagem pode-se modelar o usuário implicitamente através dos dados de uso gerados através da sua interação com a Web.

Este trabalho propõe “ONTOMUW", um framework multiagente para a personalização da Web baseado na modelagem de usuários e na mineração de uso.

O framework é composto por duas camadas onde se distribuem quatro agentes de acordo com suas responsabilidades: agente *Interfaceador*, responsável tanto pela captura das informações provenientes da navegação do usuário quanto pela execução da adaptação; agente *Modelador*, responsável por criar e atualizar tanto modelos de usuários quanto modelos de adaptação; agente *Aquisitor,* responsável pela criação e manutenção de um repositório de dados de uso contendo os modelos de usuários passados; e, finalmente, agente *Minerador*, responsável tanto pela descoberta de grupos de usuários com comportamento de navegação similar quanto pela classificação do usuário corrente nos grupos descobertos. Os agentes *Interfaceador* e *Modelador* compõem a camada de *processamento de informações do usuário*, enquanto que os agentes *Aquisitor* e *Minerador* compõem a *camada de descoberta de padrões*.

A metodologia e ferramenta utilizadas para guiar o processo de análise e projeto de domínio do ONTOMUW foram a MADEM (“*Multi-Agent Domain Engineering Methodology”*) e a ONTOMADEM respectivamente. Para a implementação do framework utilizou-se o ambiente JADE.

Palavras-chave: Modelagem de usuários, Sistemas hipermídia adaptativos, Mineração de uso da Web, Engenharia de domínio multiagente.

ABSTRACT

With the continuous migration of a great diversity of services to the Web, the need for characterizing the users in this environment increases. For that, components that can continually perceive their environment and rapidly adapt to its changes are required, thus reflecting the changes in the user behaviors.

Among the existent approaches for modeling Web users, Web usage mining appears as one of the most interesting. Through Web usage mining a user can be implicitly and automatically modeled through his/her usage data, generated from his/her interaction with the Web.

This work proposes ONTOWUM, a user modeling and usage mining-based multi-agent framework for Web personalization.

The framework comprises two layers, where four agents are distributed according to their responsibilities: *Interface* agent, responsible for both capturing the user browsing information and performing the adaptation effects; *User Modeling* agent, responsible for creating and updating both user models and adaptation models; *Acquirer* agent, responsible for creating and updating an usage data repository containing past user models; and finally, *Miner* agent, responsible for both discovering group of users with similar browsing behavior and classifying the current user in these groups. The *Interface* and *User modeling* agents belong to the *user information-processing layer*, and the *Acquirer* and *Miner* agents belong to the *pattern-discovering layer*.

The methodology and tool used to guide the analyses and design phases were MADEM ("*Multi-Agent Domain Engineering Methodology"*) and ONTOMADEM, respectively. For the implementation of ONTOWUM it was used the JADE framework.

Keywords: User modeling, Hypermedia adaptive systems, Web usage mining, Multi-agent domain engineering.

SUMÁRIO

LISTA DE TABELAS xiii
LISTA DE FIGURAS xiv
LISTA DE ABREVIATURAS xvii
1 INTRODUÇÃO 20
1.1 Contexto de pesquisa 20
1.2 Problemática 21
1.3 Objetivo do trabalho 22
1.4 Relevância do tema 23
1.5 Estrutura da dissertação 23
2 MINERAÇÃO NA WEB: UMA VISÃO GERAL 24
2.1 Mineração na Web 25
2.2 Fases do processo de descoberta de conhecimento na Web 26
2.2.1 Descobrimento de recursos 26
2.2.2 Pré-processamento 27
2.2.3 Generalização 28
2.2.4 Análise 28
2.3 Categorias da mineração na Web 29
2.4 Considerações finais do capítulo 29
3 MINERAÇÃO DE USO DA WEB 31
3.1 Aspectos gerais 31
3.2 O processo da MUW 33
3.2.1 Aquisição de dados 34
3.2.1.1 Dados do lado servidor 34
3.2.1.2 Dados do lado cliente 37
3.2.1.3 Dados intermediários 38
3.2.2 Pré-processamento dos dados 39
3.2.2.1 Filtragem dos dados 40
3.2.2.2 Identificação de usuários 42
3.2.2.3 Identificação das sessões 43

3.2.3 Descoberta de padrões 46
3.2.3.1 Agrupamento 47
3.2.3.2 Classificação 50
3.2.3.3 Descoberta de padrões seqüenciais 55
3.2.3.4 Descoberta de regras de associação 57
3.2.4 Pós-processamento do conhecimento 58
3.3 Modelagem de usuários, personalização da Web e a MUW 60
3.4 Considerações finais do capítulo 65
4 ENGENHARIA DE DOMÍNIO MULTIAGENTE 67
4.1 Aspectos gerais 68
4.2 Agentes, SMAs e a Engenharia de domínio multiagente 69
4.3 MADEM: Uma metodologia para a Engenharia de domínio multiagente 74
4.4 ONTOMUW: Um Framework Multiagente para Modelagem de Usuários baseado na Mineração de Uso 76
4.5 Considerações finais do capítulo 77
5 ANÁLISE DE DOMÍNIO DO ONTOMUW 79
5.1 Construindo o modelo de domínio 80
5.1.1 Modelagem de objetivos 80
5.1.2 Modelagem de papéis 83
5.1.2.1 Modelo do papel Monitor 84
5.1.2.2 Modelo do papel Modelador 85
5.1.2.3 Modelo do papel Aquisitor 86
5.1.2.4 Modelo do papel Minerador 88
5.1.2.5 Modelo do papel Classificador 89
5.1.2.6 Modelo do papel Personalizador 90
5.1.2.7 Modelo do papel Interfaceador 92
5.1.3 Modelagem de interações entre papéis 93
5.1.4 Modelagem de variabilidades 96
5.2 Requisitos não funcionais 97
5.3 Considerações finais do capítulo 99
6 PROJETO DE DOMÍNIO DO ONTOMUW 101

6.1 Modelo de usuários 101
6.1.1 Representação das sessões de navegação 102
6.1.1.1 Terminologia do modelo FM 103
6.1.1.2 Características do modelo FM 106
6.1.1.3 Estrutura de dados 107
6.1.1.4 Modelo de sessão 108
6.1.1.5 Modelo de grupo 110
6.2 Descoberta dos padrões 112
6.2.1 Medidas de similaridade 113
6.2.1.1 Distância euclidiana pura projetada (DEPP) 114
6.2.2 Agrupamento dinâmico X Agrupamento periódico 116
6.3 Captura da interação do usuário 118
6.4 Personalização da interface 120
6.5 Armazenamento e manutenção dos dados de uso 122
6.6 Projeto arquitetural 124
6.6.1 Reuso de padrões arquiteturais 124
6.6.2 Modelagem de agentes 127
6.6.3 Modelagem do framework 128
6.6.4 Modelagem de atividades 131
6.7 Projeto Detalhado 132
6.7.1 Refinamento dos agentes 132
6.7.2 Modelagem de conhecimento 136
6.7.3 Modelagem de comportamento 139
6.7.3.1 Modelo de comportamento do agente *Interfaceador* 139
6.7.3.2 Modelo de comportamento do agente *Modelador* 140
6.7.3.3 Modelo de comportamento do agente *Aquisitor* 142
6.7.3.4 Modelo de comportamento do agente *Minerador* 143
6.7.4 Modelo de atividades detalhado 144
6.7.5 Modelagem de interações dos agentes 146
6.8 Considerações finais do capítulo 148
7 IMPLEMENTAÇÃO DE DOMÍNIO DO ONTOMUW 151

7.1 Padrão FIPA 151
7.2 JADE 152
7.2.1 Características do JADE 153
7.2.2 JADE e FIPA 154
7.2.3 Criando um agente em JADE 155
7.2.4 Comportamento dos agentes JADE 156
7.2.5 Troca de mensagens no JADE 157
7.3 Implementação dos agentes da ONTOMUW 158
7.3.1 Camada de processamento das informações do usuário 158
7.3.2 Camada de descoberta de padrões 167
7.3.3 Exemplos de execução do ONTOMUW 172
7.4 Considerações finais do capítulo 179
8 CONCLUSÃO 181
8.1.1 Resultados e contribuições de pesquisa 182
8.1.2 Trabalhos futuros 184
8.1.3 Extensões futuras do ONTOMUW 186
8.1.4 Considerações finais 188
REFERÊNCIAS BIBLIOGRÁFICAS 190
ANEXO I 207

LISTA DE TABELAS

Tabela 1. Resumo dos algoritmos de agrupamento para a MUW 51
Tabela 2. Resumo dos algoritmos de classificação para a MUW 54
Tabela 3. Resumo dos algoritmos de descoberta de padrões seqüenciais para a MUW . 59
Tabela 4. MUW segundo a modelagem de usuários 63
Tabela 5. Fases, tarefas, atividades e produtos da MADEM 75
Tabela 6. Modelagem de variabilidades 98
Tabela 7. Resumo dos comportamentos do agente Modelador 166
Tabela 8. Resumo dos comportamentos dos agentes da camada de descoberta de padrões do ONTOMUW 172
Tabela 9. Resumo das classes auxiliares do ONTOMUW 210

LISTA DE FIGURAS

Figura 1. Processo de mineração na Web com suas entradas e saídas 27
Figura 2. Processo da MUW 34
Figura 3. MUW para personalização 60
Figura 4. Processos envolvidos no desenvolvimento de sistemas adaptativos [103] 63
Figura 5. As fases da Engenharia de domínio e da Engenharia de aplicações 69
Figura 6. As interações de um agente genérico com seu ambiente [127] 70
Figura 7. A evolução dos paradigmas de desenvolvimento para abordar a complexidade do software 71
Figura 8. Fases e produtos da Engenharia de domínio multiagente 72
Figura 9. Os insumos e os produtos da MADEM 74
Figura 10. Modelo de objetivos 83
Figura 11. Modelo do papel Monitor 85
Figura 12. Modelo do papel Modelador 86
Figura 13. Modelo do papel Aquisitor 87
Figura 14. Modelo do papel Minerador 89
Figura 15. Modelo do papel Classificador 90
Figura 16. Modelo do papel Personalizador 91
Figura 17. Modelo do papel Interfaceador 93
Figura 18. Modelo de interações do objetivo específico Modelagem de usuários 95
Figura 19. Modelo de interações do objetivo específico Modelagem de adaptação 96
Figura 20. Um algoritmo para o agrupamento dinâmico 117
Figura 21. Exemplo de um grafo semântico em RDF 123
Figura 22. Estrutura de camadas da arquitetura do ONTOMUW 125
Figura 23. Diagrama de interações mostrando as interações entre as camadas e os agentes do ONTOMUW 127
Figura 24. Modelo de agentes 129
Figura 25. Modelo do framework ONTOMUW 130
Figura 26. Modelo de atividades dos agentes 131
Figura 27. Descrição detalhada do agente Interfaceador 133

Figura 28. Descrição detalhada do agente Modelador 134
Figura 29. Descrição detalhada do agente Aquisitor 135
Figura 30. Descrição detalhada do agente Minerador 136
Figura 31. Ontologia representando o vocabulário dos agentes 138
Figura 32. Modelo de comportamento do agente Interfaceador 140
Figura 33. Modelo de comportamento do agente Modelador 142
Figura 34. Modelo de comportamento do agente Aquisitor 143
Figura 35. Modelo de comportamento do Minerador 145
Figura 36. Modelo de atividades detalhado 146
Figura 37. Diagrama de seqüência dos agentes do framework 148
Figura 38. Modelo padrão de plataforma de agentes definido pela FIPA [46] 154
Figura 39. Implementação de um agente genérico em JADE 155
Figura 40. Trecho com o código HTML necessário para a utilização do ONTOMUW 159
Figura 41. Método init contendo as incializações do applet 161
Figura 42. Método start do applet 162
Figura 43. Método stop contendo o código que é executado a cada vez que o usuário troca de página 162
Figura 44. Mensagem enviada pelo agente Interfaceador com as informações da visita do usuário corrente 162
Figura 45. Método para a captura da URL da página sendo visitada 163
Figura 46. Script que passa a URL sendo visitada ao applet Java 163
Figura 47. Trecho de código do agente Modelador 166
Figura 48. Código do agente Aquisitor 167
Figura 49. Trecho do arquivo de uso em RDF gerado pelo Aquisitor 170
Figura 50. Protocolo de comunicação FIPA-REQUEST [47] 171
Figura 51. Código do agente Minerador 171
Figura 52. Exemplo de uma página genérica com destaque para o applet sendo carregado 173
Figura 53. Console da máquina virtual Java do navegador indicando a inicialização da plataforma JADE 173

Figura 54. Utilitário do JADE mostrando uma troca de mensagens entre os agentes Interfaceador e Modelador 174
Figura 55. Segunda mensagem recebida pelo agente Modelador 175
Figura 56. Saída do console Java mostrando as matrizes de características construídas para a sessão atual do usuário 176
Figura 57. Dinâmica da sociedade de agentes do ONTOMUW 177
Figura 58. Camada de descoberta de padrões atendendo as requisições da camada de processamento de informações dos usuários 178
Figura 59. Adaptação do navegador mostrando a recomendação ao usuário 178
Figura 60. Ontologia dos agentes construída através do Protege..... 209
Figura 61. Plug-in beangenerator para a construção das ontologias dos agentes JADE 210

LISTA DE ABREVIATURAS

ACC	Canal de Comunicação de Agentes (do inglês *Agent Communication Channel*)
AID	Identificador de Agente (do inglês *Agent Identifier*)
AMS	Agente de Gerenciamento do Sistema (do inglês *Agent Management System*)
ARHP	Regra de Associação Partição de Hipergrafo (do inglês *Association Rule Hypergraph Partition*)
AUML	Linguagem de Modelagem Unificada para Agentes (do inglês *Agent Unified Modeling Language*)
CAPRI	Clementine A-Priori Intervals
CGI	Interface de Porta Comum (do ingês *Common Gateway Interface*)
DDEMAS	Projeto de Domínio para Sistemas Multiagente (do inglês *Domain Design for Multi-Agent Systems*)
DEP	Distância Euclidiana Pura
DEPP	Distância Euclidiana Pura Projetada
DF	Diretório Facilitador (do inglês *Directory Facilitator*)
FIPA	Fundação para Agentes Físicos Inteligentes (do inglês *Foundation for Intelligent Physical Agents*)
FIPA-ACL	Linguagem de Comunicação de Agentes da FIPA
FIPA-REQUEST	Protocolo de Comunicação para interações do tipo Requisição/Resposta da FIPA
FM	Matrizes de Características (do inglês *Feature Matrices*)
GESEC	Grupo de Pesquisa em Engenharia de Software e Engenharia do Conhecimento
GRAMO	Método Baseado em Ontologias para a Análise de Requisitos (do inglês *Generic Requirements Analysis Method based on Ontologies*)
GUID	Identificador Único Global (do inglês *Globally Unique Identifier*)
HTML	Linguagem de Marcação de Hipertexto (do inglês *Hypertext Markup*

	Language)
HTTP	Protocolo de Transferência de Hipertexto (do inglês *Hypertext Transfer Protocol*)
IP	Protocolo da Internet (do inglês *Internet Protocol*)
JADE	Framework para Desenvolvimento de Agentes em Java (do inglês J*ava Agent Development Framework*)
JENA	Framework para Web Semântica em Java (do inglês *Sematic Web Framework for Java*)
KDD	Descoberta de Conhecimento em Bases de Dados (do inglês *Knowledge Discovery in Databases*)
LPGL	Licença Pública Genérica Lesser (do inglês *Lesser General Public License*)
MADEM	Metodologia para a Engenharia de Domínio Multiagente (do inglês *Multi-agent Domain Engineering Methodology*)
MUW	Mineração de Uso da Web
ONTOMADEM	Ontologia genérica que representa o conhecimento da MADEM
ONTOMUW	Framework Multiagente para a Personalização na Web baseado na Modelagem de usuários e na Mineração de uso
RDF	Framework para a Descrição de Recursos (do inglês *Resource Description Framework*)
ROSA	Agentes Remotos para Sites Abertos (do inglês *Remote Open Site Agents*)
SMA	Sistema Multiagente
SQL	Linguagem de Consulta Estruturada (do inglês *Structured Query Language*)
UML	Linguagem de Modelagem Unificada (do inglês *Unified Modeling Language*)
UOL	Universo On-Line
URI	Identificador de Recurso Uniforme (do inglês *Uniform Resource Identifier*)

URL	Local de Recurso Uniforme (do inglês *Uniform Resource Location*)
W3C	Consórcio da World Wide Web (do ingles *World Wide Web Consortium*)
WUM	Minerador de Uso da Web (do inglês *Web Utilization Miner*)

1 INTRODUÇÃO

Entender e modelar o usuário através de suas interações com sistemas computacionais tem ocupado pesquisadores de diversas áreas por várias décadas. A modelagem de usuários permite, dentre outras coisas, que o sistema possa se antecipar em relação às expectativas do usuário, por exemplo, realizando recomendações de produtos baseadas nas suas preferências.

Com a incessante migração das mais variadas categorias de serviços (bancos, lojas, escolas, restaurantes, universidades, imobiliárias, etc.) para a Web, a necessidade de caracterizar os usuários nunca foi tão presente. A Web é por natureza dinâmica e heterogênea, portanto, para modelar o usuário nesse contexto são necessários componentes que sejam aptos a perceber e rapidamente se adaptarem às mudanças do ambiente. Por estarem inseridos em um ambiente, percebendo-o incessantemente através de sensores e por suas características tais como autonomia, capacidade de raciocínio e sociabilidade, o paradigma computacional dos agentes aparece como uma abordagem particularmente adequada a esse problema.

Embora já existam diversas ferramentas, ambientes e metodologias que facilitam o desenvolvimento de aplicações de software baseadas em agentes, há uma notória carência de artefatos reutilizáveis, tais como modelos de domínio e frameworks[1] multiagente. A reutilização desses artefatos traria um ganho não só quantitativo, mas também qualitativo ao desenvolvimento de tais sistemas.

1.1 Contexto de pesquisa

Este trabalho está situado nas áreas de modelagem de usuários, sistemas adaptativos, mineração de uso da Web e engenharia de domínio multiagente. O trabalho está inserido nas atividades de pesquisa do grupo GESEC (Grupo de pesquisa em Engenharia de Software e Engenharia do Conhecimento) [55] o qual busca contribuir à qualidade do software e à produtividade no seu desenvolvimento através da

[1] Um framework é uma solução computacional parcialmente completa, extensível e reutilizável para uma família de aplicações similares

sistematização de técnicas e ferramentas para a Engenharia de Software, integrando os avanços da Engenharia do Conhecimento. O grupo trabalha atualmente nos seguintes tópicos de pesquisa:

- Engenharia de domínio e de aplicações multiagente:
    - Elaboração de técnicas para a construção de abstrações de software de alto nível baseadas em ontologias (padrões, frameworks, modelos de domínio e de usuários) para o desenvolvimento de sistemas multiagente;
    - Elaboração de técnicas para análise e projeto de sistemas multiagente que enfatizam a reutilização de abstrações de software de alto nível;
    - Desenvolvimento de linguagens específicas de domínio baseadas em ontologias e sistemas de padrões para a geração de aplicações multiagente;
- Acesso à informação na Web:
    - Construção de abstrações de software de alto nível baseadas em agentes para o desenvolvimento de aplicações para o acesso à informação na Web (recuperação, mineração, filtragem e recomendação de informação);
    - Elaboração de técnicas para a modelagem implícita de usuários, baseadas na aprendizagem de máquina e sua aplicação no desenvolvimento de aplicações multiagente para recomendação.

1.2 Problemática

Através do comportamento de navegação dos usuários da Web pode-se derivar conhecimento valioso e estratégico. Mantenedores de sites podem aproveitar esse conhecimento para descobrir como os seus sites estão sendo utilizados e a partir daí tomarem decisões que aperfeiçoem os serviços oferecidos; ou então componentes podem utilizar esse conhecimento para a geração de efeitos de personalização automática.

Várias abordagens têm sido propostas com o intuito de capturar e entender o comportamento de navegação dos usuários [13], [14], [15], [20], [23], [28], [29], [30], [78], [110]. Porém, a maioria destes trabalhos está focada na construção de aplicações específicas e, geralmente, não há uma preocupação na produção de artefatos que possam ser reutilizados no futuro por uma família de sistemas[2].

No desenvolvimento de qualquer sistema, seja qual for o paradigma utilizado, é desejável a adoção de uma metodologia que guie o processo de desenvolvimento desde a análise do problema até a implementação.

Na maioria das abordagens que tratam da descoberta de comportamento de navegação dos usuários da Web, independentemente do paradigma utilizado, há uma lacuna concernente a qual metodologia de desenvolvimento é utilizada. Sendo assim, têm-se trabalhos ricos no que diz respeito à solução proposta, mas pobres no sentido da falta de sistematização para a adoção e realização da solução.

1.3 Objetivo do trabalho

O objetivo principal deste trabalho é disponibilizar um framework multiagente para uma família de sistemas que pretenda oferecer serviços personalizados na Web através da mineração de uso. O conhecimento descoberto pela mineração de uso será então utilizado por agentes autônomos com o intuito de personalização automática. O trabalho busca também uma aplicação da metodologia da Engenharia de domínio multiagente (MADEM) desenvolvidos no grupo GESEC [55] para a construção do framework.

Mais especificamente pode-se dividir o objetivo principal nos objetivos específicos descritos a seguir.

- Realizar a análise de domínio para sistemas onde o problema principal esteja ligado à modelagem de usuários e à personalização automática da Web baseado na mineração de uso;
- Realizar o projeto de domínio gerando a especificação do framework;

[2] Uma família de sistemas é um conjunto de sistemas que possuem características em comum [119]

- Realizar a implementação do framework, gerando agentes que possam ser reutilizados por aplicações específicas.

1.4 Relevância do tema

Apesar da enorme quantidade de aplicações que tratam do descobrimento de padrões a partir do comportamento de navegação dos usuários da Web, poucas delas utilizam os benefícios que a tecnologia dos agentes oferece e essas poucas que utilizam são geralmente muito específicas não permitindo uma possível reutilização por outros sistemas similares. Talvez este seja um dos motivos das poucas aplicações multiagente nesse domínio específico.

A disponibilização de abstrações de software de alto nível para o desenvolvimento de sistemas multiagente que pretendam modelar seus usuários a partir de seu comportamento de navegação promoveria uma série de benefícios tais como redução do tempo e custo de desenvolvimento. Portanto, o trabalho aqui proposto, uma vez inserido nesse contexto, tem sua relevância confirmada.

1.5 Estrutura da dissertação

O capítulo 2 apresenta uma visão geral sobre o processo de mineração de dados da Web. O capítulo 3 apresenta o processo específico da mineração de uso da Web de forma detalhada. O capítulo 4 fornece uma visão geral sobre a Engenharia de Domínio Multiagente assim como a metodologia escolhida para análise e projeto do framework. O capítulo 5 detalha a fase de análise de domínio culminando com a construção do modelo de domínio do framework aqui proposto. O capítulo 6 detalha a fase de projeto de domínio culminando com a especificação do framework ONTOMUW. O capítulo 7 detalha a fase de implementação da Engenharia de domínio culminando com agentes funcionais reutilizáveis. O capítulo 9 resume as principais contribuições do trabalho, suas limitações e trabalhos futuros a serem realizados para superá-las. O anexo I contém o código fonte das classes auxiliares e da ontologia utilizada pelos agentes do framework.

2 MINERAÇÃO NA WEB: UMA VISÃO GERAL

Somos testemunhas do enorme aumento de informações, serviços e recursos disponibilizados na Web nos últimos anos. Mais de um bilhão de páginas são indexadas pelos motores de busca todos os dias [117] e achar a informação desejada pode algumas vezes ser uma tarefa penosa. Essa abundância de informações, serviços e recursos instigaram a necessidade do desenvolvimento de técnicas e ferramentas automáticas de descoberta e análise inteligente de informações da Web.

De forma geral, a mineração na Web pode ser conceituada como a descoberta e análise inteligente de informações úteis da Web [31]. Com informações úteis queremos dizer informações que sejam interessantes aos usuários.

A mineração na Web é tradicionalmente dividida em três categorias: mineração de conteúdo, mineração de estrutura e mineração de uso.

Os usuários podem estar interessados, por exemplo, nas informações contidas dentro dos documentos da Web – mineração de conteúdo - nas informações contidas nas relações existentes entre os documentos da Web – mineração de estrutura – ou nas informações contidas nos dados gerados pela utilização ou interação com a Web – mineração de uso.

Para cada categoria são desenvolvidos conjuntos de técnicas distintas, muitas delas herdadas de outras áreas como a Aprendizagem de máquina, Banco de dados, Estatística, Inteligência artificial, Redes sociais e Recuperação e Extração da informação.

A mineração na Web já vem sendo pesquisada há algum tempo, mas tem realmente ganho importância nestes últimos anos. Podemos apontar dois fatores principais que contribuíram para isso: o aumento das transações comerciais na Web e o desenvolvimento da Web semântica [98].

O aumento das transações comerciais na Web motivou, dentre outras coisas, o desenvolvimento de técnicas complexas para o descobrimento de padrões de navegação dos usuários. Através desses padrões, os mantenedores de sites, podem aprender acerca dos perfis dos seus clientes, e através disso, montar melhores estratégias de venda e

marketing. Além disso, componentes automáticos podem utilizar esses padrões de forma a realizar adaptação e personalização de sites e serviços baseados na Web.

A Web em sua concepção foi construída de forma a atender as necessidades de visualização e consumo de seres humanos, onde os textos são quase sempre escritos em linguagem natural, o que dificulta muito o seu processamento pelas máquinas. Isso instigou o desenvolvimento de um novo conceito para a Web, chamada de Web semântica [97]. A Web semântica promete, além de outras coisas, tornar os documentos da Web processáveis, não só pelos humanos, mas também pelas máquinas, através de marcações semânticas agregadas às informações.

A mineração na Web utilizada no contexto da Web semântica promete ser uma relação de cooperação bastante benéfica e interessante para ambas as áreas. A idéia seria melhorar os resultados da mineração na Web através das novas estruturas semânticas adicionais [9].

O objetivo deste capítulo é apresentar uma visão geral sobre a mineração na Web, as fases do processo e as categorias em que se divide. A seção 2.1 apresenta os principais conceitos da mineração na Web e o processo de descoberta de conhecimento na Web. A seção 2.2 mostra as fases do processo. E por último, a seção 2.3 apresenta as categorias em que se divide a mineração na Web e o contexto de aplicação de cada uma.

2.1 Mineração na Web

A Web é uma vasta coleção de documentos heterogêneos (sons, texto, vídeos e imagens). Possui natureza dinâmica e caótica, ou seja, milhões de páginas surgem e desaparecem todos os dias sem aviso prévio. A maioria das ferramentas de busca da Web existentes não consegue lidar de forma efetiva com essa grande quantidade de informações. No contexto desse problema a mineração de dados aparece como uma abordagem natural a ser explorada, pois é uma tecnologia madura e reconhecida quando aplicada a grandes quantidades de dados [103].

A mineração de dados refere-se ao processo não trivial de identificação de padrões válidos, previamente desconhecidos e potencialmente úteis dos dados [52]. Entretanto, utilizar e compreender os dados disponíveis na Web não é uma tarefa

simples, pois esses dados são muito mais sofisticados e dinâmicos do que os sistemas de armazenamento de bancos de dados tradicionais. Enquanto estes últimos utilizam esquemas de armazenamento bem definidos e estruturados, a Web não possui qualquer estrutura ou esquema sobre as informações que armazena. Outro aspecto que diferencia a mineração de dados tradicional da mineração na Web é a existência de vínculos de hipertexto entre os seus documentos. Os vínculos de hipertexto são uma rica fonte de informações a ser explorada, pois dentre outras coisas, ajudam no processo de análise de similaridade de documentos da Web e na identificação de micro-comunidades[3].

Apesar das diferenças e particularidades entre as duas abordagens - mineração em dados tradicionais e mineração de dados da Web -, o processo de descoberta de conhecimento na Web segue os mesmos passos utilizados no processo geral de descoberta de conhecimento em bases de dados (KDD – Knowledge Database Discovery). O processo de mineração na Web é dividido em 4 sub-processos, que na verdade são análogos às fases do processo KDD [39].

Com base nas quatro fases descritas a seguir e na representação da Figura 1**,** a mineração na Web é mais bem definida como: “A utilização de técnicas de mineração de dados para a recuperação automática, extração e avaliação de informações para a descoberta de conhecimento em documentos e serviços da Web” [117]. Aqui avaliação inclui tanto “generalização” quanto ”análise”.

2.2 Fases do processo de descoberta de conhecimento na Web

2.2.1 Descobrimento de recursos

O descobrimento de recursos trata da coleta dos documentos a serem minerados. Como isso se dá vai depender da categoria de mineração a ser aplicada. Se for mineração de conteúdo ou mineração de estrutura, como veremos na seção 2.3, essa etapa irá tratar da automatização do processo de recuperação de documentos relevantes. Caso seja a mineração de uso, os documentos relevantes serão os logs de acesso dos usuários.

[3] Páginas que compartilham o mesmo tema central

Essa fase tem como entrada a necessidade de informação do usuário e como saída os documentos relevantes.

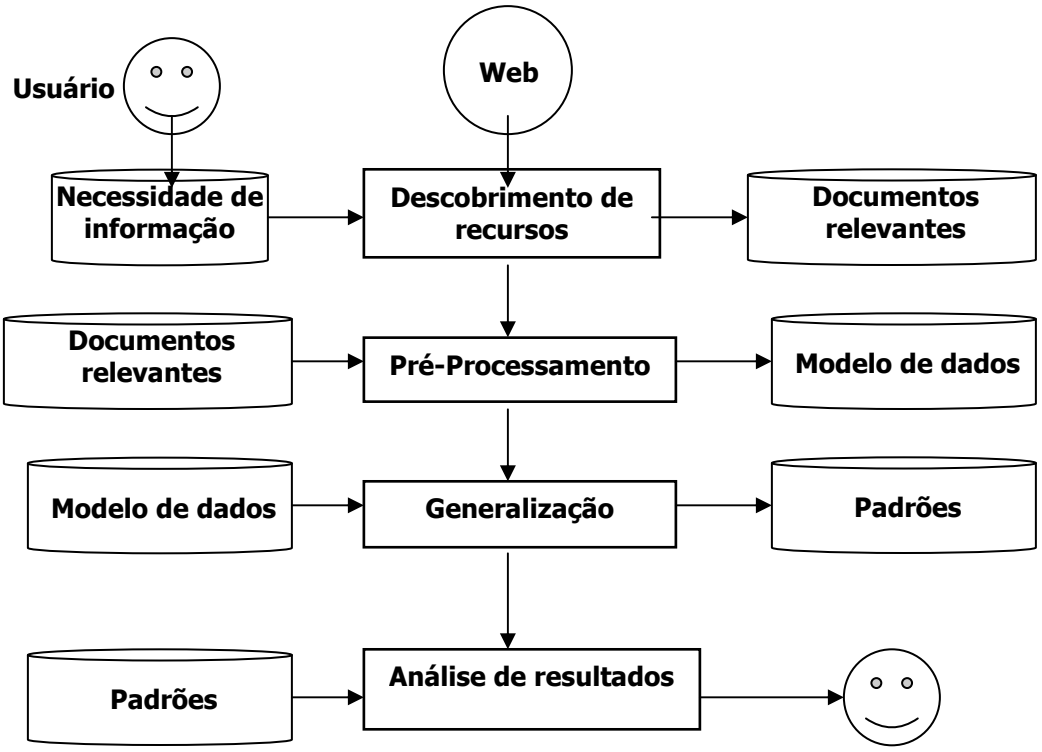

Figura 1. Processo de mineração na Web com suas entradas e saídas

2.2.2 Pré-processamento

Uma vez tendo sido os documentos recuperados, o próximo passo é transformar ou pré-processar esses documentos em um formato adequado para a realização da mineração.

Mais uma vez, as tarefas associadas ao pré-processamento dependem da categoria de mineração. No caso da mineração de conteúdo e estrutura, geralmente são utilizadas técnicas de extração de informações [53], [95], [142].

No caso da mineração de uso, o pré-processamento geralmente envolve a análise e limpeza de logs de acesso. No próximo capítulo daremos um tratamento mais pormenorizado das subtarefas associadas à mineração de uso da Web.

Essa fase tem como entrada os documentos recuperados e como saída um modelo de dados representando esses documentos.

2.2.3 Generalização

Após as informações terem sido extraídas e representadas em um modelo de dados consistente, técnicas de mineração de dados e aprendizagem de máquina são utilizadas para descobrir novo conhecimento a partir dos dados existentes. Um exemplo hipotético que nos daria uma idéia de como seria uma saída de um desses algoritmos é dado abaixo.

1) 70% das pessoas que acessam a seção sobre natação também acessam a seção sobre artes marciais;

2) 80% dos sites que abordam o tema Fórmula-1 possuem vínculos de hipertexto para sites que falam da vida de Ayrton Senna;

No primeiro exemplo, a saída poderia dar uma indicação ao mantenedor de uma loja virtual de materiais esportivos sobre as preferências e perfis de seus clientes, de forma que ele pudesse montar estratégias de vendas mais elaboradas.

No segundo exemplo, a saída descobre uma relação interessante entre os sites, podendo dar novos caminhos de pesquisa aos usuários interessados nesses tópicos.

Nessa fase, geralmente são empregados algoritmos de aprendizagem de máquina para a descoberta do conhecimento, dentre os mais utilizados estão: Agrupamento, Regras de associação, Árvores de decisão e Padrões seqüenciais.

2.2.4 Análise

Uma vez os padrões tendo sido descobertos, os analistas precisam de técnicas e ferramentas apropriadas de modo a entender, visualizar, interpretar e avaliar esses padrões. O sistema Web-Miner [110], por exemplo, propõe uma linguagem de consulta estruturada para a consulta do conhecimento descoberto (na forma de regras de associação e padrões seqüenciais).

2.3 Categorias da mineração na Web

A mineração na Web se divide em três categorias de acordo com a parte da Web a ser minerada: mineração de conteúdo, mineração de estrutura e mineração de uso.

A mineração de conteúdo aborda a mineração dos dados contidos dentro dos documentos da Web. O formato que os dados podem assumir (textos comuns, páginas HTML, imagens, áudio, vídeo, etc.) acaba dirigindo as técnicas a serem utilizadas. Como na grande maioria das vezes o formato de interesse é o textual, e a área de pesquisa conhecida como mineração em texto [106] trata exatamente da descoberta de conhecimento em texto, a mineração de conteúdo acaba sendo, na maioria das vezes, associada à mineração em texto [21], [92].

A mineração de estrutura por outro lado, aborda a mineração das informações contidas entre os documentos da Web através de seus vínculos de hipertexto. Esses vínculos escondem informações valiosas e interessantes não só sobre a topologia da Web, mas também sobre como os documentos se relacionam entre si [22], [93].

A mineração de uso, por sua vez, aborda a mineração das informações de uso da Web, que em outras palavras, são as informações sobre como o usuário utiliza ou interage com a Web. Nessa categoria são tratadas questões como personalização, interfaces adaptativas e aprendizado de perfis de usuários [122].

2.4 Considerações finais do capítulo

Este capítulo apresentou uma visão geral sobre a mineração de dados da Web. Foram apresentadas as diferentes fases do processo de mineração na Web: descobrimento de recursos, pré-processamento, generalização e análise. Também foi brevemente apresentada uma classificação das diferentes modalidades de mineração: mineração de conteúdo, mineração de estrutura e mineração de uso.

Este capítulo contribui na contextualização da mineração de uso da Web, tópico chave para o desenvolvimento do presente trabalho. Como veremos no próximo capítulo, a mineração de uso da Web segue as mesmas fases descritas na seção 2.2,

sendo que, para cada fase, são desenvolvidas conjuntos de técnicas e algoritmos específicos.

É importante notar, que apesar da mineração da Web possuir uma divisão em categorias, não significa que elas sejam mutuamente excludentes. Na verdade as aplicações modernas baseadas na mineração da Web estão, cada vez mais, integrando as três categorias (conteúdo, estrutura e uso) com o intuito de tornar mais ricos e interessantes os padrões descobertos [27], [107].

Devido a sua fundamental importância para o desenvolvimento deste trabalho, o próximo capítulo apresenta uma visão detalhada da mineração de uso da Web.

3 MINERAÇÃO DE USO DA WEB

O incessante fluxo de acessos às páginas da Web, reflete os conteúdos mais diversos, bem como costumes e necessidades pessoais ainda mais heterogêneos, resultando em padrões de uso extremamente ricos e diversificados.

Compreender esses padrões, entender as motivações que impulsionam os usuários quando estão navegando e descobrir quais os modelos subjacentes a esta navegação tem sido a tarefa de uma legião de pesquisadores em disciplinas tão diversas como Estatística, Bancos de dados, Inteligência artificial entre outras.

A Mineração de Uso da Web (MUW) é a área que se dedica à atividade de investigação de clickstreams (i.e. seqüência de visitas a páginas feitas por usuários), visando não só reconstituir os passos seguidos pelos usuários, mas principalmente descobrir quais padrões podem ser interessantes para o domínio da aplicação [8]. Estes padrões de uso podem auxiliar, por exemplo, na definição de campanhas promocionais, no planejamento de estratégias de marketing, reestruturação e adaptação automática dos sites, entre outras aplicações [17].

O capítulo tem a seguinte organização. A seção 3.1 apresenta os aspectos gerais da MUW. A seção 3.2 introduz o processo geral da MUW assim como as etapas associadas ao processo. A seção 3.3 apresenta um paralelo entre o processo de modelagem de usuários e o processo da MUW mostrando a sua relação com os sistemas de personalização da Web. E por último, a seção 3.3, apresenta as considerações finais do capítulo.

3.1 Aspectos gerais

Com o crescimento da Web e a complexidade cada vez maior das atividades de projeto e implementação dos sites, é notória a necessidade de informações sobre os padrões de uso dos usuários. Essa análise pode auxiliar na reestruturação e projeto físico dos sites, além de servir como base para ferramentas de apoio à navegação.

A MUW trata da descoberta de padrões de navegação dos usuários, através da aplicação de técnicas de mineração de dados aos dados de uso da Web. Esta idéia foi apresentada inicialmente por Chen et al. [25], Mannila & Toivonen [102] e Yan et al. [151].

Os dados de uso estão concentrados principalmente nos logs dos servidores Web, que armazenam as interações dos usuários com as páginas visitadas, mas também podem ser encontrados nas próprias estruturas dos sites Web (informações sobre as referências e vínculos entre as páginas) ou obtidos dos usuários a partir do uso de programas CGI, cookies e outros mecanismos [15].

A análise de todos esses dados pode, dessa forma, ser uma ferramenta de grande utilidade no entendimento não só do comportamento de navegação dos usuários, mas também da própria estrutura da Web, ajudando também a organizar e modificar tal estrutura.

Originalmente, o objetivo da MUW tem sido o de auxiliar os seres humanos no processo de tomada de decisões. Dessa forma, a saída do processo é geralmente um conjunto de modelos de dados que representam conhecimento implícito sobre padrões de navegação dos usuários da Web. Esses modelos são então analisados por especialistas, tais como analistas de mercado que buscam novas formas de aumentar os lucros ou administradores de sites que buscam aperfeiçoar a estrutura do site de forma a melhorar a experiência de navegação dos visitantes.

Embora originalmente o objetivo principal da MUW não esteja diretamente associado às tarefas de personalização, a sua relação com ferramentas automáticas de personalização é bastante óbvia. Os dados de uso representam as interações dos usuários com os sites da Web. A MUW fornece uma abordagem para a coleta e pré-processamento desses dados, gerando modelos que representam, dentre outras coisas, interesses e objetivos dos usuários. Esses modelos podem então servir de referência para sistemas de personalização automática da Web, dispensando assim a intervenção de especialistas humanos [122].

A MUW apresenta, então, estes três aspectos: é apropriada para analisar sistematicamente o comportamento passado dos usuários, serve como apoio na tomada

de decisões e ainda fornece conhecimento operacional que pode servir como insumo para sistemas de personalização automática da Web.

3.2 O processo da MUW

Na seção 2.1 foi mostrado o processo genérico da mineração da Web. A MUW contempla essencialmente as mesmas etapas, sendo que cada fase possui as suas respectivas especificidades intrínsecas. O processo então possui as seguintes etapas:

- *Aquisição de dados*. Nesta etapa, os dados de uso, que podem ser provenientes de várias fontes, são reunidos e o seu conteúdo e estrutura identificados. Esses dados podem ser coletados tanto de servidores Web, de máquinas clientes ou de fontes intermediárias tais como servidores proxy;
- *Pré-Processamento dos dados*. Nesta etapa, os dados são limpos de ruídos e inconsistências e são integrados de forma a serem utilizados como entrada para a próxima etapa, correspondente à *descoberta de padrões*. Isso envolve basicamente as tarefas de filtragem dos dados, identificação de usuários e identificação de sessões dos usuários;
- *Descoberta de padrões*. Nesta fase, os padrões são descobertos através da aplicação de técnicas estatísticas e de aprendizagem de máquina aos dados, tais como: agrupamento, classificação, descoberta de regras de associação e descoberta de padrões seqüenciais. Geralmente o conhecimento requerido pelos sistemas de personalização corresponde aos padrões de navegação dos usuários inferidos nessa etapa;
- *Pós-processamento do conhecimento*. Nesta última fase, o conhecimento extraído é avaliado e apresentado em um formato inteligível pelos humanos, como por exemplo, através de relatórios ou ferramentas de visualização. Para fins de personalização automática, o conhecimento descoberto é diretamente incorporado a um módulo de personalização.

A Figura 2 sumariza graficamente as etapas descritas acima. Cada uma dessas etapas apresenta vários problemas que são intrínsecos ao processo da MUW. As próximas subseções descrevem em mais detalhes cada uma dessas etapas, mostrando os problemas associados e algumas das soluções propostas pela literatura.

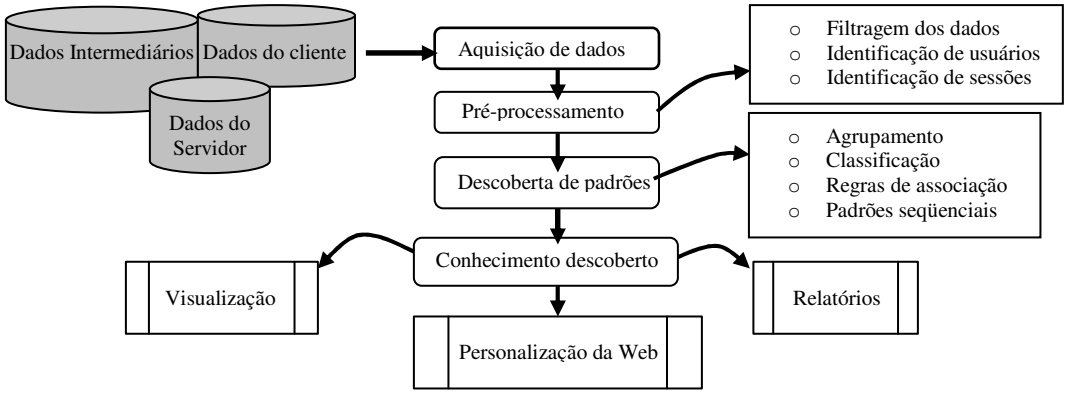

Figura 2. Processo da MUW

3.2.1 Aquisição de dados

A primeira etapa do processo consiste em reunir os dados relevantes da Web. Há basicamente duas fontes principais de dados para a MUW: os dados gerados no lado do servidor Web e os dados gerados no lado cliente. Além disso, quando dados intermediários são introduzidos na comunicação cliente-servidor eles também podem se tornar fontes de dados de uso, como, por exemplo, servidores proxy e rastreadores de pacotes ("*packet sniffers*"). Cada uma dessas fontes é examinada em maiores detalhes a seguir.

3.2.1.1 Dados do lado servidor

3.2.1.1.1 Arquivos de log do servidor

Os dados gerados no lado do servidor são coletados dos servidores de sites Web e consistem basicamente de vários tipos de arquivos de log de acesso. Esses arquivos registram as páginas e arquivos acessados pelos usuários durante suas atividades de navegação pela Web. A maioria dos servidores Web suporta como opção padrão o "Formato de Arquivo de Log Comum" (Common Log File Format), que inclui

informações sobre o endereço IP da máquina cliente que realizou as requisições, o nome da máquina cliente (*hostname*), nome de usuário (quando disponível), o instante da requisição (*timestamp*) e o nome e tamanho do arquivo requisitado. O formato de log estendido definido pela W3C, suportado atualmente pela maioria dos servidores Web (ex.: Apache), incluem informações adicionais, tais como: o endereço de "URLs referentes[4]", o nome e versão do software de navegação usado pelo usuário e o sistema operacional instalado na máquina cliente.

Os arquivos de log são tidos como a principal fonte de dados utilizada pelas aplicações baseadas na MUW. Entretanto, esses arquivos nem sempre são uma fonte confiável sobre o uso de um site. Isso se dá basicamente por dois motivos: o mecanismo de cache da Web e a interpretação errônea de endereços IP.

O mecanismo de cache da Web é utilizado para reduzir a latência e o tráfego da rede. O cache salva cópias locais das páginas requisitadas pelo usuário por certo período de tempo e, dessa forma, quando houver uma nova requisição a uma página que já foi visitada anteriormente, a cópia armazenada no cache é utilizada. Os caches da Web podem ser configurados tanto nos navegadores locais dos usuários como em servidores proxy. O problema é o mesmo para ambos, ou seja, se a página requisitada provém de um cache, a requisição do cliente não alcança o servidor correspondente e como conseqüência essa ocorrência de acesso não é gravada no arquivo de log. Isso pode causar sérias distorções nos padrões descobertos, pois algumas das páginas visitadas pelo usuário que podem indicar o seu comportamento real de navegação não estarão presentes nos arquivos de log. Para contornar esse problema, pode-se forçar o não armazenamento da página Web pelo cache, obrigando o navegador a recarregar a página sempre que ela for visitada, num processo conhecido como "*cache busting*". Entretanto, essa abordagem vai contra a principal motivação para a utilização dos caches, que é exatamente a redução da latência da Web.

O segundo problema, a interpretação errônea de endereços IP nos arquivos de log, ocorre devido a duas razões principalmente: a utilização de servidores proxy e a utilização de um mesmo IP por usuários diferentes. Os servidores proxy atribuem o

[4] Uma URL referente é a URL imediatamente anterior à URL atual

mesmo endereço IP a todos os usuários que estão acessando a Internet através da mesma rede local (LAN). Como conseqüência, todas as requisições que passam pelo servidor proxy, mesmo sendo de máquinas cliente diferentes, serão gravadas no arquivo de log como ocorrências do mesmo endereço IP. Isso pode causar uma distorção na interpretação dos dados de uso, pois algumas técnicas de identificação de usuários em arquivos de log utilizam o endereço IP como heurística e sendo assim, usuários distintos com comportamentos de navegação distintos podem ser erroneamente identificados como sendo o mesmo usuário. O mesmo efeito desse problema ocorre quando a mesma máquina cliente é usada por vários usuários distintos. O problema oposto ocorre quando, para o mesmo usuário, são atribuídos vários endereços IP diferentes, por exemplo, a atribuição dinâmica de endereços IP por provedores de Internet via conexão discada. Várias heurísticas, as quais serão discutidas na seção 3.2.5, foram propostas de forma a aliviar esses problemas.

3.2.1.1.2 Cookies

Outra técnica bastante utilizada para a aquisição dos dados de uso é a utilização de *cookies*. Os *cookies* são pequenos arquivos de texto distribuídos pelos servidores da Web e armazenados pelas máquinas cliente para uso futuro. Eles são usados, principalmente, para armazenar informações sobre a navegação do usuário.

Os servidores Web podem armazenar informações em um *cookie* que fica armazenado na máquina cliente, distribuindo assim a carga de armazenamento das informações sobre os usuários. Essas informações geralmente são identificadores (ID) criados pelos servidores para identificar um usuário unicamente. Então quando o usuário se conecta ao servidor, o *cookie* é transmitido de volta ao servidor de modo que o usuário possa ser identificado.

Os *cookies* também podem armazenar outros tipos de informação como produtos comprados em uma loja, tempo de visita em uma página, URLs das páginas visitadas, entre outros. Um *cookie* não pode ultrapassar o tamanho máximo de 4Kbytes, sendo, portanto, limitado a uma pequena quantidade de informações.

No entanto, o uso de *cookies* não é livre de problemas. Se o usuário se conectar a Internet através de diferentes máquinas, diferentes *cookies* lhe serão atribuídos. Se por outro lado, diferentes usuários utilizarem a mesma máquina, os mesmos *cookies* serão utilizados por todos os usuários. Além disso, os usuários têm a opção de desabilitar o uso de *cookies* por seus softwares de navegação. E mesmo quando os *cookies* são aceitos, os usuários tem a possibilidade de seletivamente apagá-los do sistema a qualquer momento. Os *cookies* também são numericamente limitados. Apenas vinte *cookies* são permitidos por domínio e não mais que trezentos por máquina. Se o número de *cookies* exceder esses valores, os *cookies* mais antigos cedem sucessivamente o lugar aos mais novos [122].

3.2.1.1.3 Dados explícitos de usuários

As informações que são fornecidas diretamente pelos usuários também podem ser úteis para a MUW. Esses dados são adquiridos, principalmente, através de formulários e podem conter informações importantes do usuário, tais como: informações pessoais, informações demográficas, interesses, preferências entre outras. Entretanto, esse método é intrusivo e, muitas vezes, obriga o usuário a parar as suas atividades atuais de forma a preencher extensos formulários. Isso acaba por desmotivar seriamente o retorno ao site pelo usuário. Além disso, não se pode confiar completamente nessas informações já que elas são geralmente incompletas e imprecisas. Isso se dá devido à tendência que os usuários têm de fornecer o mínimo de informações pessoais possíveis, temendo questões relacionadas à privacidade e segurança.

3.2.1.1.4 Dados externos

Os dados externos se referem aos bancos de dados contendo informações demográficas dos usuários mantidos por terceiros para algum fim específico [112]. Entretanto, questões concernentes à privacidade levaram ao surgimento de obstáculos legais no que diz respeito à distribuição desses dados.

3.2.1.2 Dados do lado cliente

Os dados de uso no lado cliente são adquiridos diretamente da máquina cliente que está acessando um site da Web. Uma das técnicas mais comuns é a utilização de

pequenos programas remotos rodando na máquina cliente, geralmente escritos em Java ou Javascript [132], [133]. Esses programas são embutidos em páginas da Web, como applets Java por exemplo, e são utilizados para a coleta de informações diretamente do cliente. Essas informações compreendem basicamente o tempo gasto na visita de uma página e o histórico de navegação do usuário.

Os dados adquiridos dessa forma são mais confiáveis que os dados coletados no lado servidor, já que eles são livres dos problemas relacionados ao cache e interpretação errônea de endereços IP. No entanto, os métodos de aquisição de dados no lado cliente também têm os seus problemas. Um desses problemas está no fato de que os diversos programas utilizados para a coleta de informações afetam o desempenho do sistema, adicionando uma carga de tempo extra para o carregamento das páginas no navegador. Além disso, esses métodos requerem a cooperação dos usuários no que diz respeito a habilitarem o suporte a Java de seus softwares de navegação.

Uma outra técnica proposta para a aquisição de dados no lado cliente foi a modificação do software de navegação Mosaic [34]. Essa modificação permitia que o navegador armazenasse as páginas visitadas pelos usuários e as enviasse juntamente com outras informações de volta ao servidor. Mas a modificação de softwares de navegação não é uma tarefa trivial, mesmo quando o código fonte está disponível. Além disso, navegadores modificados que armazenam o comportamento do usuário são considerados uma ameaça à privacidade do usuário e, dessa forma, é muito difícil fazer com a sua utilização seja aceita.

3.2.1.3 Dados intermediários

3.2.1.3.1 Servidores proxy

Um servidor proxy é um software usualmente utilizado por empresas que desejam conectar a sua rede interna (LAN) à Internet e age como um intermediário entre as máquinas cliente internas e a Internet. Os servidores proxy oferecem serviços associados a segurança, controle administrativo e cache. Apesar dos problemas que eles causam, como já discutido anteriormente, eles também podem prover uma rica fonte de dados de uso.

Similarmente aos servidores da Web, os servidores proxy também armazenam arquivos de log de acesso. A vantagem desses logs é que eles armazenam as informações de acesso dos usuários por trás do proxy. Dessa forma, os logs de proxy fornecem informações mais precisas para a correta identificação dos usuários. Mas mesmo assim, os problemas relativos ao cache e à interpretação errônea dos endereços IP ainda persistem.

3.2.1.3.2 Rastreadores de pacotes ("*packet sniffer*")

Um rastreador de pacotes é um software que monitora pacotes TCP/IP viajando em direção a um servidor Web e tem a habilidade de extrair dados a partir deles. Uma vantagem desses rastreadores em relação aos arquivos de log é que os dados podem ser adquiridos e analisados em tempo real. Outra vantagem importante diz respeito à coleta de informações em nível de rede, as quais não estão presentes nos arquivos de log convencionais. A página completa que foi requisitada também pode estar incluída nos pacotes capturados [46].

Por outro lado, o uso de rastreadores de pacotes possui sérias desvantagens em relação aos arquivos de log. Como os dados são coletados em tempo real e não são armazenados, eles podem ser perdidos para sempre caso ocorra algum problema ou com o software de rastreamento ou com a transmissão dos dados. Além disso, especialmente em sites de comércio eletrônico, os pacotes são transmitidos criptografados, dificultando assim a extração de informações úteis. Por último, as ferramentas de rastreamento de pacotes são consideradas uma séria ameaça à privacidade dos usuários já que operam diretamente sob os dados transmitidos pela Internet [122].

3.2.2 Pré-processamento dos dados

Os dados coletados na etapa anterior (aquisição de dados) são geralmente heterogêneos e volumosos. Portanto, antes da efetiva descoberta dos padrões esses dados precisam ser integrados em um modelo de dados consistente. Assim como na maioria das aplicações baseadas na mineração de dados, o pré-processamento envolve a resolução de qualquer inconsistência existente nos dados. Isso envolve principalmente a

remoção de dados redundantes e irrelevantes, tratamento de dados inconsistentes e transformação e formatação dos dados.

Na MUW essa etapa envolve basicamente a identificação dos usuários e suas sessões de navegação. Essas informações são então usadas como blocos básicos para a descoberta de padrões.

O pré-processamento dos dados também é, em alguns casos, dependente da aplicação, pois o conteúdo e a estrutura de um site podem afetar a decisão sobre quais dados devem ser considerados relevantes. Também é fortemente dependente do tipo e, sobretudo, da qualidade dos dados. Essa é sem dúvida a fase mais complexa do processo de mineração, sendo determinante para a qualidade dos padrões a serem descobertos.

3.2.2.1 Filtragem dos dados

O primeiro passo do pré-processamento é a limpeza dos dados brutos. Durante essa etapa os dados disponíveis são examinados e itens irrelevantes ou redundantes são eliminados. Isso é necessário principalmente para os logs de servidores Web e proxy, pois como eles gravam todas as interações dos usuários, eles geralmente possuem muito ruído. Os dados coletados por programas-remotos no lado cliente são geralmente limpos, pois os programas são pré-instruídos a coletar somente os dados relevantes e, dessa forma, a fase de pré-processamento é bastante aliviada. Já os eventuais dados entrados explicitamente pelo usuário (formulários de registro) precisam ser verificados, corrigidos e normalizados de modo a serem úteis na descoberta de padrões.

A principal razão da redundância nos dados encontrados nos arquivos de log se deve a natureza do protocolo HTTP [46], o qual requer uma requisição separada para cada arquivo (imagem, vídeo, etc.) contido na página. Na maioria das vezes esses arquivos são descarregados mesmo que o usuário não os tenha requisitado explicitamente, portanto, eles não são considerados como parte do comportamento real de navegação do usuário. Sendo assim, essas entradas são geralmente removidas dos arquivos de log [29]. Entretanto, como discutido acima, o pré-processamento dos dados é, algumas vezes, dependente do domínio da aplicação e dessa forma, eliminar essas

entradas pode causar uma perda no valor dos dados. Esse é o caso, por exemplo, de um site que possua predominantemente conteúdo multimídia.

As entradas nos logs também podem corresponder a requisições que não foram satisfeitas, ou seja, requisições respondidas com mensagens de erro do protocolo HTTP. Geralmente isso acontece por uma das seguintes razões: quando uma URL é digitada erroneamente no navegador; quando a página requisitada está fora do ar ou quando a página não existe mais ou mudou de lugar. Geralmente essas entradas também são retiradas do log, mas em alguns casos elas podem fornecer informações valiosas sobre a intenção do usuário, pois mesmo a requisição tendo falhado, o usuário tinha uma intenção ao requisitar a página. Além disso, as entradas geradas pelas aranhas da Web[5], isto é, programas que descarregam sites da Web de modo a atualizarem os índices de motores de busca, também devem ser eliminadas. As aranhas podem ser reconhecidas através do campo *User Agent* dos arquivos de log, pois a maioria delas se identifica através desse campo. Outra técnica é analisar o padrão de tráfego de um usuário particular. Se esse padrão indicar um tráfego contínuo e ininterrupto de visitas, como por exemplo, a visita de cada vínculo em cada página de um site, então esse padrão se caracteriza como uma aranha. Em Tan e Kumar (2002) [144] é proposto um método para a identificação de sessões de aranhas através da extração de diversas características derivadas dos acessos encontrados nos logs, como por exemplo, a porcentagem de arquivos multimídia requisitados e a porcentagem das requisições HTTP realizadas.

A MUW é geralmente utilizada para identificação do comportamento de usuários em sites específicos, sendo assim, os dados referentes a outros sites, senão os de interesse, são considerados irrelevantes e geralmente também são eliminados. Isso acontece geralmente em servidores hospedando mais de um site ao mesmo tempo, onde o log vai conter registros de acessos de todos os sites hospedados. Por exemplo, se a intenção é identificar o comportamento de navegação dos usuários em um site de compras específico, os registros no log referentes a outros sites serão irrelevantes.

[5] Comumente chamados de “Spiders” e “Crawlers”

3.2.2.2 Identificação de usuários

A identificação de usuários através dos dados de uso é crucial, especialmente para a personalização. A maioria dos sistemas comerciais de personalização existentes requer que o usuário se registre antes de usá-los. Entretanto, isso significa um fardo a mais para a navegação do usuário, o que é inaceitável por muitas aplicações. Por essa razão, muitas abordagens têm sido propostas com o intuito de automatizar o processo de identificação dos usuários, sendo que as mais importantes serão discutidas a seguir.

A abordagem mais simples consiste em atribuir um usuário a cada IP encontrado no arquivo de log. O grande problema é quando os usuários acessam a Internet através de um proxy pois, sendo assim, todos os usuários possuirão o mesmo IP. Os *cookies* também são úteis para a identificação de usuários [87]. Nessa abordagem os servidores geram e despacham um *cookie* para cada usuário, de forma que o *cookie* contenha um ID de usuário que lhe é atribuído pelo servidor. Dessa forma, quando os usuários retornam ao site o *cookie* é enviado ao servidor, e este através do ID contido no *cookie* identifica o usuário corretamente. Mas por motivos de segurança e privacidade os *cookies* podem, a qualquer momento ser desabilitados ou apagados pelo usuário. Além disso, se o usuário se conectar a Internet através de diferentes máquinas ele não poderá ser identificado corretamente. Por causa desses problemas outras heurísticas foram propostas. Uma dessas heurísticas é a utilização de serviços especiais da Internet, tais como o *inetd* e *fingerd*, os quais fornecem informações pessoais dos usuários da Web [123]. Um dos problemas dessa abordagem é que estes serviços podem estar desabilitados por questões de segurança e privacidade. Além disso, se os usuários estiverem acessando a Internet através de um proxy, esses serviços se tornam problemáticos já que sendo assim, todos os usuários possuirão o mesmo IP.

Duas outras heurísticas para tratar da identificação dos usuários são apresentadas em Cooley et al. (1999) [29]. A primeira faz uma análise dos logs no formato estendido procurando por informações relacionadas aos navegadores Web e sistemas operacionais utilizados pelas máquinas clientes (em termos de tipo e versão). A análise dessas informações pode sugerir a existência de usuários diferentes mesmo quando o endereço IP dos acessos é o mesmo. A outra heurística apresentada combina a topologia

do site com as entradas nos logs de URLs referentes. Se a requisição a uma página é feita pelo mesmo endereço IP que visitou outras páginas anteriormente, e se não há nenhum vínculo entre essas páginas, então pode se suspeitar que se trate de usuários diferentes. Mas há casos onde essas abordagens falham, por exemplo, quando o usuário abre diferentes navegadores ao mesmo tempo ou quando o usuário acessa páginas do mesmo site que não possuam vínculos entre si.

Em Schwarzkopf (2001) [135] é utilizada uma técnica diferente. Um ID é gerado para cada usuário pelo servidor Web e incluído nas URLs das páginas requisitadas pelo usuário. Em vez de armazenar o ID em um *cookie*, é solicitado que o usuário guarde uma das páginas contendo o seu ID na pasta favoritos de seu navegador. O ID é então usado para identificar o usuário todas as vezes que ele acessar o site através da página guardada na pasta favoritos (na verdade não é a página que é guardado mas só a sua URL). O acesso a essa página é então armazenado no arquivo de log substituindo-se o IP do usuário pelo seu ID. Essa técnica é bastante simples, e contorna muitos dos problemas associados aos *cookies*, como, por exemplo, a possibilidade de bloqueio dos *cookies* pelo usuário. Mas como todas as outras essa abordagem também possui os seus problemas. O problema principal é que o processo é semi-automático, pois o usuário precisa guardar uma das páginas visitadas na pasta favoritos e depois acessar o site através dela todas as vezes que precisar acessar o site, pois se o site for acessado de outra forma o servidor não terá acesso ao ID. Além disso, esta técnica se torna ineficiente quando o usuário acessa o site através de diferentes máquinas.

3.2.2.3 Identificação das sessões

Uma sessão de usuário é um conjunto delimitado de páginas visitadas por um usuário durante uma visita particular a determinado site [122]. Vários métodos têm sido utilizados para a identificação das sessões. Spiliopoulou (1999) [137] divide esses métodos de acordo com atributos relacionados ao tempo ou contexto. Um método baseado no tempo seria, por exemplo, a definição de um limite de tempo máximo o qual o usuário pode passar visitando uma página, ou o limite de tempo que determina a própria duração máxima de uma sessão. Um método baseado no contexto seria, por exemplo, o agrupamento de acessos a páginas que estejam semanticamente relacionadas [122].

Os métodos baseados no tempo têm sido amplamente utilizados pela literatura ([13], [20], [132], [149]). De acordo com essas abordagens, um conjunto de páginas visitadas pelo usuário é considerado uma sessão se as páginas tiverem sido requisitadas durante um intervalo de tempo menor do que um intervalo de tempo máximo especificado para a sessão. Esse intervalo, também conhecido como "*page viewing time*" [133], está na faixa de 25.5 minutos, e foi inicialmente proposto por Catledge e Pitkow (1995) [20]. Entretanto esse método não é muito confiável, pois não há como prever as ações dos usuários com exatidão. O usuário pode, por exemplo, passar um longo tempo lendo o mesmo documento ou pode deixar o escritório por algum tempo e depois voltar para ler o documento. Além disso, a definição da faixa de tempo do intervalo vai depender largamente do conteúdo do site sendo examinado.

Shahabi et al. (2001) [132] propõe um método baseado em applets Java para medir o tempo de permanência de um usuário em uma página. Esse applet envia ao servidor o instante em que determinada página é carregada ou descarregada no navegador do usuário. Entretanto, fatores externos tais como o tipo de navegador utilizado e o tráfego de rede introduzem importantes obstáculos a esse método. A maioria dos navegadores atuais já vem com o suporte ao Java como padrão, mas se o usuário estiver utilizando navegadores mais antigos sem esse suporte o método vai ser ineficaz. Além disso, a latência da rede pode fazer com que os tempos de visita capturados cheguem com algum atraso ao servidor, mas mesmo com esse atraso os dados continuarão precisos, pois foram capturados diretamente do sistema cliente. Como veremos no decorrer do trabalho essa foi a abordagem à qual nos inspiramos para a aquisição dos dados de uso gerados pelo usuário.

As sessões de usuários descobertas ainda podem passar por mais uma etapa de pré-processamento conhecida como descoberta de transações. Uma transação é um termo derivado da mineração de dados e é utilizado principalmente na análise de cestas de compra em supermercados [111]. Uma transação no contexto da MUW é um subconjunto de páginas relacionadas que ocorrem dentro de uma sessão. De forma a identificar as transações Cooley et al.(1997) [29] levantou a hipótese de que as transações estão fortemente associadas ao comportamento de navegação de um usuário específico, e dessa forma, podem ser identificadas através de informações do contexto de

navegação do usuário. Baseado nessa hipótese, as páginas de um site podem ser divididas em três contextos ou categorias: *páginas de navegação* ou *auxiliares*, que contém primariamente vínculos para outras páginas e são usadas para prover caminhos de navegação, *páginas de conteúdo*, contendo as reais informações de interesse dos usuários, e *páginas híbridas*, que combinam esses dois tipos de informação. Embora haja páginas que claramente pertençam a uma dessas três categorias, como as páginas principais (*home*) de um site, por exemplo, essa classificação não é muito rigorosa e depende da perspectiva do usuário. As páginas que são consideradas de navegação por um usuário, por exemplo, podem ser consideradas como páginas de conteúdo por outros.

Em Cooley et al. (1999) [27] as transações são classificadas em dois tipos basicamente: *transações conteúdo*, que contém as páginas de conteúdo propriamente ditas, e as *transações auxiliares*, que correspondem às páginas que apontam para as páginas de conteúdo. Dois métodos foram definidos para a identificação das transações desses dois tipos: *identificação por duração da referência* e *identificação por referência posterior máxima*. O método de duração por referência examina o período de tempo que um usuário passou visitando uma página. Se esse período for maior que certo limite, assume-se que a página contém informações relevantes e, sendo assim, a página deve ser incluída em uma transação do tipo *conteúdo*. De outra forma, a página deve ser incluída em uma transação do tipo *auxiliar*. Essa abordagem considera que a última página visitada por um usuário é sempre uma página de conteúdo. Se for assim, qualquer interrupção por fatores externos na navegação do usuário pode levar à identificação errônea de uma página de conteúdo. Um outro problema é a definição do limite de tempo para a sessão, pois a definição de um limite de tempo apropriado é um aspecto subjetivo dependente do conteúdo do site.

O método de identificação por referências posteriores máximas ("*maximal forward references*") foi introduzido inicialmente por Chen et al. (1996) [26]. Ele considera que uma transação é composta por uma seqüência de páginas visitadas a partir de uma página inicial, até a última página acessada imediatamente antes da próxima referência reversa.

Uma referência reversa é definida como uma página já presente no conjunto de páginas visitadas na transação. Por sua vez, uma referência posterior é uma página ainda não visitada na transação. Uma referência posterior máxima é, portanto, a última página acessada pelo usuário antes dele tentar voltar para uma página já visitada na transação, denotando assim o final desta.

O início da próxima transação corresponderá à primeira referência posterior acessada logo após o final da transação atual. Portanto, após a detecção do final da transação, ao ser atingida a primeira referência reversa, devem ser ignoradas as próximas referências reversas, até que se chegue a uma nova referência posterior, quando então se considera iniciada a nova transação. Este modelo pressupõe que as páginas de referências posteriores máximas são páginas de conteúdo, e as demais páginas que conduzem a elas são páginas auxiliares [15]. Esse método tem uma vantagem em relação ao método de duração da referência, pois ele é independente do conteúdo do site. Entretanto, ele também sofre do problema de cache das páginas, pois isso impede que a referência seja gravada no arquivo de log. O "*cache-busting*" pode contornar esse problema como já discutido anteriormente.

3.2.3 Descoberta de padrões

Nesse estágio, métodos estatísticos e de aprendizagem de máquina são utilizados de forma a extrair padrões de uso dos dados pré-processados. Inúmeros métodos de aprendizagem de máquina têm sido utilizados no contexto da MUW. Esses métodos estão compreendidos nas quatro abordagens mais comuns de toda a literatura sobre mineração de dados: Agrupamento, Classificação, Regras de associação e Padrões seqüenciais. A classificação foi um dos primeiros métodos a ser aplicado na MUW, mas a dificuldade em se classificar grandes quantidades de dados para a aprendizagem supervisionada levou a adoção de métodos de aprendizagem não supervisionada, tais como o agrupamento.

Ao contrário dos métodos para o pré-processamento dos dados, os métodos para a descoberta de padrões são independentes do domínio, ou seja, eles podem ser aplicados a qualquer site independente do seu conteúdo.

As próximas subseções são dedicadas aos métodos mais importantes de descoberta de padrões utilizados na MUW até agora. O foco das subseções é no tipo dos padrões que são buscados e não nos algoritmos de aprendizagem em si, os quais podem ser achados nos artigos originais referenciados quando oportuno.

3.2.3.1 Agrupamento

A grande maioria dos métodos que têm sido utilizados para a descoberta de padrões nos dados de uso da Web são métodos de agrupamento. O agrupamento consiste em dividir um conjunto de dados em grupos que sejam diferentes entre si e cujos membros sejam similares entre si. Han e Kamber (2001) [71] propuseram uma taxonomia para os métodos de agrupamento consistindo nas seguintes categorias:

- *Métodos de partição*, que criam *k* grupos a partir de uma partições em conjuntos de dados;
- *Métodos hierárquicos*, que decompõe um dado conjunto de dados em grupos hierárquicos;
- *Métodos baseados em modelos*, que criam grupos a partir de modelos matemáticos ou conceituais.

No contexto da mineração da Web o agrupamento tem sido utilizado, principalmente, para a agregação de usuários com comportamento de navegação similar ou para o agrupamento de páginas com conteúdo similar [35], [141]. Uma importante restrição a ser observada para a escolha do método de agrupamento é se o método permite a sobreposição de grupos. Isso é particularmente importante para a personalização, pois um usuário pode pertencer a mais de um grupo ao mesmo tempo.

Yan et al. (1996) [151] foi um dos primeiros a utilizar métodos de partição na MUW. Nesse trabalho, o algoritmo Leader é usado para o agrupamento das sessões de usuários [76]. O Leader é um algoritmo incremental que produz grupos de alta qualidade. Cada usuário da sessão é representado por um vetor de características de *n* dimensões, onde *n* corresponde às páginas visitadas na sessão. O valor de cada característica é um peso indicando o grau de interesse do usuário na página. Esse peso pode ser determinado por vários parâmetros tais como o número de vezes que uma página foi

visitada e o período de tempo gasto nas visitas. Baseado nesses vetores, sessões similares são agrupadas. Esses grupos correspondem aos padrões indicando usuários com interesses de navegação similares. A escolha dos parâmetros para a definição dos pesos não é uma tarefa trivial e depende tanto da aplicação quanto das habilidades de modelagem de um especialista humano [122]. O Leader, no entanto, tem um problema associado à ordem em que os vetores são processados pelo algoritmo. Se, por exemplo, o conjunto de vetores (a,b,c) for submetido nessa ordem ao algoritmo e outro conjunto (c,b,a) for submetido também nessa ordem, grupos diferentes serão gerados, quando na verdade os mesmos grupos deveriam ser gerados.

Um método de partição baseado na teoria dos grafos é apresentado em Perkowitz & Etzioni (1998) [120], os quais desenvolveram um sistema baseado na MUW para tornar sites da Web adaptativos. O elemento central desse sistema é um novo método algorítmico chamado "*cluster mining"*, o qual é implementado pelo algoritmo *PageGather*. O *PageGather* recebe como entrada as sessões dos usuários contendo as páginas visitadas e através dessas sessões o algoritmo cria um grafo atribuindo cada página da sessão a um vértice do grafo. Uma aresta é adicionada a dois vértices se as páginas correspondentes co-ocorrem em determinado número de sessões. A principal vantagem do *PageGather* é que ele permite a criação de grupos sobrepostos, ou seja, um usuário pode pertencer a mais de um grupo ao mesmo tempo. A principal desvantagem está no custo computacional necessário para a computação dos grafos.

Uma extensão dos métodos de partição é o agrupamento nebuloso (*"fuzzy clustering"*), o qual permite ambigüidades nos dados. Aqui cada item de dado pode estar distribuído entre os vários grupos existentes. Tal método é proposto em Joshi & Joshi (2000) [86] para o agrupamento de sessões de usuários. A topologia do site é usada para o cálculo de similaridade entre as sessões. O site é modelado em forma de árvore onde cada nó corresponde a uma URL e as arestas representam uma relação hierárquica entre as URLs. O cálculo da similaridade entre as sessões é baseado na posição relativa das URLs na árvore que pertencem às sessões. O agrupamento é implementado através de dois algoritmos: *fuzzy c-medoids* e *fuzzy c-trimmed-medoids*, os quais são variantes do método de agrupamento nebuloso *c* [11].

Um método hierárquico para o agrupamento de sessões baseado no algoritmo BIRCH [152] é apresentando por Fu et al. (1999) [54]. Nesse trabalho os dados dos logs são transformados em vetores, onde cada vetor contém o ID da página visitada e o tempo gasto na visita dessa página. De forma a melhorar o desempenho do algoritmo, as sessões são generalizadas através da hierarquia de páginas do site. Sendo assim, cada página da sessão é substituída por uma página mais genérica de acordo com a hierarquia de páginas do site, isso é feito através do método de indução orientado a atributos ("*attribute-oriented induction method*") [73]. As sessões genéricas resultantes servem como entrada para o algoritmo BIRCH. O BIRCH é um algoritmo incremental bastante eficiente para o processamento de grandes volumes de dados sendo capaz de produzir grupos de qualidade em uma só rodada. Entretanto, similarmente ao Leader, o BIRCH é dependente da ordem dos vetores de entrada.

Diversos métodos de agrupamento baseados em modelos foram utilizados por Paliouras (2000) [118] para a descoberta de modelos de comunidades, isto é, modelos representando grupos de usuários com padrões de navegação similares. Entre eles, estão: um método probabilístico, Autoclass [74], uma rede neural, Mapas de auto-organização (*"Self-Organizing Maps"*) [91], um método de agrupamento conceitual, COBWEB [49] e o seu variante não hierárquico ITERATE [12].

O Autoclass é baseado em um modelo matemático que fornece uma forma disciplinada de eliminar parte da complexidade associada aos dados. Entretanto o algoritmo é computacionalmente caro, além de requerer certo grau de conhecimento apriorístico dos dados. Os mapas auto-organizáveis são modelos de redes neurais utilizados para o mapeamento de dados multidimensionais em uma representação de duas dimensões no espaço, facilitando assim a descoberta dos grupos. Entretanto, esse método requer a especificação prévia do número de grupos a serem descobertos.

Os algoritmos de agrupamento conceituais, tais como o COBWEB e o ITERATE, medem a similaridade dos objetos de acordo com a sua adequação a descrições conceituais ao contrário dos outros que geralmente estão baseados em medidas numéricas de distância. O COBWEB é um algoritmo incremental que fornece características descritivas de cada grupo. Entretanto, o algoritmo sofre de problemas de

escalabilidade e também é dependente da ordem dos dados de entrada. O ITERATE resolve o problema de dependência da ordem de entrada, mas além de não ser incremental também não escala.

Apesar da grande variedade de métodos e algoritmos de agrupamento existentes, poucos trabalhos têm sido realizados no sentido de fornecer comparações entre esses métodos em termos de seus desempenhos.

Um método bastante simples para a validação dos grupos é proposto por Estivill-Castro (2002) [38]. Esse método envolve a definição de um conjunto de dados bem estruturados e comuns onde os resultados do agrupamento podem ser avaliados. Tanto esses dados como os métodos de avaliação devem ser públicos de forma que qualquer pesquisador possa testar e verificar a qualidade de seus algoritmos.

Paliouras et al. (2002) [118] também propõe um método para a avaliação dos grupos baseado na qualidade dos modelos de comunidades. Os modelos são analisados através de duas medidas: *distintividade*, isto é, quão diferentes os modelos são diferentes entre si, e *cobertura* do domínio, ou seja, quantas páginas do site estão contidas nos modelos.

A Tabela 1 apresenta um resumo dos algoritmos de agrupamento discutidos acima. A tabela apresenta as aplicações baseadas na MUW que utilizam esses algoritmos, a abordagem de agrupamento empregada e indicações das vantagens e desvantagens de cada abordagem no contexto da personalização na Web.

3.2.3.2 Classificação

Ao contrário do agrupamento, o objetivo da classificação é identificar as características distintas de classes/categorias de acordo com suas instâncias (membros da classe). Essas informações podem ser usadas tanto para entender os dados existentes como para prever o comportamento de novas instâncias futuras.

Algoritmo	Aplicação	Método de agrupamento	Prós	Contras
Leader [76]	Agrupamento de sessões de usuários [151]	Partição	• Incremental • Grupos de alta qualidade	• Dependente da ordem
PageGather [120]	Síntese de páginas "Index" [121]	Partição	• Sobreposição de grupos • Cada grupo corresponde a um padrão comportamental	• Alto custo computacional
Agrupamento nebuloso *c* [11]	Agrupamento de sessões de usuários [86]	Partição Agrupamento nebuloso	• Sobreposição de grupos	• Projeto da função de classificação das sessões nos grupos não trivial
BIRCH [152]	Agrupamento de sessões de usuários [54]	Hierárquico	• Incremental • Eficiente para dados multidimensionais	• Dependente da ordem
Autoclass [74]	Agrupamento de sessões de usuários [118]	Baseado em modelo	• Forte suporte matemático • Grupos de alta qualidade	• Alto custo computacional • Requer suposições apriorísticas
Mapas auto-organizáveis [91]	Agrupamento de sessões de usuários [118]	Baseado em modelo	• Visualização de dados multidimensionais	• Pré-processamento e normalização dos dados • Especificação prévia do número de grupos
COBWEB [49]	Agrupamento de sessões de usuários [118]	Baseado em modelo	• Incremental • Caracterização dos grupos	• Dependente da ordem
ITERATE [12]	Agrupamento de sessões de usuários [118]	Baseado em modelo	• Independente da ordem	• Não incremental • Não escalável

Tabela 1. Resumo dos algoritmos de agrupamento para a MUW

A classificação é um método de aprendizagem supervisionada, pois os classificadores precisam ser treinados com base em um conjunto de dados disponíveis antes de serem aplicados a novas instâncias. Esses dados são geralmente chamados dados de treino e fornecem um conjunto de instâncias e as respectivas classes as quais pertencem, por exemplo, registros de dados mostrando pacientes classificados em diabéticos ou não diabéticos baseado em seus atributos (nível de colesterol, diabéticos na família, etc.) No contexto da MUW, a classificação tem sido usada tanto para modelagem do comportamento dos usuários quanto para a categorização automática de páginas Web. Os métodos que tem sido mais comumente usados são: regras de decisão, árvores de decisão, redes neurais e classificadores Bayesianos.

A indução de regras de decisão foi uma das primeiras abordagens para a classificação na MUW. Métodos desse tipo examinam os dados de treino disponíveis e constroem conjuntos de regras que determinam as condições que as novas instâncias devem satisfazer de modo a pertencer a uma ou outra classe. A primeira tentativa de usar esse método na MUW foi feita por Ngu & Wu (1997) [113] no sistema SiteHelper. O algoritmo de indução HCV [150] foi usado recebendo como entrada ou as páginas Web extraídas do log do servidor ou um conjunto de palavras chave fornecidas pelo usuário, os quais eram consideradas como exemplos positivos pelo algoritmo. Nesse contexto o resultado do processo de indução é um conjunto de regras representando os interesses dos usuários. O algoritmo HCV se mostra eficiente para a aprendizagem de regras quando trabalha com atributos discretos, mas deteriora rapidamente diante de dados com ruído e atributos contínuos.

Uma abordagem similar foi proposta por Jording (1999) [85], onde o algoritmo CDL4 [134] foi aplicado para a criação de conjuntos de regras de decisão de forma a determinar quando um usuário está interessado em um certo assunto. O CDL4 é um algoritmo incremental para a descrição de classes a partir de uma seqüência de instâncias disponíveis (dados de treino). Entretanto o processo de indução das regras do CDL4 pode resultar em regras bastante complexas e muitas vezes difíceis de interpretar.

Tanto a indução de árvores de decisão quanto os classificadores Bayesianos também tem sido bastante utilizados para a descoberta de padrões na MUW. A indução

de árvores de decisão geralmente envolve a partição recursiva dos dados de treino, resultando numa árvore onde cada ramo percorrido desde a raiz até as folhas é uma conjunção de testes para determinar a classe da instância em questão. Os classificadores Bayesianos, por outro lado [37], são baseados no cálculo de probabilidades condicionais para cada atributo da instância dada uma classe. Essas probabilidades definem as chances de uma instância pertencer a uma determinada classe. Chan (1999) [23] examina diversos algoritmos de classificação que seguem essa abordagem.

Uma outra abordagem para a classificação é baseada na teoria conhecida como "*rough set*". Essa teoria prima pela detecção de classes equivalentes nos dados, isto é, conjuntos de instâncias idênticas de acordo com atributos descrevendo os dados. Já que isso é extremamente raro de acontecer nos dados reais, a teoria "*rough set*" é usada para aproximar tais classes equivalentes. Apesar de ser capaz de tratar dados ruidosos e imprecisos os algoritmos de "*rough set*" possuem um alto custo computacional.

A teoria dos "*rough set*" é explorada por Maheswari et al. (2001) [105] para a descrição do comportamento de navegação dos usuários. Nesse trabalho, as sessões de navegação dos usuários são identificadas através dos arquivos de log e classificadas como exemplos positivos ou negativos de determinado conceito, como a compra de determinado produto, por exemplo. Um grafo "positivo" ponderado é então criado onde os nós correspondem às páginas Web e os vértices às transições entre elas. Os pesos representam a porcentagem de sessões positivas contendo uma transição específica dentro de todo o conjunto de sessões positivas. Da mesma forma, um grafo "negativo" ponderado também é criado. Novas sessões de usuários que ainda não foram classificadas também são representadas como grafos ponderados, onde o peso do vértice é uma função do peso desse vértice de acordo com os grafos negativos e positivos dos exemplos conhecidos. Um conjunto de atributos é então usado para determinar o grau de 'positividade' ou 'negatividade' de um link particular em uma sessão, baseado nos valores dos pesos correspondentes aos links dos dois grafos. Esses atributos definem as classes equivalentes, onde cada sessão de usuários é atribuída a uma dessas classes. Sendo assim, as classes são utilizadas para definir as 'regiões' positivas e negativas do espaço de instâncias, baseado no número de sessões positivas e negativas que elas contêm. Portanto, uma sessão ainda não classificada é atribuída a uma classe equivalente

baseada na análise dos valores dos atributos no grafo da sessão, e subsequentemente classificada como positiva ou negativa se a respectiva classe pertencer à região positiva ou negativa.

O uso da classificação na MUW é bem mais limitado do que o uso do agrupamento. Isso se dá porque na classificação os classificadores precisam ser treinados com base em dados de treino disponíveis. Mas em se tratando dos dados da Web isso quase nunca é possível. Sendo assim, técnicas de aprendizagem não supervisionada, tais como o agrupamento, são mais adequadas para a maioria dos problemas tratados na área da MUW. Especialmente aqueles relacionados à personalização, pois as restrições impostas pela classificação introduzem certa tendenciosidade nos padrões descobertos.

A Tabela 2 apresenta um resumo dos algoritmos de classificação discutidos acima. A tabela apresenta as aplicações baseadas na MUW que utilizam esses algoritmos, os métodos de classificação empregados e indicações das vantagens e desvantagens de cada abordagem no contexto da personalização na Web.

Algoritmo	**Aplicação**	**Método de classificação**	**Prós**	**Contras**
HCV [150]	Extração de regras representando os interesses dos usuários [113]	Regras de decisão	• Regras compactas	• Incapaz de lidar com dados ruidosos e atributos contínuos
CDL4 [134]	Extração de regras representando os interesses dos usuários [85]	Regras de decisão	• Rápido • Incremental	• Regras complexas
Classificação Bayesiana [37]	Previsão de uma possível página de interesse [23]	Classificação Bayesiana	• Rápido • Escalável	• Suposição de independência de atributos
Teoria “Rough Set”	Classificação de sessões de acordo com conceitos [105]	Teoria “Rough Set”	• Tratamento de dados ruidosos e imprecisos	• Alto custo computacional

Tabela 2. Resumo dos algoritmos de classificação para a MUW

3.2.3.3 Descoberta de padrões seqüenciais

A descoberta de padrões seqüenciais insere o elemento “seqüência” no processo. Dois tipos de métodos têm sido utilizados para a descoberta de padrões seqüências: *métodos determinísticos*, que armazenam o comportamento de navegação dos usuários e *métodos estocásticos*, que utilizam uma seqüência de páginas visitadas de forma a prever visitas subseqüentes.

Um exemplo de método determinístico é apresentado por Spiliopoulou et al. (1999) [138] que usa a ferramenta WUM ("*Web Utilization Miner*"), de sua autoria, para a descoberta de padrões seqüenciais. O módulo processador MINT do WUM, extrai regras de seqüências a partir dos arquivos de log pré-processados. O processo de mineração é interativo, onde restrições são definidas por um especialista. Para isso é disponibilizada uma linguagem de mineração similar ao SQL (“*Structured Query Language*”). Essa linguagem suporta predicados que podem ser utilizados para a especificação do conteúdo, estrutura e estatísticas dos padrões de navegação. Por se tratar de um processo semi-automático de mineração o WUM não é adequado para a personalização automática, onde as decisões devem ser tomadas em tempo real pelo sistema.

Outro método determinístico é empregado na ferramenta Clementine [140], onde o algoritmo de descoberta de padrões seqüenciais, conhecido como CAPRI, é usado. O CAPRI (“*Clementine A-Priori Intervals*”) é um algoritmo de descoberta de regras de associação que além de descobrir o relacionamento entre os itens, nesse caso páginas Web, também acha a ordem em que foram acessadas. O processo de descoberta possui três fases: a fase *Apriori*, onde os itens freqüentes são descobertos, a fase de *descoberta*, onde árvores de seqüência são construídas, uma para cada ponto de partida possível (primeira página da sequência de navegação) e a fase de "*podagem*", onde as seqüências que se sobrepõe são eliminadas. Por exemplo, a seguinte regra poderia ser produzida: "Se os eventos A, B e C ocorrem nessa ordem, então os eventos D, E e F ocorrem logo em seguida". Apesar do CAPRI escalar bem para grandes volumes de dados, ele necessita da pré-especificação manual de diversos parâmetros de entrada (páginas de partida e fim, topologia da rede, hierarquia de conceitos, etc.), o que torna o processo semi-automático e dependente de um especialista.

Borges e Levene (1999) [13] apresentam uma abordagem estocástica para a descoberta de padrões seqüenciais através de uma gramática de hipertexto probabilística. Através da terminologia da gramática, as páginas Web são representadas por "*símbolos não terminais*", os vínculos entre as páginas como "*regras de produção*" e seqüências de páginas como "*strings*". Um algoritmo de busca em grafos é utilizado com valores pré-determinados para o suporte e confiança de modo a identificar strings, isto é, sessões de navegação de usuários a serem incluídas na gramática. O algoritmo é muito eficiente, mas a sua saída, isto é, o número de regras descobertas é extremamente dependente da escolha dos valores para o suporte e confiança.

O exemplo típico e mais adotado de métodos estocásticos são as cadeias de Markov. Isso se deve à sua adequabilidade para a modelagem de seqüências em processos. Uma das primeiras aplicações a utilizar cadeias de Markov nos dados da Web foi apresentada em Bestravos (1995) [10], onde uma cadeia de Markov escondida de primeira ordem é utilizada para prever a próxima página a ser visitada pelo usuário em determinado período de tempo. Da mesma forma, Sarukkai (2000) [129] utiliza as cadeias de Markov para a modelagem de seqüências de páginas. Aqui uma distribuição de probabilidades é feita sobre os vínculos "precedentes" no histórico de navegação do usuário, isto é, as páginas que foram visitadas no passado, de forma a atribuir pesos a esses vínculos e assim criar melhores previsões para as visitas subseqüentes. Uma abordagem similar é seguida por Zhu (2001) [153], que adicionalmente as outras abordagens explora as entradas correspondentes às “URLs referentes" no arquivo de log, isto é, a página imediatamente anterior à página atual. O método apresentado por Cadez et al. (2000) [18] também utiliza uma mistura de cadeias de Markov de primeira ordem onde diferentes cadeias são usadas para diferentes grupos de usuários de forma a caracterizar e visualizar o comportamento de navegação de vários tipos distintos de usuários.

A principal vantagem da utilização das cadeias de Markov é que elas permitem a geração de seqüências de navegação que podem ser usadas automaticamente para a previsão de visitas subseqüentes, sem necessitar nenhum processamento adicional. Sendo assim, as cadeias de Markov são bastante adequadas para a personalização. Além disso, elas são suportadas por fortes modelos matemáticos,

fornecendo assim uma base teórica bem fundamentada para a sua aplicação. A principal desvantagem é que os modelos de usuários gerados são ilegíveis para os usuários sem treino matemático.

A Tabela 3 apresenta um resumo dos algoritmos de descoberta de padrões seqüenciais discutidos acima. A tabela apresenta as aplicações baseadas na MUW que utilizam esses algoritmos, os métodos de descoberta de padrões seqüenciais empregados e indicações das vantagens e desvantagens de cada abordagem no contexto da personalização na Web.

3.2.3.4 Descoberta de regras de associação

Associações representam relações entre itens de dados que co-ocorrem em determinado conjunto de dados, tal como uma transação de compra ou uma sessão de usuário. Uma associação é geralmente representada por uma regra de associação $A \Rightarrow B$ que implica uma relação de dependência entre dois itens de dados, A e B. A união de A e B é chamada de item freqüente ("*itemset")*. As regras de associação são usadas para estimar a probabilidade de B ocorrer dado que A ocorreu. A seleção de uma regra de associação é baseada em dois critérios: o *suporte*, que é a freqüência com que o item freqüente ($A \cup B$) aparece nos dados, e a *confiança*, que é a probabilidade condicional de que B ocorra dado que A tenha ocorrido e é calculado como a razão (freqüência de $A \cup B$) / (freqüência de A). O algoritmo mais popular para a descoberta de regras de associação é o Apriori [3]. Em [77] é apresentado um tutorial e uma comparação de algoritmos para a descoberta de regras de associação no contexto da mineração de dados.

No contexto da MUW as regras de associação geralmente são usadas para descobrir associações entre as páginas Web que co-ocorrem nas sessões dos usuários [30], [108].

O trabalho de Mobasher et al. (1999) [108] utiliza regras de associação para a recomendação automática de páginas aos usuários. Uma interessante conclusão desse trabalho é que as regras de associação por si só não fornecem a precisão necessária para a geração de recomendações de qualidade [122]. Por essa razão os itens freqüentes

("*itemsets*") descobertos são agrupados, através do algoritmo K-médias, e grupos de transações são gerados. Esses grupos representam grupos de usuários com comportamento de navegação similar. Entretanto, foi constatado que grupos de transação são inapropriados para lidar com dados com muitas dimensões, nesse caso, as dimensões correspondem às páginas armazenadas nos logs de acesso. Por essa razão, páginas Web também são agrupadas em grupos de uso. Grupos de uso consistem em páginas Web que são agrupadas juntas de acordo com a freqüência da sua co-ocorrência em transações de usuários. A construção dos grupos de uso foi feita com a técnica ARHP ("*Association Rule Hypergraph Partition*") [70], que não requer cálculo de distância para a criação dos grupos [108]. Além do mais, o ARHP é eficiente para o agrupamento de dados com muitas dimensões, sem precisar utilizar técnicas de redução de dimensionalidade.

A aplicação de regras de associação para a MUW é bastante limitada, onde o foco primário é a previsão da próxima visita do usuário enquanto navegando na Web. Entretanto, esse tipo de previsão é melhor modelada como um padrão seqüencial, tópico da próxima seção. Essa é uma das principais razões para a limitada quantidade de trabalhos existentes mostrando a utilização de regras de associação na MUW.

3.2.4 Pós-processamento do conhecimento

A maioria dos trabalhos na MUW inclui um estágio de pós-processamento do conhecimento, onde os padrões descobertos são filtrados e analisados por especialistas humanos com o intuito de dar suporte à tomada de decisões. As principais abordagens para esta etapa são:

- *Geração de relatórios*, onde relatórios são gerados contendo resultados de análises estatísticas [149];
- *Visualização,* que é um método mais efetivo para a apresentação compreensiva de informações aos humanos. Essa técnica tem sido adotada pelas ferramentas de visualização WebViz [124] e Footprints [146], entre outras, para a apresentação dos padrões de navegação em forma de grafos;

- *Mecanismos de consulta* que se baseiam no SQL, e são usados para a extração de regras a partir de padrões de navegação. Essa técnica tem sido adotada nos sistemas WebMiner [111], WUM [139] e WebSIFT [30] entre outros.

Algoritmo	Aplicação	Método de descoberta de padrões seqüenciais	Prós	Contras
Spiliopoulou et al. (1999) [138]	Extração de regras seqüenciais [138]	Determinístico	• Escalável • Padrões significativos	• Procedimento semi-automático
CAPRI	Descoberta de padrões temporais de navegação	Determinístico	• Escalável • Padrões significativos	• Requer entrada complexa
Borges & Levene (1999) [13]	Extração de padrões de navegação a partir de sessões de usuários [13]	Estocástico	• Geração automática de caminhos de navegação	• Heurísticas para o refinamento da saída
Modelos Markovianos	Previsão de links [10], [129], [152]	Estocástico	• Fundamentado matematicamente • Geração automática de caminhos de navegação	• Modelos difíceis de interpretar • Alto custo computacional

Tabela 3. Resumo dos algoritmos de descoberta de padrões seqüenciais para a MUW

Uma outra abordagem bastante interessante para o pós-processamento do conhecimento é a integração da MUW ao processo de personalização da Web. Para isso os módulos de personalização responsáveis pela adaptação de um sistema Web são alimentados com o conhecimento descoberto pela MUW.

A funcionalidade oferecida por um sistema de personalização da Web depende primariamente da política de personalização adotada pelo site, isto é, as formas pelas quais as informações podem ser apresentadas aos usuários. Os sistemas de personalização baseados na MUW são tipicamente compostos por dois componentes: um componente on-line e um off-line. O componente on-line é responsável pela aquisição dos

dados de uso e realização da personalização em tempo real e automática através dos padrões descobertos. Todas as outras tarefas relacionadas à mineração de dados são geralmente feitas offline, isto é, os resultados do processo não são fornecidos em tempo real ou sob demanda, mas sim computados periodicamente e disponibilizados durante longos intervalos de tempo. Isso se dá de forma a evitar uma degradação no desempenho do sistema, pois o processo de mineração é muito caro computacionalmente. Essa abordagem para a personalização na Web segue os mesmos princípios que foram propostos por Mobasher et al. (2000) [109], ou seja, a separação do sistema em dois componentes, um on-line e outro off-line. A Figura 3 ilustra como a MUW pode ser integrada a um sistema de personalização seguindo essa abordagem.

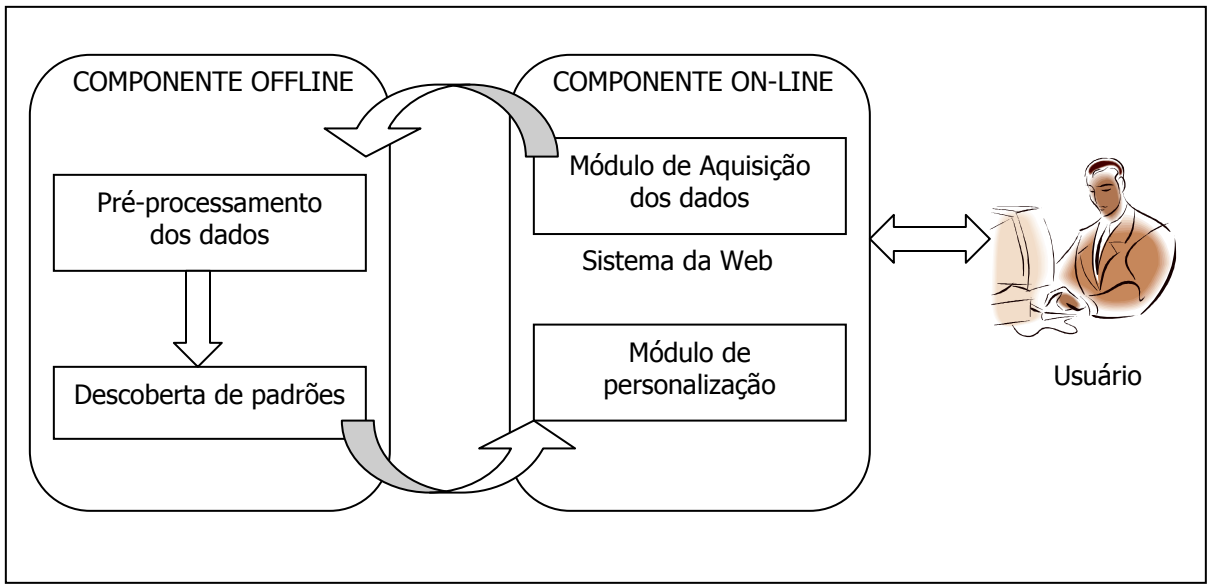

Figura 3. MUW para personalização

3.3 Modelagem de usuários, personalização da Web e a MUW

Um modelo de usuários é uma fonte de informações sobre os aspectos relevantes do usuário para um domínio de aplicação (ex.: informações demográficas, interesses e preferências). Tais modelos representam caracterizações dos usuários enquanto esses interagem com uma aplicação de software, e podem servir tanto para fornecer visualizações sobre essas características como para a adaptação automática de sistemas [90].

A modelagem de usuários é um processo geralmente realizado em três fases: *aquisição*, *representação* e *manutenção* do modelo de usuários [104].

Na *fase de aquisição* são utilizadas técnicas para a aquisição de dados relevantes dos usuários. Isso pode ser feito de forma direta (por exemplo, por meio de formulários) ou indireta (observando o comportamento do usuário, ex.: dados de uso em logs de servidores).

Na *fase de representação* são utilizadas técnicas para a representação formal das informações adquiridas sobre o usuário (ex.: estereótipos, aprendizagem de máquina, lógica, vetores do espaço vetorial e ontologias).

Por último, na *fase de manutenção*, são utilizadas técnicas para a atualização do modelo.

Fazendo um paralelo entre o processo de modelagem de usuários e o processo de MUW, nota-se que a MUW perfaz todas as etapas referentes à modelagem de usuários. A diferença básica está na nomenclatura, visto que tem-se, muitas vezes, nomes diferentes para descrever a mesma coisa em essência. Enquanto o produto final do processo de modelagem de usuários são modelos de usuários, na MUW são os padrões de navegação, sendo que esses padrões são na verdade modelos de usuários representando os interesses e preferências de usuários da Web. Abaixo é descrito sucintamente como as etapas da MUW podem ser vistas sob uma perspectiva voltada à modelagem de usuários.

Na fase de aquisição, podem ser utilizados logs de servidores Web, logs de servidores proxy e as informações geradas na máquina cliente. A aquisição dos dados aqui se constitui como sendo indireta ou implícita, ou seja, as características dos usuários são inferidas implicitamente através de suas interações com os documentos e serviços da Web.

Na fase de representação dos modelos, os dados adquiridos são limpos e formatados de forma que técnicas e algoritmos geralmente provenientes da Aprendizagem de máquina e Análise estatística sejam aplicados. O formato que os padrões/modelos vão assumir vai depender estritamente da abordagem de mineração utilizada. Se, por exemplo, abordagens de agrupamento forem utilizadas, o modelo de

usuários vai corresponder a grupos de usuários com características similares. Mas se, por outro lado, regras de associação forem utilizadas, os modelos serão constituídos de regras que representam a relação entre os itens visitados pelos usuários.

Na fase de manutenção os modelos são atualizados de acordo com os novos acessos dos usuários. O processo da MUW é geralmente executado periodicamente de acordo com algum intervalo de tempo pré-definido. Durante esse período, entre uma execução e outra, os arquivos de uso são atualizados de acordo com os novos acessos dos usuários. Sendo assim, a cada vez que o processo é executado os modelos são atualizados se mantendo assim consistentes com o passar do tempo. Além disso, algumas abordagens incluem informações provenientes de outras fontes que servem para o enriquecimento dos modelos. Em Mobasher et al. (2000) [107], por exemplo, é proposto um framework para personalização baseado na MUW e na mineração de conteúdo, onde os modelos de usuários gerados pela MUW são enriquecidos com os resultados da mineração de conteúdo do site em questão. Oberle et al. [114] propõe o enriquecimento semântico dos logs convencionais através de ontologias. As páginas visitadas pelo usuário são então mapeadas a conceitos presentes na ontologia, e sendo assim, os modelos criados terão uma maior representatividade onde cada página visitada estará associada a uma descrição conceitual. Similarmente, Dai et al. [36] propõe um framework para a incorporação de conhecimento contido em ontologias de domínio como forma de aperfeiçoar a qualidade dos modelos gerados pela MUW. A Tabela 4 sumariza as tarefas da MUW segundo a perspectiva da modelagem de usuários.

Os sistemas de hipermídia adaptativos têm estreita relação com a modelagem de usuários. Esses sistemas tentam antecipar as expectativas dos usuários a partir de modelos representando suas características. O objetivo geral de tais sistemas é prover aos seus usuários conteúdo atualizado, subjetivamente interessante, com informações pertinentes e num tamanho e profundidade adequados. O modelo de usuários então funciona como uma referência para o sistema que busca adaptar seu ambiente a ele. A Figura 4 ilustra os processos envolvidos no desenvolvimento de sistemas adaptativos baseados na modelagem de usuários.

Aquisição de informações dos usuários	Dados de uso da Web (logs de servidores Web, logs de servidores proxy e as informações geradas na máquina cliente)
Representação de modelos de usuários	Padrões de navegação dos usuários (grupos de usuários similares, regras de associação e padrões seqüenciais)
Manutenção de modelos de usuários	Dados de uso da Web
	Integração da MUW com outras categorias de mineração
	Ontologias de domínio

Tabela 4. MUW segundo a modelagem de usuários

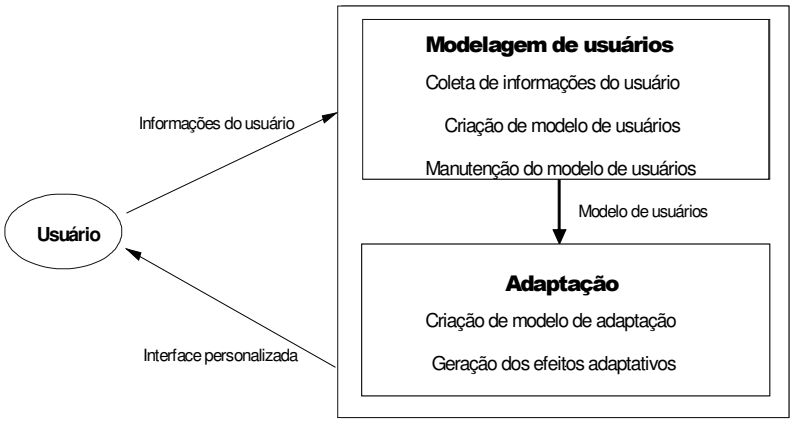

Figura 4. Processos envolvidos no desenvolvimento de sistemas adaptativos [104]

Os sistemas de personalização da Web são uma subárea da hipermídia adaptativa, e podem oferecer uma diversidade de funções, que vão desde simples saudações até funções mais complexas tais como a adaptação de interfaces. Pierrakos et al. [122] propõe um esquema genérico de classificação para as funções de personalização da Web. O esquema proposto leva em consideração o que é atualmente oferecido por sistemas comerciais e protótipos de pesquisa. Dessa forma, foram identificadas quatro classes de funções de personalização: *memorização*, *orientação*, *adaptação* e *suporte à execução de tarefas*, as quais serão sucintamente descritas a seguir.

Memorização. Esta é a mais simples das funções de personalização. O sistema armazena informações sobre o usuário, tais como seu nome e/ou histórico de navegação, para que quando o usuário retorne ao site essas informações possam ser utilizadas como um lembrete do seu comportamento passado. Um exemplo dessa função é a saudação onde o sistema identifica o usuário recorrente e mostra o seu nome na tela.

Orientação. Esta função se preocupa em assistir o usuário na rápida obtenção de informações de interesse, assim como também oferecer rotas alternativas de navegação. Isso alivia a sobrecarga de informações que o usuário pode enfrentar em um determinado site. Um exemplo dessa função é a recomendação de vínculos de hipertexto, onde um conjunto desses vínculos é recomendado de acordo com os interesses do usuário.

Adaptação. A adaptação é responsável por modificar um site da Web em termos de conteúdo, estrutura e formato de forma a refletir as preferências e interesses dos usuários. Um exemplo dessa função seria a personalização do formato de um site em termos de cor, tamanho e estilo da fonte. Esta função geralmente está associada à área conhecida como interfaces adaptativas [16].

Suporte à execução de tarefas. O suporte a execução de tarefas envolve a execução de uma determinada ação em favor do usuário. Das funções de personalização essa é mais avançada e deriva de uma categoria dos sistemas adaptativos conhecida como assistentes pessoais [16]. Como exemplo, podemos citar a personalização de negociações onde o sistema de personalização age como um negociador em favor do usuário (ex.: leilões virtuais).

Mobasher et al. (2000) [107] classifica as técnicas para a personalização na Web de acordo com três abordagens genéricas:

- *Sistemas manuais de regras de decisão*. De acordo com essa abordagem um serviço da Web é personalizado manualmente pelo desenvolvedor geralmente com a cooperação do usuário. Aqui os modelos de usuários são obtidos tipicamente de forma explicita através de um processo de cadastro. Feito isso, regras especificadas manualmente são aplicadas de forma a apresentar conteúdo personalizado de acordo com os diferentes modelos de usuários. Essa abordagem é amplamente utilizada em sistemas tutores inteligentes onde o conteúdo é apresentado de acordo com o nível de conhecimento de cada usuário [7]. Dois produtos que também utilizam essa

abordagem são: o motor de personalização do site Yahoo! [101] e o *Websphere Personalization* da IBM [81];

- *Sistemas de filtragem baseados no conteúdo*. Esse grupo de técnicas aplica métodos de Aprendizagem de máquina ao conteúdo da Web, primariamente texto, de forma a descobrir as preferências pessoais de usuários. Aqui o usuário geralmente fornece explicitamente os seus interesses de informação e o sistema, através da análise de conteúdo (textual), se encarrega de entregar documentos cujo conteúdo corresponda aos interesses especificados anteriormente. Um exemplo de ferramenta que utiliza essa abordagem é apresentado em Lang (1995) [96];
- *Sistemas de filtragem colaborativa*. O objetivo dessa abordagem é a personalização de serviços sem a necessidade da análise de conteúdo da Web. A personalização é alcançada através da identificação de características comuns entre os interesses ou preferências de diferentes usuários. Aqui o usuário geralmente expressa o interesse em uma página através de uma avaliação explícita, como por exemplo, atribuindo uma nota em uma escala de 0 a 10, onde quanto maior for a nota dada maior o interesse desse usuário na página. A grande vantagem da MUW quando utilizada para a personalização colaborativa é que os interesses são inferidos implicitamente através do comportamento de navegação do usuário, deixando o usuário livre da dificuldade de ter que avaliar cada página visitada. A loja virtual *amazon.com* [1] é o exemplo mais conhecido da utilização dessa abordagem, onde produtos são recomendados aos usuários baseado na compra de outros usuários com interesses similares.

3.4 Considerações finais do capítulo

Este capítulo apresentou os principais aspectos da MUW assim como o seu processo geral e as etapas em que se subdivide. Para cada etapa foram discutidos os principais problemas e algumas das soluções propostas pela literatura. Também foi

realizado um paralelo entre o processo da modelagem de usuários e o processo da MUW e como esses estão inseridos no contexto da personalização.

A MUW é uma tecnologia emergente que ajuda a compreender as motivações dos usuários enquanto navegando pela Web. Para isso, a MUW descobre padrões ou modelos que caracterizam os usuários segundo a sua navegação. De posse desses modelos, e de ferramentas de visualização e análise, especialistas humanos podem saber como os usuários se comportam enquanto navegando em seus sites e dessa forma aperfeiçoar os serviços oferecidos. De outra forma, esses modelos podem ser incorporados diretamente a módulos de personalização para o fornecimento de serviços automáticos de personalização.

A MUW é uma área de pesquisa bastante ativa, onde novas abordagens aparecem regularmente. Apesar disso, a MUW ainda está longe de ser considerada uma área de pesquisa madura, isso se deve aos diversos problemas técnicos ainda não resolvidos e a outras questões que continuam em aberto, algumas das quais foram apresentadas nesse capítulo. Os maiores problemas estão nas etapas de coleta e pré-processamento dos dados, onde urge novas técnicas e métodos, pois sempre que os métodos existentes resolvem um problema deixam outros tantos sem resolução. Outra questão bastante séria associada à coleta dos dados de uso está relacionada à privacidade. Uma pesquisa realizada pelo KDnuggets [79] (15/03/2000 – 30/03/2000) revelou que aproximadamente 70% das pessoas entrevistadas consideram a mineração na Web uma ameaça à privacidade. Dessa forma, é imperativo que novas ferramentas de mineração na Web sejam transparentes ao usuário, provendo acesso aos dados coletados e esclarecendo o uso desses dados, assim como os potenciais benefícios ao usuário.

Todo o estudo realizado sobre a MUW serviu então como pré-requisito para a construção do framework proposto no presente trabalho, o qual será apresentado no decorrer da dissertação.

4 ENGENHARIA DE DOMÍNIO MULTIAGENTE

No contexto da Engenharia de software, diferentes abordagens buscam melhorar a qualidade dos artefatos de software, bem como diminuir o tempo e o esforço necessários para produzi-los. Os frameworks, por exemplo, são soluções computacionais incompletas que, estendidas, permitem produzir diferentes artefatos de software. A grande vantagem dessa abordagem é a promoção do reuso de código e projeto. O reuso de componentes previamente desenvolvidos, como no caso dos frameworks, permite a redução de tempo e esforço para a obtenção de um software. Dessa forma, a reutilização de software permite, principalmente, a geração de aplicações de software com três características muito importantes: qualidade, baixo custo e rapidez no desenvolvimento.

Duas abordagens complementares caracterizam o reuso de software: a Engenharia de domínio (ou o desenvolvimento PARA o reuso) e a Engenharia de aplicações (ou o desenvolvimento COM o reuso) [63].

A Engenharia de domínio aborda a construção de abstrações de software reutilizáveis. Uma abstração de software é uma especificação de alto nível que se corresponde com uma ou várias realizações num nível maior de detalhamento. Essas abstrações são usadas na criação de aplicações específicas na Engenharia de aplicações.

A abordagem multiagente tem se mostrado particularmente apropriada para representar problemas do mundo real (ex.: MUW). Porém, a efetividade do desenvolvimento de sistemas (*com* reutilização) segundo o paradigma dos agentes dependerá de uma adequada integração das abstrações geradas na Engenharia de domínio, bem como de mecanismos apropriados para mapear especificações em realizações, seja através de mecanismos de composição ou geração automática [64].

A sistematização de técnicas e metodologias para a Engenharia de aplicações baseada em agentes é um tópico de ativa pesquisa. De forma similar ao que aconteceu com o desenvolvimento orientado a objetos, uma grande quantidade de técnicas e metodologias têm sido proposta para a Engenharia de aplicações baseadas em agentes e

ainda não há um padrão reconhecido que reúna as vantagens de cada uma delas [19], [32], [65], [116], [147], [148]. Por outro lado, é notória a escassez de técnicas e metodologias consolidadas para a Engenharia de domínio multiagente. O grupo de pesquisa GESEC [55], no qual o presente trabalho está inserido, tem desenvolvido uma metodologia baseada em ontologias para a análise e projeto de domínio multiagente, a qual foi usada e avaliada para a construção do framework aqui proposto.

Este capítulo está organizado da seguinte forma. Na seção 4.1 são apresentados os aspectos gerais da Engenharia de domínio. Na seção 4.2 são apresentados conceitos gerais sobre agentes, sistemas multiagente (SMA) e como estes se inserem no contexto da Engenharia de domínio. Na seção 4.3 é apresentada a metodologia escolhida para a construção do framework que é proposto no presente trabalho. Finalmente, na seção 4.4 é introduzido o ONTOMUW, um framework multiagente para a modelagem de usuários baseado na MUW, principal objetivo deste trabalho.

4.1 Aspectos gerais

Um dos grandes problemas associados à reutilização de software está justamente em como criar componentes reutilizáveis. Uma disciplina adequada para abordar esse problema é a Engenharia de domínio [4], [6], [5], [41], [75], [100].

A Engenharia de domínio tem como principal objetivo disponibilizar artefatos reutilizáveis, que possam ser empregados no desenvolvimento de novas aplicações.

A Análise de domínio busca a identificação de domínios e, dentro desses domínios, a captura de oportunidades de reutilização, de requisitos comuns e de variações em uma família de aplicações, tendo como produto um Modelo de Domínio.

O Projeto de domínio busca a especificação de uma solução ao problema determinado na modelagem de domínio, resultando em um ou mais frameworks e, possivelmente, em uma coleção de padrões de projeto que documentam soluções bem sucedidas ao problema tratado.

Na Implementação do domínio, por fim, são produzidas a partir dos resultados das fases anteriores, dependendo da abordagem, componentes reusáveis (se composicional) ou linguagens específicas de domínio e geradores de código (se gerativa).

A Figura 5 ilustra as três etapas da Engenharia de domínio [145].

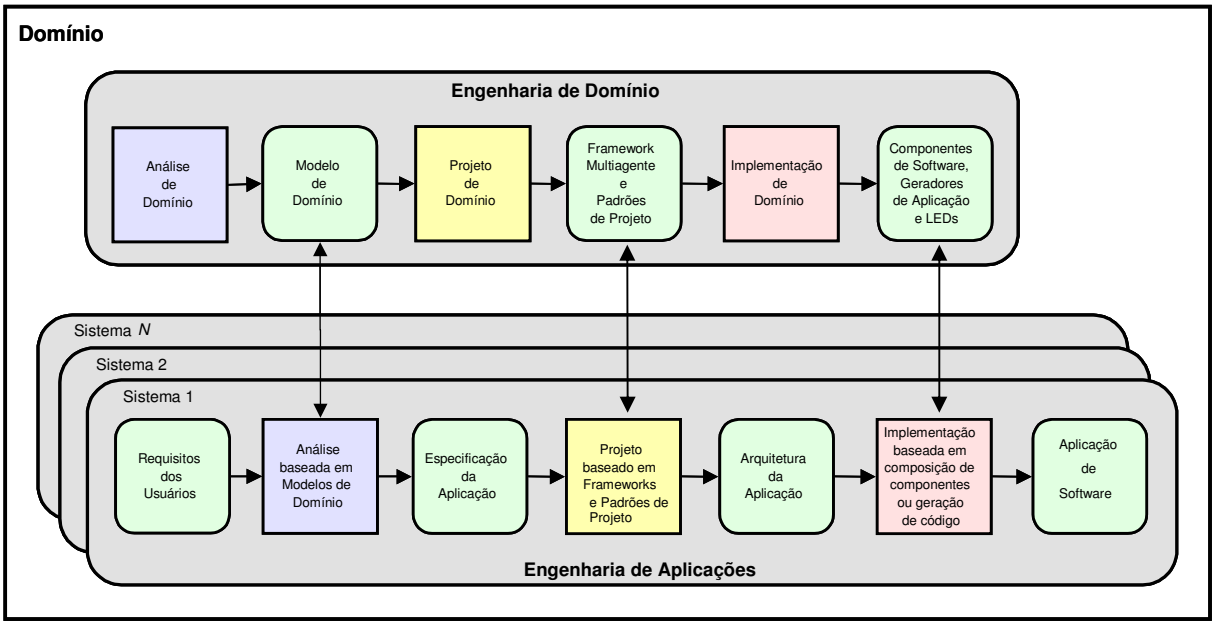

Figura 5. As fases da Engenharia de domínio e da Engenharia de aplicações

Os produtos resultantes de cada etapa da Engenharia de domínio são as também denominadas abstrações de software reutilizáveis. Uma abstração de software é uma especificação de alto nível que descreve o que o artefato de software faz, eliminando os detalhes irrelevantes e enfatizando as informações que, nesse nível, sejam importantes para o usuário da abstração [41], [66].

4.2 Agentes, SMAs e a Engenharia de domínio multiagente

Um agente é uma entidade autônoma que percebe seu ambiente através de sensores e age sobre o mesmo através de atuadores. Nos agentes humanos, por exemplo, os olhos, ouvidos e a língua são sensores enquanto que as mãos e a boca são geralmente os atuadores. A Figura 6 ilustra as interações de um agente genérico com seu ambiente [128].

A autonomia é a característica principal dos agentes. Os agentes são capazes de atuar sem intervenção humana ou de outros sistemas através do controle que eles

possuem do seu estado interno e do seu comportamento. Essa é a principal característica que distingue agentes de objetos.

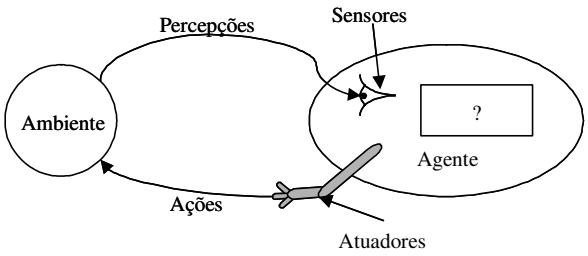

Figura 6. As interações de um agente genérico com seu ambiente [128]

Um objeto encapsula estado e comportamento, mais não pode pela sua iniciativa própria ativar o seu comportamento. Um objeto pode invocar métodos publicamente acessíveis de outros objetos. Uma vez que um método é invocado, as ações correspondentes são executadas. Neste sentido, os objetos não são autônomos porque são totalmente dependentes uns dos outros para a execução de suas ações.

Os agentes estão continuamente ativos, na execução de um ciclo de observar seu ambiente, atualizar seu estado interno e selecionar uma ação para realizar. Em contraste, um objeto é passivo a maioria do tempo, só ficando ativo quando outro objeto requer seu serviço invocando um dos seus métodos.

Um sistema multiagente (SMA) pode ser caracterizado como um grupo de agentes que atuam em conjunto no sentido de resolver problemas que estão além das suas habilidades individuais. Os agentes realizam interações entre eles de modo cooperativo para atingir uma meta.

A construção de software de alta qualidade não é tarefa simples, principalmente devido à complexidade natural do software, caracterizada pelas interações necessárias entre seus componentes. Na procura de técnicas apropriadas para abordar a complexidade do software, os paradigmas de desenvolvimento têm evoluído do paradigma estruturado, passando pela orientação a objetos e chegando então no paradigma de desenvolvimento de software baseado em agentes (Figura 7) [59].

A Engenharia de software geralmente confronta a complexidade do mundo real procurando mecanismos de abstração apropriados para mapear o mundo real ao mundo computacional, tentando reduzir a distância cognitiva entre estes dois mundos.

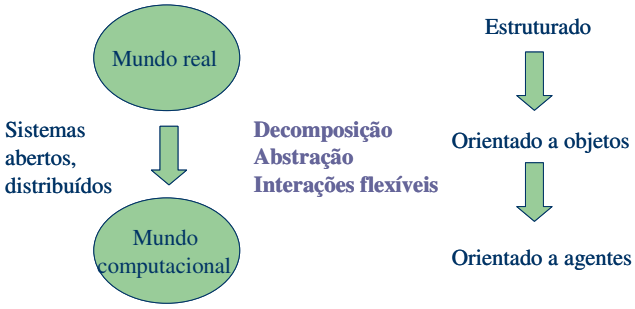

Figura 7. A evolução dos paradigmas de desenvolvimento para abordar a complexidade do software

Um agente é uma abstração de software efetiva que fornece uma boa metáfora para a modelagem de sistemas complexos devido, principalmente, à forte correspondência entre a noção de subsistema e de sociedade multiagente. Ambos envolvem um conjunto de componentes agindo e interagindo de acordo com seus papéis.

A interação entre dois componentes de software provavelmente é a característica mais importante de um software complexo. Por exemplo, em sistemas abertos é impossível saber exatamente, na fase de projeto, quais os componentes de um sistema e quais as interações entre esses componentes. Nestes sistemas são necessários componentes com comportamento autônomo e flexível.

O paradigma de agentes simplifica o desenvolvimento de sistemas complexos porque os agentes podem decidir sobre o tipo e escopo das suas interações em tempo de execução. Os agentes são capazes de iniciar uma interação e responder a outros agentes de maneira autônoma e flexível.

A autonomia permite o projeto de agentes capazes de atender requisitos não preestabelecidos. Portanto, a abordagem dos SMA é adequada para a modelagem de sistemas abertos, porque os agentes são capazes de reagir a eventos não previstos, explorando as oportunidades que surgem e realizando dinamicamente acordos com outros agentes cuja presença pode não ter sido prevista no projeto do SMA [59].

Além da complexidade das aplicações, os problemas de produtividade no desenvolvimento bem como de qualidade e manutenção dos produtos de software continuam sendo um desafio para a Engenharia de software. A reutilização de software tem apresentado boas soluções para abordar esses problemas [68]. Por isso, a Engenharia de software baseada em agentes deveria aproveitar os avanços bem como as lições aprendidas na teoria e prática da reutilização de software, em particular, a necessidade de dispor de abstrações de software efetivas para a reutilização [59] .

O grupo de pesquisa GESEC [55] tem desenvolvido um processo e uma metodologia para a Engenharia de domínio multiagente e trabalha na construção de um ambiente de desenvolvimento de software composto por um conjunto de ferramentas e bibliotecas de abstrações de software de alto nível para o desenvolvimento de SMAs.

A Figura 8 mostra o modelo proposto para a Engenharia de domínio multiagente, suas fases (Análise de domínio, Projeto de domínio e Implementação de domínio) e os produtos gerados em cada fase de acordo com o seu nível de abstração (do abstrato para concreto) e o seu nível de dependência do domínio da aplicação (do dependente do domínio para o independente do domínio).

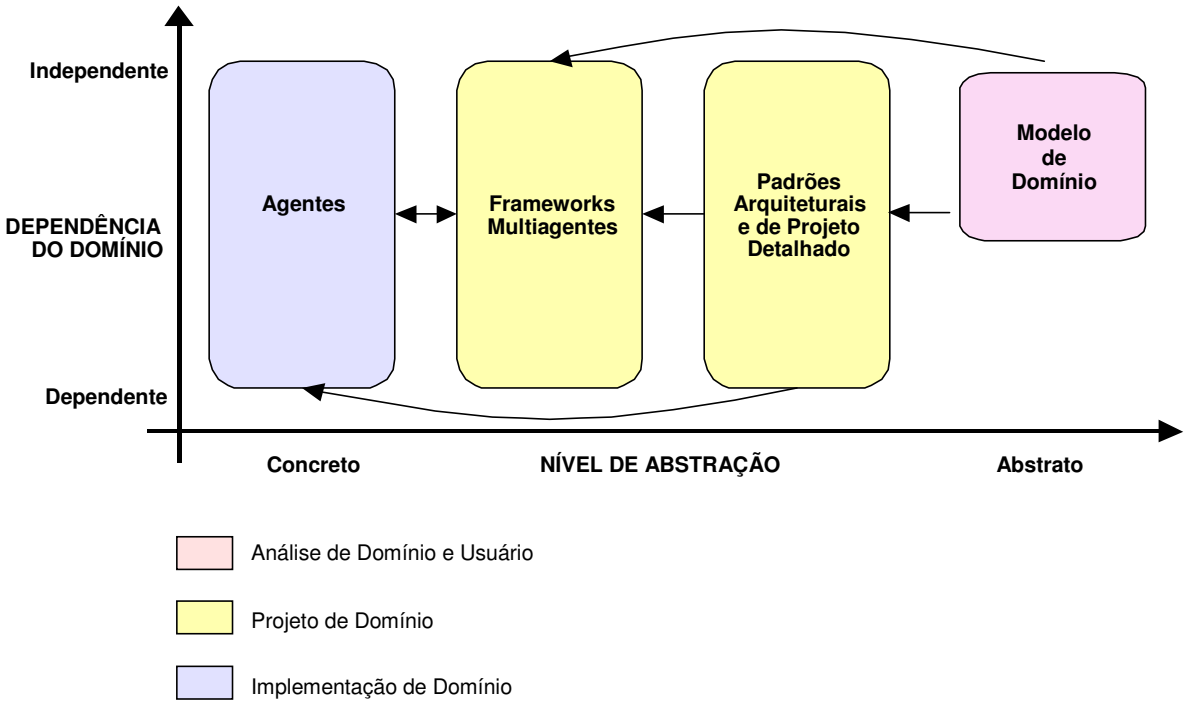

Figura 8. Fases e produtos da Engenharia de domínio multiagente

O modelo de domínio - independente do domínio da aplicação é especificado em um alto nível de abstração - representa a formulação de um problema, conhecimento e atividades do mundo real. A formulação é suficientemente genérica para representar uma família de sistemas.

O projeto de domínio na Engenharia de domínio multiagente produz uma solução computacional multiagente, reutilizável no desenvolvimento de uma família de aplicações similares em um determinado domínio. O produto desta fase consiste de um conjunto de padrões arquiteturais, padrões detalhados e frameworks multiagente.

A fase de implementação de domínio implementa os agentes componentes dos frameworks multiagente definidos no projeto de domínio.

O conhecimento das técnicas utilizadas em cada fase bem como os produtos gerados em cada uma delas é representado através de ontologias.

Uma ontologia é a representação do vocabulário de um domínio. Mais precisamente, não é simplesmente o vocabulário como tal o que qualifica a ontologia, mas os conceitos que os termos do vocabulário pretendem capturar [24], [59], [69].

As ontologias de domínio são descrições formais de classes de conceitos e relacionamentos entre esses conceitos que descrevem um domínio de aplicação [51]. Estruturas de representação de conhecimento baseadas em frames [128] são geralmente utilizadas para representar ontologias de domínio, onde os conceitos são representados por frames e os relacionamentos entre conceitos como slots dos frames.

As ontologias são particularmente úteis na representação de abstrações de software reutilizáveis de alto nível como modelos de domínio e frameworks. Elas fornecem uma terminologia não ambígua que pode ser compartilhada por todos os atores envolvidos em um processo de desenvolvimento. Por outro lado, uma ontologia pode ser generalizada, tanto quanto necessário, facilitando assim sua reutilização e adaptação.

A metodologia MADEM *("Multi-Agent Domain Engineering Methodology"*) [43] constitui o primeiro esforço no sentido de realizar o modelo proposto pelo grupo GESEC para a Engenharia de domínio multiagente. Essa metodologia foi escolhida para o desenvolvimento do presente trabalho, pois compreende técnicas que guiam tanto a fase

de análise quanto de projeto da Engenharia de domínio multiagente. A próxima subseção é dedicada à metodologia MADEM.

4.3 MADEM: Uma metodologia para a Engenharia de domínio multiagente

A MADEM é uma metodologia para a Engenharia de domínio multiagente que surgiu a partir da integração dos resultados de duas teses de mestrado desenvolvidas no contexto das atividades de pesquisa do grupo GESEC. A primeira delas descreve a GRAMO (*"Generic Requirement Analysis Method based on Ontologies"*) [40], [42], [58], [60] a qual guia a captura e especificação de requisitos de uma família de sistemas em um domínio de aplicação; e a outra descreve a DDEMAS (*"Domain Design for Multi-Agent Systems"*) [44], [45], [62] que guia a captura e especificação do projeto reutilizável para uma família de sistemas em um domínio de aplicação. O framework multiagente é o produto final gerado. Portanto, as técnicas GRAMO e DDEMAS, com algumas revisões e extensões, compõem a MADEM.

A MADEM define um roteiro para a realização das fases de análise e projeto de domínio para a Engenharia de Domínio Multiagente (Figura 9).

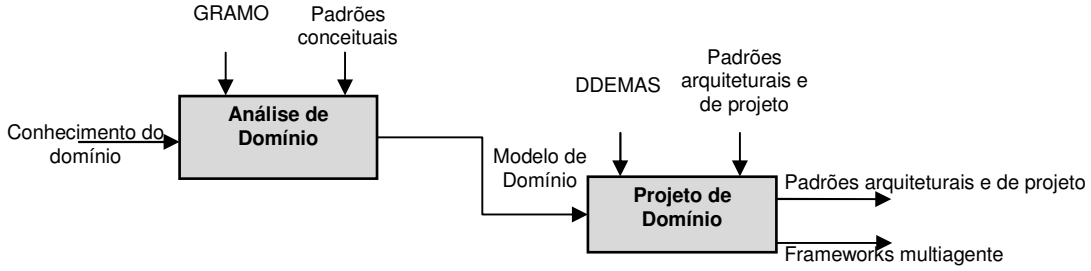

Figura 9. Os insumos e os produtos da MADEM

A Tabela 5 apresenta as fases, tarefas, atividades e produtos da metodologia MADEM.

A fase de análise de domínio é iniciada quando se identifica o que será modelado, que pode ser a formulação de um problema ou uma área de conhecimento. A fase de Análise de domínio consiste da tarefa de Modelagem de domínio. Quando a formulação de um problema é modelada são executadas as tarefas de modelagem de objetivos, de papéis, de variabilidades e de interações dos papéis. O produto desta modelagem é o modelo de domínio composto dos modelos de objetivos, papéis e

interações dos papéis. Quando uma área de conhecimento é modelada são executadas as tarefas de modelagem de conceitos e de variabilidades. O produto desta modelagem é um modelo de domínio composto do modelo de conceitos.

Fases	Tarefas	Subtarefas	Subprodutos	Produtos
Análise de Domínio	Modelagem de Domínio	Modelagem de Objetivos	Modelo de Objetivos	Modelo de Domínio
		Modelagem de Papéis	Modelo de Papéis	
		Modelagem de Variabilidades	Modelo de Objetivos e Papéis refinado	
		Modelagem de Interações dos Papéis	Modelo de Interações dos Papéis	
		Modelagem de Conceitos	Modelo de Conceitos	
Projeto de Domínio	Projeto Arquitetural	Reuso de padrões arquiteturais	Modelo arquitetural	Framework
		Modelagem de Agentes	Modelo de Agentes	
		Modelagem do Framework	Modelo de Projeto do Framework	
		Modelagem de atividades	Modelo de atividades	
	Projeto Detalhado	Refinamento dos Agentes	Modelo de Agentes Refinado	
		Modelagem de comportamento	Modelo de comportamento	
		Modelagem de interações dos agentes	Modelo de interações dos agentes	

Tabela 5. Fases, tarefas, atividades e produtos da MADEM

A fase de projeto de domínio tem como entrada modelos de domínio e sistema de padrões. Esta fase consiste de duas tarefas o projeto arquitetural e o projeto detalhado. O produto final desta fase é um framework multiagente.

A ONTOMADEM [43] é uma ontologia que representa o conhecimento da MADEM, ou seja, representa o conhecimento de metodologias tanto de análise quanto de projeto de domínio segundo o paradigma computacional dos agentes. Essa ontologia foi construída utilizando-se o editor de ontologias Protégé [125]. Esse editor provê várias facilidades para a criação de instâncias através de formulários gráficos personalizados. Dessa forma, a ONTOMADEM através do Protégé, acaba se tornando uma ferramenta intuitiva que guia o processo tanto de análise e projeto de domínio, e a criação dos modelos se resume à instanciação das classes através do preenchimento de formulários.

4.4 ONTOMUW: Um Framework Multiagente para Modelagem de Usuários baseado na Mineração de Uso

ONTOMUW é o nome dado ao framework proposto nesse trabalho. O prefixo “onto” deriva do fato de tanto o modelo de domínio quanto a especificação do projeto terem sido construídos através de instâncias de conceitos da ontologia ONTOMADEM. Sendo assim, não só os agentes finais implementados são passíveis de reutilização, mas todos os modelos construídos no decorrer da modelagem do framework também podem ser facilmente compreendidos, por estarem representados por meios de uma ontologia, e reutilizados de acordo com o nível de abstração desejado.

A idéia de utilizar a tecnologia dos agentes para a modelagem de usuários e sistemas adaptativos não é nova. Os sistemas tutores inteligentes, por exemplo, há anos vem utilizando essas idéias. Na arquitetura desses sistemas quase sempre há agentes responsáveis por construir e manter modelos de usuários e modelos de adaptação [33], [50], [84], [127], [130].

A modelagem de usuários envolve as tarefas de aquisição e representação de modelos de usuários (capacidade de aprendizagem), inferência de novas suposições sobre os usuários com base nas iniciais (autonomia e raciocínio) e disponibilização dos modelos de usuários a outros componentes do sistema (sociabilidade). Essas características são inerentes ao paradigma de agentes, daí a sua particular adequabilidade para o processo de modelagem de usuários.

Lorenz et al. (2003) [99] define duas abordagens para a modelagem de usuários baseada em agentes. Na primeira os agentes se conectam a modelos de usuários externos; e na segunda os usuários são modelados como agentes distintos trabalhando juntos em um SMA.

Na segunda abordagem, “usuário=agente”, o modelo de usuários é mapeado como o conhecimento interno do agente. Sendo assim, o conjunto das possíveis ações do usuário ativo irá refletir o próprio conjunto de ações do agente. Dessa forma, o agente acaba por representar uma metáfora computacional do usuário ativo. Além disso, sendo o agente uma entidade ativa que continuamente monitora o seu ambiente, qualquer

mudança nos interesses ou perfil do usuário pode ser rapidamente detectada e processada pelo agente. Por todos esses aspectos, essa abordagem foi escolhida como base para a modelagem de usuários ativos no ONTOMUW, ou seja, cada usuário em particular será representado por um agente específico.

A idéia de utilizar a tecnologia dos agentes para a personalização na Web através da MUW também não é nova. O sistema ROSA (*"Multi-agent System for Web Services Personalization"*), por exemplo, implementa uma sociedade de agentes responsáveis por atividades de mineração e personalização automática da Web. Na arquitetura do ROSA, similarmente ao framework proposto por Mobasher et al. (2000) [107] (vide Figura 3), os agentes são divididos em duas camadas. Na camada off-line estão distribuídos os agentes responsáveis pelas tarefas de pré-processamento e mineração e na camada on-line estão os agentes responsáveis pelas atividades que devem ser executadas em tempo real, tais como o monitoramento da sessão de navegação do usuário corrente, classificação do usuário corrente e criação e entrega da lista de recomendações ao usuário.

A idéia chave do deste trabalho não é criar uma aplicação específica para a personalização baseada na MUW, similarmente ao ROSA, mas sim um conjunto de abstrações de software que possam ser reutilizadas por famílias de aplicações que desejem oferecer serviços personalizados aos seus usuários através da MUW e do paradigma dos SMA.

Os próximos capítulos detalharão a construção do ONTOMUW, desde sua concepção na análise de domínio, passando pelo projeto e culminando com a implementação física dos agentes que compõem o framework.

4.5 Considerações finais do capítulo

Nesse capítulo foram apresentados os aspectos gerais da Engenharia de domínio e da Engenharia de domínio multiagente.

O paradigma de desenvolvimento baseado em agentes constitui uma evolução natural aos paradigmas existentes, pois a distância cognitiva, ou seja, o esforço de

mapear o mundo real ao mundo computacional é reduzido. Consequentemente a complexidade do software também é reduzida.

Muitas técnicas e metodologias têm sido propostas para a Engenharia de aplicações baseadas em agentes. Ainda não existe consenso sobre qual a melhor ou mais adequada. É sabido que a reutilização de software é uma das melhores formas de aumentar a produtividade e diminuir o custo de desenvolvimento de software. Sendo assim, o paradigma computacional dos agentes deveria, também, prover meios de lidar de forma efetiva com a reutilização. Pensando nisso o grupo de pesquisa GESEC, no qual o presente trabalho está inserido, tem trabalhado em uma metodologia e um processo baseado em ontologias para a Engenharia de domínio multiagente. Essa proposta culminou com a MADEM. A MADEM fornece técnicas de análise e projeto de domínio que guiam os desenvolvedores na árdua tarefa de construir frameworks multiagente. Apesar do produto final ser o framework multiagente, todas as abstrações geradas nas etapas intermediárias são passíveis de reutilização de acordo com o nível de abstração a que pertencem.

A potencialidade dos agentes para a modelagem de sistemas complexos há muito vem sendo explorada. Dentro desse contexto a modelagem de usuários aparece como uma área de aplicação particularmente interessante. Agentes que representam usuários reais, e que realizam ações baseado em ações reais do usuário representam uma metáfora bastante interessante, pois devido ao seu comportamento autônomo o agente pode perceber e rapidamente processar as mudanças no perfil ou interesses do usuário. Dentre os diversos métodos para a modelagem de usuários na Web a MUW aparece como um dos mais interessantes, pois a modelagem do usuário é feita de forma implícita, ou seja, o usuário não precisa informar explicitamente seus interesses ao sistema.

No desenvolvimento do ONTOMUW a proposta é então utilizar a tecnologia dos agentes em combinação com a MUW para a modelagem de usuários e personalização da Web. Mas, mais do que uma simples aplicação, propõe-se fornecer um conjunto de abstrações reutilizáveis para desenvolvedores que desejem construir aplicações específicas para a personalização na Web baseada em agentes e na MUW.

5 ANÁLISE DE DOMÍNIO DO ONTOMUW

No processo de desenvolvimento de software, a fase de análise de requisitos visa definir o que o sistema deve fazer e, de acordo com o paradigma de desenvolvimento utilizado, especificar um modelo com a representação dos requisitos do sistema.

Os métodos para a análise de requisitos de sistemas multiagente geralmente focalizam a modelagem de objetivos, papéis, atividades e interações entre indivíduos de uma organização. Apesar de ainda não existir uma definição comum desses conceitos, os significados seguintes são geralmente atribuídos aos mesmos.

Uma organização está composta de indivíduos com objetivos gerais e específicos que estabelecem o que a organização pretende alcançar. O alcance de um ou mais objetivos específicos permite alcançar o objetivo geral da organização. Por exemplo, um sistema de informação pode ter o objetivo geral de “satisfazer as necessidades de informação de uma organização” e os objetivos específicos de “satisfazer as necessidades de informação dinâmica” e “satisfazer as necessidades de informação em longo prazo” [61].

Os objetivos específicos são alcançados através do exercício de responsabilidades. Os indivíduos desempenham papéis com um certo grau de autonomia e exercitam suas responsabilidades através da execução de atividades. Para isso, eles dispõem de recursos. Por exemplo, um indivíduo pode desempenhar o papel de “recuperador de informação” com a responsabilidade de executar atividades para satisfazer as necessidades de informação dinâmicas de uma organização. Outro indivíduo pode desempenhar o papel de “filtrador de informação” com a responsabilidade de executar atividades para satisfazer as necessidades de informação a longo prazo da organização. Recursos podem ser, por exemplo, as regras da organização para acessar e estruturar suas fontes de informação [60].

Às vezes, os indivíduos têm que comunicar-se com outros indivíduos internos ou externos para cooperar na execução de uma atividade. Por exemplo, os indivíduos que fazem os papéis de “recuperador de informação” e “filtrador de informação” podem

precisar interagir com o indivíduo “gerenciador de fontes de informação” que tem a responsabilidade de armazenar e atualizar as fontes de informação da organização [59].

A fase de Análise de domínio consiste da tarefa de modelagem de domínio. Nessa fase são executadas as subtarefas de modelagem de objetivos, de papéis, de variabilidades e de interações dos papéis. O produto desta modelagem é o modelo de domínio composto dos modelos de objetivos, papéis e interações dos papéis.

Este capítulo apresenta o modelo de domínio resultante da aplicação das técnicas de análise de domínio da MADEM. O capítulo tem a seguinte organização. Na seção 5.1 é mostrada a construção do modelo de domínio através das subtarefas definidas pela MADEM. Na seção 5.2 são apresentados os requisitos não funcionais do framework. E finalmente na seção 5.3 são apresentadas as considerações finais do capítulo.

5.1 Construindo o modelo de domínio

A tarefa de modelagem de domínio definida pela MADEM contempla as seguintes subtarefas: modelagem de objetivos, modelagem de papéis, modelagem de interações entre os papéis e modelagem de variabilidades. Os produtos associados a essas subtarefas são: modelo de objetivos, modelo de papéis e modelo de interações entre papéis. As subseções a seguir detalham cada uma dessas subtarefas.

5.1.1 Modelagem de objetivos

No modelo de objetivos são identificados o objetivo geral, e os objetivos específicos do domínio a ser modelado. Um objetivo geral está associado a um problema geral. Aqui o problema geral é como oferecer serviços personalizados na Web através da MUW. A partir da definição desse problema deriva-se o objetivo geral, que é justamente *Oferecer serviços personalizados na Web baseados na MUW*.

Segundo a MADEM os objetivos específicos podem ser pensados como subproblemas, que uma vez resolvidos podem conduzir à resolução do problema geral. Um sistema adaptativo possui dois grandes subproblemas. O primeiro é como modelar os usuários e o segundo como realizar a adaptação. Daí deriva-se dois objetivos específicos: *Modelagem de usuários* e *Modelagem de adaptação*.

A MADEM também especifica que cada objetivo específico é alcançado através de uma ou mais responsabilidades. Portanto, identificou-se que o objetivo *Modelagem de usuários* é alcançado através do exercício das seguintes responsabilidades: *monitoramento do usuário, modelagem de usuário corrente, manutenção de dados de uso e MUW.* Já o objetivo específico *Modelagem de adaptação* é alcançado através das responsabilidades: *modelagem de usuário corrente*, *classificação do usuário corrente*, *criação de modelos de adaptação e adaptação da interface.* A seguir é apresentada uma breve descrição de cada uma dessas responsabilidades.

Monitoramento do usuário. O monitoramento do usuário, como o próprio nome já diz, realiza o monitoramento on-line das atividades de navegação do usuário corrente. Isso se dá através da captura das informações contidas na interação do usuário com a interface, nesse caso o navegador Web. Essas informações serão utilizadas para a criação e atualização de um modelo representando a sessão de navegação do usuário corrente. Sendo assim, como pode ser visto na Figura 10[6] essa responsabilidade alcança somente o objetivo específico *Modelagem de usuários*.

Modelagem de usuário corrente. Uma vez as informações de navegação do usuário corrente capturadas, elas devem ser processadas em um modelo representando a sessão de navegação do usuário corrente. Essa responsabilidade alcança os dois objetivos específicos, pois é através desse modelo que o sistema poderá criar o modelo de adaptação.

Manutenção dos dados de uso. Essa responsabilidade diz respeito a mecanismos que garantam a consistência dos dados de uso. Na verdade a grande maioria das aplicações para personalização na Web baseadas na MUW não possuem mecanismos responsáveis por essa tarefa. O que acontece na grande maioria dos casos é um oneroso processo de pré-processamento, limpeza e transformação de arquivos de log brutos a cada vez que o processo de mineração é executado. Tendo isso em vista, sentiu-se a necessidade de módulos separados que cuidem da aquisição e tratamento dos dados de uso, aliviando assim a etapa de pré-processamento desses dados.

[6] Cada uma das figuras representando os modelos deste capítulo foi gerada pela ferramenta ONTOMADEM

MUW. Essa responsabilidade diz respeito à descoberta dos padrões de navegação dos usuários, ou seja, modelos representando o comportamento passado de navegação de usuários. Esses modelos servirão como insumo para a personalização ou adaptação, daí o motivo dessa responsabilidade também alcançar os dois objetivos específicos como mostra a Figura 10.

Classificação do usuário corrente. A forma mais comum e mais usada de personalização baseada na MUW é a baseada na filtragem colaborativa. Nessa abordagem o mecanismo de adaptação é baseado na similaridade entre o perfil do usuário corrente e grupos de usuários com perfis similares entre si. Para efeitos de personalização, como a realização de recomendações automáticas, por exemplo, o perfil do usuário corrente é classificado, em tempo real, em algum dos grupos de usuários existentes o qual ele seja mais similar. As recomendações então são feitas baseadas nas páginas vistas pelos membros do grupo ao qual o usuário foi classificado.

Criação de modelos de adaptação. Essa responsabilidade está associada a mecanismos capazes de computar e modelar os efeitos de personalização a serem aplicados na interface do usuário. Um modelo de adaptação contém as regras que ditam como as adaptações devem ser aplicadas à interface do usuário.

Adaptação da interface. Uma vez os modelos de adaptação criados, é necessário a efetiva apresentação dessas adaptações ao usuário. Esse é o propósito dessa responsabilidade, adaptar a interface do usuário de modo a mostrar as personalizações computadas para o mesmo.

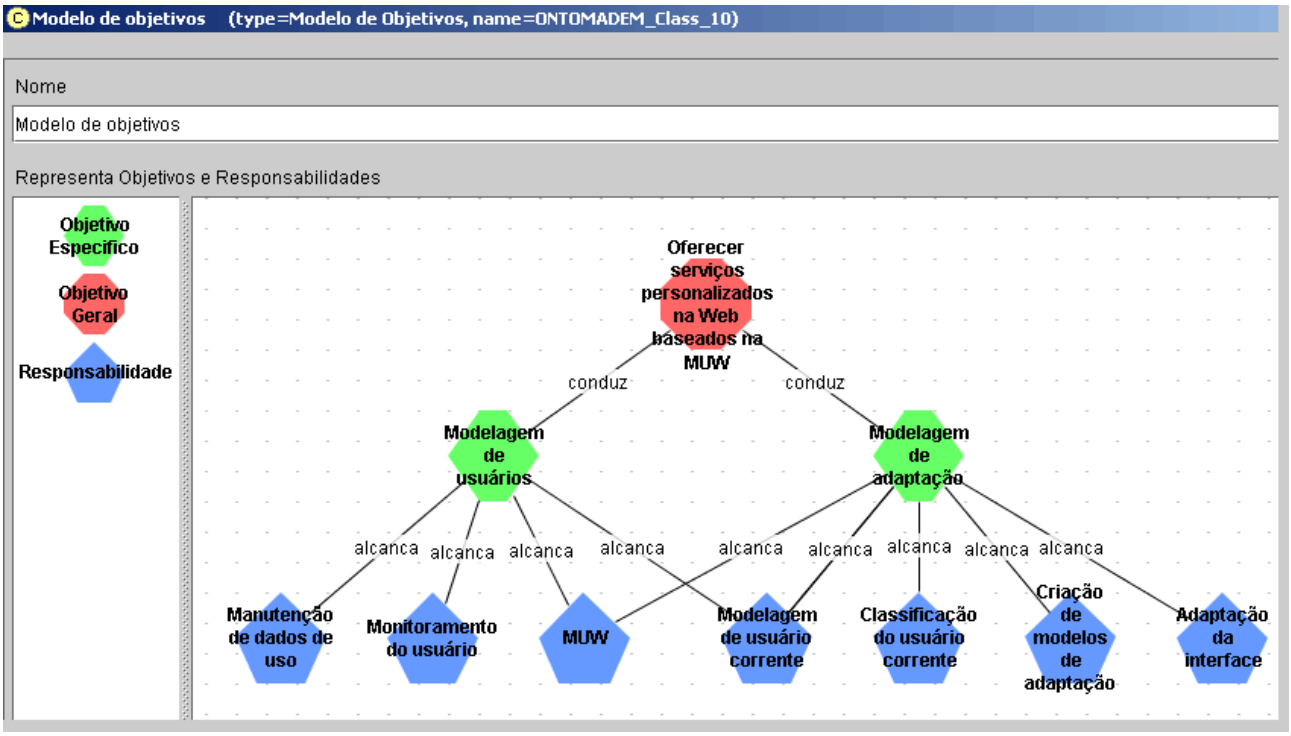

Figura 10. Modelo de objetivos

5.1.2 Modelagem de papéis

Na modelagem de papéis, cada responsabilidade identificada no modelo de objetivos é associada a papéis. Como dito antes, um papel é uma grande função que um indivíduo desempenha em uma organização. Portanto, foram identificados os seguintes papéis: *Monitor*, *Modelador*, *Aquisitor*, *Minerador*, *Visualizador*, *Classificador*, *Personalizador* e *Interfaceador*. A partir daí são identificadas as atividades que permitem o exercício de cada responsabilidade. Em seguida, são identificados as entradas, as saídas, os estados e os recursos, que cada papel usa para a realização de suas atividades. O produto dessa subtarefa é um modelo de papéis composto dos papéis do sistema com suas respectivas responsabilidades, atividades, entradas, saídas, estados anteriores, posteriores e recursos.

As atividades podem ter dados de entrada e dados de saída, onde os dados de entrada são informações necessárias para a execução da atividade e os dados de saída são os produtos resultantes da execução da atividade. Além das entradas e saídas, as atividades podem ter estados anteriores, que são os possíveis estados em que o papel se

encontra antes da execução da atividade, e estados posteriores que são os possíveis estados que o papel vai estar após a execução da atividade correspondente.

A MADEM define um recurso como uma técnica, método ou algoritmo que a atividade requisita para o cumprimento de seus objetivos. A especificação de tais técnicas e algoritmos são deixadas para a fase de projeto.

5.1.2.1 Modelo do papel Monitor

Descrição: O papel *Monitor* (Figura 11) tem a responsabilidade de monitorar o comportamento de navegação do usuário corrente.

Responsabilidade: *Monitoramento do usuário*. Através do monitoramento das atividades de navegação do usuário capturam-se as informações que caracterizam o seu comportamento de navegação.

Atividade: *Processamento do comportamento de navegação do usuário*. Isso envolve a captura de informações provenientes da interação do usuário com a Web. Exemplos dessas informações são a URL da página visitada e/ou o tempo de permanência nessa página.

- Dados de entrada: Interação do usuário como uma página Web;
- Dados de saída: Informações de navegação do usuário (ex: URL da página visitada e/ou tempo de permanência na página);
- Estado anterior: Aguardando visita de página Web pelo usuário;
- Estado posterior: Aguardando visita de página Web pelo usuário;
- Recurso: Critérios de captura de informações de navegação.

Como visto acima, tanto o estado anterior quanto o posterior são iguais. Isso significa que depois de executar a atividade o papel volta para o estado em que se encontrava antes de executar a atividade.

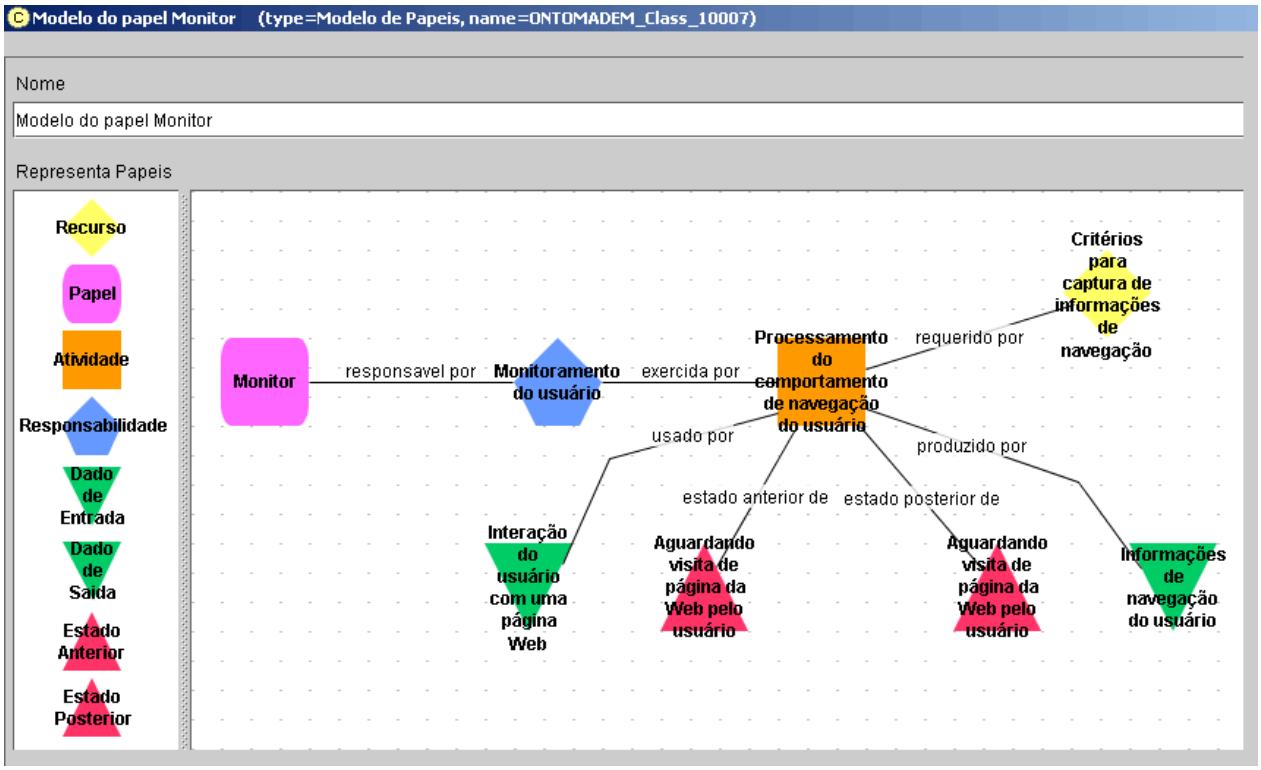

Figura 11. Modelo do papel Monitor

5.1.2.2 Modelo do papel Modelador

Descrição: O papel *Modelador* (Figura 12) tem a responsabilidade de criar e atualizar o modelo de usuários representando a sessão de navegação do usuário corrente.

Atividade: *Criação do modelo de usuários*. Essa atividade envolve a representação formal das informações de navegação do usuário capturadas pelo papel *Monitor*.

- Dados de entrada: Informações de navegação do usuário (ex: URL e/ou tempo de permanência na página visitada);
- Dados de saída: Modelo representando a sessão do usuário corrente;
- Estado anterior: Aguardando informações de navegação do usuário*;*
- Estado posterior: Aguardando informações de navegação do usuário*;*
- Recurso: Critérios para a construção de modelos de usuários.

Atividade: *Atualização do modelo de usuários*. Essa atividade envolve a atualização do modelo de usuários mediante as novas informações de navegação do usuário capturadas pelo papel *Monitor*.

- Dados de entrada: Informações de navegação do usuário (ex: URL e/ou tempo de permanência na página visitada);
- Dados de saída: Modelo de usuários atualizado;
- Estado anterior: Aguardando informações de navegação do usuário*;*
- Estado posterior: Aguardando informações de navegação do usuário*;*
- Recurso: Critérios para a atualização de modelos de usuários.

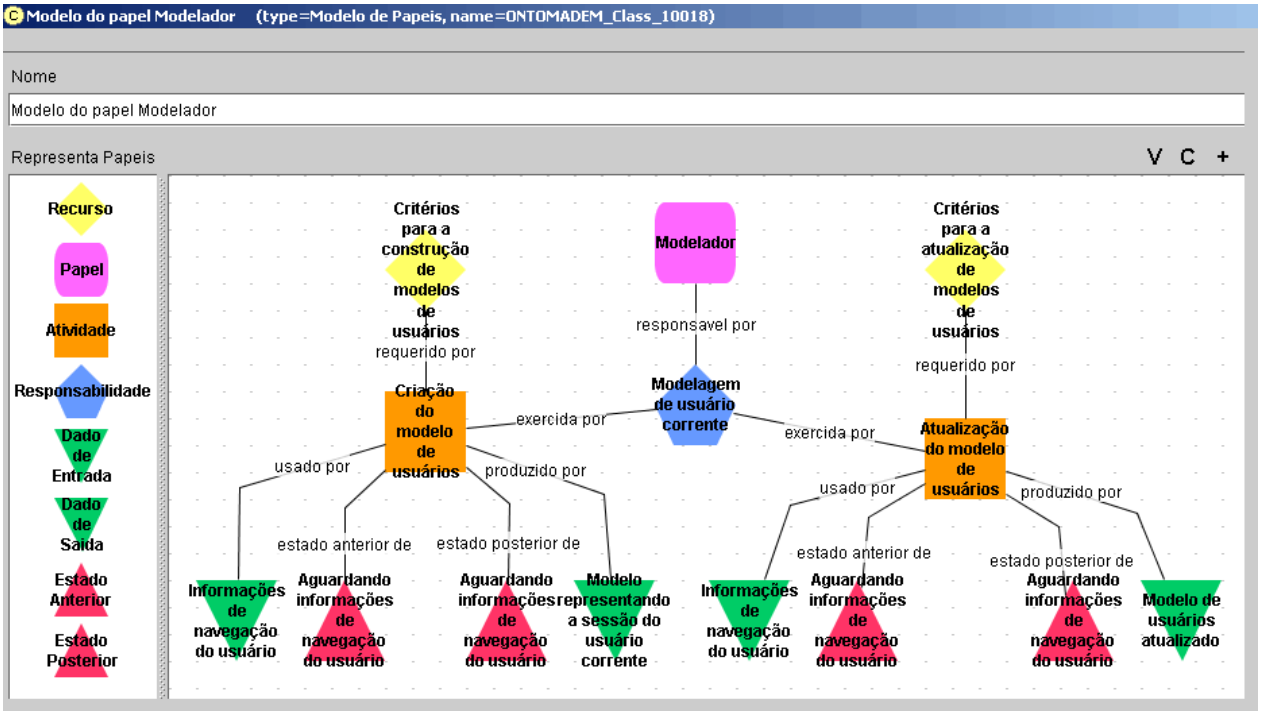

Figura 12. Modelo do papel Modelador

5.1.2.3 Modelo do papel Aquisitor

Descrição: O papel *Aquisitor* (Figura 13) tem a responsabilidade de manter consistente um repositório de dados de uso dos usuários. Essa responsabilidade representa a fase de pré-processamento da MUW (Figura 2).

Responsabilidade: *Manutenção dos dados de uso*.

Atividade: *Armazenamento dos dados de uso*.

- Dados de entrada: Modelos de usuários;
- Dados de saída: Arquivo de uso;
- Estado anterior: Aguardando modelo de usuários a ser armazenado;
- Estado posterior: Aguardando modelo de usuários a ser armazenado;
- Recurso: Critérios de armazenamento de dados de uso.

Atividade: *Atualização dos dados de uso*.

- Dados de entrada: Modelos de usuários;
- Dados de saída: Arquivo de uso atualizado;
- Estado anterior: Aguardando modelo de usuários a ser armazenado;
- Estado posterior: Aguardando modelo de usuários a ser armazenado;
- Recurso: Critérios para a atualização do arquivo de uso.

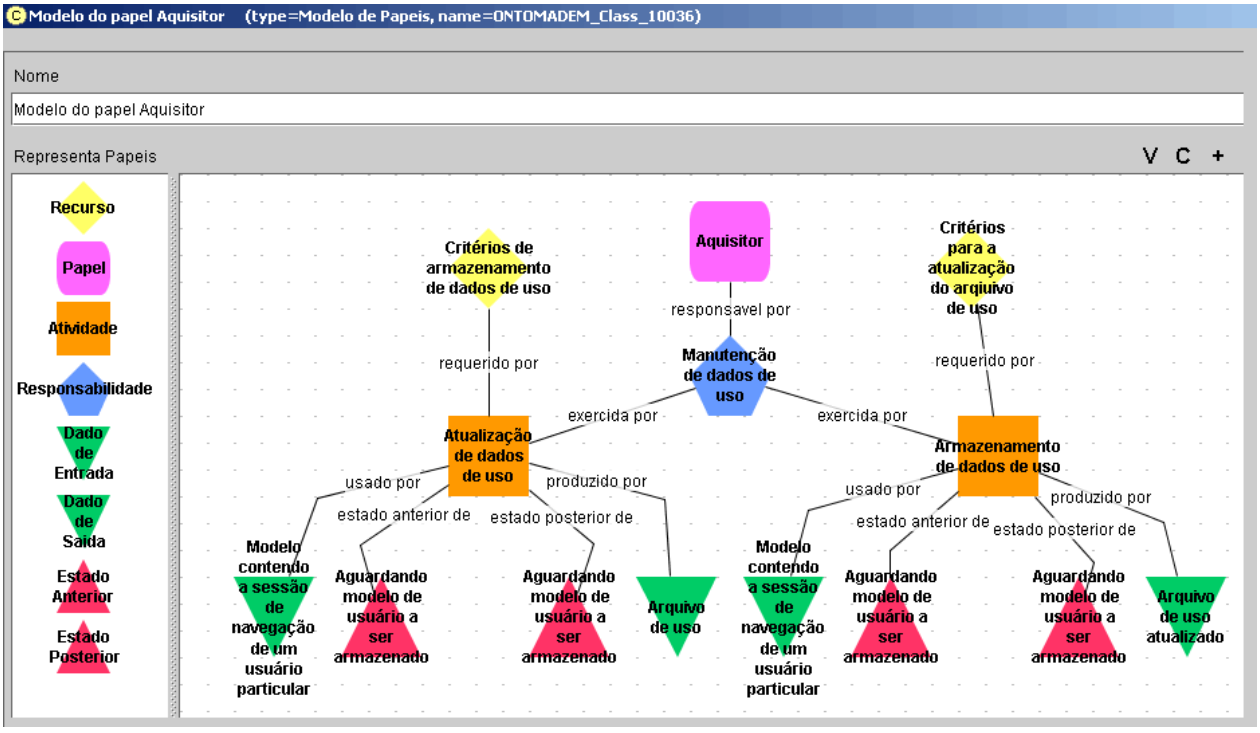

Figura 13. Modelo do papel Aquisitor

5.1.2.4 Modelo do papel Minerador

Descrição: O papel *Minerador* (Figura 14) tem a responsabilidade de construir modelos representando grupos de usuários com comportamento de navegação similar.

Responsabilidade: *MUW*.

Atividade: *Representação das sessões*. Antes dos algoritmos de mineração serem aplicados as sessões dos usuários precisam ser identificadas e representadas em um formato adequado (ex.: vetores do espaço vetorial).

- Dados de entrada: Arquivo de uso;
- Dados de saída: Conjunto de sessões de usuários;
- Estado anterior: Aguardando início do processo de mineração;
- Estado posterior: Iniciando a descoberta de padrões;
- Recurso: Critérios para a representação das sessões de usuários.

Atividade: *Descoberta de padrões*. Uma vez as sessões dos usuários devidamente representadas, é hora de aplicar os algoritmos de mineração.

- Dados de entrada: Conjunto de sessões de usuários;
- Dados de saída: Modelos representando grupos de usuários;
- Estado anterior: Aguardando a representação das sessões de usuários;
- Estado posterior: Aguardando reinício do processo de mineração;
- Recurso: Critérios para a descoberta de padrões.

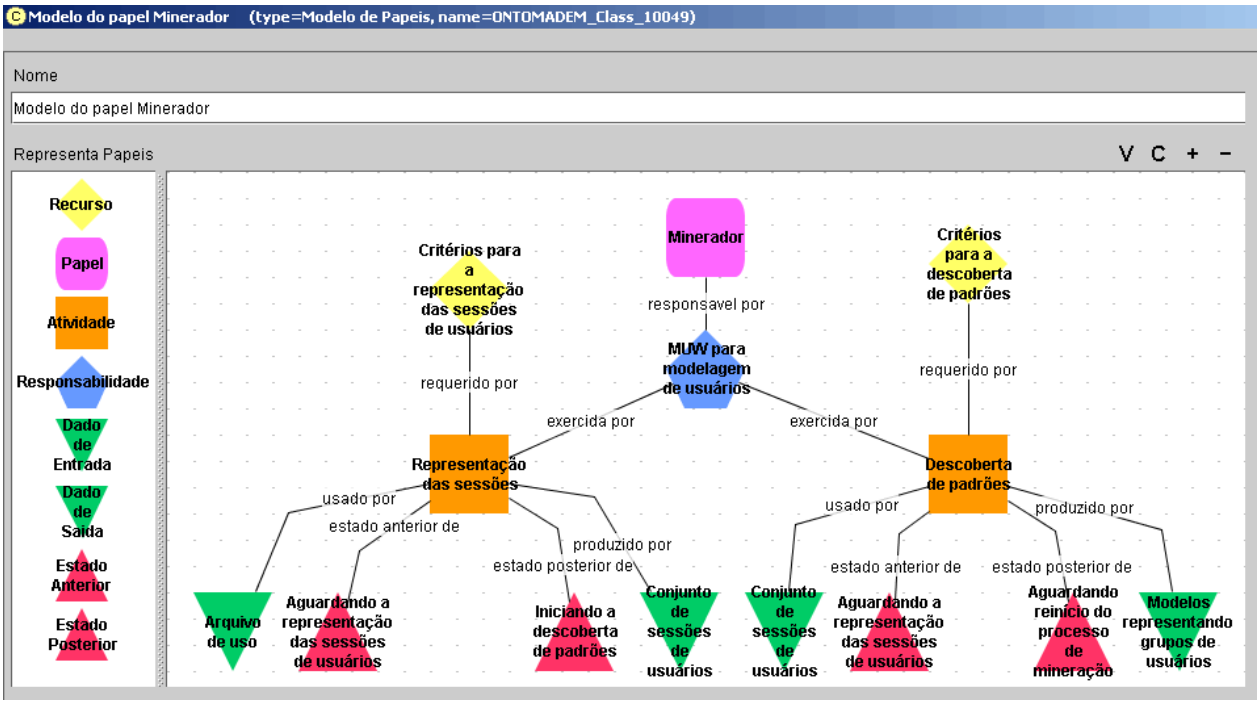

Figura 14. Modelo do papel Minerador

Nas aplicações para personalização na Web baseadas na MUW, o processo de mineração é executado periodicamente em longos intervalos de tempo. Daí o motivo do estado posterior dessa atividade ser "*Aguardando reinício do processo de mineração*"

5.1.2.5 Modelo do papel Classificador

Descrição: O papel *Classificador* (Figura 15) tem a responsabilidade de classificar o usuário ativo em um dos grupos pré-descobertos no processo de mineração.

Responsabilidade: *Classificação do usuário corrente*.

Atividade: *Classificação do usuário corrente.*

- Dados de entrada: Modelo do usuário corrente;
- Dados de saída: Grupo de usuários no qual o usuário corrente foi classificado;
- Estado anterior: Aguardando solicitação de classificação;
- Estado posterior: Aguardando solicitação de classificação*;*
- Recurso: Critérios para a classificação.

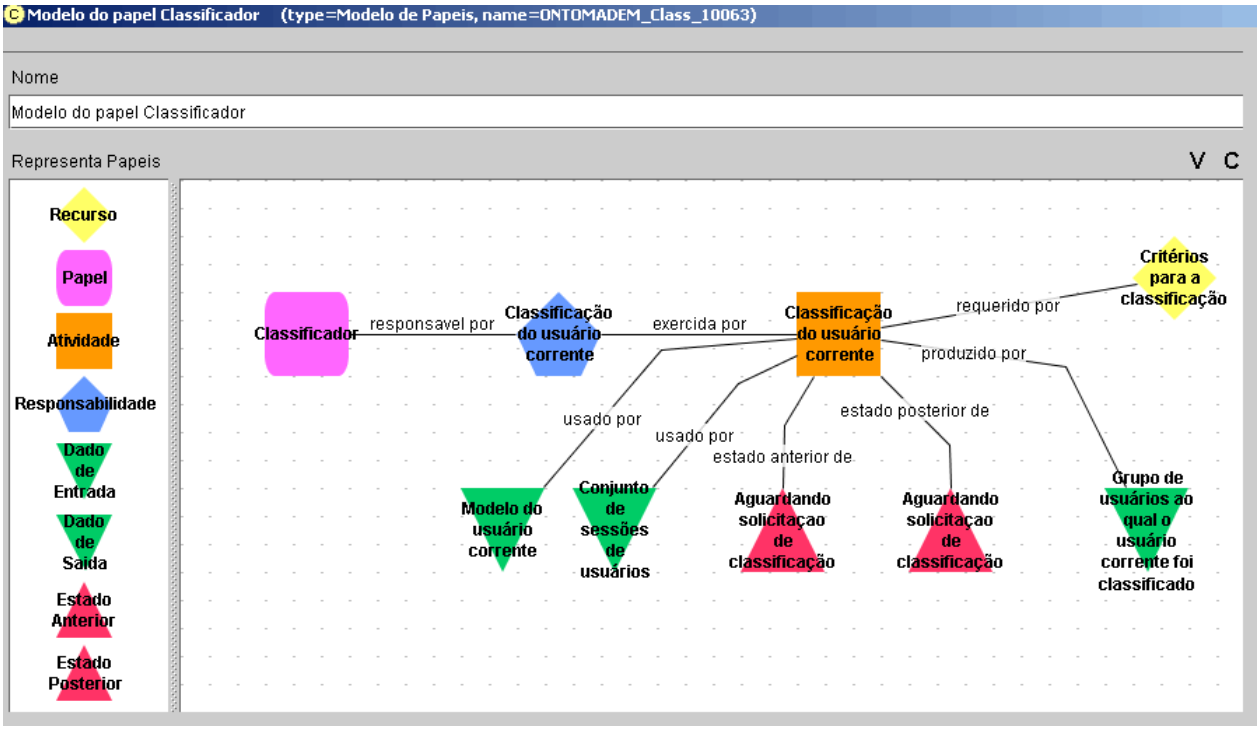

Figura 15. Modelo do papel Classificador

Em algum momento o módulo responsável pela personalização vai solicitar a classificação do usuário corrente em um dos grupos existentes. A personalização vai então ser baseada no grupo ao qual o usuário foi classificado.

5.1.2.6 Modelo do papel Personalizador

Descrição: O papel *Personalizador* (Figura 16) tem a responsabilidade de criar e atualizar os modelos de adaptação do sistema. Um modelo de adaptação contém as regras que ditam como as adaptações devem ser executadas na interface do usuário, por exemplo, regras que estabelecem o instante, ou o formato das recomendações a serem incluídas na interface do usuário.

Responsabilidade: *Criação de modelos de adaptação*.

Atividade: *Criação de modelos de adaptação*.

- Dados de entrada: Grupo no qual o usuário corrente foi classificado;
- Dados de saída: Lista de personalizações;
- Estado anterior: Aguardando a classificação do usuário corrente;

- Estado posterior: Notificando papel *Interfaceador* sobre lista de personalizações disponíveis;
- Recurso: Critérios para criação de modelos de adaptação.

Atividade: *Atualização dos modelos de adaptação*.

- Dados de entrada: Grupo ao qual o usuário foi classificado/reclassificado. O usuário pode ser reclassificado várias vezes durante sua sessão de navegação, portanto, em determinado momento ele pode ser classificado em um grupo diferente do que foi classificado anteriormente. Dessa forma, o modelo de adaptação precisa ser atualizado de modo a refletir essas mudanças;
- Dados de saída: Nova lista de personalizações;
- Estado anterior: Aguardando reclassificação do usuário corrente;
- Estado posterior: Notificando papel *Interfaceador* sobre nova lista de personalizações disponíveis;
- Recurso: Critérios para atualização de modelos de adaptação.

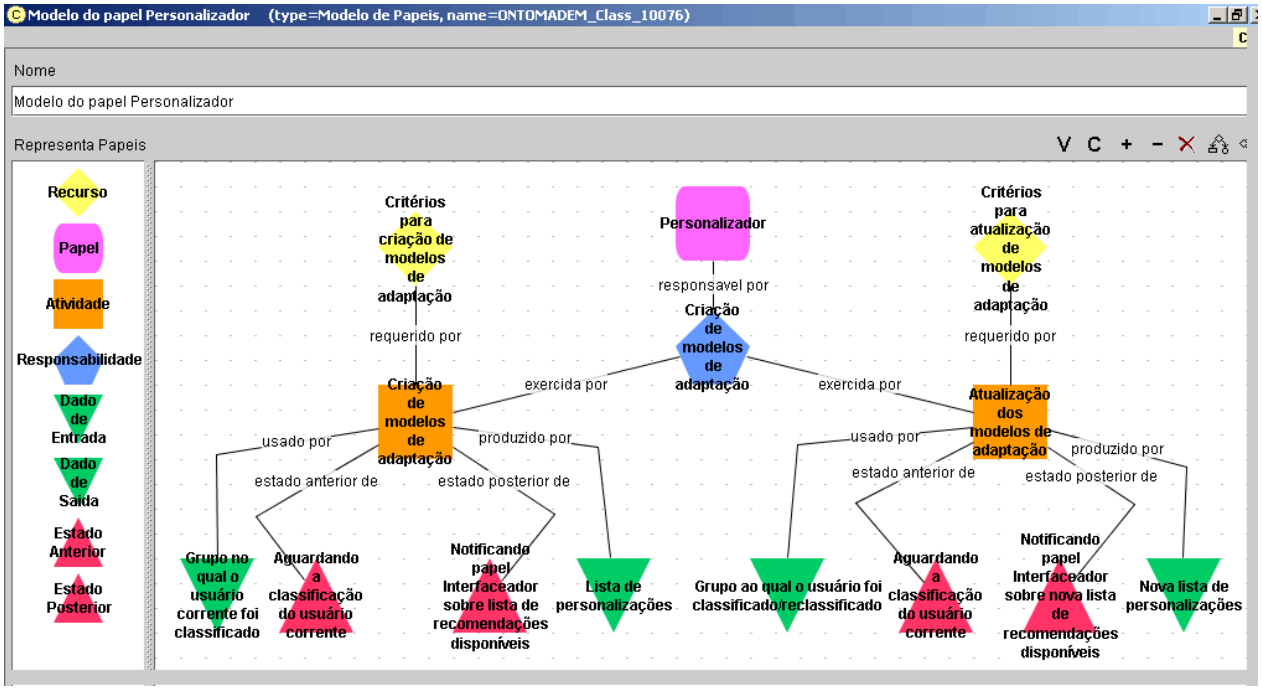

Figura 16. Modelo do papel Personalizador

5.1.2.7 Modelo do papel Interfaceador

Descrição: O papel *Interfaceador* (Figura 17) tem a responsabilidade de manipular a interface do usuário de modo a apresentar os efeitos de personalização gerados pelo sistema, por exemplo, incluindo vínculos de hipertexto na página sendo visitada.

Responsabilidade: *Adaptação da interface*.

Atividade: *Apresentação das personalizações*. Isso envolve a formatação das personalizações em um formato adequado à visualização dos usuários (ex.: HTML).

- Dados de entrada: Lista de personalizações previamente processadas pelo papel *Personalizador;*
- Dados de saída: Interface adaptada com a inclusão das personalizações;
- Estado anterior: Aguardando lista de personalizações;
- Estado posterior: Aguardando lista de personalizações;
- Recurso: Critérios para a apresentação das personalizações.

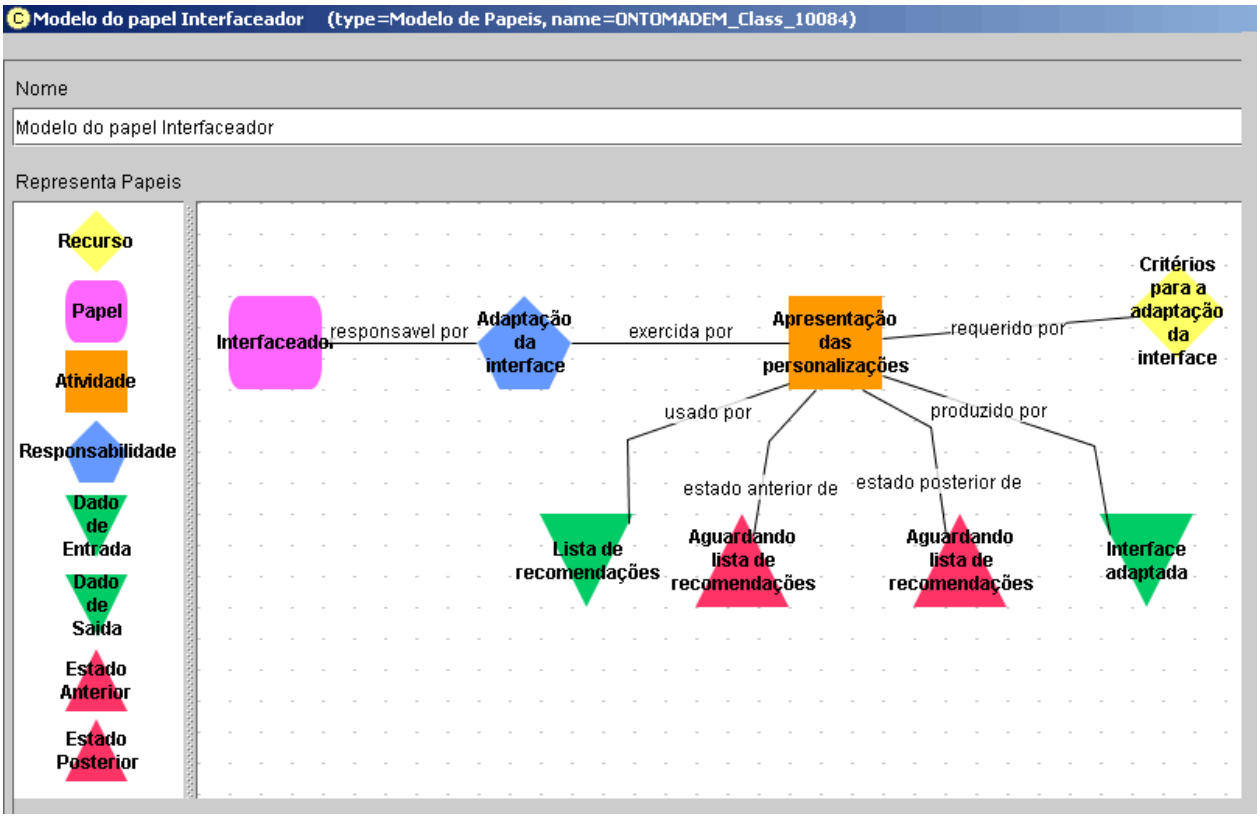

Figura 17. Modelo do papel Interfaceador

5.1.3 Modelagem de interações entre papéis

Para cada objetivo específico define-se um modelo de interações entre papéis. Nesses modelos são representadas as interações que ocorrem entre os papéis e as entidades externas ao sistema. Dessa forma, teremos dois modelos de interações, um para o objetivo específico *Modelagem de usuários* e outro para o objetivo específico *Modelagem de adaptação*.

No primeiro (Figura 18), são mostradas as interações que são realizadas com o a finalidade de modelar o usuário, seja através de suas interações em tempo real ou através de seus dados de uso. O modelo de interações definido pela MADEM é inspirado no diagrama de colaboração da UML, onde as interações são ordenadas numericamente.

Abaixo se descreve as interações entre os papéis com a finalidade de atingir o objetivo *Modelagem de usuários*:

1. O usuário fornece comportamento de navegação;

2. O papel *Monitor* captura as informações de navegação do usuário e as envia ao papel *Modelador*;
3. O papel *Modelador* então cria/atualiza um modelo de usuários a partir das informações recebidas do papel *Monitor*. Depois que é identificado o término da sessão de navegação do usuário o modelo é enviado ao papel *Aquisitor;*
4. Uma vez que o papel *Aquisitor* recebe um modelo do papel *Modelador*, ele o formata adequadamente e o armazena de forma consistente;
5. Em seguida, existindo dados a serem minerados, o papel *Minerador* pode então iniciar o processo de mineração. Isso culmina com a descoberta de padrões.

No segundo diagrama (Figura 19), são mostradas as interações que são realizadas com a finalidade de oferecer serviços personalizados aos usuários da Web baseados na MUW:

1. O usuário fornece comportamento de navegação;
2. O papel *Monitor* captura as informações de navegação do usuário e as envia ao papel *Modelador;*
3. O papel *Modelador* então cria/atualiza um modelo de usuários a partir das informações recebidas do papel *Monitor*. O papel *Modelador*, de acordo com critérios a serem definidos na fase de projeto, pode solicitar a classificação do usuário corrente num dos grupos descobertos durante a MUW;
4. De posse dos padrões descobertos e do modelo do usuário corrente, o papel *Classificador* realiza a classificação do usuário corrente em algum dos grupos descobertos pelo minerador e envia esse mesmo grupo ao papel *Personalizador*, para que este possa construir o modelo de adaptação para o usuário ativo;

5. Após a construção do modelo de adaptação o papel *Personalizador* notifica o papel *Interfaceador* sobre a lista de personalizações disponíveis*;*
6. Após o recebimento da lista de personalizações o papel *Interfaceador* as formata e as apresenta ao usuário.

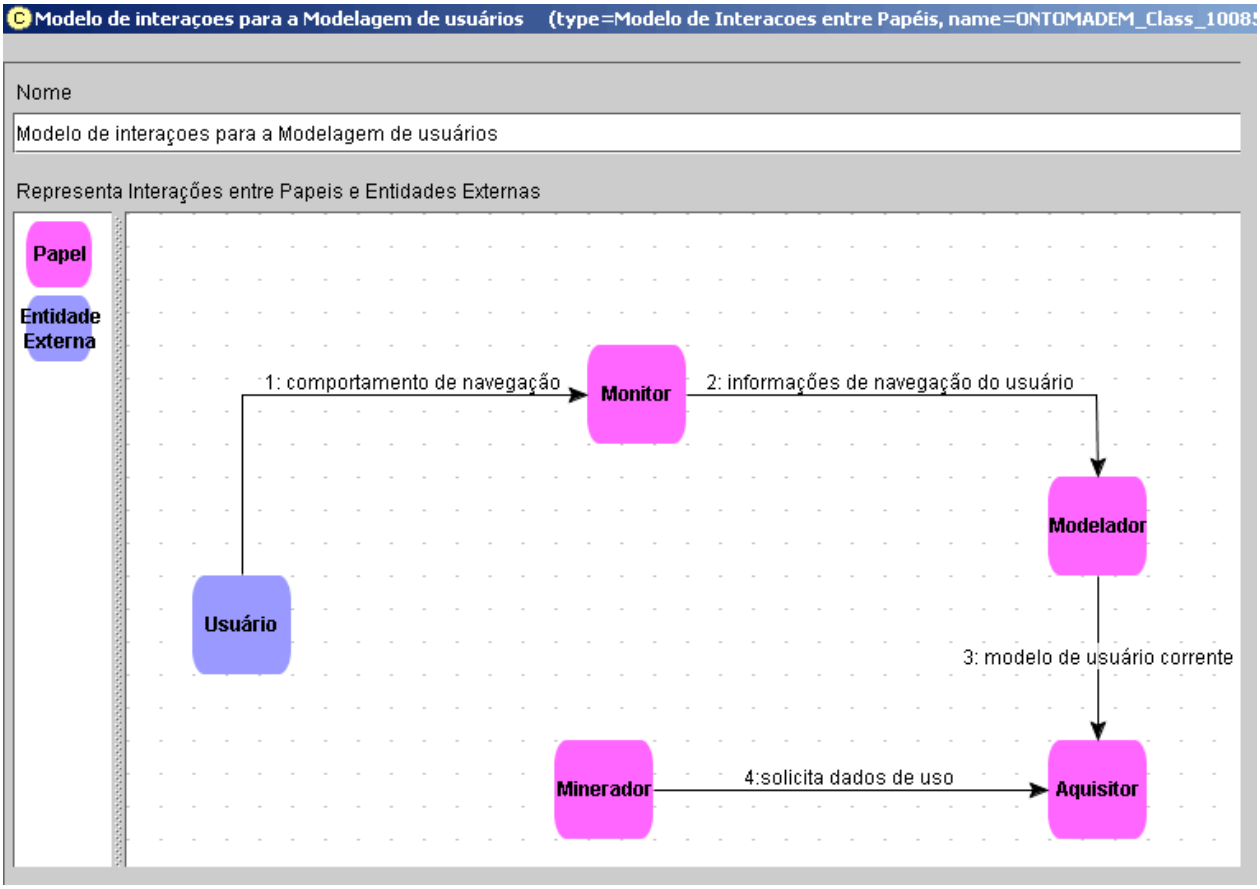

Figura 18. Modelo de interações do objetivo específico Modelagem de usuários

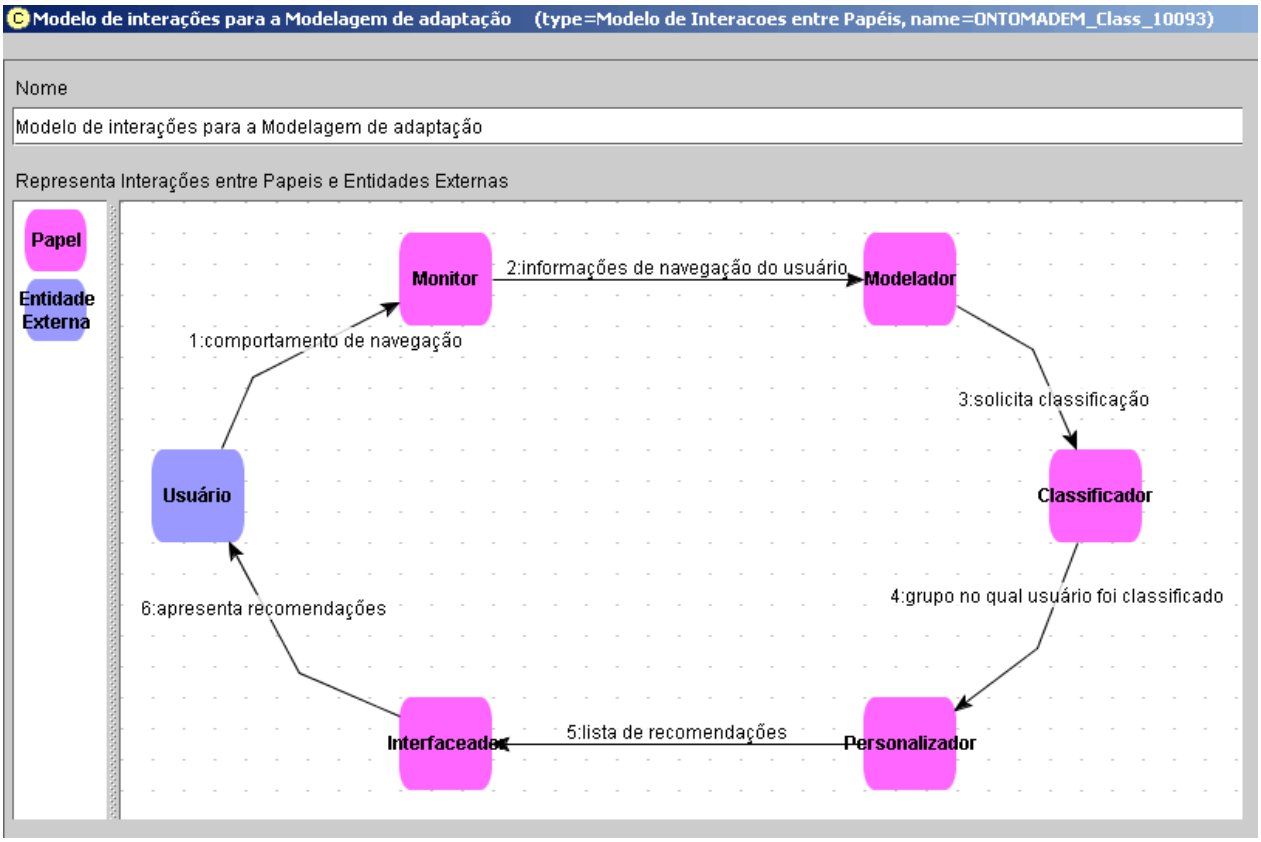

Figura 19. Modelo de interações do objetivo específico Modelagem de adaptação

5.1.4 Modelagem de variabilidades

Na Modelagem de variabilidades é feita uma classificação das instâncias dos conceitos (objetivos geral e específicos, responsabilidades, papéis, atividades e recursos) em fixos e variáveis. Conceitos fixos são conceitos que estão presentes em todos os sistemas de uma família de sistemas e representam suas características comuns. Conceitos variáveis são conceitos que podem estar presentes ou não em sistemas de uma família de sistemas e representam suas características particulares. Os conceitos são classificados de acordo com as regras apresentadas a seguir [43]:

O objetivo geral é fixo. De acordo com o modelo de objetivos, um objetivo geral deve ser definido e alcançado por todos os sistemas de uma família de sistemas. Sendo assim, o objetivo geral é um conceito fixo.

Os objetivos específicos são variáveis. De acordo com o que é definido no modelo de objetivos, para que o objetivo geral seja alcançado, pelo menos um dos

objetivos específicos tem que ser alcançado. Sendo assim, um objetivo específico é um conceito que pode estar presente ou não no desenvolvimento de cada sistema de uma família de sistemas e, portanto, é variável.

As atividades podem ser fixas ou variáveis. Os papéis exercem suas responsabilidades através da execução de atividades. Atividades são fixas se elas devem ser executadas em todos os sistemas da família. Do contrário, as atividades são variáveis.

As responsabilidades podem ser fixas ou variáveis. Os objetivos específicos são alcançados através do exercício de suas responsabilidades. Se a responsabilidade contribui para o alcance de todos os objetivos específicos e é exercida através da execução de atividades fixas, então a responsabilidade é classificada como fixa. Do contrário, as responsabilidades são classificadas como variáveis.

Os papéis podem ser fixos ou variáveis. As responsabilidades são exercidas pelos papéis. Então os papéis são classificados de acordo com suas responsabilidades: papéis são variáveis se exercem responsabilidades variáveis. Do contrário, os papéis são fixos.

Os recursos podem ser fixos ou variáveis. Para a execução de atividades os papéis dispõem de recursos. Um recurso é classificado como fixo se for requerido por pelo menos, uma atividade fixa. Caso contrário, o recurso é classificado como variável.

A Tabela 6 mostra a classificação as características fixas e variáveis do modelo de domínio do ONTOMUW.

5.2 Requisitos não funcionais

A MADEM no estágio atual não contempla técnicas para a captura dos requisitos não funcionais do sistema. Entretanto, é imperativa a explicitação de tais requisitos, pois eles serão responsáveis por regular as decisões da fase de projeto.

Como visto nos modelos de papéis construídos na seção 5.1.2, as atividades geralmente requerem recursos. Esses recursos estão geralmente associados a técnicas ou algoritmos a serem definidos e justificados na fase de projeto. O que vai determinar a escolha dessas técnicas e algoritmos, em parte, são justamente os requisitos não

funcionais do sistema. Esses requisitos são definidos levando-se em conta os objetivos do sistema.

Características fixas				
Objetivo Geral	**Responsabilidades**	**Papéis**	**Atividades**	**Recursos**
Oferecer serviços personalizados na Web	Manutenção dos dados de uso	Aquisitor	- Armazenamento de dados de uso -Atualização de dados de uso	-Critérios para o armazenamento dos dados de uso -Critérios para a atualização dos dados de uso
	Monitoramento do usuário	Monitor	- Processamento do comportamento de navegação	- Critérios para captura de informações de navegação
	Modelagem de usuário corrente	Modelador	- Criação do modelo de usuários - Atualização do modelo de usuários	- Critérios para a construção de modelos de usuários - Critérios para a atualização de modelos de usuários
	MUW	Minerador	- Representação das sessões dos usuários - Descoberta de padrões	- Critérios para a representação de sessões - Critérios para a descoberta de padrões
Características variáveis				
Objetivo específico	**Responsabilidades**	**Papéis**	**Atividades**	**Recursos**
Modelagem de adaptação	Criação de modelos de adaptação	Personalizador	- Criação de modelos de adaptação - Atualização de modelos de adaptação	- Critérios para criação de modelos de adaptação - Critérios para a atualização de modelos de adaptação
	Classificação do usuário corrente	Classificador	- Classificação do usuário corrente	- Critérios para a classificação
	Adaptação da interface	Interfaceador	- Apresentação das personalizações	- Critérios para a adaptação da interface

Tabela 6. Modelagem de variabilidades

Portanto, baseado nos objetivos definidos na fase de análise e no que é oferecido pela maioria das aplicações baseadas na MUW, chegou-se a seguinte lista de requisitos não funcionais:

- A captura e representação das informações derivadas do comportamento de navegação do usuário devem ser feitas em tempo real;
- O usuário deve ser considerado anônimo. Uma das principais vantagens dos sistemas baseados na MUW é justamente a desnecessidade de não requerer do usuário o preenchimento de questionários ou a solicitação explícita de feedback sobre as páginas visitadas;
- Sendo os usuários anônimos, as informações utilizadas para a construção do modelo de usuários devem ser inferidas diretamente do seu comportamento de navegação;
- O modelo de usuários deve ser flexível o suficiente de forma a suportar uma ou mais características de navegação do usuário (como, por exemplo, a URL da página visitada e o tempo de permanência na página);
- O mesmo modelo utilizado para a representação de uma sessão individual deve ser utilizado para a representação de grupos de sessões;
- A classificação do usuário corrente em grupos de usuários pré-determinados deve ser rápida o suficiente para ser realizado em tempo real sem atrasos perceptíveis para o usuário, independentemente do número de grupos existentes e da cardinalidade (número de membros do grupo) dos mesmos;
- A aplicação das recomendações deve ser realizada em tempo real e aplicadas transparentemente ao usuário durante suas atividades correntes de navegação;

O próximo capítulo apresenta detalhadamente as técnicas e algoritmos escolhidos respeitando esses requisitos.

5.3 Considerações finais do capítulo

Esse capítulo apresentou a construção do modelo de domínio do ONTOMUW. O modelo de domínio foi construído segundo as tarefas de modelagem definidas pela MADEM (modelagem de objetivos, papéis, interações entre papéis e variabilidades). Os modelos gerados são instâncias da ontologia ONTOMADEM e foram gerados através do

preenchimento de formulários personalizados no editor de ontologias Protégé [125]. Sendo assim, devido à flexibilidade e alto grau de representação que as ontologias oferecem, desenvolvedores podem tanto facilmente compreender quanto reutilizar ou estender esse modelo de domínio.

O levantamento dos requisitos se deu mediante todo o estudo e pesquisa realizados sobre a mineração na Web e em especial sobre a MUW.

É importante notar que não foram dadas soluções aos problemas levantados na fase de análise; apenas investigou-se “o que” deve ser feito para que os objetivos do domínio sejam alcançados. O “como” deve ser feito é uma questão do projeto, tema do próximo capítulo.

A MADEM ainda possui uma lacuna referente à falta de técnicas para a captura de requisitos não funcionais. A definição desses requisitos vai fornecer restrições sobre as técnicas ou algoritmos a serem utilizados. Sendo assim, a explicitação de tais requisitos se deu de forma um tanto quanto informal, onde se levou em conta o que se espera da maioria dos sistemas de personalização para a Web baseados na MUW.

6 PROJETO DE DOMÍNIO DO ONTOMUW

Enquanto na fase de análise o foco estava na investigação do domínio, onde se definiu “o que” deve ser feito para alcançar os objetivos, na fase de projeto o foco está na solução, onde se deve definir o “como” fazer.

A fase de projeto de domínio da MADEM produz uma especificação do projeto que pode ser reutilizada para o desenvolvimento de aplicações multiagente específicas pertencentes a uma família de sistemas em um domínio de aplicação. É nessa fase que as técnicas e algoritmos são especificados com o intuito de fornecer soluções funcionais para os requisitos funcionais levantados na fase de análise.

Na MADEM, a fase de projeto de domínio tem como insumos modelos de domínio e coletâneas de padrões (ver Figura 9, cap. 4). Esta fase consiste de duas tarefas: o *Projeto Arquitetural e* o *Projeto Detalhado*. O produto final desta fase é a especificação formal de um framework multiagente.

Este capítulo apresenta o projeto arquitetural e detalhado do ONTOMUW. A seção 6.1 apresenta as técnicas e formalismos escolhidos para a construção tanto de modelos de usuários individuais quanto de grupos de usuários. A seção 6.2 apresenta os algoritmos escolhidos para a descoberta dos padrões de navegação dos usuários. A seção 6.3 descreve a estratégia definida para a captura do comportamento de navegação dos usuários. A seção 6.4 apresenta a estratégia para a personalização da interface do usuário. A seção 6.5 mostra as técnicas escolhidas para o armazenamento e manutenção dos dados de uso dos usuários. A seção 6.6 detalha o *Projeto Arquitetural*. A seção 6.7 apresenta o *Projeto Detalhado*, mostrando cada uma de suas tarefas e modelos gerados. Por último, a seção 6.8 apresenta as considerações finais do capítulo.

6.1 Modelo de usuários

Como visto no capítulo 3, a modelagem de usuários envolve a aquisição, representação e manutenção das características dos usuários. Como exemplos de características podemos citar: informações demográficas (nome, idade, sexo, etc.), preferências (cor, tamanho de fonte, etc.), interesses (sistemas multiagente, ontologias,

etc.), nível[7], e etc. Quais características capturar é uma decisão dependente da aplicação. Aqui, para os propósitos do ONTOMUW, as características a serem capturadas são as informações provenientes da navegação do usuário. Como o usuário é anônimo, ou seja, ele não se identifica previamente no sistema, as informações devem ser inferidas através de sua navegação pela Web. Desse modo, o modelo de usuários representará a sessão corrente de navegação dos usuários. Aqui serão necessários dois tipos de modelos, um modelo representando uma sessão de navegação individual e um modelo representando grupos de sessões com comportamento de navegação similar. Pois, de forma a classificar o usuário corrente em um grupo pré-determinado, precisa-se tanto do modelo do usuário corrente quanto dos modelos representando os grupos. A representação usada, tanto para uma sessão individual quanto para um grupo, deve ser a mesma, pois isso possibilita um entendimento e uma visão comum dos modelos.

6.1.1 Representação das sessões de navegação

Dentre os formalismos existentes, optou-se pelo modelo de matrizes de características idealizado por Shahabi et al. (2003) [131]. Esse modelo oferece um framework conceitual, formalizado matematicamente e experimentalmente testado, para a modelagem de usuários e grupo de usuários anônimos. Dentre as principais vantagens desse formalismo pode-se citar:

- Flexibilidade, pois se pode adicionar quantas características de navegação (ex.: URL visitada, número de visitas, tempo de permanência, etc.) forem desejadas sem comprometer a estrutura de dados do modelo;
- O modelo lida com usuários anônimos, onde um usuário é representado por uma sessão de navegação em particular;
- O mesmo formalismo utilizado para a modelagem de um usuário individual é utilizado para a modelagem de grupos de usuários;

[7] Geralmente, os sistemas tutoriais definem estereótipos indicando os níveis de conhecimento do usuário. Então a partir do grau de conhecimento do usuário ele pode ser classificado em diferentes níveis, por exemplo, iniciante, intermediário ou avançado [7]

- Para o cálculo de similaridade entre os modelos de usuários individuais, ou entre modelos individuais e modelos de grupo podem ser utilizadas quaisquer medidas de similaridade conhecidas para vetores do espaço vetorial, pois o modelo de matrizes é uma extensão desse modelo;
- O cálculo da similaridade entre modelos individuais e modelos de grupos não depende da cardinalidade dos grupos (número de membros de grupo), permitindo classificações bastante rápidas.

O modelo de matrizes de características (FM) é uma abordagem que provê um formalismo baseado no modelo do espaço vetorial para a representação tanto de sessões de navegação de usuários individuais quanto de grupos de usuários. Esse modelo foi idealizado, proposto e formalizado por Shahabi et al. (2003) [131]. A idéia chave é a construção de um conjunto de matrizes onde cada matriz representa uma característica de navegação particular do usuário. De forma a quantificar essas características considera-se um conjunto universal de segmentos, isto é, um conjunto de subcaminhos de navegação em um espaço conceitual como base do espaço de sessões. Essa idéia é análoga à definição de base em um espaço vetorial, onde um conjunto de vetores linearmente independentes constroem o espaço vetorial. No caso do modelo FM, um conjunto universal de segmentos constrói o espaço de segmentos possíveis.

Vale lembrar que não se tem a intenção, com o presente trabalho, de definir novos métodos ou algoritmos para a MUW, mas sim utilizar os que se encontram em estado da arte segundo os requisitos funcionais e não-funcionais levantados na fase de análise.

As seções subseqüentes apresentam uma descrição da terminologia básica do modelo FM assim como a estrutura de dados que o suporta.

6.1.1.1 Terminologia do modelo FM

Site da Web: Um site da Web pode ser visto como um conjunto de páginas estáticas ou dinâmicas.

Espaço Conceitual: Cada site da Web fornece informações sobre um ou mais tópicos ou conceitos. O site do universo on-line (UOL[8]), por exemplo, inclui conceitos tais como turismo, política, esportes e etc. Dessa forma, as páginas de um site da Web podem ser categorizadas de acordo com os conceitos os quais representam. Portanto, um espaço conceitual em um site da Web pode ser definido como o conjunto de páginas que contêm informações sobre certo conceito. Pode acontecer também, de uma determinada página incluir mais de um conceito ao mesmo tempo, por exemplo, uma determinada página que fala sobre a política esportiva inclui os conceitos política e esporte. Dessa forma, o espaço conceitual de um site não é formado por conjuntos disjuntos, pois a mesma página pode pertencer a mais de um conceito. A identificação do espaço conceitual de um site pode ser realizada de forma manual, automática, ou híbrida onde parte do processo é automática e parte manual. Um exemplo onde a abordagem híbrida pode ser aplicada seria a utilização de técnicas de análise de conteúdo, tais como os métodos comumente utilizados pelos motores de busca para classificar e categorizar as páginas em diferentes conceitos baseados em seu conteúdo. Depois, se necessário, um especialista humano pode, manualmente, melhor ajustar o espaço conceitual de acordo com a aplicação.

No presente trabalho, considerou-se que todas as páginas de um site sempre estarão representando um único conceito. Essa decisão se deu pelo fato de que para trabalhar com um espaço conceitual são necessárias técnicas associadas à análise de conteúdo, o que está fora do escopo do deste trabalho. A adição da mineração de conteúdo para o enriquecimento dos padrões descobertos pela MUW é uma extensão desejável ao ONTOMUW, e fica como sugestão para trabalhos futuros.

Apesar dessa restrição o modelo continua apresentando resultados satisfatórios, como pode ser visto em Shahabi (2003) [131], idealizador do modelo FM, o qual demonstrou através de seus experimentos bons resultados mesmo trabalhando com apenas um conceito.

Caminho: Um caminho em um site da Web é um conjunto finito de páginas:

$$x_1 \rightarrow x_2 \rightarrow ... \rightarrow x_i \rightarrow ... \rightarrow x_s$$

[8] Universo On-Line (UOL) - http://www.uol.com.br

onde x_i é uma página pertencente a um site da Web. As páginas visitadas em um caminho não são necessariamente distintas, pois se pode visitar a mesma página mais de uma vez durante um caminho.

Características de um caminho: Qualquer atributo espacial ou temporal de um caminho é denominado *característica do caminho* ou simplesmente *característica*. O número de visitas em determinada página, o tempo de permanência em uma página e a posição espacial de uma página no caminho são exemplos de características.

Sessão: É denominada *sessão* o caminho percorrido por um usuário enquanto navegando em um espaço conceitual. Toda vez que o usuário deixa um espaço conceitual, por exemplo, entrando em uma página que não seja membro do conceito corrente, a sessão é considerada terminada. Como uma página pode pertencer a mais de um conceito, várias sessões de vários conceitos podem estar presentes num único caminho. Outra possibilidade é a presença de várias sessões do mesmo conceito em um caminho, pois o usuário pode sair e re-entrar no espaço conceitual repetidamente. Para a identificação do comportamento de navegação do usuário podem-se analisar todas as sessões presentes em um caminho ou dar uma prioridade aos conceitos e analisar somente as sessões que pertencem aos conceitos de mais alta prioridade. Também se pode reduzir a complexidade da análise escolhendo somente as maiores sessões do caminho. De qualquer forma, os resultados da análise em diferentes sessões podem ser integrados de forma a fornecer o resultado final.

Como aqui se considerou que todas as páginas representam o mesmo conceito, a sessão de navegação do usuário será constituída de todas as páginas visitadas no site durante determinado intervalo de tempo, ou até que o usuário feche o software de navegação.

Espaço de sessões: É denominado *espaço de sessões* o conjunto de todas as sessões possíveis em um espaço conceitual.

Segmento do caminho: Um segmento de um caminho, ou simplesmente segmento, é uma *n*-tupla E de páginas: $(x_1, x_2, ..., x_i, ..., x_n)$. Define-se *n* como a ordem do segmento onde $\mathrm{E}(n \geq 1)$. Há uma correspondência de um para um entre as tuplas e as

seqüências de páginas: $(x_1, x_2, ..., x_i, ..., x_n) \equiv x_1 \rightarrow x_2 \rightarrow ... \rightarrow x_i \rightarrow ... \rightarrow x_n$. A representação de tupla é utilizada para facilitar a discussão. Qualquer subseqüência de páginas em um caminho pode ser considerada um segmento do caminho. O caminho $x_1 \rightarrow x_3 \rightarrow x_2 \rightarrow x_5 \rightarrow x_2$, por exemplo, contém vários segmentos como, segmentos de primeira ordem (x_1), segmentos de segunda ordem (x_3, x_2) e segmentos de quarta ordem (x_3, x_2, x_5, x_2). O modelo FM explora a noção de *segmentos* como os blocos construtores das sessões.

Conjunto universal de segmentos (ε_C^n): Um conjunto universal de segmentos de ordem-*n* é um conjunto de segmentos rerpesentados por *n*-tuplas em um espaço conceitual *C*. Como já dito, assumiu-se que qualquer site representa apenas um único conceito, independente do número de páginas do site, sendo assim, daqui em diante o *C* será suprimido da notação.

Grupo: Um *grupo* é definido como um conjunto de sessões similares. A similaridade é definida quantitativamente de acordo com uma medida de similaridade apropriada. No decorrer da seção serão dados mais detalhes sobre as possíveis medidas de similaridade.

6.1.1.2 Características do modelo FM

As sessões são definidas através das seguintes características:

- *Visitas (V)*: Uma visita é uma característica espacial que reflete quais páginas são visitadas durante uma sessão. O modelo FM captura *V* identificando quantas vezes cada segmento é encontrado em um caminho de uma sessão. Na verdade, *V* pode ser considerado como uma generalização do conceito convencional de "contador". Os contadores contam o número de visitas por página, que são segmentos de ordem um;
- *Seqüência (S)*: Esta característica espacial diz respeito à posição relativa de uma página em um caminho de uma sessão. *S* é capturado através da identificação da posição relativa de cada segmento contido no conjunto de segmentos contidos na sessão. Se um segmento é repetidamente visitado

em uma sessão, *S* é aproximado através da média aritmética de todas as ocorrências. Dessa forma, *S* não captura a seqüência exata de segmentos, seqüências exatas podem ser capturadas através de ordens mais altas de *V*;

- *Tempo de visita (T)*: O tempo de visita captura o tempo gasto em cada segmento contido em uma sessão. Ao contrário de *V* e *S, T* é uma característica temporal.

As características de cada sessão são capturadas em termos das características dos segmentos contidos na sessão.

O modelo FM é um modelo aberto. Ele permite a inclusão de qualquer outra característica significativa além das mencionadas acima. A mesma estrutura de dados pode ser utilizada para a inclusão de novas características, ou seja, a estrutura do modelo não precisa ser alterada para a inclusão de novas características. Apesar da inclusão de novas características tornarem o modelo mais rico, os idealizadores do modelo FM comprovaram, através de experiências, que as características mencionadas acima são apropriadas e completas o suficiente para a detecção de similaridades e dissimilaridades entre sessões [131].

6.1.1.3 Estrutura de dados

Seja ε^n a base para capturar a característica *F* para a sessão *U*. Constrói-se uma matriz *n*-dimensional $M_{r^n}^F$ para armazenar os valores da característica *F* para todos os segmentos de ordem-*n* de *U*. A matriz *n*-dimensional M_{r^n} é uma generalização da matriz quadrada bi-dimensional M_{r*r}. Cada dimensão de M_{r^n} tem *r* linhas onde *r* é a cardinalidade do espaço conceitual. Por exemplo, a matriz M_{4*4*4} é um cubo com quatro linhas em cada uma das suas três dimensões, e é uma matriz de características para um espaço conceitual de quatro páginas, tendo ε^3 como base. Assume-se que a ordem das dimensões da matriz esteja pré-determinada. O valor de *F* para cada segmento de ordem-*n* $(x_\alpha, x_\beta, ..., x_\omega)$ é gravado no elemento $a_{\alpha\beta...\omega}$ de $M_{r^n}^F$. Para simplificar o entendimento dessa estrutura, o leitor deve assumir que as linhas em todas as dimensões

da matriz são indexadas em uma ordem única de páginas do espaço conceitual. Dessa forma o valor da característica do segmento de ordem-*n* $(x_\alpha, x_\beta, ..., x_\omega)$ está localizado na interseção da linha x_α na primeira dimensão, na linha x_β na segunda dimensão e na linha x_ω na *n*-ésima dimensão da matriz de característica. Deve-se notar que M_{r^n} cobre todos os segmentos de ordem-*n* pertencentes a ε^n, por exemplo, em um espaço conceitual de cem páginas tendo ε^2 como base, é gerada uma matriz M_{100^2} de 10.000 elementos. Por outro lado, o número de segmentos existentes em uma sessão está, geralmente, na ordem de dez. Portanto M_{r^n} é geralmente uma matriz esparsa. Para os elementos nos quais não há segmentos correspondentes na sessão é atribuído zero.

De forma a mapear uma sessão ao modelo FM equivalente, as matrizes de características apropriadas são extraídas para as características contidas na sessão. O conjunto inteiro de matrizes de características gerado para uma sessão em particular constitui o seu modelo FM:

$$U^{fm} = \left\{ M_{r^{n_1}}^{F_1}, M_{r^{n_2}}^{F_2}, M_{r^{n_3}}^{F_3}, ..., M_{r^{n_m}}^{F_m} \right\}.$$

Se $n = \max(n_1, n_2, ..., n_m)$ então U^{fm} é um modelo FM de ordem-*n*.

6.1.1.4 Modelo de sessão

Nas seções anteriores foram mostradas as características de uma sessão particular e a estrutura de dados que a representa, mas não foi mostrado como os valores de diferentes características de uma sessão são extraídos para formar as matrizes de características do modelo FM. Vale relembrar que as características de uma sessão são armazenadas em termos das características de seus segmentos, dessa forma, é suficiente mostrar como extrair características de um segmento fictício *E*:

- Para a *Visita (V)*, conta-se o número de vezes que *E* ocorre na sessão $(V \geq 0)$. Segmentos podem parcialmente se sobrepor, então contanto que não haja nenhuma página sobreposta em dois segmentos, diz-se que esses segmentos são distintos. Por exemplo, a sessão $x_1 \to x_2 \to x_2 \to x_2 \to x_1$ tem

um total de quatro segmentos de ordem dois incluindo uma ocorrência de (x_1, x_2), duas ocorrências de (x_2, x_2) e uma ocorrência de (x_2, x_1);

- Para a *Seqüência (S),* acha-se a posição relativa de cada ocorrência de *E* e armazena-se a média aritmética como *S* para *E* $(S > 0)$. De forma a encontrar a posição relativa dos segmentos, eles devem ser numerados de acordo com a ordem em que aparecem na sessão. Na sessão $x_1 \overset{1}{\rightarrow} x_2 \overset{2}{\rightarrow} x_2 \overset{3}{\rightarrow} x_2 \overset{4}{\rightarrow} x_1$ por exemplo, os valores de *S* para os segmentos (x_1, x_2), (x_2, x_2) e (x_2, x_1) são 1, 2.5 $(= (2+3)/2)$, e 4 respectivamente;
- Para o *Tempo de visita (T),* adiciona-se o tempo gasto em cada ocorrência de *E* na sessão $(T \geq 0)$.

Exemplo 1. Nesse exemplo é ilustrado como extrair o modelo FM de uma sessão fictícia. Assume-se que a sessão *U* tenha sido capturada em um espaço conceitual $Z = \{x_1, x_2, x_3, x_4, x_5\}$ como segue:

$$x_1 \rightarrow x_3 \rightarrow x_2 \rightarrow x_2 \rightarrow x_2 \rightarrow x_1 \rightarrow x_3 \rightarrow x_5 \rightarrow x_3$$

Assume-se que o usuário tenha gastado 20 segundos em cada página durante suas atividades de navegação. Para simplificar, assume-se ε^1 como a base de *T* e ε^2 como a base tanto para *V* como para *S:*

$$\varepsilon^1 = \{(x_1), (x_2), (x_3), (x_4), (x_5)\},$$

$$\varepsilon^2 = \begin{Bmatrix} (x_1,x_1)\ (x_1,x_2)\ (x_1,x_3)\ (x_1,x_4)\ (x_1,x_5) \\ (x_2,x_1)\ (x_2,x_2)\ (x_2,x_3)\ (x_2,x_4)\ (x_2,x_5) \\ (x_3,x_1)\ (x_3,x_2)\ (x_3,x_3)\ (x_3,x_4)\ (x_3,x_5) \\ (x_4,x_1)\ (x_4,x_2)\ (x_4,x_3)\ (x_4,x_4)\ (x_4,x_5) \\ (x_5,x_1)\ (x_5,x_2)\ (x_5,x_3)\ (x_5,x_4)\ (x_5,x_5) \end{Bmatrix}$$

Para achar as posições relativas dos segmentos dentro da sessão, primeiro numeram-se os segmentos:

$$x_1 \overset{1}{\rightarrow} x_3 \overset{2}{\rightarrow} x_2 \overset{3}{\rightarrow} x_2 \overset{4}{\rightarrow} x_2 \overset{5}{\rightarrow} x_1 \overset{6}{\rightarrow} x_3 \overset{7}{\rightarrow} x_5 \overset{8}{\rightarrow} x_3.$$

Para a seqüência (x_1, x_3) por exemplo, *V* e *S* são computados como $V = 1+1 = 2$ e $S = (1+6)/2 = 3.5$. O valor de *T* para o segmento (x_1) por exemplo, é calculado como $T = 20+20 = 40$. $U^{fm} = \{M_{5^2}^V, M_{5^2}^S, M_5^T\}$ é o modelo FM final extraído de U:

$$M_{5^2}^V = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \; M_{5^2}^S = \begin{bmatrix} 0 & 0 & 3.5 & 0 & 0 \\ 5 & 3.5 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 \end{bmatrix}, \; M_5^T = \begin{bmatrix} 40 & 60 & 60 & 0 & 20 \end{bmatrix}.$$

6.1.1.5 Modelo de grupo

Para a modelagem de grupos é necessária uma medida de similaridade para quantificar a distância entre as sessões e um algoritmo de agrupamento para a construção dos grupos. Mais do que isso o modelo dos grupos deve ser escalável. Sites da Web populares são visitados por um enorme número de usuários e nesse contexto pode-se usar qualquer medida de similaridade ou algoritmo de agrupamento, mas somente agrupar as sessões não é o suficiente. Se um grupo for modelado como somente um conjunto de sessões, qualquer análise posterior vai depender da cardinalidade do grupo, ou seja, do número de sessões contidas no grupo, o que não é uma solução escalável. Particularmente para a classificação em tempo real de sessões em grupos pré-determinados, o modelo do grupo deve ser um modelo "condensado" de tal forma que a complexidade e o tempo da classificação independam do número de membros do grupo.

Na abordagem proposta pelos idealizadores do modelo FM [131], os valores das características de todas as sessões pertencentes a um grupo são agregadas numa única *sessão virtual* chamada de centróide. O centróide pode ser visto como uma representação de todas as sessões do grupo, sendo na verdade o próprio modelo do grupo. Conseqüentemente, a complexidade atrelada a qualquer análise posterior do grupo vai ser independente de sua cardinalidade.

De forma a agregar as características das sessões nas correspondentes características do modelo do grupo é suficiente agregar as características de cada

segmento. Assumindo que o valor da característica *F* para cada segmento *E* é dado por *F(E)*, para achar o valor agregado de *F(E)* aplica-se a média aritmética simples para os valores de *F(E)* em todas as sessões do grupo. Dessa forma se M^F é a matriz de características para a característica *F* para o modelo do grupo e M_i^F é a matriz de características para a característica *F* da *i-ésima* sessão do grupo, cada elemento de M^F é computado através da agregação dos elementos correspondentes de todas as matrizes M_i^F. Dessa forma a matriz de características agregada para cada característica *F* do modelo de grupo C^{fm} é computada como:

$$M^F = \frac{1}{N}\sum_{i=1}^{N} M_i^F$$

onde *N* é a cardinalidade do grupo *C.* A mesma função de agregação pode ser utilizada de forma incremental, quando o modelo do grupo já estiver sido criado e só se desejar atualizá-lo assim que uma nova sessão U_j se juntar ao grupo:

$$M^F \leftarrow \frac{1}{N+1}(N \times M^F + M_j^F).$$

Essa propriedade é chamada pelos autores de agrupamento dinâmico e ele permite que os grupos descobertos se tornem adaptativos.

Esse procedimento é repetido para cada característica do modelo FM. O resultado final da agregação é um conjunto de matrizes de características agregadas que constituem o modelo FM do grupo:

$$C^{fm} = \left\{M^{F_1}, M^{F_2}, ..., M^{F_n}\right\}.$$

Portanto o modelo FM pode tanto modelar sessões quanto grupos.

Exemplo 2. Esse exemplo ilustra como extrair o modelo FM de um grupo simples. Assumindo que U_1 e U_2 são os únicos membros do grupo *C*, e M_1^F e M_2^F são as suas matrizes de características para a característica *F*. As seguintes equações são auto-explicativas:

$$M_1^F = \begin{bmatrix} 0 & 2 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad M_2^F = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$M^F = \frac{1}{2}(M_1^F + M_2^F) = \begin{bmatrix} 0.5 & 1.5 & 0 \\ 0.5 & 0 & 0.5 \\ 1 & 0 & 0.5 \end{bmatrix}.$$

A única desvantagem em se agregar sessões de um grupo em um centróide, é a perda de precisão, pois o grupo perde a capacidade de representar cada sessão individualmente. Quanto mais extensiva for a agregação menos preciso é o modelo, mas se não for assim qualquer análise ou classificação posterior dependerá da cardinalidade do grupo e, dessa forma, o modelo não escala. Esse é o preço que se paga pela escalabilidade. Mas o modelo FM é completo o suficiente de modo a permitir um certo balanceamento desse compromisso [131].

6.2 Descoberta dos padrões

O agrupamento é considerado como o principal método de aprendizagem não supervisionada, e tem o objetivo de achar padrões em conjuntos de dados não rotulados. Dados rotulados referem-se a possíveis categorias às quais registros de dados podem pertencer. Por exemplo, registros de dados médicos que categorizam um paciente em diabético ou não diabético. As sessões capturadas dos usuários anônimos da Web não possuem qualquer rótulo, pois não se sabe previamente as categorias às quais elas podem estar associadas.

O modelo FM é composto por um conjunto de matrizes, que para o cálculo de similaridade são convertidas em vetores do espaço (ver Exemplo 3), sendo assim, as possíveis medidas de similaridade são baseadas na distância geométrica ou euclidiana entre elas. A classe de algoritmos aplicáveis nesse contexto é denominada *agrupamento baseado na distância*.

As seções subseqüentes introduzem a medidas de similaridade e os algoritmos escolhidos para o agrupamento das sessões no ONTOMUW.

6.2.1 Medidas de similaridade

Uma medida de similaridade é uma métrica que quantifica a noção de "similaridade". De forma a capturar o comportamento de usuários da Web, sessões são agrupadas de acordo com o grau de similaridade existente entre elas. Como o modelo FM é um modelo baseado na distância, as funções de distância freqüentemente utilizadas para o modelo vetorial, tais como a função co-seno e a distância euclidiana, podem ser utilizadas. No entanto há um problema de superestimação com a distância euclidiana pura, portanto uma nova medida chamada distância euclidiana projetada foi definida de forma a amenizar esse problema [131].

Independente da medida de similaridade usada, cada matriz de característica deve ser utilizada como uma matriz unidimensional de forma a facilitar o cálculo. Para ilustrar isso, assume-se que todas as linhas de uma matriz de características são concatenadas em uma ordem pré-determinada de linhas e dimensões. O resultado será uma lista unidimensional de valores de características. Essa lista ordenada é na verdade um vetor de valores de características em $R^{(r^n)}$, onde *r* é a cardinalidade do espaço conceitual. Supondo que se queira medir a similaridade entre duas sessões U_1^{fm} e U_2^{fm}, cada modelo de sessão é então composto de um conjunto de vetores de características, um para cada característica capturada no modelo FM. Para cada característica F_i, a medida de similaridade é aplicada aos dois vetores de características F_i de U_1^{fm} e U_2^{fm} de forma a computar a similaridade D^{F_i}. Como a similaridade entre U_1^{fm} e U_2^{fm} deve ser baseada em todas as características de FM, a similaridade total é computada como uma média ponderada de todas as similaridades de todas as características:

$$D^F = \sum_{i=1}^{m} \omega_i \times D^{F_i} \qquad (\sum_{i=1}^{m} \omega_i = 1)$$

onde *m* é o número de características do modelo FM. O peso ω_i é dependente da aplicação e determina a importância relativa de cada característica para o cálculo da similaridade entre as sessões.

6.2.1.1 Distância euclidiana pura projetada (DEPP)

Como o modelo FM é convertido em vetores para os cálculos de similaridade entre as sessões, qualquer função de distância tradicionalmente usada com vetores pode ser aplicada, onde as mais comuns são a função co-seno e a distância euclidiana pura. No entanto os autores do modelo FM mostram experimentalmente que a distância euclidiana pura sofre de um problema de superestimação da dissimilaridade entre as sessões. Isso ocorre porque na maioria das vezes os usuários navegam um caminho similar a apenas um subconjunto do padrão de navegação representado pelo grupo C^{fm}. De forma a calcular a similaridade entre a sessão U^{fm} e o grupo C^{fm} deve-se evitar comparar as partes de C^{fm} não cobertas por U^{fm}, pois, de outra forma, a sua dissimilaridade será superestimada.

De forma a amenizar esse problema o modelo FM suporta uma outra medida de similaridade baseada na distância euclidiana pura chamada de distância euclidiana projetada.

Assumindo que $\vec{A}$ e $\vec{B}$ são dois vetores de características do mesmo tipo, pertencentes a uma sessão e a um modelo de grupo respectivamente, e que cada vetor é composto de *N* elementos, para calcular a similaridade entre $\vec{A}$ e $\vec{B}$, $DEPP$ calcula a distância euclidiana pura entre $\vec{A}$ e a projeção de $\vec{B}$ nas coordenadas nas quais $\vec{A}$ tenha elementos diferentes de zero:

$$DEP\left(\vec{A},\vec{B}\right)=\left(\sum_{i=1,a_i\neq 0}^{N}\left(a_i-b_i\right)^2\right)^{\frac{1}{2}},$$

onde $DEPP\in[0,\infty]$ sendo que $DEPP$ não é comutativa.

Os elementos de $\vec{A}$ que são diferentes de zero correspondem àqueles segmentos que existem na sessão. Os elementos cujo valor é zero, por outro lado, estão relacionados ao restante dos segmentos do conjunto universal de segmentos. Sendo assim, a parte do grupo que não é coberta pela sessão é excluída da comparação de forma a evitar a superestimação. No exemplo a seguir é mostrado o uso da distância

euclidiana pura e depois o impacto do uso da distância euclidiana pura projetada na correção do problema da superestimação.

Exemplo 3. Assume-se que os grupos C e $C^{'}$ são representados pelos seguintes centróides:

$$C : x_2 \to x_3 \to x_2 \to x_3 \to x_2 \to x_1$$

$$C^{'} : x_3 \to x_1 \to x_3$$

e a sessão U é capturada como:

$$U : x_2 \to x_3 \to x_2 .$$

O objetivo então é selecionar o grupo que seja mais similar a U. Assumindo que a única característica capturada é *S,* os vetores da característica *S* de C, $C^{'}$ e U são mostrados a seguir:

$$M_C^S = \begin{bmatrix} 0 & 0 & 0 \\ 5 & 0 & 2 \\ 0 & 3 & 0 \end{bmatrix} \Rightarrow V_C^S = [0 \;\; 0 \;\; 0 \;\; 5 \;\; 0 \;\; 2 \;\; 0 \;\; 3 \;\; 0],$$

$$M_{C^{'}}^S = \begin{bmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \Rightarrow V_{C^{'}}^S = [0 \;\; 0 \;\; 2 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0],$$

$$M_U^S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 2 & 0 \end{bmatrix} \Rightarrow V_U^S = [0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 2 \;\; 0].$$

Como observado da seqüência de páginas, U é um subcaminho de C, enquanto que não tem um segmento sequer em comum com $C^{'}$. Mas mesmo assim, utilizando a distância euclidiana pura (DEP[9]), acha-se erroneamente que U é mais similar a $C^{'}$ do que C:

---

[9] $DEP(\vec{A}, \vec{B}) = \left| \vec{A} - \vec{B} \right| = \left(\sum_{i=1}^{N} (a_i - b_i)^2 \right)^{\frac{1}{2}}$

$$DEP\left(V_U^S, V_C^S\right) \approx 5.20 > DEP\left(V_U^S, V_{C'}^S\right) \approx 3.16.$$

Utilizando a distância euclidiana pura projetada esse problema é corrigido, como demonstrado abaixo:

$$DEPP = \left(V_U^S, V_C^S\right) = \sqrt{(1-2)^2 + (2-3)^2} \approx 1.41$$

$$DEPP = \left(V_U^S, V_{C'}^S\right) = \sqrt{(1-0)^2 + (2-0)^2} \approx 1.73$$

$$\Rightarrow DEPP\left(V_U^S, V_C^S\right) < DEPP\left(V_U^S, V_{C'}^S\right).$$

Já que $DEPP$ pode comparar sessões de tamanhos diferentes, essa medida é particularmente adequada para comparação em tempo real onde apenas uma parte da sessão está disponível em determinado instante. Como vimos, $DEPP$ é superior à DEP, pois além de corrigir o problema da superestimação ainda aperfeiçoa o tempo de classificação. Em Shahabi et al. (2003) [131] é demonstrada a superioridade de $DEPP$ também em relação a função co-seno.

6.2.2 Agrupamento dinâmico X Agrupamento periódico

Como visto na seção 6.2.5, o modelo FM de grupo é independente da cardinalidade, dessa forma, qualquer manipulação no modelo tem baixa complexidade. Uma vantagem dessa propriedade é que os grupos FM podem ser atualizados dinamicamente em tempo real.

Através do agrupamento dinâmico os sistemas baseados na MUW podem se adaptar às mudanças de comportamento dos usuários em tempo real. Novos grupos podem ser gerados dinamicamente e os grupos existentes podem se adaptar às novas tendências dos usuários. A Figura 20 mostra o procedimento necessário para o agrupamento dinâmico assim que uma nova sessão é capturada.

O agrupamento periódico é a abordagem tradicional para a atualização dos grupos. Essa abordagem resulta em alta precisão, mas não pode ser realizada em tempo real. De acordo com as experiências realizadas por Shahabi et al. (2003) [131] o agrupamento dinâmico mostra menor precisão em relação ao re-agrupamento periódico para a atualização dos grupos. Na verdade com o agrupamento dinâmico troca-se a

precisão pela adaptabilidade. O agrupamento dinâmico deve ser utilizado de forma híbrida, ou seja, o conjunto contendo todos os grupos deve ser atualizado em longos intervalos através de re-agrupamento, de forma a evitar divergências entre os grupos e o comportamento real dos usuários. Enquanto isso, o agrupamento dinâmico pode ser aplicado em tempo real de forma a adaptar os grupos para as mudanças on-line no comportamento de navegação dos usuários.

```
1.  Ache a distância/similaridade entre a sessão corrente e todos os
    grupos existentes no conjunto de grupos, através de qualquer
    medida de similaridade adequada;

// Qualquer uma das medidas de distância discutidas pode ser
// aplicada (função co-seno, distância euclidiana pura, distância
// euclidiana projetada)
// Essas medidas de similaridade são definidas baseadas na estrutura
// de dados do modelo FM

2.  Se não houver nenhum grupo mais próximo à sessão do que TDC {

       cria um novo grupo e usa o modelo FM da nova sessão como o
       modelo do grupo;

     } senão {

       atualiza o grupo mais próximo à sessão incluindo a sessão ao
       modelo do grupo;

       }

// TDC é um valor limiar específico do agrupamento dinâmico. Se a
// distância entre uma nova sessão e os grupos for maior que TDC,
// então é razoável criar um novo grupo porque nesse caso um novo
// comportamento foi identificado
```

Figura 20. Um algoritmo para o agrupamento dinâmico

Para o agrupamento periódico optou-se pelo K-Médias [143]. O algoritmo K-Médias é um algoritmo de partição bastante simples de implementar e funciona da seguinte forma. Primeiro determina-se o número de grupos que se deseja gerar. Em seguida escolhem-se aleatoriamente tantos indivíduos quantos forem o número de grupos escolhidos, esses indivíduos serão tidos como os grupos iniciais. Em seguida cada indivíduo é classificado em cada um desses grupos de acordo com a similaridade entre seus centróides. O processo é repetido até que todos os indivíduos tenham sido classificados nos grupos. Após isso o centróide do grupo é recalculado. No nosso caso os indivíduos são as sessões de usuários, e a atualização do centróide do grupo é feita

através da média aritmética entre todos os membros do grupo, como visto na seção 6.1.1.5.

O principal problema desse algoritmo é a determinação apriorística dos grupos. Isso é uma decisão dependente da aplicação e deve refletir a estrutura do site em questão. Se, por exemplo, determinado site compreende dois tópicos de interesse distintos, seria razoável determinar a criação de dois grupos, refletindo assim, esses dois interesses específicos. Fica como sugestão para trabalhos futuros a determinação de métodos que possam aprender a quantidade de grupos a serem gerados inicialmente de forma automática.

6.3 Captura da interação do usuário

Os dados de uso podem ser coletados de várias fontes incluindo caches do navegador no lado do cliente, cookies, logs de servidores Web ou logs de servidores proxy. Em se tratando de usuários anônimos, quanto mais preciso for o dado de uso, ou seja, quanto mais fiel o comportamento do usuário estiver representado nos dados de uso melhor será esta fonte para a mineração.

A captura dos dados no lado cliente não só fornece informações precisas como também elimina a necessidade de pré-processamento dessas informações, pois todas aquelas tarefas típicas do pré-processamento como limpeza de dados, identificação de usuários e sessões é feita transparentemente no lado cliente. Em Shahabi et al. (2001) [132] é introduzido uma solução baseada em applets Java remotos para a captura do comportamento de navegação do usuário no lado cliente. Os dados capturados pelos applets são enviados e armazenados no servidor como unidades semânticas distintas, não precisando, portanto, que as sessões dos usuários sejam re-identificadas posteriormente.

As principais desvantagens associadas a essa abordagem estão: na necessidade que os usuários têm de cooperarem no sentido de terem o Java habilitado em seus navegadores e o tempo extra no descarregamento de cada página, já que o applet é descarregado junto com cada página. Mas nos dias de hoje praticamente todos os navegadores já vem com a máquina virtual do Java habilitada como padrão, e, além

disso, o descarregamento de pequenos applets é quase imperceptível, dependendo da velocidade da conexão. Considerando as vantagens que a captura do comportamento de navegação no lado cliente traz, e com advento da internet rápida, argumentamos que esses pequenos gargalos tendem a sumir.

A abordagem a ser apresentada foi inspirada na mesma idéia proposta por Shahabi et al. (2001) [132], a principal diferença está no fato de que em vez de apenas um applet fazendo o serviço de captura do comportamento de navegação do usuário, tem-se dois agentes autônomos, um agente chamado de *Interfaceador* e um agente chamado de *Modelador*. Isso funciona da seguinte forma:

- Cada página do site a ser monitorado precisa ter uma chamada a um applet que vai ser responsável por criar os agentes e servir de elo entre os agentes e o navegador. Para isso, basta incluir uma tag HTML em cada página do site referenciando o applet. Isso pode ser feito com a ajuda de ferramentas off-line que percorram o diretório do site e incluam a tag em cada página encontrada automaticamente. O applet precisa ser pequeno e leve, de forma a ser rapidamente carregado pelo navegador. A tag HTML deve parecer com:

  ```
  <APPLET CODEBASE="/java"
  CODE="AppletAgents" WIDTH=1 HEIGHT=1>
  <PARAM NAME="PAGE_NAME"
  VALUE="http://maae.deinf.ufma.br/principal.htm">
  </APPLET>.
  ```

- Assim que o usuário acessa a primeira página do site, sendo que primeira aqui não significa necessariamente a página principal (“*home*”), o applet é carregado e instruído a criar os agentes *Interfaceador* e *Modelador* na máquina virtual que acompanha o navegador. Perceba que cada usuário vai possuir dois agentes rodando transparentemente, no lado cliente, em seu favor. O applet Java roda na máquina virtual embutida no próprio navegador, então por natureza, tem habilidades para perceber quando um usuário entra e deixa uma página, ou ainda

quando o usuário fecha o navegador. Além disso, uma vez o applet e agentes carregados pela primeira vez na máquina virtual do navegador eles não precisam ser re-criados a cada página visitada. Os agentes podendo se comunicar com o applet, terão todas as informações que precisam, tais como as interações do usuário com o navegador, tempo de permanência na página do usuário, a URL visitada, e ainda saber quando o usuário fecha o software de navegação.

6.4 Personalização da interface

Como visto na seção 3.3 do capítulo 3, existem várias funções de personalização (*memorização*, *adaptação*, *orientação* e *suporte* à *execução de tarefas*). Para esta primeira versão do ONTOMUW escolheu-se trabalhar com a função de *orientação*. Essa função é responsável por assistir o usuário na rápida obtenção de informações de interesse, assim como também oferecer rotas alternativas de navegação. No caso do ONTOMUW isso vai se dar em forma de recomendações de vínculos de hipertexto.

A abordagem escolhida para personalização na ONTOMUW foi baseada na filtragem colaborativa, onde a personalização independe da análise de conteúdo e é realizada através da identificação de características comuns entre os usuários. Isso é bastante comum na MUW, onde primeiro descobre-se grupos de usuários com características similares e depois se classifica o usuário corrente em um desses grupos (Figura 3). Nesse caso, as recomendações são baseadas nas páginas visitadas por outros usuários e no grau de interesse dos mesmos nessas páginas.

O modelo de adaptação é criado através do grupo no qual o usuário foi classificado. Isso se dá da seguinte forma. A primeira coisa a fazer é identificar a página atual do usuário, pois a partir daí pode-se saber em que posição ele está na matriz universal de segmentos. Vale lembrar que a matriz universal de segmentos possui todos os segmentos possíveis do site em questão, e sendo assim, serve como um mapa para localizar a posição de navegação do usuário corrente. A partir daí pode-se recomendar página(s) baseado no comportamento de navegação dos membros do grupo, comportamentos esses quantificados através das matrizes de características. Para isso,

basta verificar os maiores valores dos segmentos das matrizes do grupo de acordo com o segmento sendo visitado pelo usuário corrente. Em qual característica vai se basear a recomendação é uma questão dependente da aplicação, pois se pode, por exemplo, basear as recomendações nos segmentos que passaram mais tempo sendo visitados, ou então nos segmentos mais visitados pelos membros do grupo. Caso haja empate entre dois ou mais segmentos pode-se utilizar os valores das outras matrizes como critério de desempate. O Exemplo 4 abaixo ajuda a esclarecer essas idéias.

Exemplo 4. Para este exemplo deve-se considerar o mesmo espaço de páginas do Exemplo 1 e a base ε^2 para a matriz de visitas. A matriz de segmentos para esse espaço de páginas é visualizada a seguir.

$$\varepsilon^{(2)} = \begin{Bmatrix} (x_1,x_1)\,(x_1,x_2)\,(x_1,x_3)\,(x_1,x_4)\,(x_1,x_5) \\ (x_2,x_1)\,(x_2,x_2)\,(x_2,x_3)\,(x_2,x_4)\,(x_2,x_5) \\ (x_3,x_1)\,(x_3,x_2)\,(x_3,x_3)\,(x_3,x_4)\,(x_3,x_5) \\ (x_4,x_1)\,(x_4,x_2)\,(x_4,x_3)\,(x_4,x_4)\,(x_4,x_5) \\ (x_5,x_1)\,(x_5,x_2)\,(x_5,x_3)\,(x_5,x_4)\,(x_5,x_5) \end{Bmatrix}$$

Consideremos agora, que o usuário tenha sido classificado em determinado grupo, e que após a classificação o usuário se encontra na página x_3. A partir daí, deve-se identificar os possíveis segmentos em que esse usuário está de acordo com a página atual. Como a ordem da matriz de segmentos é dois, os únicos possíveis segmentos em que o usuário atual pode estar se encontram na linha três da matriz de segmentos. Feito isso, basta recomendar a próxima página a ser visitada baseada no segmento mais visitado da matriz de visitas do grupo. Como a ordem do segmento é dois, só se pode recomendar a próxima página do segmento. De acordo com a matriz de visitas do grupo, visualizada logo abaixo, se percebe que na linha três o segmento mais visitado é (x_3,x_1), pois possui valor 4.1, maior valor entre todos os segmentos da linha. Como o usuário já está na página x_3, a página a ser recomendada é x_1.

$$C^{(V)} = \begin{Bmatrix} 2.3 & 4.1 & 1 & 3 & 1.4 \\ 1.6 & 2.1 & 4 & 3 & 3.1 \\ 4.1 & 1.3 & 1 & 2 & 2.1 \\ 1.3 & 3.3 & 3 & 1 & 4.1 \\ 1.2 & 2.1 & 2 & 5 & 6.1 \end{Bmatrix}$$

Essa estratégia pode ser estendida para qualquer ordem escolhida pelo desenvolvedor. Recomendações mais interessantes podem surgir através de ordens mais altas da matriz de segmentos.

Apesar da simplicidade dessa abordagem, deve-se ter alguns cuidados de modo a evitar recomendações óbvias. Se, por exemplo, o segmento mais visitado da matriz de grupos fosse o segmento (x_3, x_3), a página recomendada seria x_3, justamente a página que o usuário está visitando no momento, o que não faria muito sentido. Mas isso pode ser resolvido através de heurísticas simples, como por exemplo, nunca analisar segmentos constituídos somente de páginas repetidas.

6.5 Armazenamento e manutenção dos dados de uso

Como se pôde notar na análise de domínio, os dados de uso são criados e mantidos por um papel em particular, o papel *Aquisitor*. Este papel será associado a um agente mais adiante nesse capítulo o qual estará continuamente recebendo modelos de usuários dos vários usuários do site sendo monitorado. Esses modelos são armazenados em um arquivo de uso quando recebidos.

Um agente de software possui, por natureza, habilidades para realizar tarefas de natureza concorrente, como por exemplo, o tratamento de requisições simultâneas. Esse é justamente o caso aqui. Cada vez que uma sessão de usuário termina, o modelo representando esse usuário é enviado ao agente *Aquisitor.* O agente *Aquisitor* é responsável por receber esses modelos, tratá-los e mantê-los adequadamente em um arquivo de uso. Podem acontecer situações onde duas ou mais sessões terminem ao mesmo tempo, e sendo assim, vários modelos podem ser enviados simultaneamente ao agente *Aquisitor*.

O formato que os dados de uso vão assumir deve ser antes de tudo um formato padrão, que forneça interoperabilidade, independência de plataforma e de fabricante. Sendo assim optou-se pelo RDF ("*Resource Description Framework*").

O RDF é um padrão (tecnicamente uma recomendação do W3C) para a descrição de recursos. Um recurso pode ser pensado como qualquer coisa identificável, como por exemplo, carro, mesa, cadeira, uma página Web, uma pessoa, etc.

O RDF baseia-se na idéia de identificar objetos usando URIs ("*Uniform Resource Identifiers*"), e descrever recursos em termos de propriedades simples. Isto permite ao RDF representar proposições simples sobre estes recursos como grafos semânticos (Figura 21).

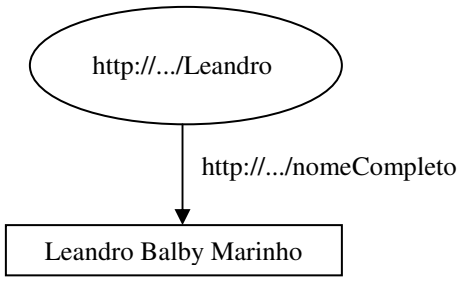

Figura 21. Exemplo de um grafo semântico em RDF

A figura acima mostra o seguinte. O recurso Leandro é mostrado em uma elipse e representado pela URI *http://.../Leandro*. A URI serve basicamente para identificar o recurso unicamente, não sendo, portanto, necessário que ela aponte para algum endereço real da internet.

Os recursos têm propriedades. Nesse caso, está-se interessado na propriedade "*nomeCompleto*" do recurso. As propriedades são representadas por arcos rotulados com o nome da propriedade e também são representadas por URIs.

Os recursos têm valores. Nesse pequeno exemplo o valor corresponde a Leandro Balby Marinho. Aqui esse valor é representado por um literal, mas há casos onde o valor pode ser outro recurso. Literais são mostrados por retângulos.

O RDF também suporta a definição de esquemas para os dados. Por ser simples e fornecer um modelo de dados rico e bem definido, o RDF está sendo usado como um dos pilares da emergente Web Semântica [97]. Através dos esquemas pode-se criar vários tipos de arquivos de log diferentes, de acordo com o propósito específico da

aplicação. No decorrer da dissertação será mostrado um exemplo de log em RDF gerado pelo agente *Aquisitor*.

6.6 Projeto arquitetural

O *Projeto Arquitetural* consiste das seguintes subtarefas: definição da arquitetura através do reuso de padrões arquiteturais, modelagem de agentes, modelagem do framework e modelagem de atividades. Os produtos desta tarefa são: um modelo de agentes, um modelo do framework e um modelo de atividades.

6.6.1 Reuso de padrões arquiteturais

Padrões de software são desejáveis para o desenvolvimento de software baseado na reutilização. Em particular para o desenvolvimento de sistemas multiagente, pois eles constituem-se de soluções melhoradas para problemas recorrentes e podem ser aplicados tanto no nível conceitual, arquitetural ou de implementação.

Um padrão representa uma solução bem sucedida a um problema recorrente. Ele mostra não só a solução, como também as suas restrições e o contexto em que se deve aplicar essa solução. Um padrão possui uma forma ou estrutura identificável e reconhecida, em diferentes domínios, que ajuda a diminuir a complexidade do software em várias fases do seu ciclo de vida [126].

Um padrão arquitetural apresenta o esquema ou organização estrutural fundamental de sistemas de software ou hardware; ele provê um conjunto de subsistemas pré-definidos, especifica suas responsabilidades e inclui regras e guias para organizar os relacionamentos entre eles [126], [127], [136].

Não foi encontrado nenhum padrão formal na literatura para o problema específico aqui tratado. Sendo assim, o ONTOMUW se baseou na arquitetura genérica definida por Mobasher et al. (2000) [109] para os sistemas de personalização na Web baseados na MUW (Figura 3). De certa forma essa arquitetura representa um padrão, pois além de definir uma boa solução para a estruturação dos módulos do sistema, também tem sido amplamente utilizada pela literatura, comprovando assim a sua aceitação e aplicabilidade. O que se tem que fazer então é estruturar os agentes do ONTOMUW de acordo com essa arquitetura.

A arquitetura de Mobasher et al. (2000) [109] é dividida em duas camadas. Uma camada responsável pelo monitoramento, modelagem do usuário corrente e execução da adaptação na interface do usuário e outra camada responsável pelo pré-processamento dos dados de uso e descoberta de padrões de navegação. A camada de descoberta de padrões de navegação fornece serviços para a primeira camada, que são justamente os padrões de uso descobertos, necessários para a realização da personalização. A divisão dos agentes nas camadas se dá de acordo com a similaridade entre suas responsabilidades. Sendo assim, similarmente a Mobasher et al. (2000) [109] tem-se duas camadas, uma chamada de *camada de processamento de informações do usuário* e outra chamada de *camada de descoberta de padrões de navegação* (Figura 22).

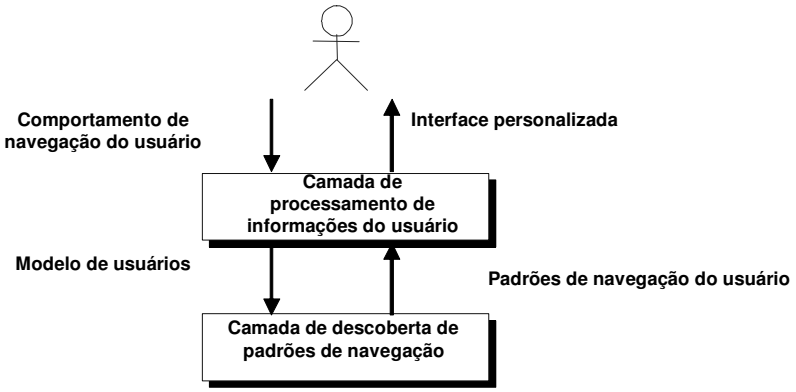

Figura 22. Estrutura de camadas da arquitetura do ONTOMUW

Um grupo de agentes é associado a cada camada, sendo que cada camada está estruturada da seguinte forma.

Camada de processamento de informações do usuário.

Nessa camada, um agente de interface e um agente de modelagem de usuários são criados para cada usuário. Esses agentes possuem as seguintes responsabilidades:

- Construção e apresentação de interface personalizada;
- Captura implícita do perfil do usuário[10];

[10] Um perfil de usuário designa a coleção de interesses, objetivos, planos e preferências de um usuário e é formalmente representado em um modelo de usuários.

- Criação e atualização de um modelo representando o perfil do usuário.

Camada de descoberta de padrões de navegação do usuário.

Essa camada é composta por um agente minerador e um agente aquisitor de dados, e dentre as suas responsabilidades estão:

- Armazenamento e atualização de um repositório de dados de uso;
- Descoberta de grupos de usuários com comportamento de navegação similar e a identificação de qual grupo o usuário corrente pertence.

É necessário ressaltar que a primeira camada está distribuída entre os usuários do site sendo monitorado, pois cada usuário terá a primeira camada da arquitetura rodando transparentemente na máquina virtual do seu navegador. Enquanto que a camada de descoberta de padrões se situa em algum servidor remoto, sem que necessariamente seja o mesmo servidor em que o site está hospedado. Tem-se então vários agentes modeladores e interfaceadores requisitando serviços de somente um agente *Aquisitor* e um agente *Minerador*.

Essa arquitetura sugere um protocolo de interação similar ao FIPA-REQUEST [48], onde agentes solicitam ações ou serviços a serem executados por outros agentes. A Figura 23 mostra um diagrama de seqüência baseado na AUML [115] onde é apresentada a comunicação interna entre os agentes de cada camada, assim como a comunicação entre os agentes de camadas diferentes. Na fase de projeto detalhado os rótulos das mensagens devem ser substituídos por performativas de uma linguagem de comunicação entre agentes, tal como a FIPA-ACL [47], por exemplo.

O fluxo de interações se inicia com o agente *Interfaceador* capturando as informações provenientes das atividades de navegação do usuário. Uma vez essas informações capturadas elas são enviadas ao agente *Modelador*. O agente *Modelador*, por sua vez, utiliza essas informações para criar ou atualizar o modelo representando a sessão do usuário corrente. De posse desse modelo, o agente *Modelador* pode solicitar ao agente *Aquisitor*, caso a sessão do usuário tenha terminado, a gravação do modelo contendo a sessão do usuário corrente. De outra forma, ele pode solicitar a classificação do usuário corrente ao agente *Minerador*.

Uma vez o usuário classificado em um dos grupos pré-descobertos, o agente *Minerador* notifica o agente *Modelador* sobre o grupo no qual o usuário corrente foi classificado. Feito isso, o agente *Modelador* constrói o modelo de adaptação para o usuário corrente e o repassa ao agente *Interfaceador*, que será responsável por aplicar os efeitos adaptativos na interface do usuário.

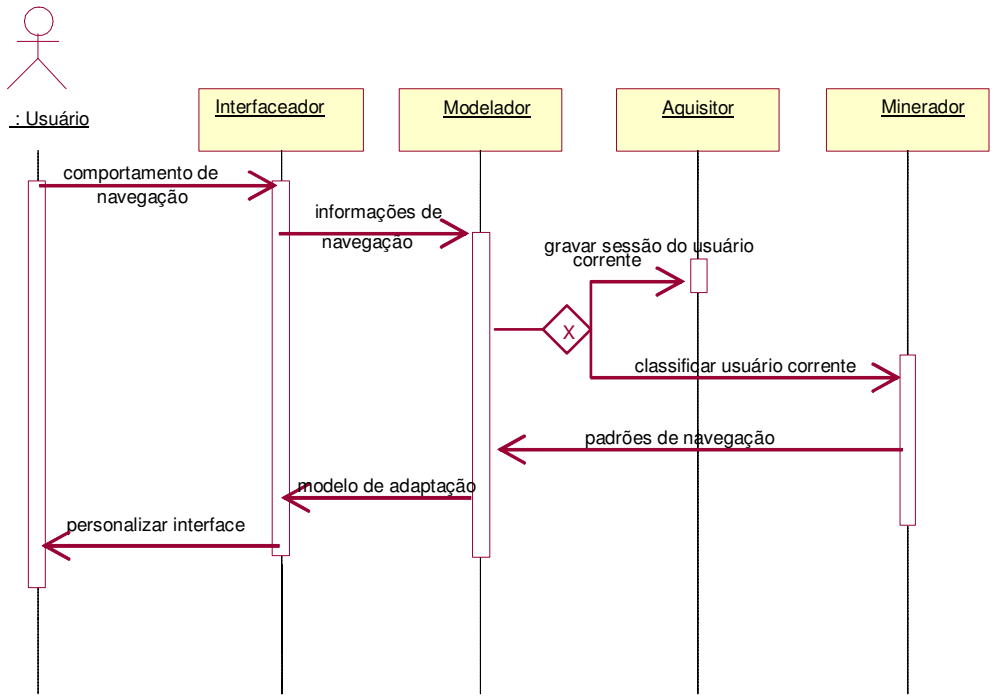

Figura 23. Diagrama de interações mostrando as interações entre as camadas e os agentes do ONTOMUW

6.6.2 Modelagem de agentes

Na *Modelagem de Agentes* são identificados os agentes que irão compor o modelo do framework. Os agentes são identificados a partir dos papéis especificados no modelo de papéis. Inicialmente um papel é desempenhado por um agente. Porém, podem existir situações onde um agente pode ser composto de um ou mais papéis ou situações onde um papel pode originar mais de um agente, de forma a atender os requisitos de coesão, desempenho e reusabilidade.

Cada agente irá realizar um conjunto de atividades necessárias para o cumprimento de suas responsabilidades. Durante esse processo de refinamento pode ser identificado que alguma atividade ou um conjunto relacionado de atividades são

executados por vários agentes. Neste caso pode ser apropriada a criação de um agente independente. O produto dessa subtarefa é um modelo de agentes composto de agentes e dos seus respectivos papéis desempenhados. O modelo de agentes é um organograma de dois níveis: o primeiro nível é composto do agente e o segundo nível é composto do papel.

O modelo de agentes é bastante simples como mostra a Figura 24. Como se pode ver, o agente *Interfaceador* desempenha os papéis: *Monitor e Interfaceador*. O agente *Modelador* os papéis: *Modelador e Personalizador* O agente *Minerador*, por sua vez, desempenha os papéis: *Minerador e Classificador*. Por último, o agente *Aquisitor* exerce somente um papel: *Aquisitor*.

A decisão de juntar mais de um papel em um agente é tomada de acordo, principalmente, com requisitos de coesão funcional. Então no caso do agente *Interfaceador*, por exemplo, tanto os papéis *Interfaceador* (adapta a interface para a visualização das recomendações) quanto o *Monitor* (monitora o usuário através de suas interação), lidam diretamente com a interface do usuário e, portanto, faz sentido agrupá-los em um mesmo agente.

6.6.3 Modelagem do framework

De posse dos agentes e de como eles se organizam, através modelo arquitetural, pode-se construir o modelo do framework.

A *Modelagem do framework* tem o propósito de integrar os agentes modelados para formar uma solução computacional ao problema tratado. Esses agentes são organizados de forma adequada, a fim de que o sistema resolva eficientemente o problema. O produto dessa subtarefa é o modelo do framework.

A MADEM apresenta o modelo do framework como um diagrama de colaboração baseado no diagrama de colaboração da UML, mostrando os agentes estruturados segundo o padrão arquitetural escolhido e as interações entre os agentes e entre os agentes e as entidades externas ao sistema.

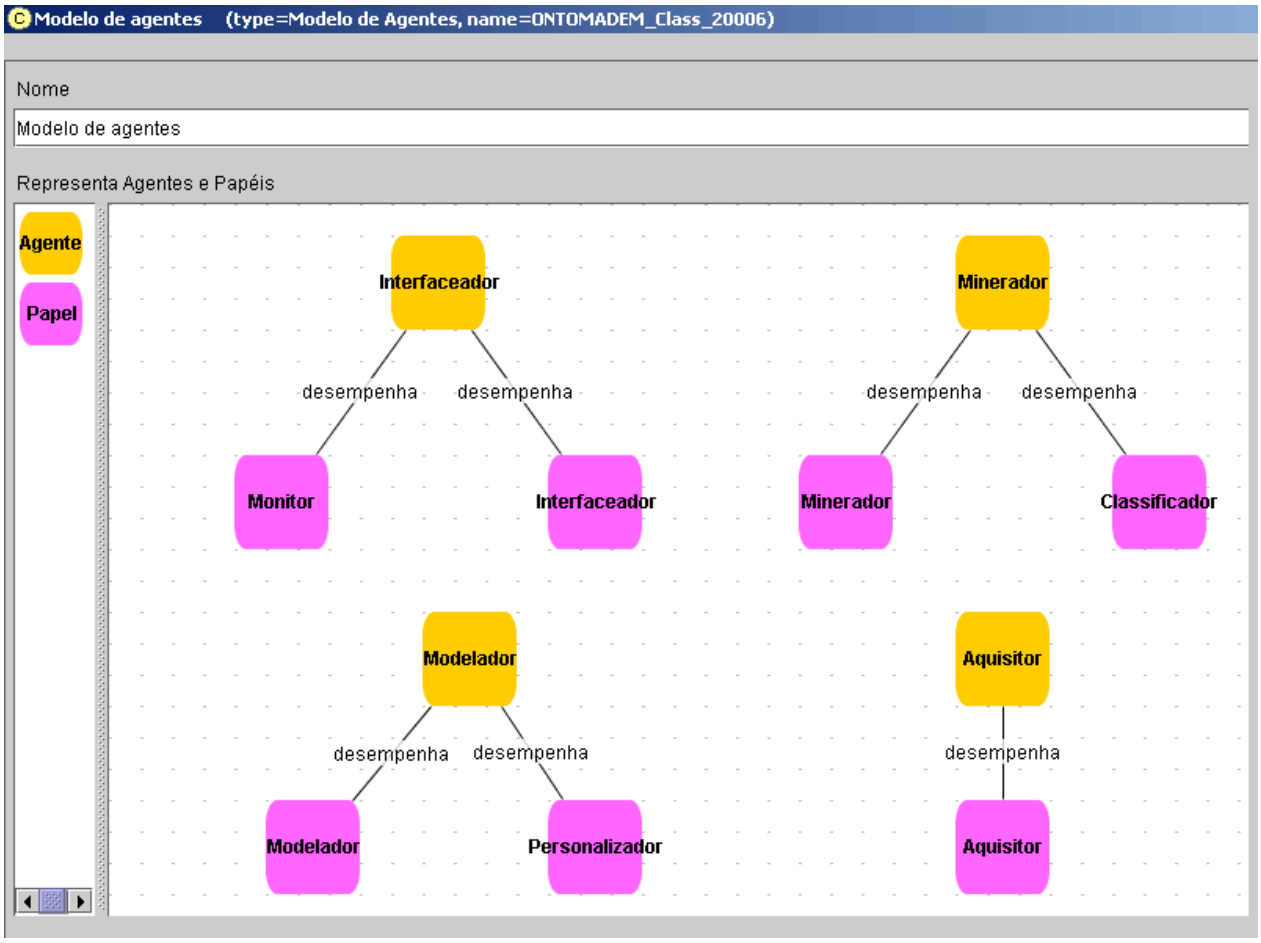

Figura 24. Modelo de agentes

A comunicação entre os agentes (Figura 25) é iniciada com a navegação do usuário (fluxo 1), o agente *Interfaceador* captura essas informações e as repassa ao agente *Modelador* (fluxo 2) que então as representa de acordo com o modelo FM (ver seção 6.2). Nesse ponto o agente *Modelador* pode realizar duas ações excludentes, ou (a) ele envia o modelo de usuários ao agente *Aquisitor* para que este possa armazenar o modelo em um arquivo de dados de uso, ou então (b) ele envia o modelo ao agente *Minerador* solicitando a classificação deste em algum dos grupos de usuários pré-existentes (fluxo 3). Se a sessão do usuário terminar, ou porque ele fechou o navegador, ou porque o tempo determinado para a sessão expirou, o *Modelador* envia o modelo ao *Aquisitor*. A condição que determina quando o *Modelador* irá envia o modelo ao *Minerador* é dependente da aplicação e deixada para o desenvolvedor. Essa condição geralmente está associada ao número de páginas visitadas, por exemplo, o desenvolvedor pode definir que a cada três páginas visitadas o *Modelador* deve enviar o modelo ao *Minerador*.

Dependendo para qual agente a mensagem for enviada o fluxo de ações vai tomar um caminho diferente.

(a) No caso da mensagem ter sido enviada ao *Aquisitor*, este vai armazenar os dados recebidos em um arquivo de dados de uso no formato RDF (fluxo 4). O agente *Minerador* possui um relógio interno que regula os intervalos em que o agente deve realizar o processo de mineração, então uma vez existindo dados no arquivo de uso, o *Minerador* vai acessar esses dados, e a partir deles, descobrir e representar os padrões na forma de grupos de usuários com comportamento de navegação similar (fluxo 5).

(b) Caso o *Modelador* solicite a classificação do usuário para o agente *Minerador* (fluxo 3), este último depois de classificar o usuário, retorna o grupo no qual o usuário foi classificado ao agente *Modelador* (fluxo 4), que então será responsável por criar o modelo de adaptação. Feito isso, o modelo de adaptação é enviado ao agente *Interfaceador* para que este possa aplicar as adaptações cabíveis na interface do usuário (fluxo 6).

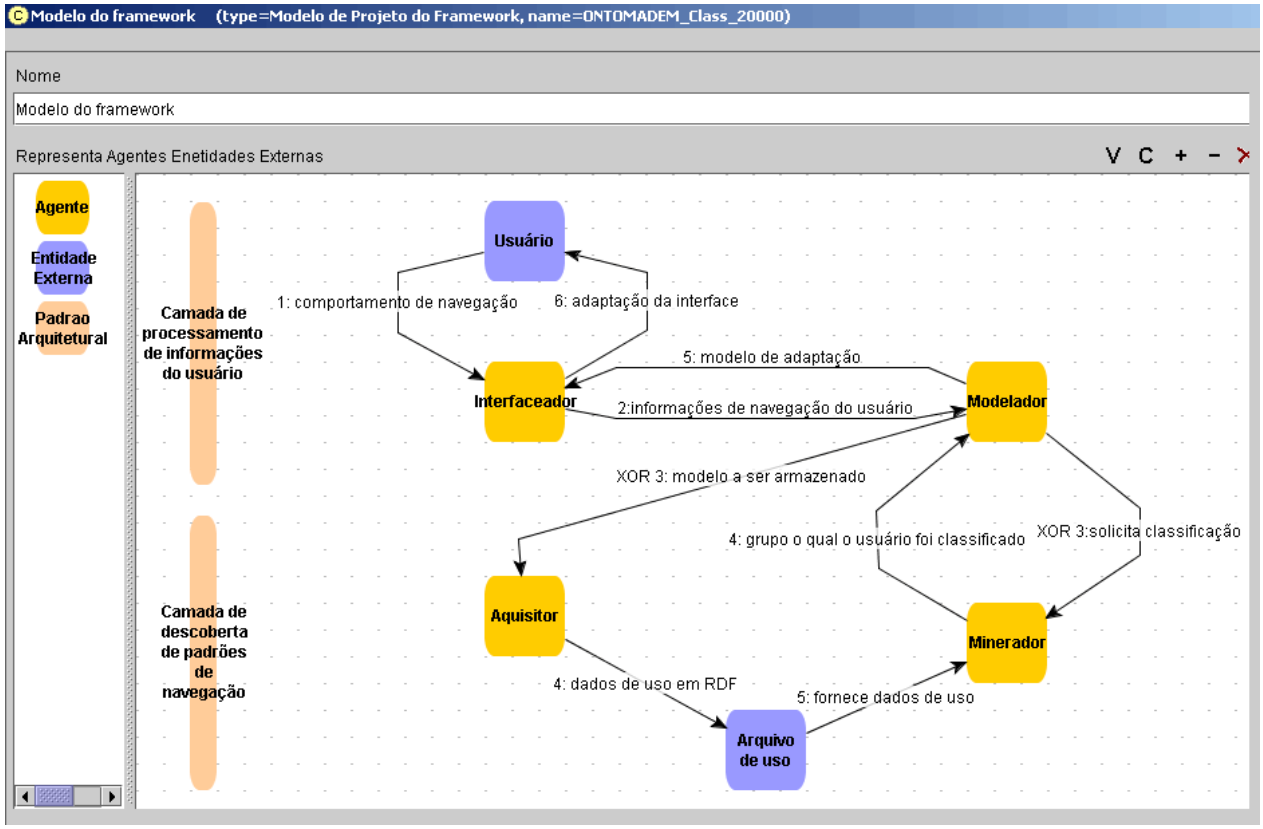

Figura 25. Modelo do framework ONTOMUW

6.6.4 Modelagem de atividades

A *Modelagem de atividades* representa as atividades dos agentes e suas interações. As atividades deste modelo são obtidas do modelo de agentes, pois no modelo de agentes estão os papéis desempenhados. O modelo de atividades é obtido utilizando-se a UML e é composto dos agentes, suas atividades e as interações entre as atividades (Figura 26). No projeto detalhado é apresentado um refinamento desse modelo de atividades, onde cada atividade mostrada aqui é explodida em subatividades refletindo as técnicas e algoritmos escolhidos para cada agente. A seguir o fluxo dessas atividades é brevemente comentado.

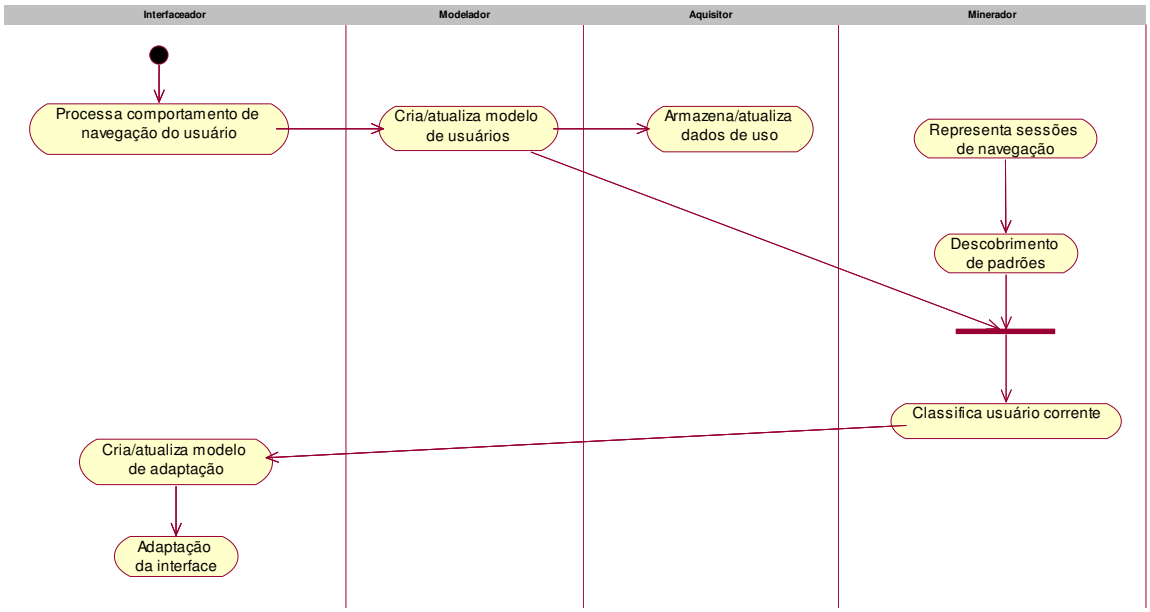

Figura 26. Modelo de atividades dos agentes

O *Interfaceador* é um agente fundamental no sistema, pois dele dependem, direta ou indiretamente, todos os outros agentes do sistema. Só é possível criar o modelo de usuários, armazenar os dados de uso e consequentemente realizar a mineração, através das informações capturadas pelo agente *Interfaceador.* Uma vez essas informações capturadas, o *Modelador* as utiliza para criar/atualizar o modelo de usuários. Os dados a serem armazenados pelo *Aquisitor* provêm justamente do modelo criado pelo agente *Modelador*, pois quando a sessão de navegação do usuário termina, esse modelo é enviado ao agente *Aquisitor*. O agente *Minerador* então utiliza o arquivo criado pelo

agente *Aquisitor* para iniciar o processo de mineração. O processo se inicia com a representação de todas as sessões contidas no arquivo em um formato apropriado (modelo FM) para a descoberta dos padrões. Uma vez os padrões descobertos eles podem ser utilizados para a personalização. Note que os padrões só podem ser utilizados se, obviamente, eles existirem. Por isso as barras de sincronização no diagrama. Elas indicam justamente que os padrões precisam ter sido descobertos antes que a classificação possa ser realizada.

6.7 Projeto Detalhado

A tarefa de projeto detalhado consiste das seguintes subtarefas: refinamento dos agentes, modelagem de comportamento e modelagem de interações entre agentes. Os produtos desta tarefa são: modelo de agentes refinado, modelo de comportamento e de interações entre agentes.

6.7.1 Refinamento dos agentes

O propósito dessa subtarefa é o detalhamento de cada um dos agentes que compõem o framework. Para o refinamento dos agentes é necessário realizar os seguintes passos: seleção do padrão de projeto detalhado e classificação dos agentes.

A seleção do padrão de projeto detalhado é feita da seguinte forma: a) analisa-se o problema ou subproblema que o agente pretende resolver; b) seleciona-se um ou mais padrões disponíveis que combinem com o problema e o contexto tratados, além de analisar as forças destes padrões. É notória a falta de catálogos, coletâneas e sistemas de padrões para sistemas multiagente. Não encontramos padrões que se apliquem aos nossos problemas, fica então como sugestão para trabalhos futuros a extração de padrões a partir da nossa solução.

Os agentes podem ser classificados segundo dois tipos básicos: reativo e deliberativo [128]. Um agente reativo é uma entidade que age usando um tipo de comportamento baseado em estímulo/resposta, seu comportamento é reduzido à execução de um conjunto de regras de condição-ação (do tipo se...então). Um agente deliberativo é uma entidade que possui um modelo simbólico interno. Ele elabora e seleciona planos ou ações, através de um processo baseado em raciocínio lógico, para

que possa atingir sua meta no contexto da situação em questão. As figuras seguintes (Figura 27, Figura 28, Figura 29, Figura 30) apresentam um resumo de cada agente do framework. As figuras mostram o nome do agente, os papéis desempenhados, o possível padrão de projeto utilizado, o tipo de agente (reativo, deliberativo), a descrição do agente e os seus comportamentos.

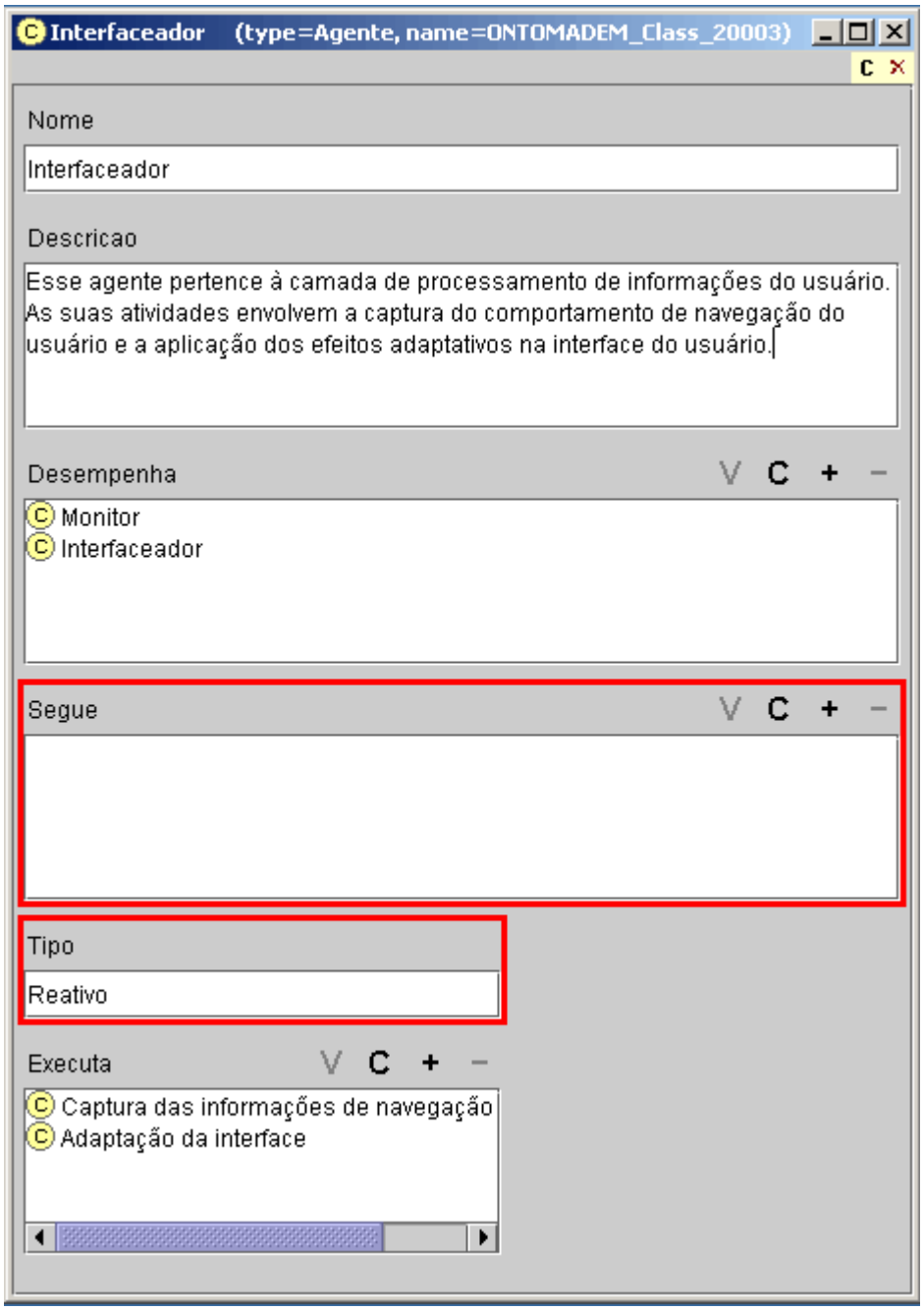

Figura 27. Descrição detalhada do agente Interfaceador

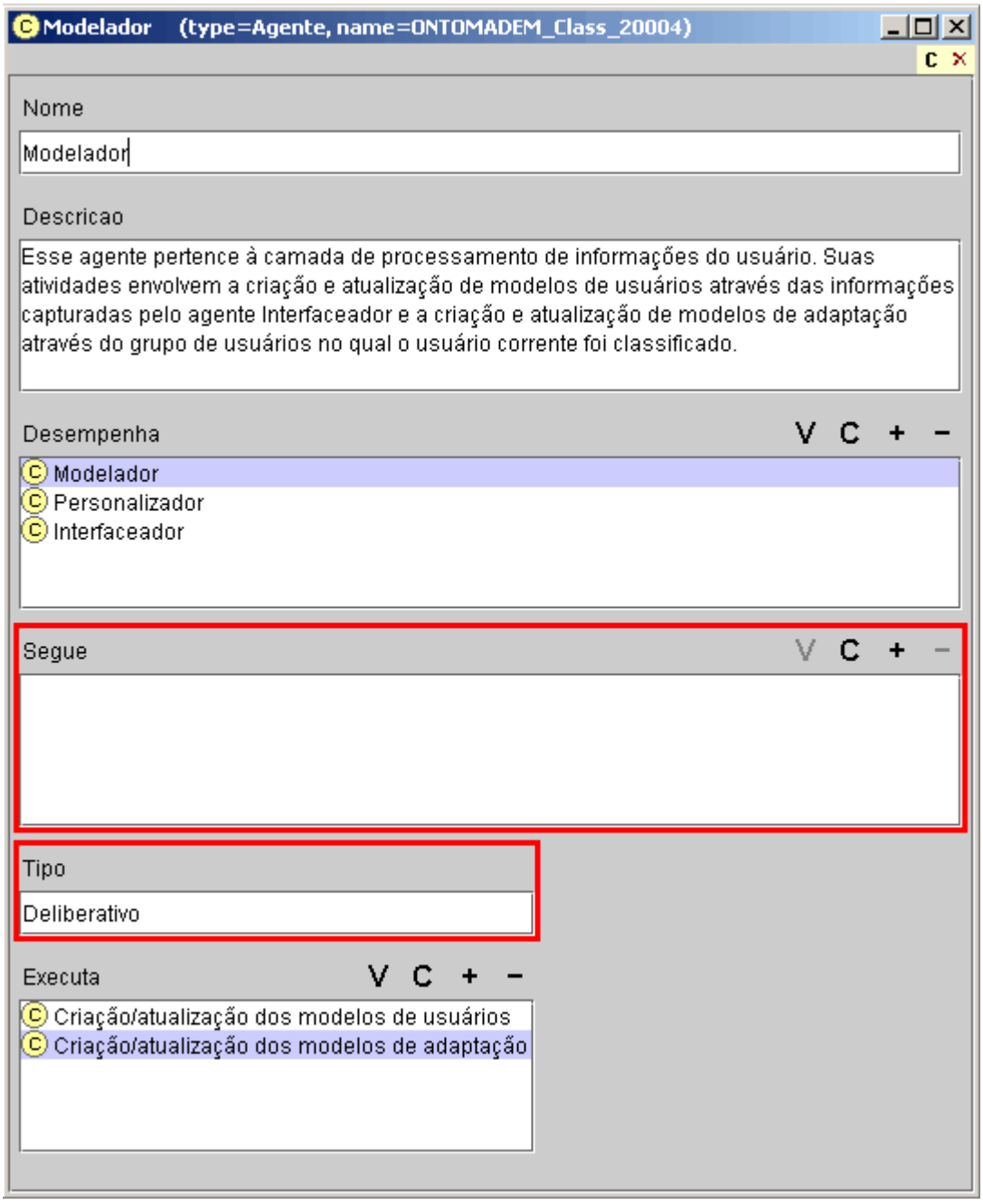

Figura 28. Descrição detalhada do agente Modelador

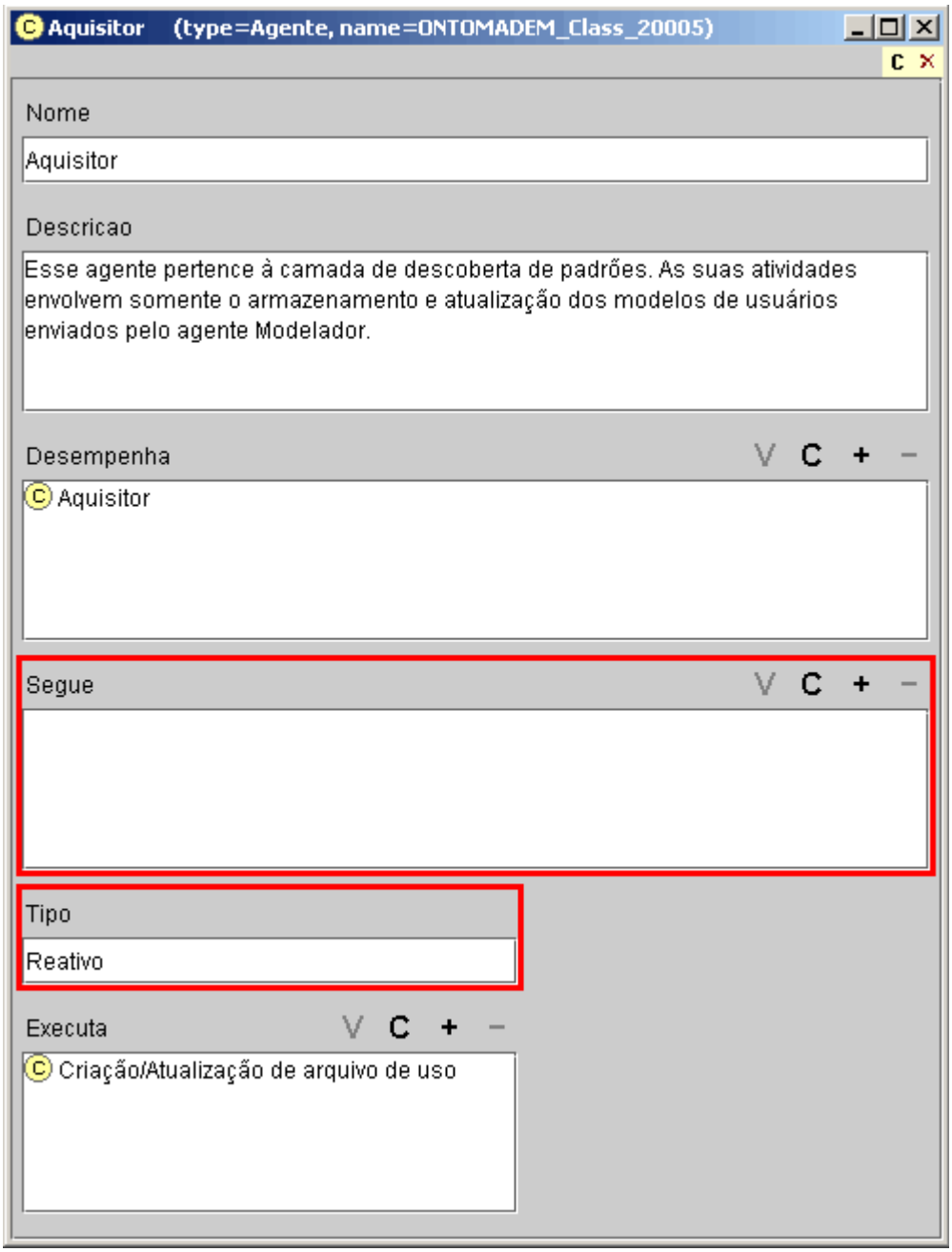

Figura 29. Descrição detalhada do agente Aquisitor

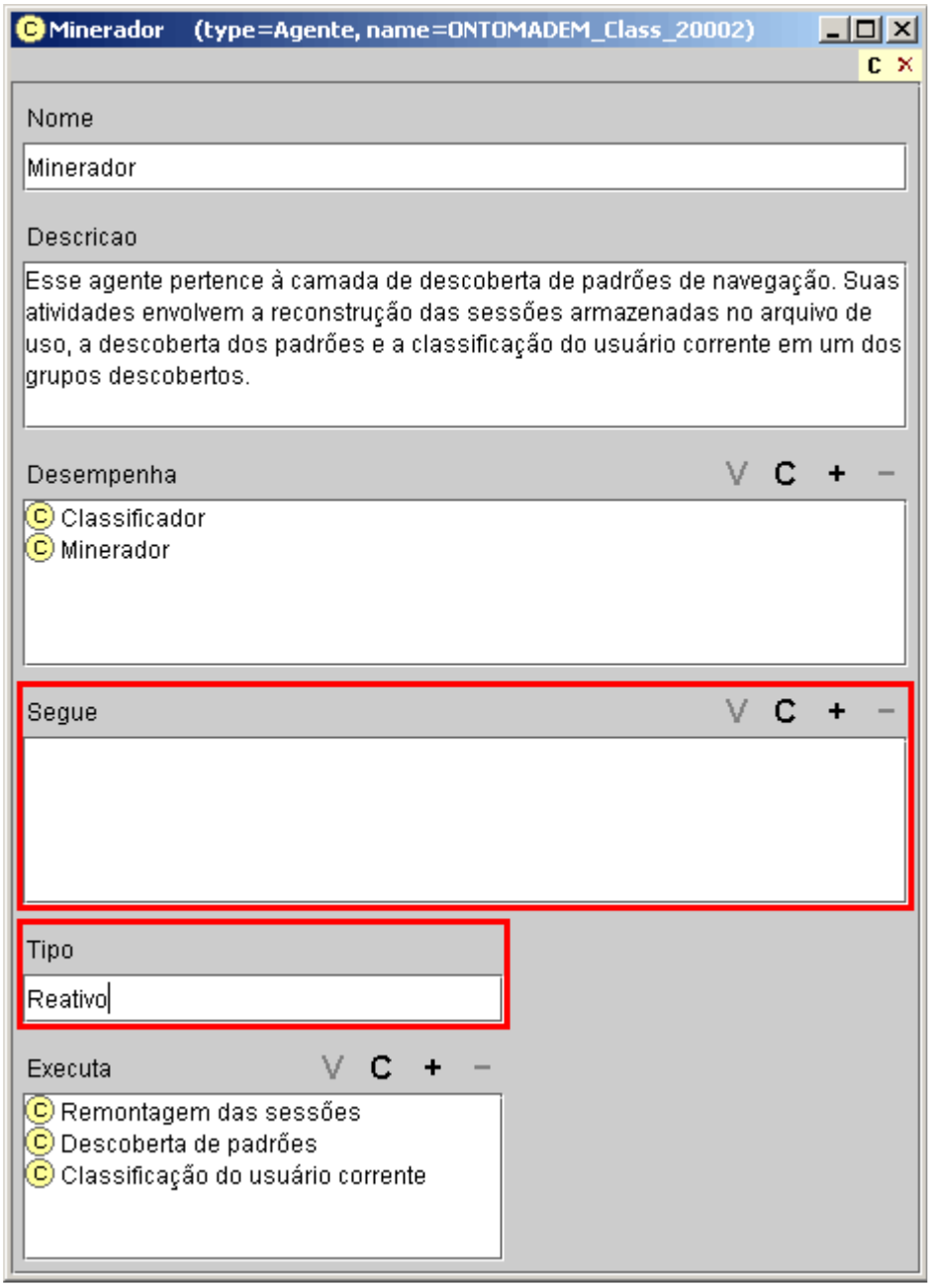

Figura 30. Descrição detalhada do agente Minerador

6.7.2 Modelagem de conhecimento

Quando os agentes se comunicam eles trocam informações entre eles. Essas informações precisam ter um significado compartilhado de forma que os agentes possam compreender o significado das informações trocadas dentro de uma mesma perspectiva.

Em outras palavras, os agentes precisam compartilhar um vocabulário de termos associados ao domínio em questão.

A MADEM em seu estágio atual não define uma tarefa associada à modelagem do conhecimento dos agentes. Sendo assim, devido à sua fundamental importância para a efetiva interação dos agentes decidiu-se por incluir aqui essa tarefa. A decisão de incluí-la na fase de projeto detalhado justifica-se pelo fato dos conceitos atrelados ao domínio estarem fortemente atrelados às decisões de projeto, como as técnicas e algoritmos utilizados, por exemplo.

Por padrão, o vocabulário dos agentes é definido através de ontologias. Uma ontologia provê uma representação formal tanto dos conceitos de um domínio quanto de seus relacionamentos entre si. Um conceito representado em uma ontologia adquire uma unicidade em sua interpretação eliminando assim tanto desentendimentos quanto interpretações ambíguas. Se, por exemplo, os agentes estiverem “conversando” sobre o conceito “manga” e manga estiver definida em uma ontologia de frutas, os agentes não correrão o risco de interpretar o conceito “manga” como “manga de camisa”, por exemplo.

O agente *Interfaceador* precisa enviar as visitas dos usuários ao agente *Modelador*. Uma visita aqui está representada por uma página e o tempo de permanência nessa página. Uma página por sua vez é caracterizada pelo seu endereço Web (URL). Vale ressaltar que a interpretação que foi dada a esses conceitos está atrelada aos objetivos específicos do ONTOMUW, não significando, portanto, que esses conceitos devam ser universalmente entendidos tal como definidos aqui.

Uma sessão de navegação é representada pela lista de visitas realizadas pelo usuário e pela ordem do segmento das visitas. É através da sessão de navegação e da ordem de segmento que o agente *Modelador* constrói as matrizes de características.

O conceito *MatrizCaracteristicas* representa uma matriz de características genérica e é caracterizada pelo número de linhas e colunas da matriz e de um atributo denominado “matriz”, contendo efetivamente as linhas e colunas com os elementos da matriz.

O modelo de usuários é construído e mantido (como conhecimento interno) pelo agente *Modelador.* Esse modelo pode ser enviado para o agente *Aquisitor*, quando

uma sessão for terminada, para a gravação do modelo no arquivo de uso, ou então pode ser enviado ao agente *Minerador*, de forma que o usuário corrente seja classificado em algum dos grupos de usuários pré-existentes. Aqui um modelo de usuários é então representado pela sessão de navegação do usuário e por três matrizes de características, uma para a característica visitas (*V*), uma para a característica tempo (*T*) e outra para a seqüência (*S*).

Assim que o usuário corrente é classificado, o agente *Minerador* envia o grupo no qual esse usuário foi classificado ao agente *Modelador*. Aqui um grupo de usuários é caracterizado pela cardinalidade do grupo, pelo número do grupo que funciona como um identificador único do grupo e o centróide do grupo, que corresponde à média aritmética entre todas as matrizes de todos os membros do grupo.

A Figura 31, gerada pelo Protégé, sumariza graficamente a ontologia que define o vocabulário dos agentes. A figura mostra os conceitos do domínio e seus relacionamentos através de seus atributos.

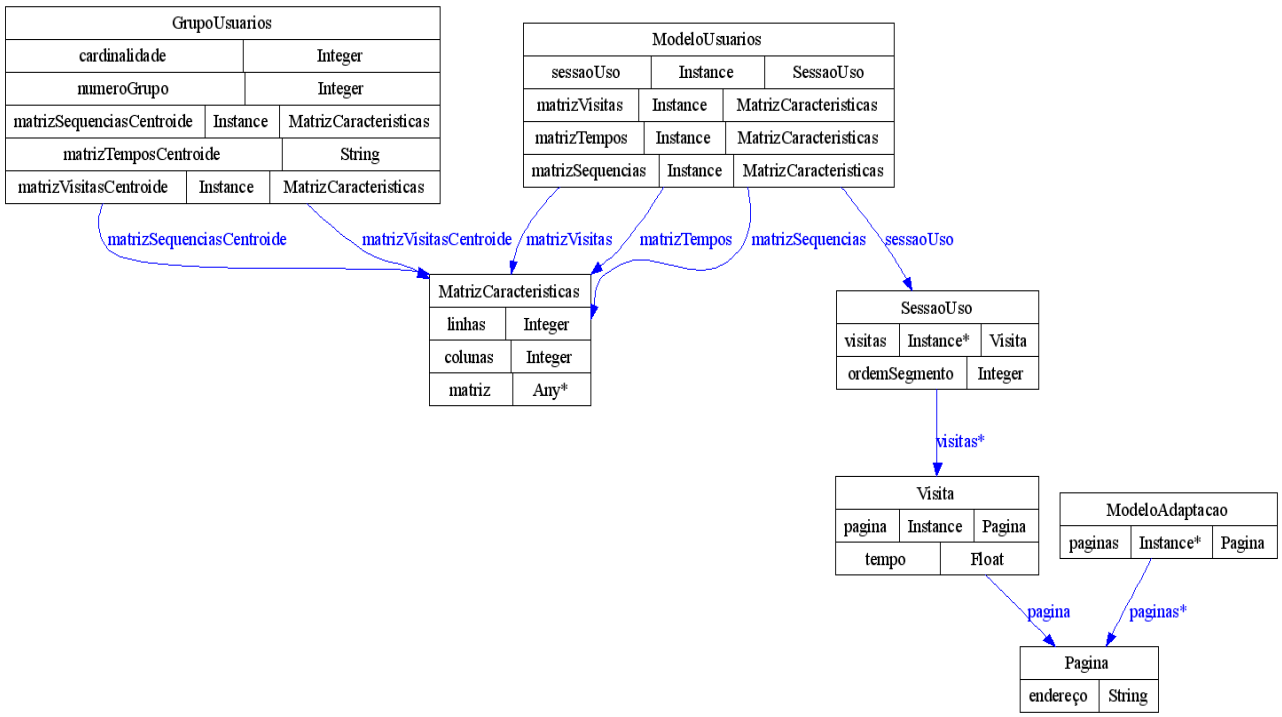

Figura 31. Ontologia representando o vocabulário dos agentes

6.7.3 Modelagem de comportamento

Os comportamentos dos agentes são, na verdade, um refinamento das atividades identificadas no modelo de papéis da análise de domínio e do modelo de atividades do projeto detalhado. Os comportamentos são identificados através das atividades definidas na fase de análise e das técnicas e algoritmos definidos na de projeto. A MADEM propõe, inicialmente, o mapeamento de uma “atividade” para um “comportamento”. Porém, podem existir situações onde um comportamento pode ser composto de uma ou mais atividades ou situações onde uma atividade pode originar mais de um comportamento. Para a construção do modelo de comportamento utilizou-se o diagrama de estados da UML, que é composto dos estados e das transições que ocasionam a mudança de estados dos agentes.

Tanto na UML como na AUML o diagrama de estados é utilizado para descrever o comportamento interno das entidades do sistema, portanto é adequado para descrever o comportamento dos agentes. Nas figuras a seguir (Figura 32, Figura 33, Figura 34, Figura 35) estão os diagramas de estado representando o comportamento de cada agente do framework.

6.7.3.1 Modelo de comportamento do agente *Interfaceador*

Como mostra a Figura 32, logo que o agente *Interfaceador* é criado ele entra no estado *Capturando comportamento de navegação*. Quando o usuário solicita outra página (do mesmo site), o agente sai desse estado, envia as informações da visita do usuário ao agente *Modelador* e, assim que a nova página solicitada é carregada, ele retorna ao estado *Capturando comportamento de navegação*. Nesse estado o agente executa as seguintes ações:

1. A primeira ação a ser realizada ao entrar no estado é a captura do instante em que a página é carregada;
2. Logo em seguida a URL da página visitada também é capturada;
3. Antes de sair desse estado o agente captura o instante em que o usuário solicita outra página e processa o tempo total gasto na página subtraindo o instante final menos o inicial.

A qualquer momento o agente *Modelador* pode solicitar a classificação do usuário corrente.

Uma vez que o agente *Interfaceador* recebe modelo de adaptação criado pelo agente *Modelador*, o agente entra no estado *Adaptando interface* e realiza as seguintes ações:

1. Formatação da lista de URLs na linguagem HTML;
2. Inserção da lista de URLs formatadas na página corrente ao qual o usuário está visitando.

Vale lembrar que a qualquer momento o usuário pode deixar o site ou fechar o navegador, portanto o agente pode a qualquer momento a partir de qualquer estado terminar suas atividades e ser destruído.

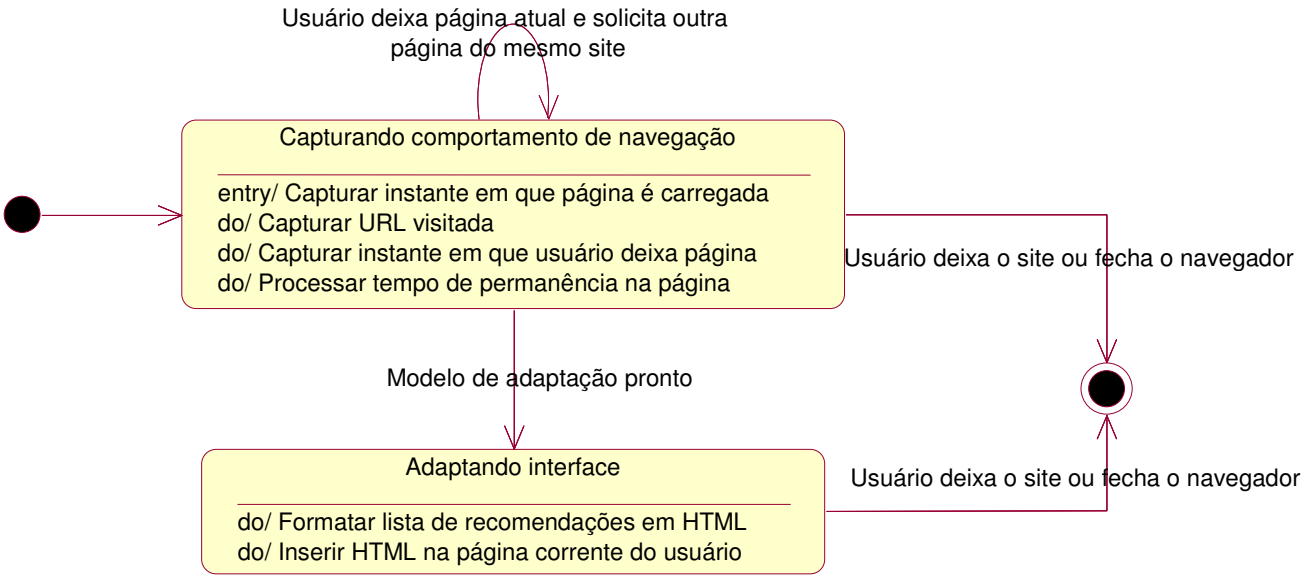

Figura 32. Modelo de comportamento do agente Interfaceador

6.7.3.2 Modelo de comportamento do agente *Modelador*

Como mostra a Figura 33, assim que o agente *Modelador* é criado ele entra no estado *Aguardando mensagem do Interfaceador*. Quando chega uma mensagem do agente *Interfaceador* com as informações de uma visita do usuário corrente, o agente

Modelador entra no estado *Criando/Atualizando modelo de usuários*. Nesse estado o agente realiza as seguintes ações:

1. Extração dos itens contidos na mensagem recebida (tempo e/ou URL);
2. Atualização da sessão de uso incluindo a nova página visitada;
3. Construção das matrizes de características. Esse processo envolve a criação de uma matriz para cada característica de navegação capturada do usuário. O Exemplo 1 da sessão 6.1.1.4 ilustra bem esse processo.

Se por outro lado o agente *Modelador* receber o grupo de usuários no qual o usuário foi classificado ele entra no estado Criando/Atualizando modelo de adaptação. Uma vez nesse estado o agente executa as seguintes ações:

1. Mapeamento das matrizes do grupo recebido em uma lista de páginas;
2. Eliminação das páginas já visitadas pelo usuário;
3. Criação de uma lista de páginas a serem recomendadas. Lembrando que essa lista de páginas é o próprio modelo de adaptação.

A condição que determina quando solicitar a classificação do usuário é dependente da aplicação e é deixada para o desenvolvedor. Quanto mais páginas forem visitadas pelos usuários, melhor caracterizado será o seu comportamento de navegação e consequentemente mais precisa será a classificação. Por outro lado, quanto mais se espera para classificar o usuário mais tempo ele pode ficar sem recomendações. O desenvolvedor deve ser astuto de forma a balancear esses dois lados. Então, quando o limite do número de visitas definido pelo desenvolvedor for alcançado o agente solicita a classificação do usuário enviando o seu modelo de usuários ao agente *Minerador*. Embora isto não esteja representado no diagrama, é importante entender sobre quais condições é realizada a solicitação de classificação do usuário corrente.

O desenvolvedor também é responsável por definir um tempo limite para a sessão do usuário, uma vez esse tempo atingido o agente envia o modelo de usuários ao agente *Aquisitor*. O fechamento do navegador também denota o término de uma sessão e consequentemente causa o envio do modelo a ser armazenado pelo agente *Aquisitor*.

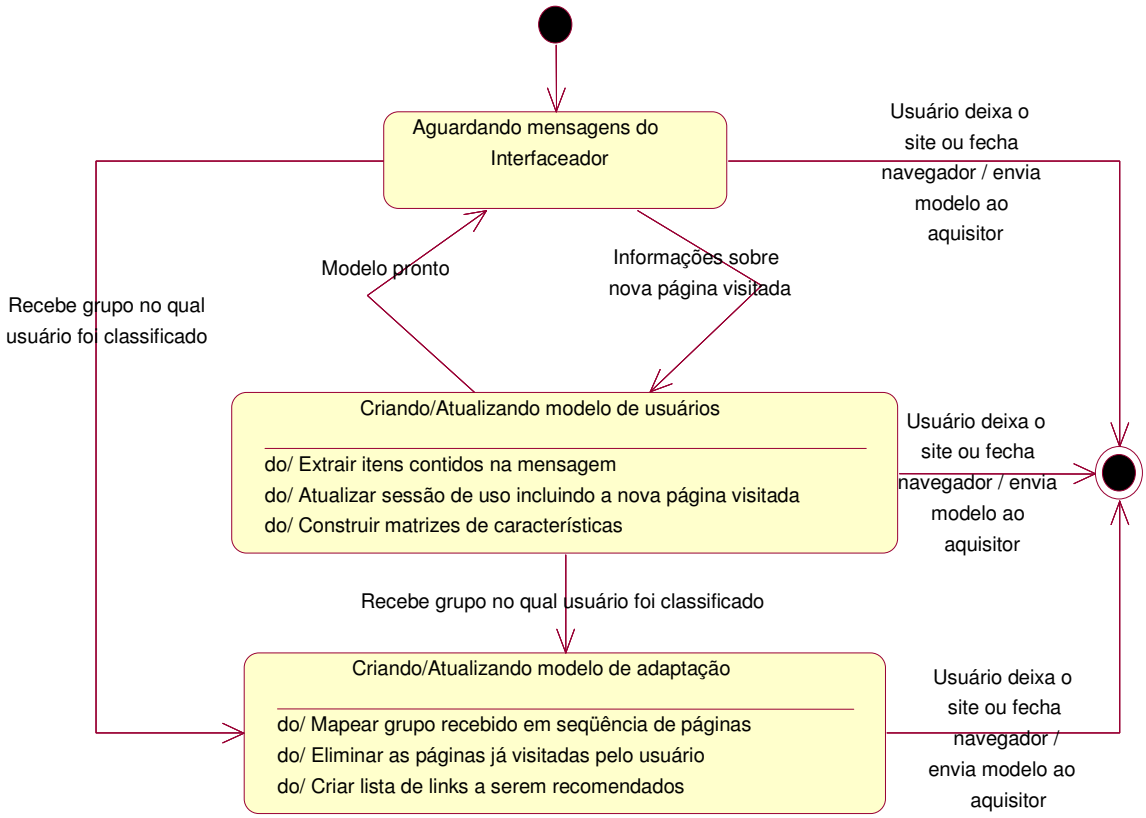

Figura 33. Modelo de comportamento do agente Modelador

6.7.3.3 Modelo de comportamento do agente *Aquisitor*

Como mostra a Figura 34, o agente *Aquisitor* depois de criado, fica aguardando os modelos de usuários dos agentes *Modeladores*. Quando chega um novo modelo o agente entra no estado *Criando/Atualizando dados de uso* e realiza as seguintes ações:

1. Extração dos itens contidos na mensagem;
2. Formatação dos itens extraídos no formato RDF;
3. Armazenamento do modelo no arquivo.

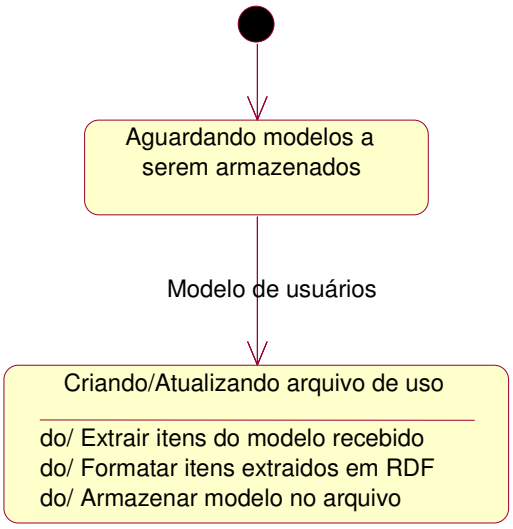

Figura 34. Modelo de comportamento do agente Aquisitor

6.7.3.4 Modelo de comportamento do agente *Minerador*

Como mostra a Figura 35, assim que o agente é criado ele entra no estado *Aguardando expiração do agendamento*. Como visto na sessão 6.2.2 adotou-se a utilização do agrupamento dinâmico para a adaptação dos grupos em tempo real e o reagrupamento periódico em intervalos definidos pelo desenvolvedor para que os grupos se mantenham consistentes com o passar do tempo. Assim que o intervalo definido pelo desenvolvedor expira, o agente entra no estado *Remontando modelos de usuários*. Nesse estado são realizadas as seguintes ações:

1. Abertura do arquivo de uso para leitura;
2. Construção do conjunto de modelos armazenados no arquivo.

Uma vez o conjunto de modelos devidamente construído, o agente entra no estado *Criando modelo de grupos*. É nessa fase que o conhecimento ou os padrões de navegação são efetivamente descobertos. Uma vez nesse estado o agente executa as seguintes ações:

1. Aplicação do algoritmo de agrupamento K-Médias ao conjunto de modelos, onde a medida de similaridade escolhida é $DEPP$ (ver sessão 6.2.6.1);

2. Criação do modelo FM para cada grupo. Isso corresponde ao cálculo do centróide de cada grupo (ver Exemplo 2 da sessão 6.2.5).

Uma vez os grupos construídos e representados pelo modelo FM o agente entra no estado *Aguardando requisição do Modelador*. Uma vez recebida a mensagem contendo a requisição do *Modelador* o agente entra no estado *Classificando usuário corrente*. O agente realiza as seguintes ações nesse estado:

1. Comparação do usuário corrente com o conjunto contendo os grupos descobertos;
2. Classificação do usuário corrente no grupo mais similar. O algoritmo utilizado para a classificação é o agrupamento dinâmico e a medida de similaridade $DEPP$.

Note que o agente não precisou em nenhum momento realizar tarefas associadas ao pré-processamento dos dados de uso, pois estes já se encontram devidamente limpos, estruturados e consistentes.

A qualquer momento o intervalo definido pelo desenvolvedor pode expirar. Toda vez que isso acontecer o agente volta ao estado *Criando modelo de usuários* e o processo de mineração é reiniciado.

6.7.4 Modelo de atividades detalhado

De forma a complementar o modelo de comportamento é apresentado um modelo de atividades detalhado. Nesse modelo, cada atividade mostrada no modelo de atividades do projeto arquitetural é dividida em subatividades refletindo as técnicas e algoritmos escolhidos.

Enquanto o diagrama de estado fornece uma visão do comportamento interno dos agentes baseado em seus possíveis estados, o diagrama de atividades fornece uma visão do fluxo de controle das ações executadas pelos agentes, mostrando os passos seqüenciais e/ou concorrentes envolvidos na dinâmica da sociedade. A Figura 36 mostra o diagrama de atividades detalhado.

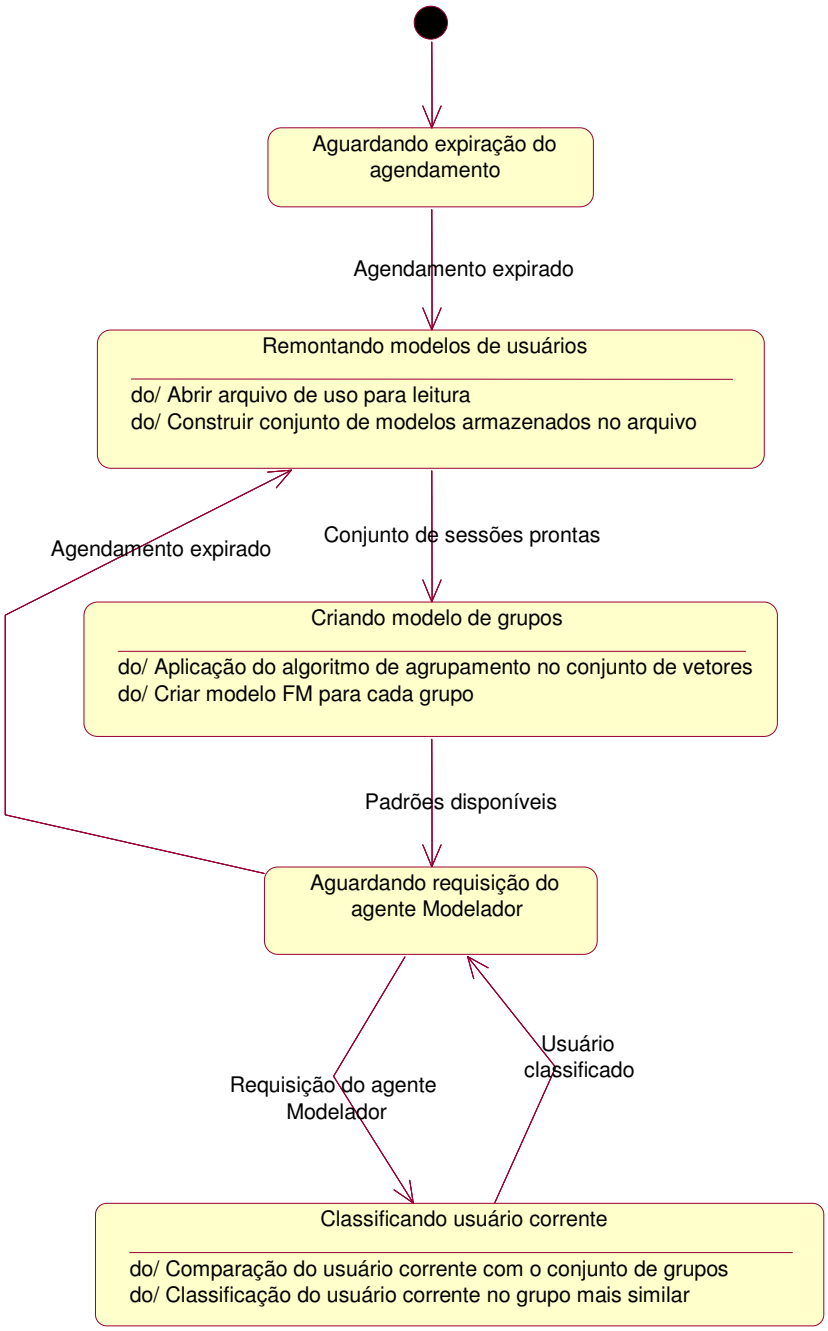

Figura 35. Modelo de comportamento do Minerador

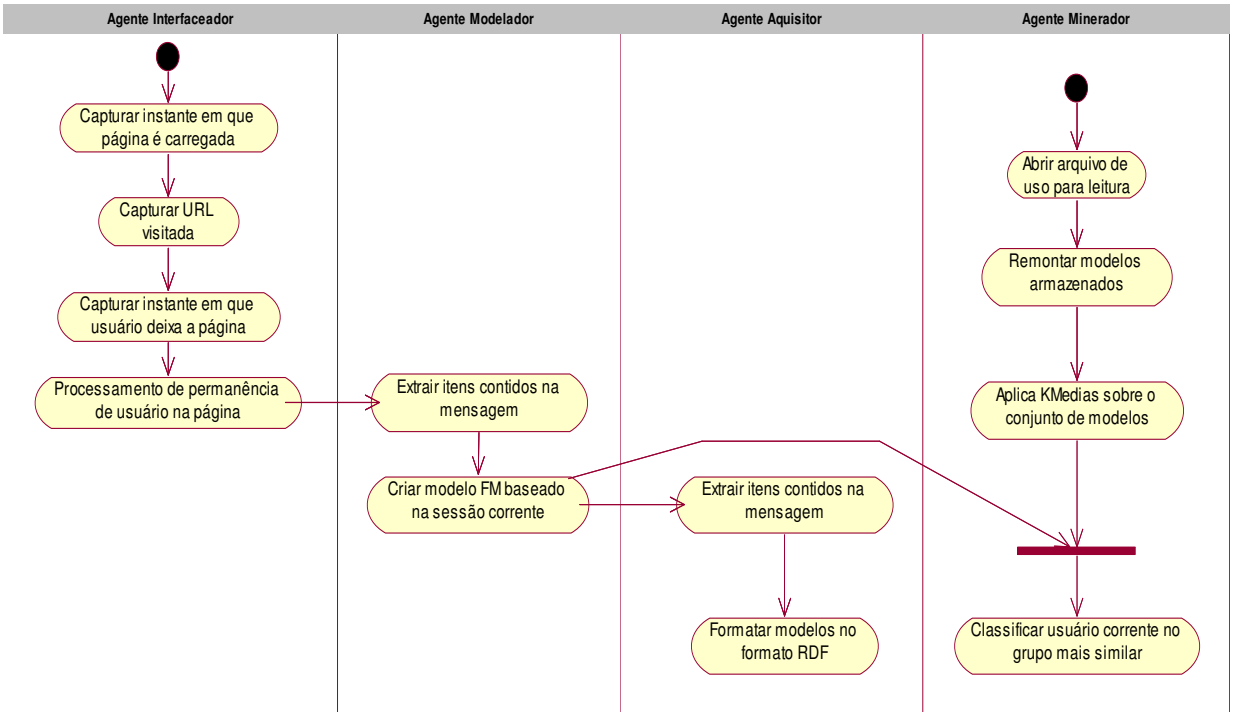

Figura 36. Modelo de atividades detalhado

6.7.5 Modelagem de interações dos agentes

O produto dessa subtarefa é um modelo de interações que especifica as interações que ocorrem entre os agentes do framework. Para a construção do modelo de interações é utilizada a AUML. As mensagens são rotuladas com performativas da linguagem de comunicação entre agentes FIPA-ACL [47]. A FIPA-ACL é um padrão internacional para a interoperabilidade entre os agentes, esse padrão define vários atributos, em particular:

- O *remetente* da mensagem;
- A lista de *destinatários*;
- A intenção da comunicação (também chamada de “performativa”) que indica a intenção do remetente ao enviar a mensagem. A performativa pode ser REQUEST, se o remetente deseja que o destinatário execute uma ação, INFORM, se o remetente quer informar o destinatário sobre um fato, QUERY_IF, se o remetente deseja saber se determinada

condição é verdadeira, CFP, ("*call for proposal*"), PROPOSE, ACCEPT_PROPOSAL, REJECT_PROPOSAL, se o remetente e o destinatário estão engajados em alguma negociação;

- O *conteúdo*, que é a informação incluída na mensagem (ex: a ação a ser executada numa mensagem REQUEST, ou o fato a ser informado na mensagem INFORM);
- A *ontologia*, que é o vocabulário usado no conteúdo da mensagem indicando o seu significado (tanto o remetente quanto o destinatário devem atribuir o mesmo significado às mensagens trocadas para que a comunicação possa ser efetiva);
- Por último, existem alguns atributos usados para o controle de conversas concorrentes entre os agentes e limites de tempo para o recebimento de respostas tais como *conversation-id*, *reply-with*, *in-reply-to* e *reply-by*.

A AUML [115] permite que os possíveis papéis desempenhados pelos agentes sejam mostrados durante as interações, através das linhas de tempo do diagrama. Embora não esteja representada no diagrama subentende-se que o vocabulário dos agentes é a ontologia definida na seção 6.6.2.

A Figura 37 mostra o agente *Interfaceador*, desempenhando inicialmente o papel de *Monitor*, enviando uma mensagem informando o *Modelador* sobre uma nova visita do usuário (conceito: *Visita*). O *Modelador* então pode enviar uma mensagem, ou informando o agente *Aquisitor* sobre o modelo ao qual ele deve armazenar, ou então solicitar, através da performativa *request*, a classificação do usuário corrente (conceito: *ModeloUsuarios*). O losango com o X no diagrama representa um OU exclusivo, onde só uma ou outra mensagem pode ser enviada de acordo com alguma pré-condição. Após isso, o *Minerador* informa o *Modelador*, agora desempenhando o papel de *Personalizador*, sobre o grupo que o usuário foi classificado. O *Modelador* de posse desse grupo cria o modelo de adaptação e o envia ao agente *Interfaceador* para que este, agora desempenhando o papel de *Interfaceador*, possa adaptar a interface. Lembrando que aqui

o modelo de adaptação é simplesmente um conjunto de páginas a serem recomendadas ao usuário.

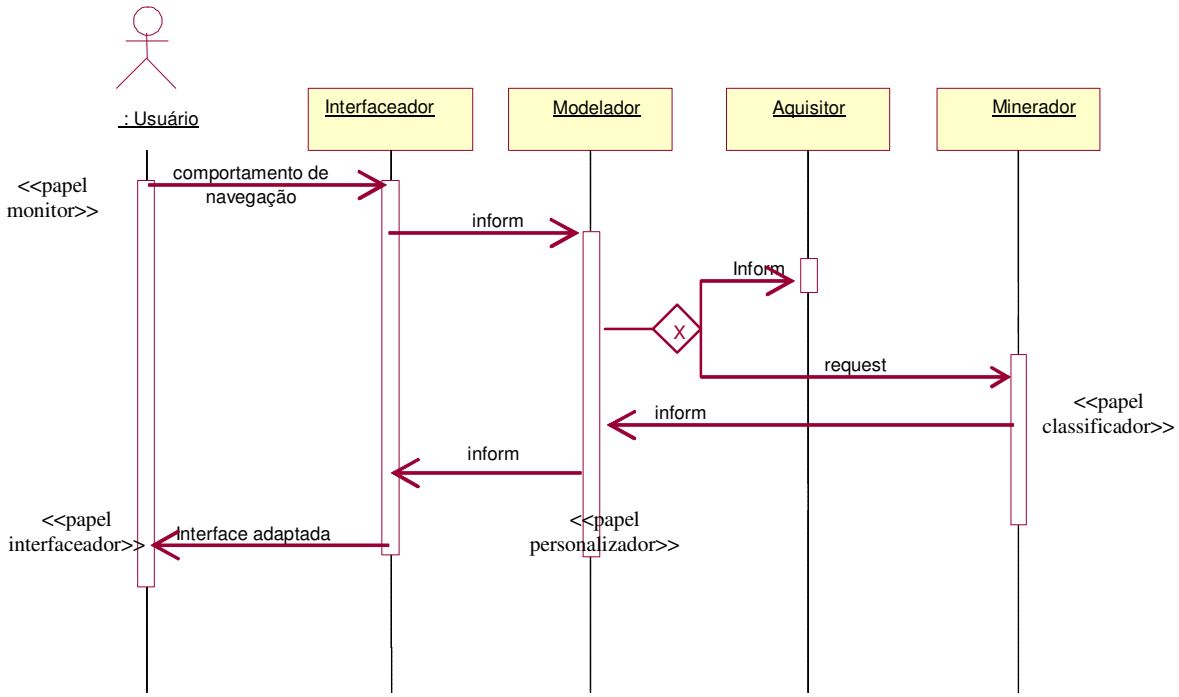

Figura 37. Diagrama de seqüência dos agentes do framework

O próximo capítulo apresenta a realização do framework através de agentes funcionais implementados segundo o ambiente de desenvolvimento de SMAs JADE [82].

6.8 Considerações finais do capítulo

Esse capítulo apresentou a fase de projeto do framework da Engenharia de domínio multiagente em termos da MADEM e ONTOMADEM.

O projeto de domínio na Engenharia de domínio multiagente produz uma solução computacional multiagente, reutilizável no desenvolvimento de uma família de aplicações similares em um determinado domínio.

Para a representação do modelo de usuários e de grupos de usuários escolhe-se o modelo de matrizes de características idealizado por Shahabi et al. (2003) [131]. Nesse modelo as características de navegação dos usuários são representadas e quantificadas por matrizes, baseadas em uma matriz universal de segmentos, que gera o

espaço de matrizes. A escolha desse modelo se deu baseado na simplicidade, flexibilidade e poder de representação.

Como estratégia de mineração optou-se pelo agrupamento dinâmico como algoritmo de classificação e o reagrupamento periódico como forma de tornar os grupos consistentes com o tempo. Quando se realiza o agrupamento dinâmico trabalha-se com sessões incompletas de usuários, ou seja, sessões que provavelmente ainda não tenham terminado. Dessa forma, com o passar do tempo a qualidade dos grupos é degradada, por isso então a necessidade do reagrupamento periódico.

Para o algoritmo de reagrupamento periódico optou-se, inicialmente, pelo K-Médias. Essa escolha se deu basicamente pela simplicidade do algoritmo do ponto de vista da implementação. Outro fator que influenciou essa decisão é o fato desse algortimo ser bastante aplicado para vetores do espaço vetorial, e sendo as matrizes generalizações dos vetores, fica fácil a sua adaptação para as matrizes do modelo FM. O maior problema do K-Médias é o fato de se ter que determinar aprioristicamente o número de grupos iniciais a serem construídos, o que é uma decisão bastante dependente da aplicação, requerendo então o auxílio de especialistas de domínio do site em questão.

A captura das informações derivadas da navegação do usuário também foi inspirada em uma idéia de Shahabi et al. (2001) [132]. Aqui cada página do site monitorado possui um applet que serve de interface entre o navegador do usuário e os agentes (*Interfaceador* e *Modelador*). Os applets são programas especiais que executam na máquina virtual do navegador e possuem habilidades para capturar eventos do navegador do usuário (ex.:URL visitada).

Para a estruturação dos agentes do framework utilizou-se a arquitetura genérica definida por Mobasher et al. (2000) [109] para sistemas de personalização na Web baseados na MUW. Apesar de não ser um padrão arquitetural formal, essa arquitetura é bastante aceita na comunidade de pesquisa e serviu como guia para a estruturação dos agentes do ONTOMUW. Nessa arquitetura os componentes do sistema são divididos em duas camadas de acordo com a coesão funcional entre suas responsabilidades. Uma camada é responsável pela modelagem do usuário corrente e adaptação da interface e a outra é responsável pela descoberta dos padrões que servirão

de insumo para a primeira camada. Segundo essa perspectiva, definiu-se dois agentes para a primeira camada, *Interfaceador* e *Modelador*, denominada *camada de processamento de informações do usuário* e outros dois para a segunda camada, *Aquisitor* e *Minerador*, denominada *camada de descoberta de padrões de navegação*.

A MADEM em seu estágio atual não define uma tarefa responsável pela modelagem de conhecimento dos agentes. Esse conhecimento é responsável por definir o vocabulário de domínio dos agentes. Sendo assim, devido a sua fundamental importância para o desenvolvimento de aplicações multiagente efetivas, adicionou-se uma tarefa de modelagem de conhecimento na fase de projeto detalhado. Essa decisão se justifica pelo fato dos conceitos descritos na ontologia estarem ligados às decisões de projeto.

Para algumas tarefas da MADEM, como a modelagem do comportamento, atividades e de interações dos agentes, a ONTOMADEM não foi utilizada, pois outras linguagens de representação como UML e AUML são mais representativas e descritivas que os modelos atualmente gerados pela ONTOMADEM.

7 IMPLEMENTAÇÃO DE DOMÍNIO DO ONTOMUW

A implementação de domínio é a fase da Engenharia de domínio responsável por criar componentes de software reutilizáveis, baseados tanto no modelo de domínio, definido na fase de análise de domínio, quanto na especificação do framework, definida na fase de projeto de domínio. Na Engenharia de domínio multiagente, os componentes de software são na verdade os próprios agentes de software.

A MADEM em seu estágio atual não contempla a fase de implementação de domínio. Até o momento da conclusão deste trabalho não se tinha encontrado metodologias que contemplassem a fase de implementação de domínio da Engenharia de domínio multiagente. Portanto, a implementação de domínio a ser descrita aqui, não está apoiada por uma metodologia formal e sistemática para a implementação dos agentes (componentes) do framework.

A estratégia que se seguiu então foi escolher um ambiente de desenvolvimento de SMAs, que fosse implementado segundo o padrão FIPA [47], que fosse código aberto e que fosse suficientemente maduro e bem documentado. De acordo com esses critérios optou-se pelo ambiente JADE ("*Java Agent Development Framework*") [82]. O JADE é um framework escrito na linguagem Java que facilita a construção de SMAs.

O capítulo está organizado como segue. A seção 7.1 apresenta uma breve descrição do padrão FIPA [47]. A seção 7.2 introduz o ambiente JADE. A seção 7.3 descreve a implementação dos agentes do ONTOMUW juntamente com exemplos de execução de cada um. Finalmente, a seção 7.4 apresenta as considerações finais do capítulo.

7.1 Padrão FIPA

A FIPA ("*Foundation for Intelligent Physical Agents*") [47] é uma fundação sem fins lucrativos direcionada à produção de padrões para a interoperabilidade de agentes heterogêneos e interativos e sistemas baseados em agentes. A sua missão básica é facilitar a interligação de agentes e SMAs entre múltiplos fornecedores de ambientes. A missão oficial é: "A promoção de tecnologia e especificações de interoperabilidade que

facilitem a comunicação entre sistemas de agentes inteligentes no contexto comercial e industrial moderno" [47]. Em suma, interoperabilidade entre agentes autônomos.

Segundo dados encontrados no site oficial, antes da FIPA [47] havia cerca de 60 sistemas proprietários de agentes em competição, sendo que a maioria destes eram sistemas "fechados" e incompatíveis. Fatos tais que atrasaram o desenvolvimento da tecnologia de agentes. Logo, fica claro que a necessidade de padronização torna-se indispensável.

7.2 JADE

O JADE [82] é um ambiente código aberto (LGPL) para o desenvolvimento de aplicações baseadas em agentes, conforme as especificações da FIPA para interoperabilidade entre agentes totalmente implementados em Java.

Vale ressaltar que o ambiente JADE é bastante extenso, portanto, um aprofundamento nesse ambiente acabaria por desviar o foco do objetivo principal do trabalho. Sendo assim, é apresentado apenas o suficiente do JADE de forma a contextualizar o leitor no ambiente em que os agentes do ONTOMUW foram implementados.

O principal objetivo do JADE é simplificar e facilitar o desenvolvimento de SMAs, garantindo um padrão de interoperabilidade entre agentes através de um abrangente conjunto de serviços [2]. Dentre estes serviços estão: serviço de nomes ("*naming service*"), páginas amarelas ("*yellow-page service*"), transporte de mensagens, serviços de codificação e decodificação de mensagens e uma biblioteca de protocolos de interação (padrão FIPA) pronta para ser usada.

O JADE foi escrito em Java devido a algumas características particulares dessa linguagem, como por exemplo, suporte à programação distribuída, suporte à programação concorrente ("*multithreaded*") e portabilidade [2].

A escolha do JADE para a implementação do ONTOMUW deu-se por dois motivos basicamente: o fato de já se possuir um treino prévio na linguagem Java, o que possibilitou um rápido aprendizado do ambiente JADE, e pela extensa documentação

existente, com artigos, tutoriais, listas de discussões e exemplos funcionais disponíveis junto com a distribuição.

7.2.1 Características do JADE

Seguem abaixo algumas características oferecidas pelo JADE para o desenvolvimento de SMAs:

- *Plataforma distribuída de agentes*. Os agentes do JADE podem estar distribuídos em várias máquinas independentemente de sua posição geográfica. O JADE possui uma coleção de protocolos de transporte de mensagens prontos para serem utilizados pelos agentes, incluindo o HTTP;
- *Interface gráfica*. Interface visual que gerencia vários agentes e containeres de agentes inclusive remotamente;
- *Ferramentas de depuração*. Ferramentas que ajudam na depuração de aplicações multiagente baseados em JADE;
- *Suporte á execução de atividades concorrentes*. Através do modelo de comportamento do JADE as ações dos agentes podem ser realizadas de forma concorrente;
- *Ambiente de agentes complacente a FIPA*;
- *Transporte de mensagens*. Transporte de mensagens em conformidade com o padrão FIPA-ACL.
- *Biblioteca de protocolos FIPA*. Para interação entre os agentes, o JADE dispõe de uma biblioteca de protocolos prontos para serem utilizados;
- *Automação de registros*. Registro e cancelamento automático de agentes na plataforma;
- *Serviços de nomes em conformidade aos padrões FIPA*. Quando os agentes são inicializados eles obtêm seus GUID (“*Globally Unique Identifier*”) diretamente da plataforma. Esses GUIDs funcionam como identificadores únicos dos agentes;

- *Integração*. Mecanismo que permite a comunicação de aplicações externas com com os agentes JADE.

Além das características acima citadas, o JADE também possui algumas ferramentas muito úteis, que simplificam a administração da plataforma e o desenvolvimento de aplicações. Como o "*sniffer agent"*, que permite monitorar graficamente as trocas de mensagens entre os agentes e o "*introspector agent"* que permite o monitoramento dos comportamentos internos dos agentes através de uma interface gráfica.

7.2.2 JADE e FIPA

O modelo de plataforma padrão especificado pela FIPA, que pode ser visto na Figura 38, é composto pelas seguintes estruturas definidas a seguir.

O *Agente*, que é o agente propriamente dito e cujas tarefas serão definidas de acordo com o objetivo da aplicação. Este se encontra dentro de uma plataforma de agentes e realiza toda sua comunicação com outros agentes através de troca de mensagens. O agente também pode se relacionar com aplicações externas (*Software*).

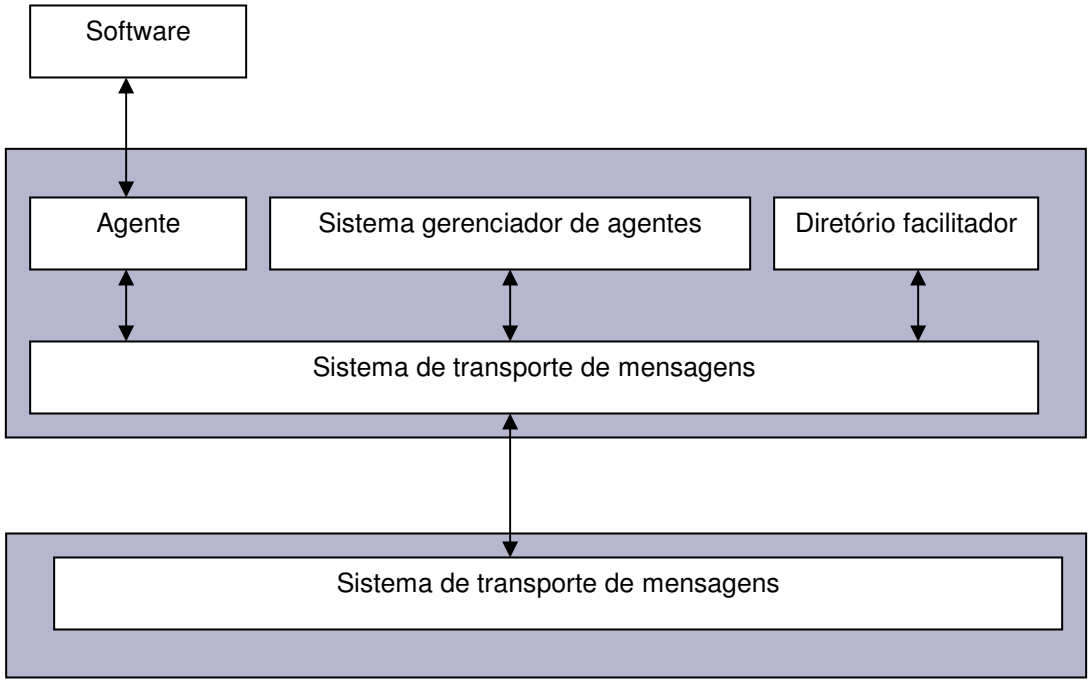

Figura 38. Modelo padrão de plataforma de agentes definido pela FIPA [47]

O sistema gerenciador de agentes ou AMS (“*Agent Management System*”), é responsável pela supervisão do acesso e uso da plataforma de agentes. Apenas um AMS pode existir em uma plataforma. Ele fornece guia de endereços (páginas brancas) e controle do ciclo de vida dos agentes, mantendo um diretório de identificadores de agentes AID (“*Agent Identifier*”) e estados de agentes. Ele é o responsável pela autenticação dos agentes e pelo controle de registro dos agentes na plataforma. Cada agente tem que se registrar no AMS para obter um AID válido.

O diretório facilitador ou DF (“*Directory Facilitator*”) é responsável por fornecer o catálogo de serviços da plataforma (páginas amarelas).

O sistema de transporte de mensagens ou ACC (“*Agent Communication Channel*”) é responsável por prover toda a comunicação entre os agentes dentro e fora da plataforma. Todos os agentes, inclusive o AMS e o DF, utilizam esse canal para a comunicação.

O JADE cumpre totalmente com a especificação dessa arquitetura. Sendo que, no carregamento da plataforma JADE, o AMS e o DF são criados e o ACC é configurado para permitir a comunicação através de mensagens [2].

7.2.3 Criando um agente em JADE

Para criar um agente JADE basta estender a classe *jade.core.Agent* e implementar o método *setup* () (Figura 39).

```
import jade.core.Agent;
public class MeuAgente extends Agent {
        protected void setup() {
                System.out.println("O agente "+getAID().getName()+" está pronto!");
                }
            }
```

Figura 39. Implementação de um agente genérico em JADE

O método *setup* () é responsável por incluir as inicializações do agente. As ações ou atividades dos agentes são executadas através de comportamentos ("*behaviours*"). A próxima subseção introduz brevemente os comportamentos dos agentes JADE.

7.2.4 Comportamento dos agentes JADE

De forma a implementar o comportamento dos agentes o desenvolvedor deve instanciar objetos de classes que herdem da classe *jade.core.behaviours.Behaviours*, e adicionar esses objetos ao agente. Isso é feito através do método *addBehaviour* () do agente. Um comportamento pode ser adicionado a um agente a qualquer momento, tanto na inicialização do agente (no método *setup* ()) como a partir de outros comportamentos.

Cada classe que herda da classe *jade.core.behaviours.Behaviours* precisa implementar o método *action* (), que é onde as operações vão de fato ser efetuadas enquanto o comportamento estiver ativo, e o *método done* () (retorna um valor booleano), que especifica quando um comportamento terminou de executar, de forma que ele possa ser retirado da fila de comportamentos .

A estrutura de comportamentos do JADE se dá através de um escalonador interno que gerencia automaticamente o escalonamento dos comportamentos.

O JADE possui vários tipos especiais de comportamentos que podem ser aplicados em situações específicas, como por exemplo:

- *jade.core.behaviours.CyclicBehaviour* – Essa classe pode ser herdada para criação de comportamentos que se mantém executando continuamente. Nesse caso, o método *done* () herdado da super classe *jade.core.behaviours.Behaviours.Behaviour*, sempre retorna *false*, o que faz com que o comportamento se repita como se estivesse em um "*loop*" infinito;
- *jade.core.behaviours.TickerBehaviour* – Essa classe implementa um comportamento que executa periodicamente de acordo com um intervalo de tempo definido pelo desenvolvedor. Nesse caso, o desenvolvedor redefine o método *onTick* () e inclui as ações que devem ser executadas

periodicamente pelo comportamento. O período de "*ticks*" é definido no construtor da classe em milisegundos;

- *jade.core.behaviours.WakerBehaviour* - Comportamento que é executado depois que determinado intervalo de tempo expira. A diferença para o *TickerBehaviour* é que aqui a execução do comportamento não se dá de forma periódica e sim atômica, ou seja, executa somente uma vez. As ações do comportamento são inseridas no método *handleElapsedTimeout* () o qual é chamado depois que o intervalo de tempo é transcorrido.

7.2.5 Troca de mensagens no JADE

Toda a troca de mensagens realizada no JADE é feita através de instâncias da classe *ACLMessage*. Esta classe possui um conjunto de atributos que estão em conformidade com as especificações da FIPA-ACL [47]. Assim, um agente que pretenda enviar uma mensagem deve instanciar um objeto da classe *ACLMessage*, preenchê-lo com as informações necessárias e chamar o método *send* () da classe *jade.core.Agent*. Quando o agente for receber mensagens, o método *receive* () ou *blockingReceive* () da classe *jade.core.Agent* deve ser chamado. Outro meio de enviar ou receber mensagens no JADE é através do uso das classes de comportamentos *SenderBehaviour* e *ReceiveBehaviour*. Através desses comportamentos as trocas de mensagens passam a ser escalonadas como atividades independentes de um agente.

A classe *ACLMessage* define um conjunto de constantes (ACCEPT_PROPOSAL, AGREE, CANCEL,CFP, CONFIRM, etc) que são as performativas da FIPA-ACL. Essas constantes são passadas ao construtor da classe com o objetivo de definir o tipo da mensagem.

Segue abaixo os principais métodos da classe *ACLMessage*:

- *public void addReceiver (AID idAgente)* – Adiciona o AID de um agente como receptor ou destinatário da mensagem. Em outras palavras, determina quem receberá a mensagem;

- *public void removeReceiver (AID idAgente)* – Remove o AID do agente da lista de receptores;
- *public ACLMessage createReply* () - Cria uma mensagem ACL de resposta;
- *public static java.lang.String[] getAllPerformativeNames* () - Retorna um array de strings com todas as performativas;
- *public Iterator getAllReceiver* () – Retorna um "*iterator*" contendo todos os AIDs dos agentes receptores da mensagem;
- *public java.lang.String getContent* () - Retorna o conteúdo da mensagem;
- *public void setContent (java.lang.String conteúdo)* – Define o conteúdo da mensagem a ser enviada;
- *public void setOntology (java.lang.String ontologia)* – Define a ontologia representando o vocabulário dos agentes.

7.3 Implementação dos agentes da ONTOMUW

A idéia aqui é disponibilizar agentes, implementados através do JADE, que possam ser reutilizados ou estendidos por desenvolvedores que desejem construir aplicações multiagente específicas para a personalização na Web baseadas na MUW.

As próximas subseções apresentarão a implementação dos agentes de acordo com as camadas em que o framework está dividido.

7.3.1 Camada de processamento das informações do usuário

Como visto na fase de projeto, a estratégia dessa camada é embutir um applet Java em cada uma das páginas do site em questão (Figura 40). Esse applet é então responsável por criar a plataforma JADE e os agentes *Interfaceador* e *Modelador*, servindo de elo entre o software de navegação, capturando os eventos disparados pela interação do usuário com o navegador, e os agentes. Para fins didáticos, pode-se considerar o applet como sendo um braço ou uma extensão do agente *Interfaceador*.

Sendo que na prática, a classe que implementa o applet possui uma referência à classe que implementa o agente *Interfaceador* e vice-versa. Tanto a plataforma JADE, que é onde efetivamente os agentes estarão residindo, quanto o applet que os cria, estarão rodando transparentemente na máquina virtual do software de navegação do usuário.

```
<html>
<head>
<title>MineracaoUsoWeb #1</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body onload="document.applets[0].setURL(document.URL)"
onunload="document.applets[0].setURLAtual(document.activeElement.href)">
<applet code="interfaceador.ClientApplet.class"
archive=".\lib\clientapplet.jar, .\lib\jade.jar, .\lib\Base64.jar, .\lib\iiop.jar, .\lib\jadeTools.jar,
.\lib\http.jar" width="1" height="1">
  <!-- the port of the jade platform -->
  <param name="jadePort" value="2002">
  <!-- the applet container port-->
  <param name="appletContainerPort" value="2004">
</applet>
</body>
</html>
```

Figura 40. Trecho com o código HTML necessário para a utilização do ONTOMUW

O ciclo de vida de um applet Java é definido por quatro métodos: *init* (), *start* (), *stop* () e *destroy* (). A seguir é apresentada uma breve descrição de cada um desses métodos.

- *init* () – Esse método é usado para todas as inicializações que forem necessárias ao applet. Funciona de forma similar a um construtor, ou seja, é chamado automaticamente pelo sistema quando o Java executa o applet pela primeira vez;
- *start* () – Esse método é chamado automaticamente depois que o Java chama o *init* (). Ele é chamado também sempre que o usuário retornar à página contendo o applet, depois de visitar outras páginas. Isso significa que o método *start* () pode ser chamado repetidamente, o que não ocorre com o método *init* (). Por esse motivo deve-se colocar o código que é executado só uma vez no método *init* () e não no *start* ();

- *stop* () – Esse método é chamado automaticamente quando o usuário sai da página em que o applet está. Ele pode, portanto, ser chamado repetidamente no mesmo applet. Seu propósito é dar a chance de parar alguma atividade que ocupe muito o sistema quando o usuário não estiver mais dando atenção ao applet;
- *destroy* () – A linguagem Java garante chamar esse método quando o navegador termina normalmente. Como os applets são feitos para operar dentro de páginas HTML, não é necessário preocupar-se sobre como destruir o applet. Isso acontece normalmente quando o navegador encerra suas atividades.

A Figura 41 apresenta o trecho de código contendo o método *init* do applet utilizado no ONTOMUW. Tanto a plataforma JADE quanto os agentes (*Interfaceador* e *Modelador*) são criados e inicializados pelo applet através desse método. Como esses agentes são criados somente uma vez durante a navegação do usuário é adequado então que essas operações sejam postas no método *init*. Note que algumas das informações necessárias para a criação dos agentes, como o número da porta, por exemplo, são capturadas diretamente da página HTML através das tags especiais “<PARAM>”.

A Figura 42 mostra o método “*start*” sendo utilizado para capturar o instante em que o usuário carrega uma nova página no navegador. Note que o tempo é capturado em milisegundos. Esse método também é responsável por adaptar a interface, nesse caso o navegador. Quando chegam as personalizações processadas e enviadas pelo agente *Modelador*, o *Interfaceador* atualiza a variável *urlRecomendada* do applet e atualiza uma variável flag indicando que essa é uma nova recomendação. Assim que o usuário entra em uma nova página a recomendação é disparada, através do método start do applet. Isso se dá da seguinte forma, cada página do site tem uma função Javascript responsável por mostrar um *layer* na página atual com a recomendação. Essa função recebe a url da página a ser recoemendada através de uma chamada do applet. Esse layer fica por defautl invisível, sendo ativado através de uma chamada do applet.

A Figura 43 apresenta o método “*stop”* sendo utilizado para capturar o instante em que o usuário deixa a página atual. Logo em seguida é computado o tempo de

permanência total do usuário na página através da subtração do instante final menos o inicial, e o envio de uma mensagem ao agente *Modelador* com as informações capturadas para essa visita do usuário.

A Figura 44 mostra o método do applet que dispara uma mensagem do agente *Interfaceador* ao agente *Modelador* com as informações da visita do usuário. Esse método sempre é chamado no método *stop* do applet, pois através desse método o applet é capaz de saber quando o usuário encerrou uma determinada visita.

```
        public void init() {
                try {
                        // captura o nome da máquina cliente
                jadeHostName = InetAddress.getLocalHost().getHostName();
                } catch (UnknownHostException e1) {
                        e1.printStackTrace();
                                }
                        //captura os parâmetros contidos na página HTML
                jadePort = Integer.parseInt(getParameter("jadePort"));
                clientPort = Integer.parseInt(getParameter("appletContainerPort"));
                Specifier s = new Specifier();
                s.setClassName("jade.mtp.http.MessageTransportProtocol");
                Object a[] = new Object[1];
                a[0] = "http://"+jadeHostName+":"+clientPort+"/test";
                s.setArgs(a);
                jade.util.leap.List l = new ArrayList(1);
                l.add(s);
                Profile profile = new ProfileImpl(jadeHostName, jadePort, null);
                profile.setSpecifiers(Profile.MTPS, l);
AgentContainer mc = rt.createMainContainer(profile);
                        try {
                                interfaceador = new Interfaceador();
                                modelador = new Modelador(2,80000,4);
                        mc.acceptNewAgent("Interfaceador", interfaceador).start();
                                //Adiciona e incializa os agentes no container
                                mc.acceptNewAgent("Modelador", modelador).start();
                        } catch (StaleProxyException e) {
                                e.printStackTrace();
                        }
                }
```

Figura 41. Método init contendo as incializações do applet

```
public void start() {
        LoadTime = System.currentTimeMillis();
        JSObject win = JSObject.getWindow(this);
        String[] recomendacao = {URL_Recomendada, URL_Atual};
        win.call("setUrls",recomendacao);
        String[] timeLine = {"Recomendar"};
        win.call("MM_timelinePlay",timeLine);
}
```

Figura 42. Método start do applet

```
public void stop() {
        UnloadTime = System.currentTimeMillis();
        Time = UnloadTime-LoadTime;
        sendMessage(ClientApplet.url,Time);
}
```

Figura 43. Método stop contendo o código que é executado a cada vez que o usuário troca de página

```
public void sendMessage(String endereco, long tempo) {
    //preparando mensagem
    ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
    AID saAID = new AID();
    //Adicionando destinatário
saAID.setName(getFullAgentName(MODELADOR_AGENT_NAME, jadeHostName,
jadePort));
    msg.addReceiver(saAID);
    msg.setLanguage(codec.getName());
    msg.setOntology(ontology.getName());
    //Instanciando conceitos da ontologia
Pagina pagina = new Pagina();
    pagina.setEndereco(endereco);
    VisitaUsuario visita = new VisitaUsuario();
    visita.setPagina(pagina);
    visita.setTempo(tempo);
    FoiRealizada visitaRealizada = new FoiRealizada();
    visitaRealizada.setVisita(visita);
    try {
        //Adicionando conteúdo da mensagem em termos da ontologia de domínio
interfaceador.getContentManager().fillContent(msg, visitaRealizada);
//Enviando mensagem
        interfaceador.send(msg);
....
```

Figura 44. Mensagem enviada pelo agente Interfaceador com as informações da visita do usuário corrente

Para a captura da URL da página sendo visitada é utilizado um método do applet Java a ser chamado por um pequeno script Javascript contido no evento onload[11] da página Web. O script captura a URL da página visitada e a repassa ao applet. Como o applet tem uma referência cruzada ao agente *Interfaceador*, fica fácil passar essas informações ao agente (Figura 45, Figura 46).

```
public void setURL(String url) {

        ClientApplet.url = url;
  }
```

Figura 45. Método para a captura da URL da página sendo visitada

```
<html>
<title>página Web</title>
<body onload="document.applets[0].setURL(document.URL)">
…
```

Figura 46. Script que passa a URL sendo visitada ao applet Java

A mesma estratégia é utilizada para a captura da URL da página atual do usuário. É fundamental que se saiba a página atual em que o usuário está, pois assim não se corre o risco de recomendar a mesma página que o usuário está visitando no momento. A diferença é que aqui o método *setURLAtual* () do applet é chamado a partir do evento unload[12] da página; nesse evento é definido um pequeno script Javascript passando o endereço da página selecionada pelo usuário

Na verdade o agente *Interfaceador* não possui comportamentos. Um agente JADE, mesmo sendo escrito em Java não possui habilidades para a captura de eventos do software de navegação. Isso cabe aos applets. Mas applets Java não são agentes, são apenas programas Java especiais com um ciclo de vida particular e habilidades especiais para interação com softwares de navegação através de suas máquinas virtuais. Sendo assim, por causa de uma impossibilidade prática de implementação, não é o agente *Interfaceador* quem monitora o usuário ou apresenta as recomendações, mas sim um applet. Por outro lado, um applet por não ser um agente, não tem habilidades de

[11] Esse evento é disparado assim que uma nova página é carregada no navegador
[12] Esse evento é disparado assim que a página que está sendo visitada no momento é substituída por outra

comunicação segundo uma linguagem de comunicação entre agentes ou ontologias. Então o que acontece na prática é que o agente *Interfaceador* só vai atuar na formatação das informações capturadas pelo applet segundo a ontologia definida para os agentes, no envio dessas informações ao agente *Modelador* e no recebimento do modelo de adaptação contendo a lista de páginas a serem recomendadas.

O applet só consegue passar ou receber informações do agente *Interfaceador* porque, como já dito antes, possui uma referência cruzada a esse agente.

Já o agente *Modelador* precisa realizar uma série de inicializações ao ser criado. Ele precisa instanciar o espaço de páginas[13], criar a matriz universal de segmentos, inicializar o modelo de usuários e inicializar a sessão de navegação. Alguns desses conceitos estão presentes na ontologia de domínio (espaço de páginas, sessão de navegação e modelo de usuários) e outros são classes auxiliares (matriz de segmentos). Essas inicializações são incluídas no construtor do agente, portanto, ao instanciar o agente, o desenvolvedor precisa passar os parâmetros específicos de sua aplicação por meio do construtor. O agente *Modelador* é criado com três parâmetros correspondentes ao tempo de duração da sessão de navegação, a ordem dos segmentos e a janela de visitas[14].

Vale lembrar que o desenvolvedor é responsável por instanciar o espaço de páginas adicionando cada página contida no site a ele, isso é fundamental para a construção correta das matrizes de características. Nessa primeira versão do ONTOMUW esse processo é manual, ou seja, o desenvolvedor tem que manualmente construir uma lista de todas as páginas contidas no diretório do site, instanciar um conceito *Página* para cada uma delas atribuindo a URL correspondente e depois adicioná-las ao espaço de páginas, que nesse caso corresponde a um objeto da classe *EspaçoPáginas*.

Como se pode ver na Figura 47, o agente possui dois comportamentos adicionados na inicialização (*setup* ()), *CriaModeloUsuarios (*do tipo *"CyclycBehavior")*,

[13] Está-se considerando todos os sites como pertencendo ao mesmo conceito, portanto substituiu-se a terminologia espaço conceitual por espaço de páginas, ou seja, o conjunto de todas as páginas contidas no site

[14] Janela de visitas é o nome dado ao critério que determina quando o usuário deve ser classificado. Se por exemplo a janela de visitas for igual a quatro, o usuário será classificado a cada quatro visitas, recebendo assim as recomendações a cada quatro visitas.

responsável por criar e atualizar o modelo de usuários através das informações provenientes do *Interfaceador* e *FinalizaSessaoUsuario* (do tipo "*TickerBehavior*"), que verifica se o tempo limite para a sessão terminou e se for o caso envia o modelo de usuários ao agente *Aquisitor*. Há mais um comportamento, *SolicitaClassificacaoUsuario* (do tipo "*SimpleBehavior*"), que não é adicionado na inicialização do agente. Esse comportamento implementa um dos protocolos de interação da biblioteca de protocolos do JADE para interações do tipo requisição/resposta ("*AchieveREInitiator/AchieveResponder*") e é adicionado ao agente pelo comportamento *CriaModeloUsuarios*. Isso se dá porque a decisão de quando solicitar a classificação do usuário está atrelada ao número de páginas ou segmentos visitados, e como o comportamento *CriaModeloUsuarios* é cíclico, pode-se testar a cada momento se o usuário já visitou o número de páginas/segmentos definidos de modo a solicitar a classificação. Esse comportamento (*SolicitaClassificacaoUsuario*) é atômico e só termina quando a requisição é respondida. Quando isso acontece, o comportamento inclui o comportamento *ConstruirModeloAdaptacao* que vai então ser responsável por construir o modelo de adaptação do usuário corrente e enviá-lo ao agente *Interfaceador*. Esse comportamento é do tipo *OneShotBehavior*. Esse tipo de comportamento é atômico e é adicionado ao agente sempre que o usuário for classificado e o grupo recebido.

Os agentes do JADE definem um método que é responsável por executar um trecho de código antes que o agente seja destruído, o *takeDown* (). Esse método é útil, pois antes que o agente seja destruído quando o usuário fecha o navegador, o agente deve enviar o modelo de usuários a ser armazenado pelo agente *Aquisitor*. A Figura 47 mostra um trecho de código do agente *Modelador* com os detalhes mais relevantes desse agente.

```
public class Modelador extends Agent {
        private int ordemSegmento;
        private long periodoSessao;
        protected EspacoPaginas espacoPaginas;
        protected MatrizSegmentos matrizSegmentos;
        protected ModeloUsuarios modeloUsuarios;
        protected SessaoUso sessao;
        public Modelador(int ordemSegmento, long periodoSessa) {
                this.ordemSegmento=ordemSegmento;
                this.periodoSessao=periodoSessao;
//Inicializa o espaço de páginas
                espacoPaginas = new EspacoPaginas();
                //inicaliza a matriz universal de segmentos
                matrizSegmentos = new MatrizSegmentos(espacoPaginas,ordemSegmento);
                //Incializa o modelo de usuários
                modeloUsuarios = new ModeloUsuarios();
                //Inicializa a sessão de navegação do usuário
                sessao = new SessaoUso();
//Adiciona os comportamentos dos agentes
}
protected void setup() {
                addBehaviour(new CriaModeloUsuarios(this));
                addBehaviour(new FinalizaSecaoUsuario(this, periodoSessao));
   }
```

Figura 47. Trecho de código do agente Modelador

A Tabela 7 mostra um resumo dos comportamentos do agente *Modelador* segundo as classes de comportamento do JADE.

Agente	Comportamento	Tipo
Modelador	CriaModeloUsuários	CyclicBehavior (cíclico)
	SolicitaClassificaçãoUsuário	SimpleBehavior (atômico, implementa protocolo "AchieveREinitiator/AchieveResponder")
	FinalizaSecaoUsuário	TickerBehavior (periódico)
	CriaModeloAdaptação	OneShotBehavior (atômico)

Tabela 7. Resumo dos comportamentos do agente Modelador

7.3.2 Camada de descoberta de padrões

Esta camada possui dois agentes (*Aquisitor* e *Minerador*) executando em um servidor remoto e atendendo continuamente a requisições da camada de *processamento de informações dos usuários* distribuída pela Web.

Cada agente JADE possui uma caixa de correio interna, onde as mensagens provenientes de outros agentes são armazenadas até que elas possam ser processadas. Isso os torna bastante adequados para o atendimento de requisições, pois, dessa forma, as mensagens que chegarem e não puderem ser atendidas imediatamente ficam aguardando em uma lista de mensagens pendentes. Isso garante que a mensagem não seja descartada se o agente estiver ocupado atendendo a outra requisição.

O agente *Aquisitor* é o mais simples dos agentes, ele possui somente um comportamento (*ArmazenaModeloUsuarios*) que fica continuamente aguardando por modelos de usuários a serem recebidos dos agentes *Modeladores*. Assim que um novo modelo chega, o agente o formata em RDF e o inclui no arquivo de uso. Para a formatação e recuperação dos arquivos RDF utilizou-se o Jena (“A Semantic Web Framework for Java”) [83]. O Jena é um framework código aberto escrito em Java para a construção de aplicações para a Web Semântica. O Jena tem uma poderosa API não só para a leitura/escrita de arquivos RDF, mas como também para a recuperação e consulta de dados contidos nesses arquivos. A Figura 48 mostra o código fonte do agente *Aquisitor*. Note que o nome do arquivo de log é passado no construtor da classe que implementa o agente.

```
public class Aquisitor extends Agent {
        private String nomeArquivo;
public Aquisitor (String valor) {
        this.nomeArquivo=valor;
}
protected void setup() {
        addBehaviour(new ArmazenaModeloUsuarios(this, nomeArquivo));
        }
}
```

Figura 48. Código do agente Aquisitor

Vale lembrar que o modelo de usuários é representado pela sessão de uso do usuário e as suas respectivas matrizes de características, os quais são gravados na íntegra pelo agente *Aquisitor*.

Para recuperar os modelos gravados no arquivo de uso o agente *Minerador* não precisa realizar nenhum processamento adicional, pois os dados estão livres de ruído e inconsistências.

O RDF é bastante adequado para a representação de ontologias simples. Tendo isso em vista, o conhecimento dos agentes quando mapeados ao RDF não perde sua estrutura semântica. O conceito *ModeloUsuarios,* por exemplo, quando mapeado ao RDF se transforma numa fotografia do que era enquanto estava na memória do agente. Isso significa que os dados de uso aqui são justamente o conhecimento que os agentes *Modeladores* possuíam de seus usuários enquanto ativos. Metaforicamente é como se fotografias do conhecimento interno desses agentes fossem capturadas e guardadas para uso futuro pelo agente *Minerador*.

Sendo assim, além de tudo, o RDF provê um mecanismo adequado para a persistência do conhecimento interno dos agentes em forma de dados de uso. A Figura 49 mostra um trecho de código do log de uso gerado pelo agente *Aquisitor*.

Como visto na fase de projeto, o agente *Minerador* possui dois comportamentos, *CriaGrupos* e *ClassificaUsuarioCorrente*. O primeiro é do tipo *TickerBehaviour* e é responsável por periodicamente consultar o arquivo de uso retirando de lá os modelos armazenados e através deles descobrir os grupos de usuários. Depois dos grupos construídos, os seus centróide são calculados. O intervalo de tempo para o reagrupamento, a ser definido pelo desenvolvedor, é passado no construtor da classe que implementa o agente.

Para o agrupamento periódico implementou-se uma classe auxiliar chamada KMeans, que implementa o algoritmo K-Médias (Anexo I). Nessa classe também foi embutido um método responsável pelo agrupamento dinâmico do usuário.

O outro comportamento, *ClassificaUsuarioCorrente*, assim como o comportamento *SolicitaClassificacaoUsuario* do agente *Modelador*, implementa o protocolo *AchieveREInitiator/AchieveResponder* disponível no JADE para conversações

entre agentes do tipo requisição/resposta baseado no FIPA-REQUEST [48] (Figura 50). Esse comportamento é cíclico e fica continuamente esperando por requisições do agente *Modelador*. Então assim que chega uma requisição, o comportamento classifica o usuário no grupo mais similar, atualiza o centróide do grupo e envia o grupo ao agente *Modelador*. A Figura 51 mostra o código fonte do agente *Minerador*. Note que o nome do arquivo de uso e o número de grupos a serem criados são passados no construtor do comportamento.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.0="http://maae.deinf.ufma.br/ONTOWUM/modeloUsuarios#" >
<rdf:Description rdf:nodeID="A6">
    <j.0:matrizTempos rdf:nodeID="A7"/>
    <j.0:sessaoUso rdf:nodeID="A8"/>
    <j.0:matrizVisitas rdf:nodeID="A9"/>
    <rdf:type
rdf:resource="http://maae.deinf.ufma.br/ONTOWUM/modeloUsuarios#ModeloUsuari
os"/>
    <j.0:mSequencia rdf:nodeID="A10"/>
  </rdf:Description>
<rdf:Description rdf:nodeID="A13">

<j.0:endereco>http://maae.deinf.ufma.br/temp/wum/mineracao_uso_web_3.htm</j
.0:endereco>
    <rdf:type
rdf:resource="http://maae.deinf.ufma.br/ONTOWUM/modeloUsuarios#Pagina"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A14">
<rdf:Description rdf:nodeID="A29">
    <j.0:pagina rdf:nodeID="A30"/>
    <rdf:type
rdf:resource="http://maae.deinf.ufma.br/ONTOWUM/modeloUsuarios#VisitaUsuari
o"/>
    <j.0:tempo>3125.0</j.0:tempo>
  </rdf:Description>
<rdf:Description rdf:nodeID="A113">
    <j.0:ordemSegmento>2</j.0:ordemSegmento>
    <rdf:type
rdf:resource="http://maae.deinf.ufma.br/ONTOWUM/modeloUsuarios#SessaoUso"/>
    <j.0:visitas rdf:nodeID="A114"/>
  </rdf:Description>
<rdf:Description rdf:nodeID="A4">
    <rdf:_4>0.0</rdf:_4>
    <rdf:_5>0.0</rdf:_5>
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Seq"/>
    <rdf:_2>8593.0</rdf:_2>
    <rdf:_3>0.0</rdf:_3>
    <rdf:_1>9188.0</rdf:_1>
  </rdf:Description>
<rdf:Description rdf:nodeID="A171">
    <j.0:colunas>5</j.0:colunas>
    <j.0:linhas>5</j.0:linhas>
    <rdf:type
rdf:resource="http://maae.deinf.ufma.br/ONTOWUM/modeloUsuarios#MatrizCaract
eristicas"/>
    <j.0:matrizTempos rdf:nodeID="A172"/>
  </rdf:Description>
```

Figura 49. Trecho do arquivo de uso em RDF gerado pelo Aquisitor

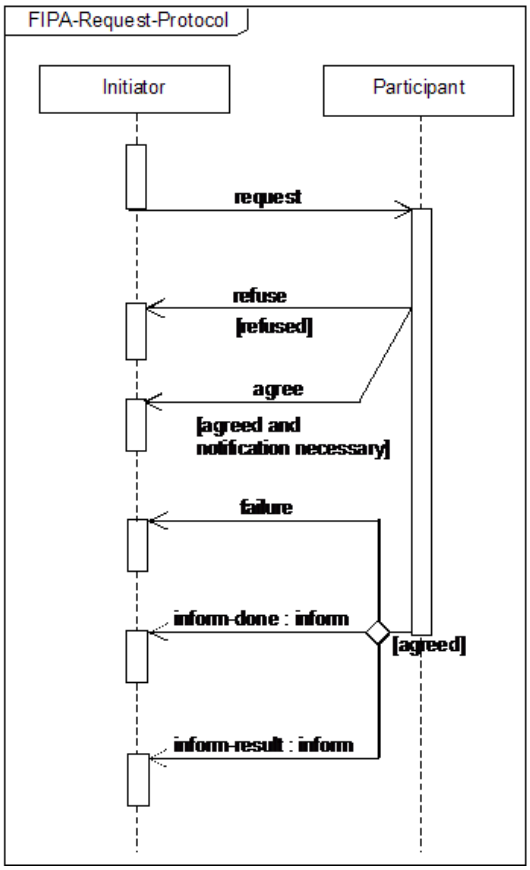

Figura 50. Protocolo de comunicação FIPA-REQUEST [48]

```
public class Minerador extends Agent {
        private long periodoReagrupamento;
        protected KMeans kMeans;

public Minerador (long valor) {
        this. periodoReagrupamento=valor;
}
protected void setup() {
        MessageTemplate mt =
AchieveREResponder.createMessageTemplate(FIPAProtocolNames.FIPA_REQUEST);
                addBehaviour(new ClassificaUsuarioCorrente(this, mt));
                addBehaviour(new CriaGrupos(this, periodoReagrupamento, “Log.rdf”, 2));
        }

}
```

Figura 51. Código do agente Minerador

A Tabela 8 a seguir resume os comportamentos dos agentes da camada de descoberta de padrões do ONTOMUW.

Agente	Comportamento	Tipo
Aquisitor	ArmazenaModeloUsuários	CyclicBehavior (cíclico)
Minerador	RemontaModeloUsuários	TickerBehavior (periódico)
	ClassificaUsuárioCorrente	CyclicBehavior (cíclico, implementa protocolo "AchieveREinitiator/AchieveResponder")

Tabela 8. Resumo dos comportamentos dos agentes da camada de descoberta de padrões do ONTOMUW

7.3.3 Exemplos de execução do ONTOMUW

Para ilustrar o funcionamento dos agentes foi construído um pequeno site, constituído de cinco páginas e sem conteúdo ou propósito específico. Sendo que cada página possui um vínculo para cada uma das outras páginas do site, incluindo para si mesma. Os segmentos foram definidos como sendo de ordem dois para a construção das matrizes. A Figura 52 mostra uma das páginas desse site. Note o destaque na barra de status do navegador indicando que o applet foi iniciado na página.

A Figura 53 apresenta o console da máquina virtual Java do navegador indicando a inicialização da plataforma JADE. É importante relembrar que a plataforma JADE só é carregada uma vez durante uma sessão de navegação do usuário. Isso é feito sempre que o usuário acessa a primeira página do site em questão, sendo que a primeira aqui pode ser qualquer página do site, e não somente a página principal ("*home*").

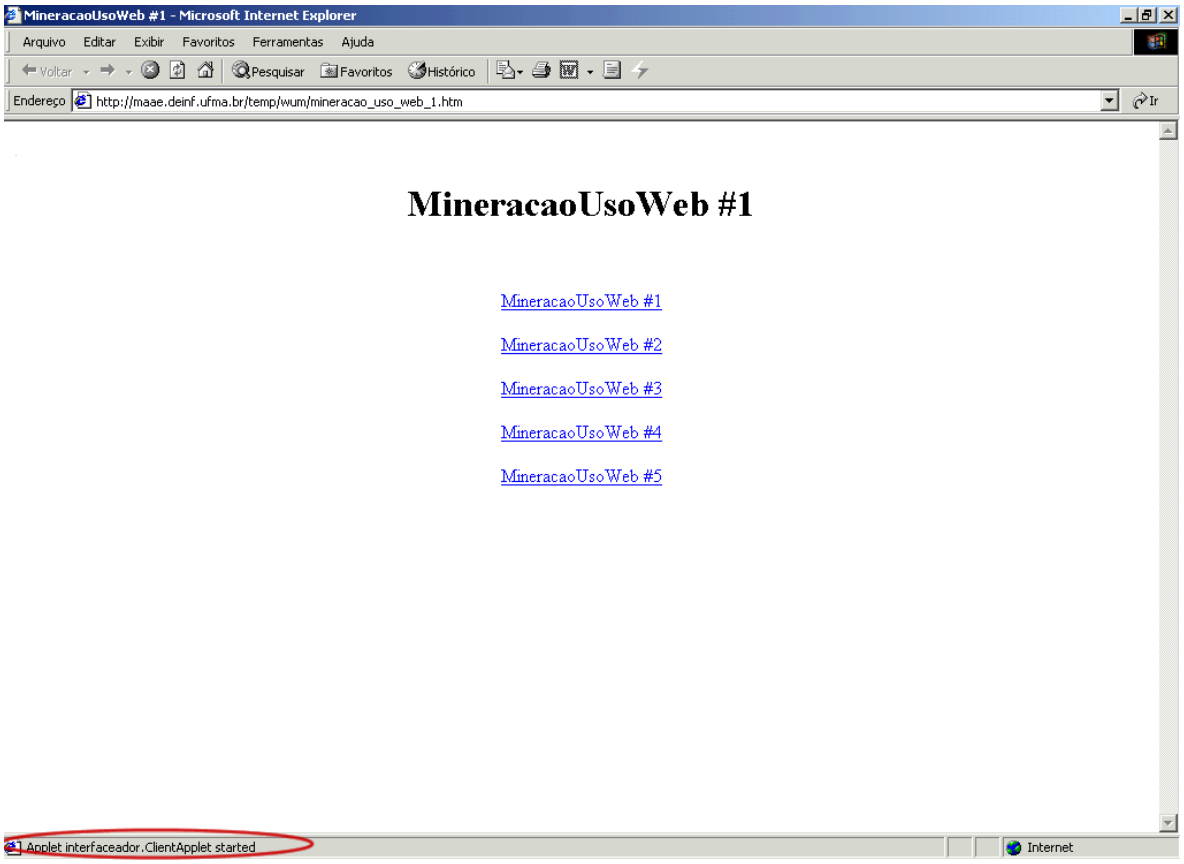

Figura 52. Exemplo de uma página genérica com destaque para o applet sendo carregado

Figura 53. Console da máquina virtual Java do navegador indicando a inicialização da plataforma JADE

A Figura 54 mostra o utilitário do JADE "*snifferAgent*" mostrando a passagem de uma mensagem entre o agente *Interfaceador* e o agente *Modelador*. Essa figura mostra a primeira mensagem passada entre os agentes, ou seja, a mensagem correspondente à primeira visita realizada pelo usuário (conceito visita da ontologia). A cada visita realizada pelo usuário uma mensagem é enviada ao agente *Modelador* pelo agente *Interfaceador*, contendo as respectivas informações dessa visita. A partir dessas informações o agente *Modelador* atualiza a sessão de navegação do usuário e pode construir as matrizes de características para a sessão de navegação atual. Como aqui a ordem dos segmentos foi definida como dois, o agente só pode construir as matrizes de visitas assim que o usuário tiver realizado pelo menos duas visitas no site. De forma a relembrar a terminologia do modelo FM, como por exemplo, segmentos e ordem de segmentos o capítulo 6 deve ser reconsultado.

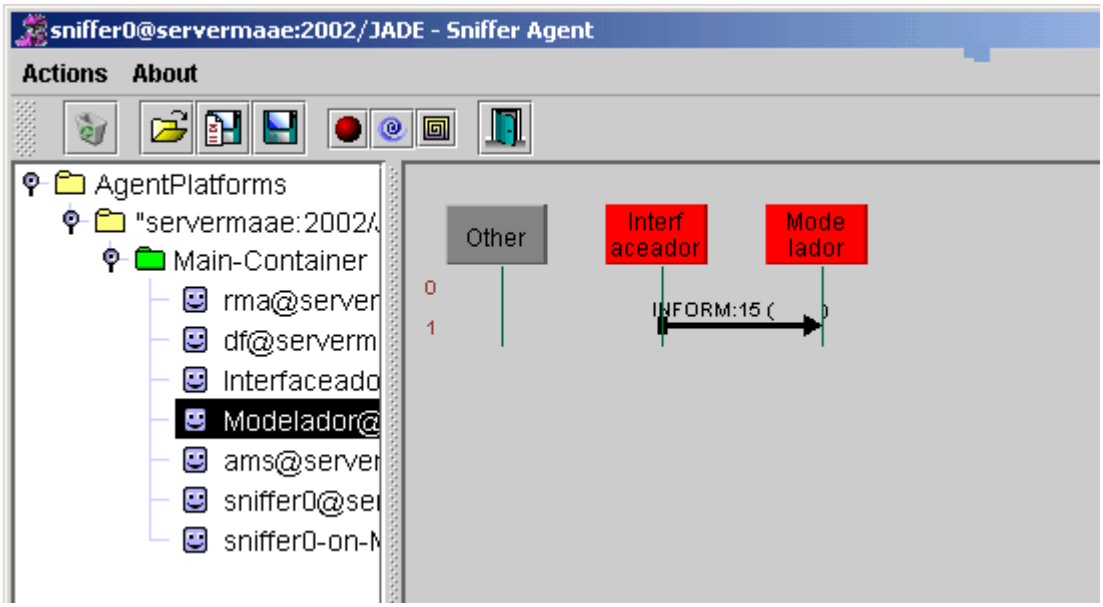

Figura 54. Utilitário do JADE mostrando uma troca de mensagens entre os agentes Interfaceador e Modelador

A Figura 55 ilustra a segunda mensagem recebida pelo agente *Modelador* (segunda visita do usuário no site). A partir daí o agente *Modelador* já pode construir as matrizes de características para a sessão atual. A Figura 56 mostra a saída do console da máquina virtual Java do navegador mostrando as matrizes construídas. Vale relembrar que a partir daí sempre que chegar uma nova visita as matrizes serão reconstruídas,

mantendo assim o modelo de usuários atualizado. Como o usuário só visitou um segmento do site, a maioria das posições das matrizes estão zeradas.

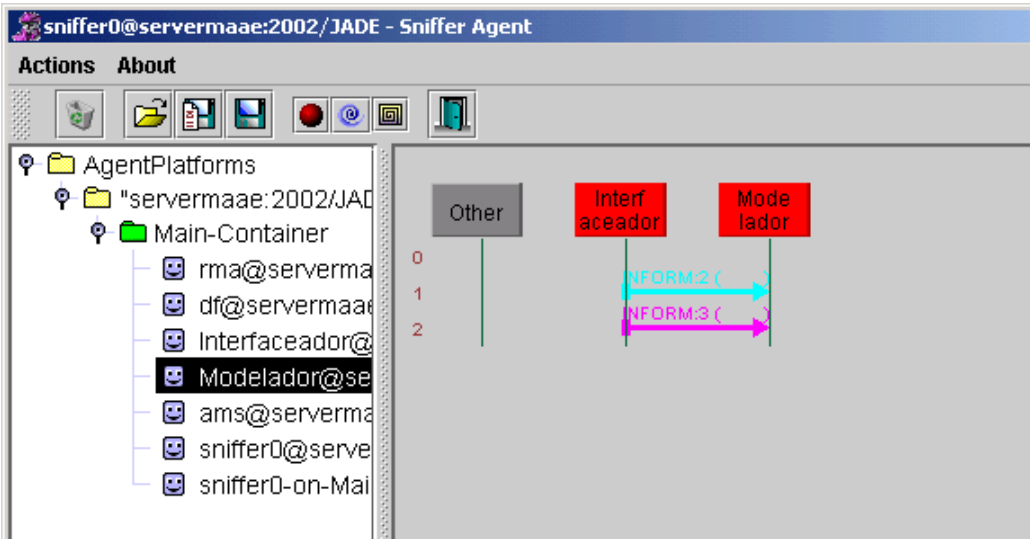

Figura 55. Segunda mensagem recebida pelo agente Modelador

A janela de visitas foi definida com valor quatro, ou seja, a cada quatro páginas o usuário é classificado/reclassificado, e consequentemente, a recomendação é realizada a cada quatro visitas. Mas nem sempre se tem a garantia de que a recomendação será realizada. Isso se dá, por exemplo, quando o usuário é o primeiro a utilizar o site, ou seja, como ainda não existem dados de uso de outros usuários, consequentemente não existirão grupos nos quais o usuário atual possa ser classificado. Esse problema é conhecido na literatura como "*cold start*", e o tratamento adequado desse problema fica como sugestão para trabalhos futuros. Outro caso onde não acontecem recomendações é quando nenhum membro do grupo ao qual o usuário foi classificado visitou o segmento atual em que o usuário atual está, ou seja, não há como basear a recomendação em um segmento por onde ninguém jamais passou.

A Figura 57 mostra a dinâmica da sociedade de agentes da ONTOMUW. Na figura o retângulo rotulado como "*Other*" representa um agente que reside em outra plataforma que não a atual, nesse caso os agentes *Minerador* e *Aquisitor*. Além das mensagens rotuladas com "REQUEST" solicitando a classificação do usuário atual, se vê uma mensagem saindo da plataforma com a performativa "INFORM". Isso indica o

término de uma sessão de usuário pela expiração do tempo pré-definido pelo desenvolvedor e o envio do modelo a ser armazenado pelo *Aquisitor*.

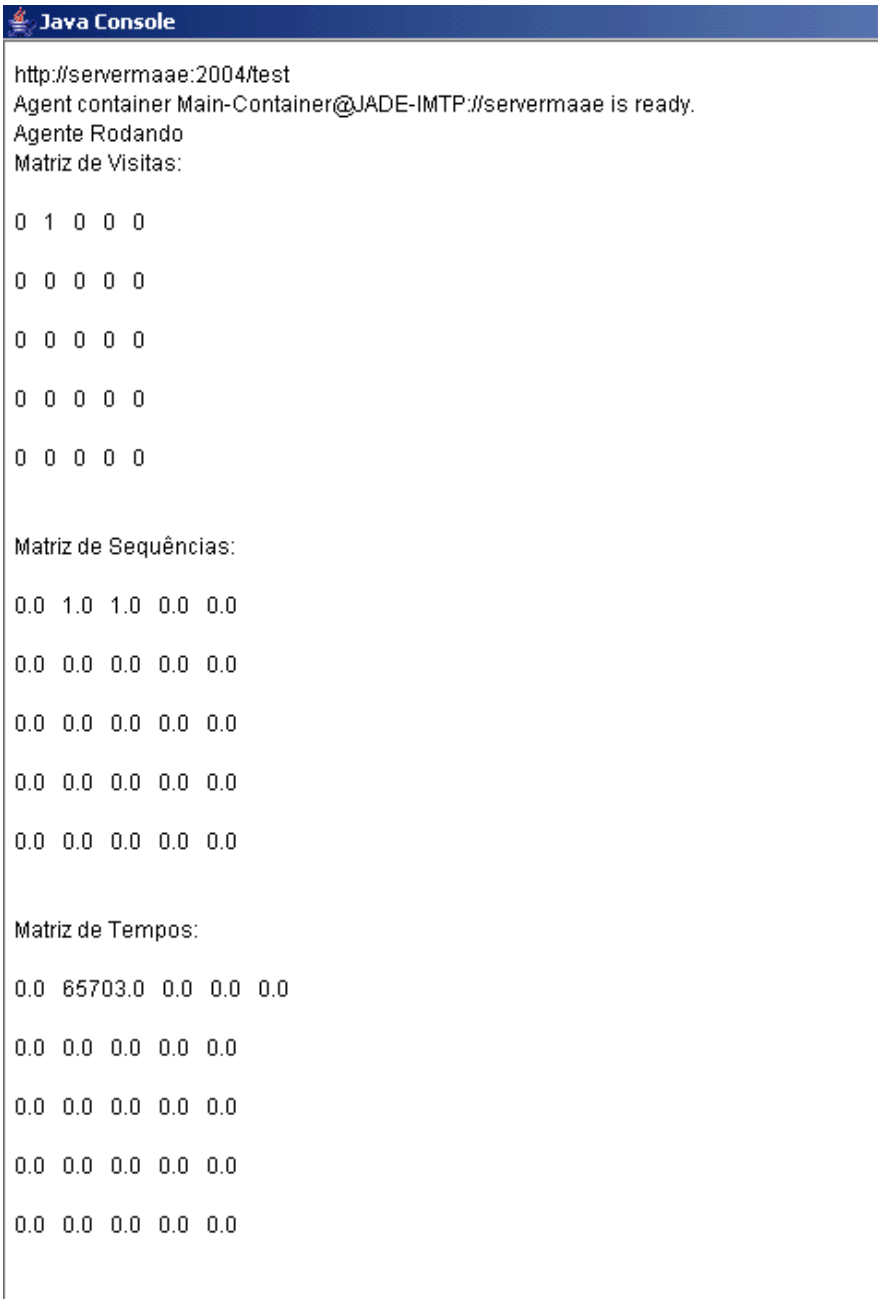

Figura 56. Saída do console Java mostrando as matrizes de características construídas para a sessão atual do usuário

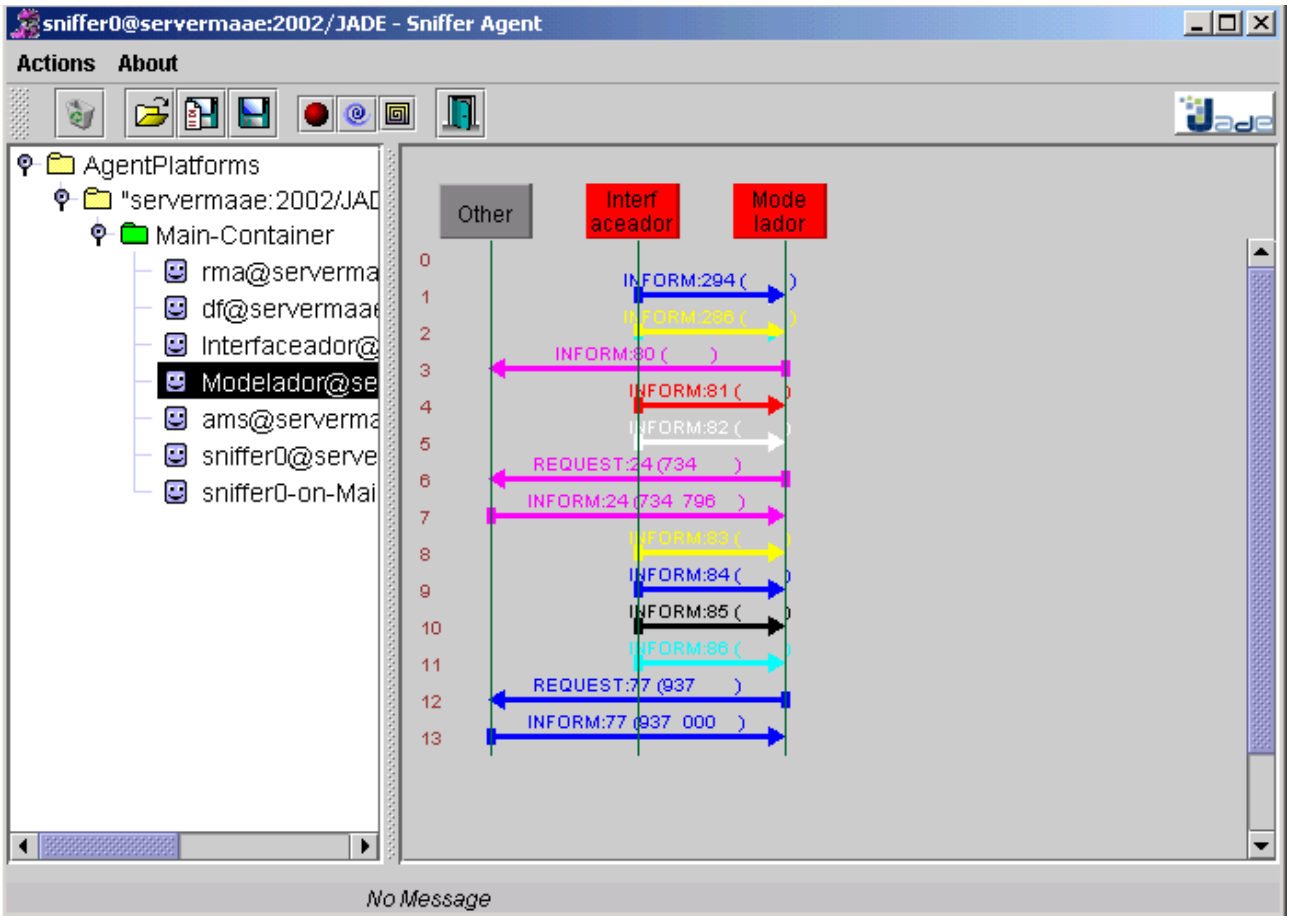

Figura 57. Dinâmica da sociedade de agentes do ONTOMUW

A Figura 58 mostra a camada de descoberta de padrões atendendo as requisições da camada de processamento de informações dos usuários.

A Figura 59 ilustra uma recomendação sendo realizada assim que a janela de visitas é alcançada. A recomendação toma a forma de uma pequena janela que desliza sobre a página visitada, fica um tempo na tela e depois desaparece. Com já dito isso é feito através de uma chamada do applet à uma função Javascript contida na página.

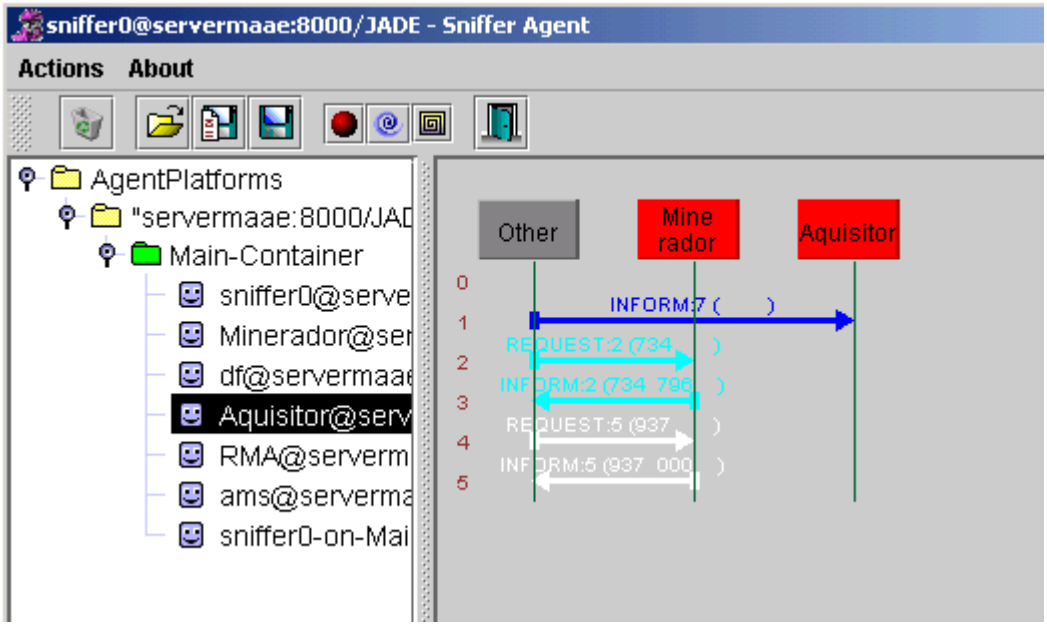

Figura 58. Camada de descoberta de padrões atendendo as requisições da camada de processamento de informações dos usuários

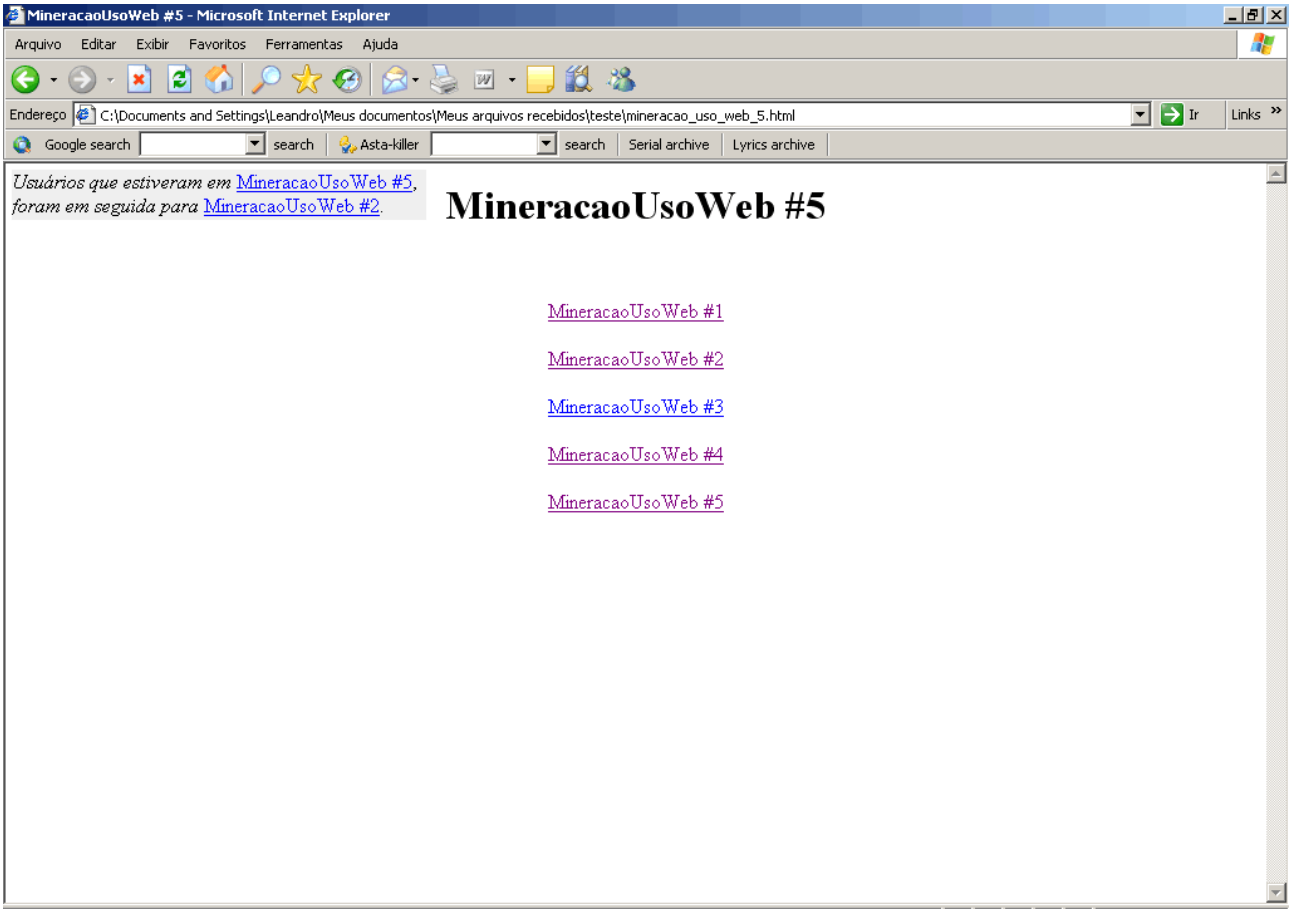

Figura 59. Adaptação do navegador mostrando a recomendação ao usuário

7.4 Considerações finais do capítulo

Esse capítulo apresentou a implementação de domínio do ONTOMUW. A metodologia MADEM, em seu estágio atual, ainda não contempla a fase de implementação de domínio da Engenharia de domínio multiagente. Sendo assim, optou-se pelo ambiente de desenvolvimento multiagente JADE [82]. O JADE é código aberto, bem documentado, escrito em Java e está em total conformidade com a FIPA, razões essas que justificam a sua escolha.

A implementação da ontologia dos agentes e algumas classes auxiliares são mostradas no Anexo I.

Apesar de não se ter utilizado nenhuma metodologia formal para a implementação do domínio, o mapeamento das especificações da fase de projeto para a implementação foi bastante natural.

O applet que é descarregado na primeira visita do usuário ao site sendo monitorado ainda sofre de um problema de desempenho, pois de modo a criar os agentes é necessário que as classes do ambiente JADE também sejam descarregadas, o que torna a execução do applet, na primeira visita, bastante lento. Uma forma de resolver esse problema no futuro seria uma recompilação do JADE eliminando as classes consideradas desnecessárias para os propósitos do ONTOMUW. O JADE possui muitas classes, e a maioria delas é dispensável no caso dos agentes *Interfaceador* e *Modelador* por serem agentes relativamente simples.

De forma que o ONTOMUW possa ser utilizado em um site real, as páginas do site precisam ser modificadas de modo a incluírem a chamada ao applet que cria os agentes e alguns scripts do Javascript. Isso é um gargalo, pois se o site for grande, o desenvolvedor vai ter bastante trabalho em modificar cada página individualmente. Uma possível melhoria nesse sentido seria o desenvolvimento de uma ferramenta automática responsável por percorrer o diretório do site em o que o ONTOMUW seria implantado, inserindo automaticamente o código HTML necessário.

De modo a reutilizar os produtos gerados pela Engenharia de domínio precisa-se de técnicas e metodologias da Engenharia de aplicações. Nesse sentido o grupo

GESEC [55] está construindo uma metodologia para a Engenharia de aplicações que define técnicas para a construção de aplicações específicas a partir dos produtos gerados pela Engenharia de domínio. A partir daí, o ONTOMUW poderá ser mais bem avaliado, pois uma metodologia formal será utilizada para a sua efetiva reutilização em uma aplicação real.

8 CONCLUSÃO

Com a incessante migração das mais variadas categorias de serviços para a Web, a necessidade de caracterizar os usuários nunca foi tão necessária. A Web é por natureza dinâmica e heterogênea, portanto, para modelar o usuário nesse contexto são necessários componentes que sejam aptos a perceber e rapidamente se adaptarem às mudanças do ambiente. Por estarem inseridos em um ambiente, percebendo-o incessantemente através de sensores e por suas características tais como autonomia, capacidade de raciocínio e sociabilidade, o paradigma computacional dos agentes aparece como uma abordagem particularmente adequada a esse problema.

A MUW oferece uma solução bastante interessante ao problema da modelagem de usuários. Na MUW, o modelo de usuários é construído automaticamente através dos dados de uso gerados pela interação do usuário com a Web. Esses modelos, que podem representar interesses, preferências e objetivos do usuário, podem ser utilizados automaticamente por módulos de personalização automática para a adaptação da interface do usuário.

Pela naturalidade que os agentes oferecem para tratar a complexidade do software, a junção entre a tecnologia dos agentes e a MUW é uma abordagem adequada para a construção de sistemas de personalização automática para a Web baseada na MUW.

Mas, para construir tais aplicações é necessário que as boas práticas de desenvolvimento da Engenharia de software, especialmente àquelas concernentes à reutilização, sejam aplicadas também ao paradigma computacional dos agentes. Nesse sentido, a Engenharia de domínio multiagente tem contribuído no sentido de fornecer direcionamentos para a construção de abstrações de software reutilizáveis para o paradigma dos agentes. Em particular, o grupo de pesquisa GESEC, tem desenvolvido a MADEM, uma metodologia para as fases de Análise e Projeto de domínio da Engenharia de domínio mutiagente. A MADEM guiou as fases de Análise e projeto de domínio da ONTOMUW.

8.1.1 Resultados e contribuições de pesquisa

Seguem abaixo as principais contribuições deste trabalho.

- *Uma avaliação da metodologia MADEM através da sua aplicação na construção do framework ONTOMUW*. A MADEM é uma metodologia em consolidação e, portanto, ainda existem algumas lacunas, as quais foram identificadas durante a construção do ONTOMUW. Dentre elas as mais importantes são:
  - Falta de técnicas para o levantamento dos requisitos não funcionais na fase de Análise de domínio;
  - Falta de uma tarefa concernente à modelagem do conhecimento dos agentes na fase de Projeto de domínio;
  - A modelagem de variabilidades ainda não está muito bem definida na MADEM, pois, as regras que definem a identificação da variabilidade do framework são muito empíricas e por esse motivo carecem de maior formalismo;
  - Alguns dos modelos gerados pela ONTOMADEM na fase de projeto carecem de maior poder de representação, principalmente se comparados à AUML;
  - A MADEM ainda não contempla uma fase referente à implementação de domínio.
- *Construção de um modelo de domínio para sistemas de personalização da Web baseados na MUW*. O modelo foi construído segundo as atividades definidas pela MADEM e gerado através da ONTOMADEM. O modelo de domínio comporta os requisitos funcionais e não funcionais do ONTOMUW, e especifica, fundamentalmente, o que deve ser feito de forma a alcançar os objetivos definidos;
- *Especificação de projeto do framework ONTOMUW*. Para as decisões de projeto, tais como que algoritmos e técnicas utilizar, contou-se com o

conhecimento adquirido durante a pesquisa sobre a MUW, e optou-se pelos que se mostraram mais completos e flexíveis dados os objetivos do ONTOMUW. A flexibilidade é particularmente importante, pois, um framework é por natureza, um trabalho inacabado e por esse motivo sempre deve estar apto a permitir extensões;

- *Enriquecimento dos logs de uso com semântica*. Implementou-se uma forma bastante elegante de aumentar o poder de interpretação dos logs de uso dos usuários através do mapeamento do conhecimento interno dos agentes para grafos semânticos do RDF;
- *Implementação de domínio dos agentes no ambiente JADE*. Os agentes foram implementados em um ambiente que: está em conformidade com a FIPA; é muito bem documentado, é código aberto e é escrito em Java. Essas características (padronização, portabilidade e acesso ao código fonte do ambiente) facilitam bastante a reutilização desses agentes em aplicações específicas reais;
- Extração de um sistema de padrões de projeto arquitetural e detalhado a partir do ONTOMUW [57].
- Publicações
    - Ribeiro I., Marinho, Leandro B. and Girardi, R.: Padrões baseados em agentes para a Modelagem de Usuários e Adaptação de Sistemas, Proceedings of the Fourth Latin American Conference on Pattern Languages of Programming: SugarLoafPLoP 2004, Fortaleza - CE, Brasil. August 10 - 13, 2004.
    - Marinho, Leandro B., Girardi, R., et al.: A Development Experience on Agent-based User Modeling, Proceedings of the Workshop on Architectures and Methodologies for Building Agent-Based Learning Environments, held in conjunction with SBIA, SBRN, IEEE MLSP, Ed. Livro Rápido, pp. 17-28. São Luís, Maranhão, Brasil. September 28th, 2004.

- Girardi, R., Faria, C. and Marinho, L.: Ontology-based Domain Modeling of Multi-Agent Systems. Proceedings of the Third International Workshop on Agent-Oriented Methodologies at International Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2004), Ed. Cesar Gonzalez-Perez, pp. 51-62. Vancouver, Canada. October 24th to 28th, 2004.
- Girardi, R., Marinho, Leandro B.: A System of Agent-based Patterns for User Modeling based on Usage Mining, Interacting with Computers Journal, 2005. (A ser publicado)
- Marinho, Leandro B. and Girardi, R.: An Ontology-based Domain Model for Usage Mining-based Web Personalization Systems. Information Systems Journal, 2005. (Submetido)

8.1.2 Trabalhos futuros

Em relação à modelagem de variabilidades, a MADEM apesar de descrever uma técnica para a fase de análise, não faz o mesmo para a fase de projeto. E mesmo essa técnica definida para a fase de análise é bastante informal, baseada em regras bastante empíricas, o que acaba por comprometer um pouco o grau de formalismo dessa tarefa.

A variabilidade do framework ONTOMUW fica bastante evidente em se tratando dos papéis a serem desempenhados pelos agentes. Apesar da grande maioria das aplicações do ONTOMUW serem voltadas para a personalização automática da Web, podem haver aquelas onde a personalização é dada de forma estática, ou seja, o usuário especialista toma conhecimento dos padrões de uso de seu site e altera manualmente a estrutura do mesmo baseado no conhecimento contido nos padrões. Nesse tipo de aplicação, o agente *Interfaceador* só precisa desempenhar o papel de *Monitor* e o agente *Minerador* por sua vez, não precisa desempenhar o papel de *Classificador*. Mas em contrapartida precisa-se de mecanismos de visualização dos padrões descobertos, de forma a apresentá-los amigavelmente aos usuários especialistas. Nesta primeira versão,

o ONTOMUW não possui esse último mecanismo, o que pode ser uma extensão para trabalhos futuros.

O grupo GESEC está trabalhando atualmente em uma metodologia para a Engenharia de aplicações multiagente que deve permitir uma sistematização na utilização dos frameworks gerados na Engenharia de domínio multiagente. O próximo passo natural é então oferecer ao usuário, não só o framework tal como foi apresentado aqui, mas toda uma metodologia formal de como reutilizá-lo para a construção de uma aplicação específica real. Isto servirá então como uma avaliação para o ONTOMUW, mostrando o seu real grau de efetividade.

Até o presente momento não foi encontrado nenhum trabalho similar que pudesse ser comparado qualitativamente com o framework aqui proposto. Praticamente todos os trabalhos encontrados na literatura são voltados para a Engenharia de aplicações, mesmo que na maioria das vezes isto não esteja explicitado [15], [78], [110].

O applet que é criado na primeira visita do usuário ainda tem um sério problema de desempenho. Isso se dá porque de forma a criar a plataforma JADE e os agentes *Interfaceador* e *Modelador*, o applet precisa descarregar as classes do ambiente JADE, o que torna o carregamento da primeira página bastante lento. Isso pode ser amenizado no futuro através de uma recompilação do JADE, onde se pode eliminar as classes desnecessárias do ambiente, deixando que só as classes realmente usadas pelos agentes sejam carregadas. Essa é também uma das vantagens de se estar trabalhando com um ambiente de código aberto, ou seja, a flexibilidade em se poder alterar o ambiente de acordo com as necessidades.

Para que o ONTOMUW possa ser utilizado em um site real, as páginas do site precisam ser modificadas de modo a incluírem a chamada ao applet que cria os agentes e alguns scripts do Javascript. Isso é um sério gargalo, pois se o site for grande, o desenvolvedor vai ter bastante trabalho em modificar cada página individualmente. Uma possível melhoria nesse sentido seria o desenvolvimento de uma ferramenta automática responsável por percorrer o diretório do site em o que o ONTOMUW seria implantado, inserindo automaticamente o código HTML necessário.

Como foi explicitado ao longo de todo o manuscrito muitas das idéias usadas para a construção do framework foram extraídas do trabalho de Shahabi et al. (2003) [131]. O modelo FM proposto por ele preenche um dos principais requisitos para a construção de um framework, ou seja, a flexibilidade, além, é claro, de ser comprovadamente um modelo muito poderoso para a representação do comportamento de navegação de usuários anônimos.

8.1.3 Extensões futuras do ONTOMUW

- *Outras técnicas para a modelagem de usuários*. Existem inúmeras técnicas para a modelagem de usuários, as quais podem ser adicionadas ao framework com o intuito de oferecer mais opções ao desenvolvedor, pois como já dito, um framework é por natureza um trabalho inacabado e, portanto, sempre deve estar apto a suportar melhorias e extensões;
- *Outros algoritmos de agrupamento*. Outra extensão bastante desejável diz respeito aos algoritmos de agrupamento. O K-Médias foi escolhido inicialmente por causa de sua simplicidade do ponto de vista da implementação, mas o K-Médias não está na lista dos melhores algoritmos para o agrupamento existentes. Com certeza seria desejável que o desenvolvedor pudesse ter um leque maior de opções à sua disposição;
- *Suporte a outras categorias de mineração na Web*. A MUW vem cada vez mais sendo integrada às outras categorias da mineração na Web (mineração de conteúdo e mineração de estrutura) de forma a aumentar o poder de representação dos padrões descobertos [107]. Nesse trabalho focou-se única e exclusivamente na MUW, e, portanto, trabalhos propondo a extensão do ONTOMUW nesse sentido seriam muito bem vindos;
- *Suporte a outras funções de personalização*. Como visto no capítulo 3 existem diversas funções de personalização, como a memorização, a

orientação e a adaptação, por exemplo. A versão atual da ONTOMUW só oferece a função de orientação, onde links são recomendados aos usuários de acordo com os seus interesses. Seria desejável a extensão do framework de forma que o desenvolvedor pudesse escolher a função de personalização mais adequada à sua aplicação;

- *Suporte à Web Semântica*. Uma outra extensão bastante importante e interessante diz respeito à Web Semântica. Como se pôde notar, o ONTOMUW foi construído de forma a trabalhar no contexto da Web tradicional. Recentemente tem surgido uma nova classe de métodos, técnicas e algoritmos para a mineração aplicada à Web Semântica [9]. Nesse contexto, os sites e suas páginas contêm significado, significado esse definido através de ontologias. Sendo assim, um novo mundo de oportunidades se abre para a MUW, especialmente quando as tarefas da mineração são executadas por agentes. Se antes o conhecimento dos agentes era definido de forma prévia pelo desenvolvedor, devido principalmente à falta de significado da Web tradicional do ponto de vista dos agentes, na Web Semântica, esse conhecimento pode ser construído dinamicamente pelos agentes através das ontologias contidas nos sites e documentos da Web. O modelo FM já foi construído de forma a suportar a semântica contida nas páginas visitadas pelo usuário. Na verdade, como pode ser visto no capítulo 6, o modelo FM define um espaço conceitual, que são todos os conceitos possíveis existentes em um site. E daí, a matriz de segmentos é construída não baseada em segmentos de visitas, mas sim em segmentos de conceitos. Aqui, por não se ter trabalhado nem com análise textual (mineração de conteúdo) nem com a Web Semântica, essa terminologia foi mudada para espaço de páginas, onde assumiu-se todas as páginas de determinado site como pertencentes ao mesmo conceito. Portanto, o modelo FM continua sendo válido também para a Web Semântica.

8.1.4 Considerações finais

É importante ressaltar mais uma vez, que com esse trabalho não se buscou a definição de novos métodos para as tarefas associadas à MUW ou à personalização da Web baseada na MUW, como é mais comum na literatura, mas sim, construir artefatos de software reutilizáveis segundo o paradigma computacional dos agentes que contemplem as soluções em estado da arte para a personalização da Web baseada na MUW.

Como já dito, a MADEM, por estar em fase de consolidação, ainda não define de maneira completa algumas tarefas que são consideradas essenciais para a Engenharia de domínio multiagente. Sendo assim, estas foram incluídas à medida que se foi sentindo a necessidade.

Uma prática que está se tornando bastante comum na área da MUW é o enriquecimento semântico dos arquivos de log de uso [114]. Já está mais do que comprovado que os arquivos de log de uso, puramente, não são suficientes para a descoberta de padrões relevantes. Nesse sentido, e inspirados no trabalho de Shahabi et al. (2001) [132], definiu-se um novo meio de armazenar e manter os dados de uso dos usuários. Aqui o comportamento de navegação do usuário é formalizado por um modelo de usuários. Esse modelo consiste precisamente da sessão de navegação do usuário contendo todas as visitas realizadas até o término da sessão, e todas as matrizes de características geradas até então. Todos esses conceitos, modelo de usuários, páginas, visitas, sessão e matriz de características estão definidos em uma ontologia e, portanto, fazem parte do conhecimento interno dos agentes. O RDF é uma linguagem de descrição de recursos (conceitos) padronizada pela W3C que é bastante adequada para a descrição de ontologias simples, como é o caso da ontologia do ONTOMUW.

Pode-se pensar no RDF como um modelo de dados, onde se tem um esquema para o modelo (ontologia) e os registros de dados de acordo com esse esquema (instâncias da ontologia). Então ao armazenar os modelos de usuários, se está guardando fotografias do conhecimento interno dos agentes (Modeladores) enquanto ativos. Além de ser uma forma bastante elegante de armazenamento dos dados de uso do usuário, ainda existem diversas ferramentas, na maioria código aberto, para a manipulação e visualização gráfica de arquivos RDF [83].

Por último, mas não menos importante, conseguiu se observar na prática o poder da abstração agente para a modelagem de problemas complexos. Basta observar, que se estivéssemos trabalhando com qualquer outro paradigma computacional na construção do ONTOMUW, se estaria falando em componentes, programação baseada em sockets, chamadas a procedimentos remotos e programação concorrente. Não que o paradigma multiagente elimine completamente a necessidade de se trabalhar com programação concorrente ou programação baseada em sockets, pois podem existir situações práticas onde se faça necessária tais abordagens. Mas a intenção é que com o amadurecimento da tecnologia dos agentes isso fique transparente ao desenvolvedor, de forma que a discussão se baseie em agentes de software (componentes), troca de mensagens (programação baseada em sockets) e comportamentos (programação concorrente), onde a distância cognitiva é reduzida e com isso também a complexidade atrelada ao desenvolvimento de software.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Amazon.com, disponível em: http://www.amazon.com, acesso em: 31/11/2004.

[2] AYRES, L., 2003, **"Estudo e Desenvolvimento de Sistemas Multiagentes usando JADE: Java Agent Development Framework",** Monografia de Conclusão de Curso, Universidade de Fortaleza – UNIFOR, Centro de Ciências Tecnológicas – CCT, Curso de Informática, Fortaleza – Ceará, junho de 2003.

[3] AGRAWAL, R., SRIKANT, R., 1994, **"Fast algorithms for mining association rules"**, In: Proceedings of the 20th VLDB Conference, Santiago Chile, pp. 487-499.

[4] ARANGO, G., 1998, **"Domain Engineering for Software Reuse"**, Ph.D. Thesis, Department of Information and Computer Science, University of California, Irvine.

[5] ARANGO, G., 1994, **"Domain Analysis"**, Encyclopedia of Software Engineering, Volume 1, John J. Marciniak, Editor-in-chief, John Wiley & Sons, Inc., pp. 424-434.

[6] ARANGO, G., PRIETO-DIAZ, R., 1991, **"Domain Analysis Concepts and Research Directions"**, In Domain Analysis and Software Systems Modeling, IEEE Computer Society Press.

[7] BEAUMONT, I., 1994, **"User Modeling in the Interactive Anatomy Tutoring System ANATOM-TUTOR",** User Modeling and User-Adapted Interaction Journal Vol 4. nº 1 pp. 21-45.

[8] BECKER, K., VANZIN, M., 2004, **"Mineração do uso da Web"**, Proceedings of the 19th Brazilian Symposium on Databases, October 18-22, Brasília, Brazil.

[9] BERENDT, B., HOTHO, A., STUMME, G., 2002, **"Towards semantic web mining"**, Proceedings of the 1st International Semantic Web Conference (ISWC-02), pp. 264-278. Springer-Verlag, 2002.

[10] BESTAVROS, A., 1995, **"Using Speculation to Reduce Server Load and Service Time on the WWW"**, In: Proceedings of CIKM'95: The 4th ACM International Conference on Information and Knowledge Management, Baltimore, Maryland, pp. 403-410.

[11] BEZDEK, J. C., 1981, **"Pattern Recognition with Fuzzy Objective Function Algorithms"**, Plenum Press: New York.

[12] BISWAS, G., WEINBERG, J. B., FISHER, D., 1998, **"ITERATE: A conceptual clustering algorithm for data mining"**, IEEE Transactions on Systems, Man and Cybernetics, 28, pp.100-111.

[13] BORGES, J., LEVENE, M., 1999, **"Data mining of user navigation patterns"**, In: Proceedings of Workshop on Web Usage Analysis and User Profiling (WEBKDD), in conjunction with ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA., pp.31-36.

[14] BORGES, J., LEVENE, M., 1998, **"Mining association rules in hypertext databases"**, In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98). New York City, New York, USA.

[15] BOULLOSA, J. R. F., 2002, **"Um ambiente para mineração de utilização da Web"**, Dissertação de mestrado – Área: Banco de dados, Universidade Federal do Rio de Janeiro, COPPE/UFRJ.

[16] BRUSILOVSKY, P., MAYBURY, M., 2002, **"From Adaptive Hypermedia to the Adaptive Web"**, Communications of ACM, Vol. 45, nº 5, pp. 31-33, 2002.

[17] BUCHNER, A. G., MULVENNA, M. D., 1999, **"Discovering Internet Marketing Intelligence through Online Analytical Web Usage Mining"**, SIGMOD Record, 27(4), pp. 54-61.

[18] CADEZ. I, HECKERMAN, D., MEEK, C., SMYTH, P., WHITE, S., 2000, **"Visualization of Navigation Patterns on a Web Site Using Model Based Clustering"**, Technical Report MSR-TR-00-18, Microsoft Research.

[19] CASTRO, J., KOLP, M., MYLOPOULOS, J., 2001, **"A Requirement-Driven Software Development Methodology"**, In: Proceedings of the 13th

International Conference on Advanced Information Systems Engineering (CAISE 2001), Interlaken, Switzerland, Springer-Verlag, pp. 108-123.

[20] CATLEDGE, L. D., PITKOW, J. E., 1995, **"Characterizing Browsing Strategies in the World Wide Web"**, Computer Networks and ISDN Systems 27(6), Elsevier Science, pp.1065-1073, 1995.

[21] CHAKRABARTI, S., 2000, **"Data mining for hypertext"**, ACM SIGKDD Explorations, vol. 1, no. 2, pp. 1-11.

[22] CHAKRABARTI, S., DOM, B., GIBSON, D., et al., 1999, **"Mining the Link Structure of the World Wide Web",** Computer Journal, Vol. 32, nº 8, pp. 60-67.

[23] CHAN, P. K., 1999, **"A non-invasive learning approach to building Web user profiles"**, In: Proceedings of 5th ACM SIGKDD International Conference, Workshop on Web Usage Analysis and User Profiling, Springer, pp. 7-12.

[24] CHANDRASEKARAN, B., JOSEPHSON, J. R., BENJAMINS, V. R., 1999, **"What are ontologies, and why do we need them?"**, IEEE Intelligent Systems, v.14 n.1, pp .20-26, January 1999.

[25] CHEN, M., PARK, J.S., YU, P.S., 1998, **"Efficient data mining for path traversal patterns"**, IEEE Transactions on Knowledge and Data Engineering, v.10, n. 2 (Mar), pp. 209-221.

[26] CHEN, M. S., PARK, J. S., YU, P. S., 1996, **"Data Mining for Path Traversal Patterns in a Web Environment"**, In: Proceedings of the 16th International Conference on Distributed Computing Systems, pp. 385-392.

[27] COOLEY, R., 2003, **"The use of web structure and content to identify subjectively interesting web usage patterns"**, ACM Trans. Inter. Tech., Vol.3, nº 2, pp. 93-116, ACM Press.

[28] COOLEY, R. W., 2000, **"Web usage mining: Discovery and application of Interesting Patterns from Web data"**, PhD thesis, Dept. of Computer Science, University of Minnesota.

[29] COOLEY, R., MOBASHER, B., SRIVASTAVA, J., 1999, **"Data preparation for**

mining world wide web browsing patterns", Knowledge and Information Systems.

[30] COOLEY, R., TAN, P. N., SRIVASTAVA, J., 1999, **"WebSIFT: The Web Site Information Filter System"**, In: Proceedings of the Web Usage Analysis and User Profiling Workshop (WEBKDD'99).

[31] COOLEY, R., MOBASHER, B., SRIVASTAVA, J., 1997, **"Web mining: information and pattern Discovery on the World Wide Web"**, Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence, 1997.

[32] COSSENTINO, M., BURRAFATO, P., LOMBARDO, S., SABATUCCI, L., 2002, **"Introducing Pattern Reuse in the Design of Multi-agent Systems"**, In Proceedings of the Agent Technologies, Infrastructures, Tools, and Applications for E-Services (NODe 2002), Agent-Related Workshops, Efurt, Germany, Springer-Verlag, pp. 107-120.

[33] COSTA, E. B., Lopes, M. A., FERNEDA, E., 1995, "**Mathema: A Learning Environment Based on a Multi-Agent Architecture**", In: J Wainer and A. Carvalho, editors, Proc. of 12th Brazilian Symposium on Artificial Inteligence, volume 991 of Lecture Notes in Artificial Intelligence, pp. 141-150. Spring-Verlag, Campinas, Brazil, October 1995.

[34] CUNHA, C. A., BESTAVROS, A., CROVELLA, M. E., 1995, **"Characteristics of WWW Client-based Traces"**, Technical Report TR-95-010, Boston University, Department of Computer Science.

[35] CUTTING, D.D., KARGER, J., PEDERSON, J., et al., 1992, **"A cluster based approach to browsing large document collections"**, Proceedings of the Fifteenth International Conference on Research and Development in Information Retrieval.

[36] DAI, H., MOBASHER, B., 2002, **"Using ontologies to discover domain-level Web usage profiles"**, In: Proceedings of the Second Semantic Web Mining Workshop at PKDD 2001, Helsinki, Finland, August 20, 2002.

[37] DUDA, R., HART, P., 1973, **"Pattern Classification and scene analysis"**, Journal of Documentation, New York: Wiley, 35, pp. 285-295.

[38] ESTIVILL-CASTRO, V., 2002, **"Why so many clustering algorithms - A Position Paper"**, SIGKDD Explorations, 4(1), 65-75.

[39] ETZIONE, O., 1996, **"The World Wide Web Quagmire or gold mine"**, Communications of the ACM, vol.39, no.11, pp. 65-68.

[40] FARIA, C., GIRARDI, R., 2003, **"GRAMO: Uma Técnica para a Construção de Modelos de Domínio Reutilizáveis no Desenvolvimento de Sistemas Multiagente"**, Anais do XII Seminário de Computação (SEMINCO 2003), Centro de Convenções, Willy Sievert, PROEB, Blumenau, Santa Catarina, Brasil, pp. 71-84, Ed. FURB. 05 a 08 de agosto de 2003.

[41] FARIA, C., OLIVEIRA, I., GIRARDI, R., 2003, **"Uma Ontologia Genérica para a Análise de Domínio e Usuário na Engenharia de Domínio Multiagente"**, Anais do Simpósio de Informática da Região Centro/RS (SIRC/RS 2003), Santa Maria, Rio Grande do Sul, Brasil, Ed. SIRC/RS 20 a 22 de agosto de 2003.

[42] FARIA, C., 2004, **"Uma Técnica para a Aquisição e Construção de Modelos de Domínio e Modelos de Usuários Baseados em Ontologias para a Engenharia de Domínio Multiagente"**, Dissertação de mestrado (Mestrado em Engenharia de Eletricidade) – Área: Ciência da Computação, Universidade Federal do Maranhão – UFMA, 2004.

[43] FARIA, C., 2004, **"Ferramentas para a Engenharia de Domínio e de Aplicações no desenvolvimento baseado em Agentes"**, Trabalho de pesquisa, UFMA/DTI-7G-CNPq.

[44] FERREIRA, Steferson, 2003, **"Uma Técnica para o Projeto de Domínio de Sistemas Multiagente"**, Anais do II Workshop de Informática do IMESA, Assis, São Paulo, Brasil, 13 a 17 de outubro de 2003

[45] FERREIRA, Steferson, 2004, **"Uma Ferramenta e Técnica para o Projeto Global e Detalhado de Sistemas Multiagente"**, Dissertação de mestrado (Mestrado em Ciência da Computação) – Área: Ciência da Computação, Universidade Federal do Maranhão – UFMA, 2004.

[46] FIELDING, R., GETTYS, J., MOGUL, J., MASINTER, L., LEACH, P., et al., 1999, RFC 2616 - Hypertext Transfer Protocol - HTTP/1.1.

[47] FIPA, 2004, Disponível em: http://www.fipa.org, acesso em 20/04/2004.

[48] FIPA Request Interaction Protocol Specification, 2004, Disponível em: http://www.fipa.org/specs/fipa00026/SC00026H.html, acesso em 22/01/2005.

[49] FISHER, D., 1987, **"Knowledge acquisition via incremental conceptual clustering"**, Machine Learning, 2, pp.139-172.

[50] FLORES, C., D., SEIXAS, L., GLUZ., J., et al., 2004, "**AMPLIA Learning Environment Architecture**", In: Workshop on Architectures and Methodologies for Building Agent-based Learning Environments, 2004, São Luis. Simpósio Brasileiro de Inteligência Artificial - SBIA2004, São Luis: UFMA, 2004.

[51] FOURO, A. M. M., WERNER, C. M. L., 2001, **"Modelos de Domínio ou Ontologias?"**, RTInfo – Revista da Tecnologia da Informação, 2001.

[52] FRAWLEY, W. J., PIATETSKY-SHAPIRO, G., MATHEUS, C. J., 1991, **"Knowledge discovery in databases: an overview"**. In: G. Piatetsky-Shapiro & W.J. Frawley, editors, "*Knowledge Discovery in Databases"*, AAAI / MIT Press.

[53] FREITAG, D., 1998, **"Information Extraction from HTML: Application of a General Machine Learning Approach"**, American association for Artificial Intelligence.

[54] FU, Y., SANDHU, K., SHIH, M. Y., 1999, **"Clustering of Web Users Based on Access Patterns"**, In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Springer: San Diego.

[55] GESEC, 2004, Grupo de pesquisa em Engenharia de Software e Engenharia do Conhecimento, disponível em: http://maae.deinf.ufma.br, acesso em Dezembro de 2004.

[56] GHANI, R., JONES, R., MLADENIC, D., et al., 2000, **"Data mining on symbolic knowledge extracted from the Web"**, In Proceedings of the Sixth International Conference on knowledge Discovery and Data mining (KDD-2000).

[57] GIRARDI, R., MARINHO, L. B., **"A System of Agent-based Patterns for User Modeling based on Usage Mining",** Interacting with Computers Journal, a ser

publicado.

[58] GIRARDI, R., FARIA, C., MARINHO, L., 2004, **"Ontology-based Domain Modeling of Multi-Agent Systems"**, Proceedings of Third International Workshop on Agent-Oriented Methodologies at International Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2004). 24 a 28 de outubro de 2004.

[59] GIRARDI, R., 2004, **"Engenharia de Software baseada em Agentes"**, Anais do IV Congresso Brasileiro de Ciência da Computação (CBCOMP 2004), Ed. UNIVALI, pp. 913-937, Itajai, Santa Catarina, Brasil. 08 a 12 de outubro de 2004.

[60] GIRARDI, R., SERRA, I., 2004, ''**Using Ontologies for the Specification of Domain-Specific Languages in Multi-Agent Domain Engineering"**, Proceedings of the Sixth International Bi-Conference Workshop on Agent-oriented Information Systems (AOIS-2004) at The 16th International Conference on Advanced Information Systems Engineering (CAISE'04), Ed. Janis Grundspenkis and Marite Kirikova (Eds.), pp. 295-308. Riga, Latvia. 07 a 11 de junho de 2004.

[61] GIRARDI, R., FARIA, 2004, **"An Ontology-Based Technique for the Specification of Domain and User Models in Multi-Agent Domain Engineering",** CLEI Electronic Journal, V. 7, N. 1, Pap. 7. Junho de 2004.

[62] GIRARDI, R., 2003, **"Engenharia de Domínio Multiagente"**, Anais do 3 Ibero-Americano Symposium on Software Engineering and Knowledge Engineering (JIISIC 2003), Scientific Cooperation, Universidad Austral de Chile.

[63] GIRARDI, R., FARIA, C., 2002, **"A Generic Ontology for the Specification of Domain Models"**, Proceedings of 1st International Workshop on Component Engineering Methodology (WCEM 2003) at Second International Conference on Generative Programming and Component Engineering, pp. 41-50, Ed. Sven Overhage and Klaus Turowski. Efurt, Germany, September 2003.

[64] GIRARDI, R., 2002, **"Reuse in Agent-based Application Development"**, In: Proceedings of 1º International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'2002), May 2002.

[65] GIRARDI, R., SODRÉ, A. C. S., 2002, **"A Methodology for Multi-Agent Application Development, In 6th International Conference on Intelligent**

Tutoring Systems", Proceedings of the ITS´2002 Workshops - Architectures and Methodologies for Building Agent-Based Learning Environments, Biarritz, v. 1, pp. 58-66.

[66] GIRARDI, R., 2001, **"Software Abstractions in Agent-Based Application Engineering"**, In: Joint Meeting of the 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001), Orlando, Florida.

[67] GIRARDI, R., 1998, **"Main Approaches to Software Classification and Retrieval"**, En las actas del curso Ingeniería del Software y reutilización: Aspectos Dinámicos y Generación Automática. Editores J. L. Barros y A. Domínguez. (Universidad de Vigo - Ourense, del 6 al 10 de julio de 1998). Julio, 1998.

[68] GIRARDI, R., IBRAHIM, B., 1993, **"New Approaches for Reuse Systems"**, In: Position Paper Collection of the 2nd. International Workshop on Software Reuse (IWSR-2).

[69] GUARINO, N., 1998, **"Formal Ontology and Information Systems"**, Proceedings of FOIS'98, Trento, Italy. Amsterdan, IOS Press, pp. 3-15, 6-8 June, 1998.

[70] HAN, E. H., KARYPIS, G., KUMAR, V., et al., **"Clustering based on association rule hyper graphs"**, In : Proceedings of SIGMOD'97Workshop on Research issues in Data Mining and Knowledge Discovery, 9-13, 1997.

[71] HAN, J., KAMBER, M., 2001, **"Data Mining: Concepts and Techniques"**, Morgan Kaufmann Publishers.

[72] HAN, J., 2000, **"OLAP Mining: An integration of OLAP with Data Mining"**, School of Computing Science, Simon Fraser University, British Columbia, Canada.

[73] HAN, J., CAI, Y., CERCONE, N., 1002, **"Knowledge discovery in databases: An attribute oriented approach"**, In : Proceedings of 18th International Conference on Very Large Databases, Vancouver, Canada, pp. 547-559, 1992.

[74] HANSON, R., STUTZ, J., CHEESEMAN, P., 1991, **"Bayesian classification theory"**, TR-FIA-90-12-7-01, AI Branch, NASA Ames Research Center, CA.

[75] HARSU M., 2002, **"A Survey on Domain Engineering"**. Report 31, Institute of Software Systems, Tampere University of Technology, December 2002, 26 pp.

[76] HARTIGAN, J., 1975, **"Clustering Algorithms"**, John Wiley.

[77] HIPP, J., GUNTZER, U., NAKHAEIZADEH, G., 2000, **"Algorithms for Association Rule Mining"**, A General Survey and Comparison, SIGKDD Explorations, 2(1), 58-64.

[78] KAZIENKO, P., KIEWRA, M., 2003, **"ROSA - Multi-agent System for Web Services Personalization"**, In Proc. of the First International Atlantic Web Intelligence Conference, AWIC 2003, Madrid, Spain, May 5-6.

[79] KDnuggets, 2004, disponível em: http://www.kdnuggets.com/, acesso em 03/12/2004.

[80] KLEINBERG, J.M., 1998, **"Authoritative Sources in a Hyper-linked Environment"**, In: Proceedings of ACM-SIAM Symposium on Discrete Algorithms.

[81] IBM WebSphere Personalization Server, Disponível em: http://www-306.ibm.com/software/genservers/personalization/, acesso em: 31/11/2004.

[82] JADE, disponível em: http://jade.tilab.com/, acesso em: 23/01/2005.

[83] JENA, disponível em: http://jena.sourceforge.net/, acesso em: 24/01/2005.

[84] JOHNSON, W.L., SHAW, E., **"Using Agents to Overcome Difficulties in Web-Based Courseware"**, In: AI-ED'97 Workshop on Intelligent Educational Systems on the World Wide Web. Retrieved September 23, 2003, from http://www.isi.edu/isd/ADE/papers/ aied97/aied97-www.htm.

[85] JORDING, T., 1999, **"A Temporary User Modeling Approach for Adaptive Shopping on the Web"**, In: Proceedings of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW, UM'99, Banff, Canada, 75-79.

[86] JOSHI, A., JOSHI, K., 2000, **"On Mining Web Access Logs"**, In: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge

Discovery, pp. 63-69.

[87] KAMDAR, T., JOSHI, A., 2000, **"On Creating Adaptive Web Sites using Web Log Mining"**, Technical Report TR-CS-00-05. Department of Computer Science and Electrical Engineering University of Maryland, Baltimore County.

[88] KAZIENKO, P. KIEWRA, M., 2003, **"ROSA - Multi-agent System for Web Services Personalization"**, In: Proceedings of the First International Atlantic Web Intelligence Conference, AWIC 2003, Madrid, Spain, May 5-6, 2003, pp. 297-306.

[89] KOBSA, A., 1999, **"Personalized Hypermedia presentation techniques for improving online customer relationships"**, GMD Report 66.

[90] KOBSA, A., 1994, **"User Modeling and User-Adapted Interaction"**, Conference Companion on Human Factors in Computing Systems, Boston, Massachusetts, United States, pp. 415-416, April 24-28.

[91] KOHONEN, T., 1997, **"Self-organizing Maps (second edition)"**, Springer Verlag: Berlin.

[92] KOSALA, R., BLOCKEEL, H**.,** 2000**, "Web Mining Research: a Survey",** SIG KDD Explorations, Vol.2, pp. 1-15.

[93] KUMAR, R., RAGHAVAN, P., RAJAGOPALAN, S., et al., 2002, **"The Web and Social Networks"**, IEEE Computer, vol.35, no.11, pp.32-36.

[94] KUMAR, S.R., RAGHAVAN, P., RAJAGOPALAN, S., et al., 1999, **"Trawling the web for emerging cyber communities"**, In: Proceedings of the Eighth WWW Conference.

[95] KUSHMERICK, N., 1997, **"Wrapper Induction for Information Extraction"**, Doctoral thesis. University of Washington, Department of Computer Science and Engineering.

[96] LANG,K., 1995, **"NEWSWEEDER: Learning to filter news"**, In: Proceedings of the 12th International Conference on Machine Learning, Lake Tahoe, CA: Morgan Kaufmann, pp. 331-339.

[97] LEE, T. B., HENDLER, J., LASSILA, O., 2001, **"The Semantic Web"**, Scientific American, May 2001.

[98] LEVY, A., WELD, D., 2000, **"Intelligent Internet Systems",** Artificial Intelligence, Vol.118, nº 1-2.

[99] LORENZ, A., DEZINGER, J., 2003, "**Functional Agent Systems for User Modeling**", Workshop on Artificial Intelligence, In Proceedings of 18th International Joint Conference On Artificial Intelligence, Acapulco, Mexico, August 11, 2003.

[100] MAGNAN, M. A. S., MURTA, L. G. P., SOUZA, J. M., et al., 2001, **"Modelos de domínio e Ontologias: uma comparação através de um estudo de caso prático em hidrologia"**, IV International Symposium on Knowledge Management/Document Management (ISKM/DM'2001). Curitiba, Agosto, 2001.

[101] MANBER, U., PATEL, A., ROBISON, J., 2000, **"Experience with Personalization on Yahoo"**, Communications of the ACM, 43(8), pp.35-39.

[102] MANNILA, H., TOIVONEN, H., 1996, **"Discovering Generalized Episodes Using Minimal Occurrences"**, In: Proceedings of 2nd ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (KDD96), pp. 146-151, Portland, OR, USA, Aug. 1996.

[103] MARINHO, Leandro B., GIRARDI, R., 2003, "**Mineração na Web"**, Revista Eletrônica de Iniciação Científica da SBC, Vol. 3, nº 2, Junho 2003.

[104] MARINHO, L. B., GIRARDI, R., RIBEIRO, I., et al., 2004, **"A Development Experience on Agent-based User Modeling"**, Proceedings of the Workshop on Architectures and Methodologies for Building Agent-Based Learning Environments, held in conjunction with SBIA, SBRN, IEEE MLSP, Ed. Livro Rápido, pp. 17-28. São Luís, Maranhão, Brasil. September 28th, 2004.

[105] MAHESWARI, U., SIROMONEY, V. A., MEHATA, K. M., 2001, **"The Variable Precision Rough Set Model for Web Usage Mining"**, In : Proceedings of the First Asia-Pacific Conference on Web Intelligence (WI'2001), Maebashi City, Japan, Oct 2001, Lecture Notes in Computer Science, 2198, pp. 520-524, Springer Verlag.

[106] MLADENIC, M., GLOBELNIM M., 1998, **"Efficient text categorization"**, In: Proceedings of Text Mining Workshop on the 10th European Conference on Machine Learning.

[107] MOBASHER, B., DAI, H., LUO, T., et al., 2000, **"Integrating Web usage and content mining for more effective personalization"**, In: Proceedings of the International Conference on E-Commerce and Web Technologies (ECWeb), LNCS - Lecture Notes in Computer Science, v. 1875, Springer-Verlag, pp. 165-176, London, Sep. 2000.

[108] MOBASHER, B., COOLEY, R., SRIVASTAVA, J., 1999, **"Creating adaptive Web sites through usage based clustering of URLs"**, In: Proceedings of the 1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99), 143-153.

[109] MOBASHER, B., COOLEY, R., SRIVASTAVE, J., 1999, **"Automatic personalization based on Web usage mining"**, Technical Report TR99010, Department of Computer Science, DePaul University.

[110] MOBASHER, B., JAIN, N., HAN, E.H., et al., 1997, **"Web Mining: Patterns from WWW transactions"**, Tech. Rep. TR96-050, Dept. of Computer Science, University of Minnesota.

[111] MOBASHER, B., JAIN, N., HAN, E., et. al., **"Web Mining: Pattern Discovery, from World Wide Web Transactions"**, TR-96050, Department of Computer Science. DePaul University, 1996.

[112] MULVENNA, M. D., BUCHNER, A. G., 1997, **"Data mining and electronic commerce, Overcoming"**, Barriers to Electronic Commerce, (OBEC '97), Malaga, Spain, 1-7.

[113] NGU, D. S. W., WU, X., 1997, **"SiteHelper: A localized agent that helps incremental exploration of the World Wide Web"**, Computer Networks and ISDN Systems: The International Journal of Computer and Telecommunications Networking, 29(8), 1249-1255.

[114] OBERLE, D., BERENDT, B., HOTHO, A., et al., 2003, **"Conceptual User Tracking"**, In the Proceedings of the First International Atlantic Web Intelligence Conference (AWIC 2003), Madrid, Spain.

[115] ODELL, J., PARUNAK, H. V. D., BAUER, B., 2000, **"Extending UML for Agents"**, In Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence

[116] OLIVEIRA DE ALMEIDA, H., 2004, **"COMPOR: Desenvolvimento de Software para Sistemas Multiagente"**, Dissertação (Mestrado em Ciência da Computação), Departamento de Ciência da Computação, Universidade Federal de Campina Grande – UFCG.

[117] PAL, S. K., TALWAR, V., MITRA, P., 2000, **"Web Mining in Soft Computing Framework: Relevant, State of the Art and Future Directions"**.

[118] PALIOURAS, G., PAPATHEODOROU, C., KARKALETSIS, V. et al., 2000, **"Clustering the Users of Large Web Sites into Communities"**, In: Proceedings of International Conference on Machine Learning (ICML), Stanford, California, pp. 719-726.

[119] PARNAS, D., 1976, **"On the design and development of program families"**, IEEE Transactions on Software Engineering, SE-2(1), pp.1-9.

[120] PERKOWITZ, M., ETZIONI, O., 1998, **"Adaptive sites: Automatically synthesizing Web pages"**, In: Proceedings of the 15th National Conference on Artificial Intelligence. Madison, Wisconsin, pp. 727-732.

[121] PERKOWITZ, M., ETZIONI, O., 2000, **"Adaptive Web Sites"**, Communications of the ACM, 43(8),152-158.

[122] PIERRAKOS, D., PALIOURAS, G., PAPATHEODOROU, C., SPYROPOLOUS, C. D., **"Web Usage Mining as a Tool for Personalization: A Survey"**, User Modeling and User-Adapted Interaction 13: 311-372, 2003.

[123] PITKOW, J., PIROLLI, P., 1999, **"Mining longest repeating subsequences to predict WWW surfing"**, In: Proceedings of the 1999 USENIX User Annual Technical Conference, pp. 139 – 150.

[124] PITKOW, J., BHARAT, K., 1994, **"WEBVIZ: A Tool for World-Wide Web Access Log Visualization"**, In: Proceedings of the 1st International World-Wide Web Conference. Geneva, Switzerland, pp. 271-277.

[125] PROTEGÉ PROJECT (2004), disponível em: http://protege.stanford.edu, acessado em Novembro de 2004.

[126] RIBEIRO, I., 2004, **"Um Sistema de Padrões baseados em Agentes para a Modelagem de Usuários e Adaptação de Sistemas"**, Dissertação de mestrado (Mestrado em Engenharia de Eletricidade) – Área: Ciência da Computação, Universidade Federal do Maranhão – UFMA, 2004.

[127] RIBEIRO, I., GIRARDI, R., 2003, **"Padrões Arquiteturais e de Projeto para a Modelagem de Usuários baseada em Agentes"**, In: Proceedings of The Third Latin American Conference on Pattern Languages of Programming - SugarLoafPlop, Porto de Galinhas.

[128] RUSSELL, S, NORVIG, P., 1995, **"Artificial Intelligence: A Modern Approach"**, Prentice-Hall.

[129] SARUKKAI, R.R., 2000, **"Link Prediction and Path Analysis Using Markov Chains"**, In: Proceedings of the 9th World Wide Web Conference. Amsterdam.

[130] SERRA JR, G. C., "**Agente de Modelagem do Aprendiz para o sistema MATHNET de Ensino Inteligente Cooperativo Computadorizado**", Dissertação (Mestrado em Ciência da Computação) – Curso de Pós-Graduação em Engenharia de Eletricidade, Universidade Federal do Maranhão, São Luís-MA, 2001.

[131] SHAHABI, C., BANAEI-KASHANI, F., 2003, **"Efficient and Anonymous Web Usage Mining for Web Personalization"**, INFORMS Journal on Computing - Special Issue on Data Mining, Vol.15, No.2, Spring.

[132] SHAHABI, C., BANAEI-KASHANI, F., FARUQUE, J., 2001, **"A Reliable, Efficient, and Scalable System for Web Usage Data Acquisition"**, In: WebKDD'01Workshop in conjunction with the ACMSIGKDD 2001, San Francisco, CA, August 2001.

[133] SHAHABI, C., ZARKESH, A.M., ADIBI, J., SHAH, V., 1997, **"Knowledge discovery from users web page navigation"**, Proceedings of the IEEE RIDE97 Workshop, Birmingham, England. 20-31.

[134] SHEN, W. M., 1996, **"An Efficient Algorithm for Incremental Learning of**

Decision Lists", Technical Report, USC-ISI-96-012, Information Sciences Institute, University of Southern California.

[135] SCHWARZKOPF, E., 2001, **"An adaptive web site for the UM2001 conference"**, In: Proceedings of the UM2001 Workshop on Machine Learning for User Modeling, pp. 77 - 86.

[136] SILVA, G. B., 2003, **"Padrões Arquiteturais para o Desenvolvimento de Aplicações Multiagente"**, Dissertação (Mestrado em Ciência da Computação) – Curso de Pós-Graduação em Engenharia de Eletricidade, Universidade Federal do Maranhão.

[137] SPILIOPOULOU, M., 1999, **"Tutorial: Data Mining for the Web"**, PKDD'99, Prague, Czech Republic.

[138] SPILIOPOULOU, M., FAULSTICH, L. C., WILKLER, K., 1999, **"A data miner analyzing the navigational behavior of Web users"**, In: Proceedings of the Workshop on Machine Learning in User Modeling of the ACAI99,Chania, Greece, 54-64.

[139] SPILIOPOULOU, M., FAULSTICH, L. C., 1998, **"WUM: A Web Utilization Miner"**, In International Workshop on the Web and Databases, Valencia, Spain, March 1998.

[140] SPSS Clementine, disponível em: http://www.spss.com/clementine, acesso em 01/12/2004.

[141] SRIVASTAVA, J., COOLEY, R., DESHPANDE, M. et al., 2000, **"Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data"**, SIGKDD Explorations, 1(2), 12-23.

[142] SODERLAND, S., 1999, **"Learning Information Extraction Rules for Semi-structured and Free Text"**, Machine Learning 1-44. Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

[143] STEINBACH, M., KARYPIS, G., KUMAR, V., 2000, **"A comparison of document clustering techniques"**, In KDD Workshop on Text Mining, 2000.

[144] TAN, P. N., KUMAR, V., 2002, **"Discovery of Web Robot Sessions Based**

on their Navigational Patterns", Data Mining and Knowledge Discovery, 6(1), 9-35.

[145] WERNER, C. M. L., BRAGA, R. M. M., 2000, **"Desenvolvimento Baseado em Componentes"**, XIV Simpósio Brasileiro de Engenharia de Software, Mini-curso, João Pessoa, Outubro, 2000.

[146] WEXELBLAT, A., MAES, P., 1997, **"Footprints: History-rich Web browsing"**, In: Proceedings Conference Computer-Assisted Information Retrieval (RIAO), pp. 75-84.

[147] WOOD, M., DELOACH, S., 2001, **"An Overview of the Multiagent Systems Engineering Methodology"**, In: Agent-Oriented Software Engineering – Proceedings of the First International Workshop on Agent-Oriented Software Engineering, 10th June 2000, Limerick, Ireland. P Cicarini, M. Woodridge, editors. Lecture Notes in Computer Science, v. 1957, Springer Verlag, Berlin.

[148] WOOLDRIDGE, M., JENNINGS, N., KINNY, D., 2000, **"The Gaia Methodology for Agent-Oriented Analysis and Design"**, International Journal of Autonomous Agents and Multi-Agent Systems, v. 3.

[149] WU, K., YU, P. S., BALLMAN, A., 1998, **"Speed tracer: A Web usage mining and analysis tool"**, IBM Systems Journal, 37(1), 8 pp. 9-105.

[150] WU, X., 1993, **"The HCV Induction Algorithm"**, In: Proceedings of the 21st ACM Computer Science Conference, ACM Press, pp. 169-175.

[151] YAN, T. W., JACOBSEN, M., GARCIA-MOLINA H., et al., 1996, **"From user access patterns to dynamic hypertext linking"**, Fifth International World Wide Web Conference, Paris, 47 France. 1007-1118.

[152] ZHANG, T., RAMAKRISHNAN, R., LIVNY, M., 1996, **"BIRCH: an efficient data clustering method for very large databases"**, In: Proceedings ACM-SIGMOD International Conference in Management of Data, Montreal, Canada, pp. 103-114.

[153] ZHU, T., 2001, **"Using Markov Chains for Structural Link Prediction in Adaptive Web Sites"**, UM 2001, LNAI2109, pp. 298-300.

ANEXO I

As ontologias do JADE são definidas de acordo com a classificação definida pela FIPA-ACL, que divide os elementos do domínio em termos ou predicados.

Predicados são expressões que indicam um fato do mundo real (ex.: *Página Visitada: http://maae.deinf.ufma.br*). Os predicados são geralmente usados no conteúdo das mensagens do tipo INFORM.

Já os termos possuem uma subclassificação em *conceitos*, *ações*, *primitivas*, *variáveis* e *agregados.* Dentre estes, só o termo do tipo conceito foi utilizado no ONTOMUW. Conceitos são expressões que indicam entidades complexas que podem ser definidas através de estruturas baseadas em frames, como por exemplo, (*Pessoa : nome John : idade 33*).

As ontologias especificadas para o JADE assumem a forma de *beans* do Java. O editor de ontologias Protege [125] suporta um plug-in chamado *beangenerator* que automatiza o processo de criação de ontologias do JADE a partir de ontologias definidas no Protege. A vantagem é que o Protege oferece várias facilidades para a criação e edição gráfica de ontologias. Cada conceito ou predicado definido no Protege é então transformado em uma classe do Java pelo *beangenerator*. A Figura 60 mostra a ontologia dos agentes construída através do Protege e a Figura 61 mostra o *beangenerator* sendo utilizado para a construção da ontologia dos agentes do JADE. Abaixo é mostrado o código fonte de um dos predicados definidos na ontologia que foi transformado em uma classe do Java pelo *beangenerator*.

```
public class ModeloUsuarios implements Predicate {

  /**
  * Protege name: matrizTempo
  */
  private MatrizCaracteristicas matrizTempo;
  public void setMatrizTempo(MatrizCaracteristicas value) {
   this.matrizTempo=value;
  }
  public MatrizCaracteristicas getMatrizTempo() {
    return this.matrizTempo;
```

```
}
/**
* Protege name: matrizVisitas
*/
private MatrizCaracteristicas matrizVisitas;
public void setMatrizVisitas(MatrizCaracteristicas value) {
 this.matrizVisitas=value;
}
public MatrizCaracteristicas getMatrizVisitas() {
  return this.matrizVisitas;
}
/**
* Protege name: matrizSequencia
*/
private MatrizCaracteristicas matrizSequencia;
public void setMatrizSequencia(MatrizCaracteristicas value) {
 this.matrizSequencia=value;
}
public MatrizCaracteristicas getMatrizSequencia() {
  return this.matrizSequencia;
}
private SessaoUso sessaoUso;
public void setSessaoUso(SessaoUso value) {
        this.sessaoUso=value;
}
public SessaoUso getSessaoUso() {
         return this.sessaoUso;
         }
}
```

A partir do código fonte acima, percebe-se que um modelo de usuários na ONTOMUW é composto pelas três matrizes de características (*visitas*, *tempo* e *seqüência*) e pela sessão de navegação correspondente. Esse predicado será usado tanto na comunicação entre os agentes *Modelador* e *Aquisitor*, para o armazenamento do modelo ao término de uma sessão, quanto na comunicação entre os agentes *Modelador* e *Minerador*, para a classificação do usuário corrente.

Para cada *slot* definido na ontologia são criados atributos na classe Java correspondente, assim como métodos do tipo "*set*" e "*get"* para o acesso e manipulação desses atributos.

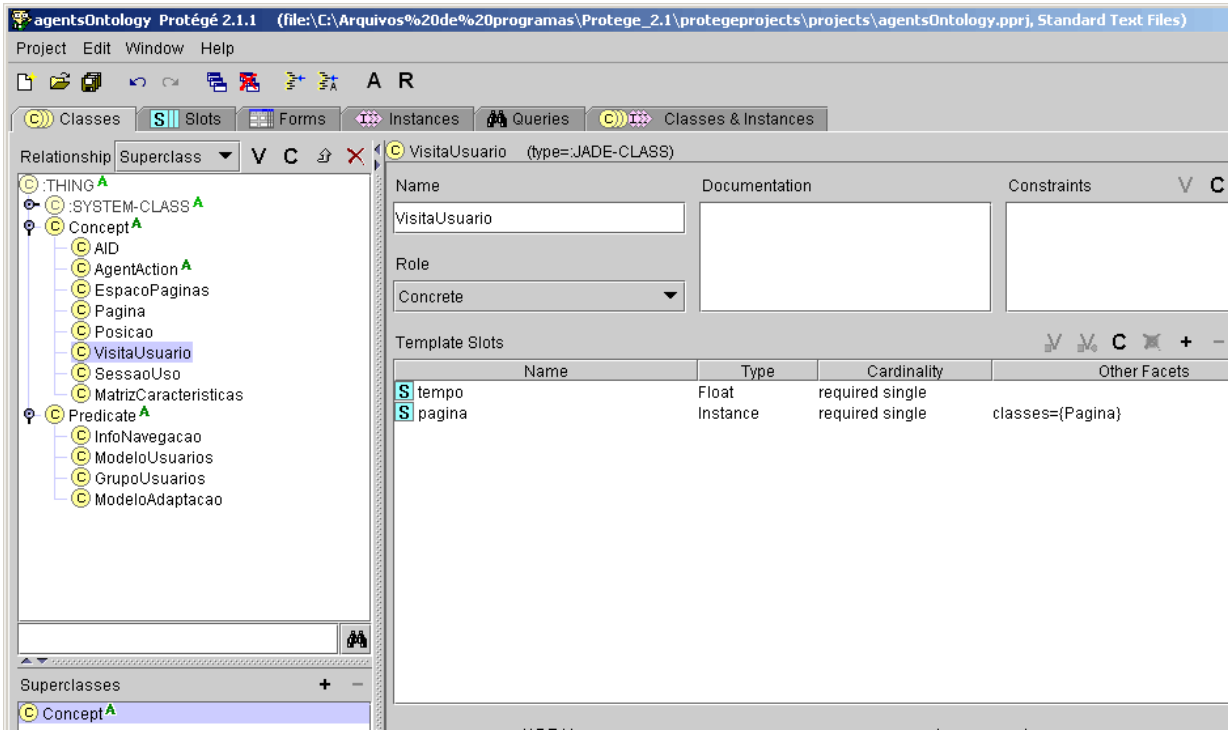

Figura 60. Ontologia dos agentes construída através do Protege

Alguns dos conceitos definidos na ontologia nunca chegam a ser utilizadas em conteúdos de mensagens de fato, mas servem como conceitos auxiliares para algum processamento específico. O conceito *Posição*, por exemplo, encapsula as informações sobre a posição de um segmento em uma matriz. Como vários agentes lidam com a manipulação de matrizes, é coerente que todos tenham um entendimento comum sobre esse conceito, mesmo que não troquem mensagens se referindo a ele.

Além das ontologias, foram construídas algumas classes auxiliares que automatizam algumas das tarefas a serem executadas pelos agentes. A Tabela 9 abaixo resume essas classes, descrevendo a sua funcionalidade básica e que agentes as utilizam.

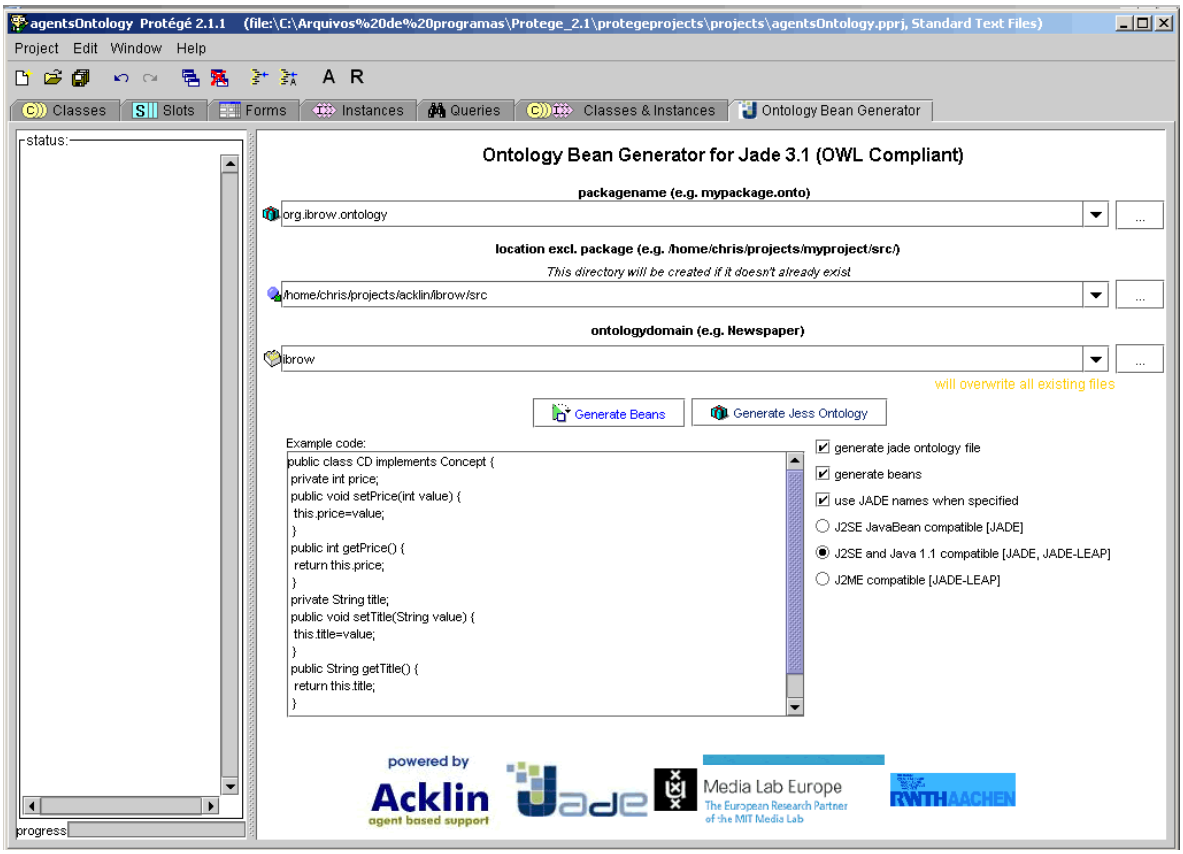

Figura 61. Plug-in beangenerator para a construção das ontologias dos agentes JADE

Classe	Função	Agente
MatrizSegemntos	Essa classe encapsula métodos para a criação da matriz de segmentos.	Modelador
ModeloCaracteristicas	Classe responsável pela construção das matrizes de características. Dentre os métodos mais importantes estão: construirMatrizVisitas (), construirMatrizTempo (), construirMatrizSequencia ()	Modelador
LogHandler	Encapsula métodos para a recuperação de dados no arquivo de uso. Para isso essa classe utiliza a API do Jena	Minerador
KMeans	Classe que implementa o algoritmo de agrupamento KMédias e o agrupamento dinâmico	Minerador

Tabela 9. Resumo das classes auxiliares do ONTOMUW

Abaixo se fez uma seleção de código de alguns dos métodos mais importantes dessas classes.

Classe: *MatrizSegmentos* Método: *construirMatrizSegmentos* ()

```
private Vector construirMatrizSegmentos(int ordemSegmento) {

        Vector matriz = new Vector();
        if(ordemSegmento == 1) {
                for(int i = 0; i < this.espacoPaginas.getNumeroPaginas(); i++) {
                        Vector vetor = new Vector();
                        SegmentoUso segmentoUso = new SegmentoUso();
                        VisitaUsuario visita = new VisitaUsuario();
                        visita.setPagina(this.espacoPaginas.getPagina(i));
                        visita.setTempo(0);
                        segmentoUso.addVisitas(visita);
                        vetor.add(segmentoUso);
                        matriz.add(vetor);
                }
        }else if(ordemSegmento == 2) {
                for(int i = 0; i < this.espacoPaginas.getNumeroPaginas(); i++) {
                        Vector vetor = new Vector();
                        for(int j = 0; j < this.espacoPaginas.getNumeroPaginas(); j++) {
                                SegmentoUso segmentoUso = new SegmentoUso();
                                VisitaUsuario visita1 = new VisitaUsuario();
                                VisitaUsuario visita2 = new VisitaUsuario();
                                visita1.setPagina(this.espacoPaginas.getPagina(i));
                                visita2.setPagina(this.espacoPaginas.getPagina(j));
                                visita1.setTempo(0);
                                visita2.setTempo(0);
                                segmentoUso.addVisitas(visita1);
                                segmentoUso.addVisitas(visita2);
                                vetor.add(segmentoUso);
                        }
                        matriz.add(vetor);
                }
        }else {
            for(int i = 0; i < this.espacoPaginas.getNumeroPaginas(); i++) {
                Vector matrizAuxiliar = construirMatrizSegmentos(ordemSegmento - 1);
                        Vector vetor = new Vector();
                        for(int j = 0; j < matrizAuxiliar.size(); j++) {
                          Vector vetorAuxiliar = (Vector) matrizAuxiliar.get(j);
                         for(int k = 0; k < vetorAuxiliar.size(); k++) {
                            SegmentoUso segmentoAuxiliar = (SegmentoUso)
                                                                vetorAuxiliar.get(k);
                            SegmentoUso segmentoUso = new SegmentoUso();
                            VisitaUsuario visita = new VisitaUsuario();
                            visita.setPagina(this.espacoPaginas.getPagina(i));
                            visita.setTempo(0);
                            segmentoUso.addVisitas(visita);
                        for(int l = 0; l < segmentoAuxiliar.getNumeroVisitas(); l++) {
                        VisitaUsuario visitaUsuario = segmentoAuxiliar.getVisita(l);
```

```
                                segmentoUso.addVisitas(visitaUsuario);
                                        }
                            vetor.add(segmentoUso);
                                    }
                                }
                                matriz.add(vetor);
                        }
                    }
                    return matriz;
            }
```

Classe: *ModeloCaracteristicas* Método: *construirMatrizVisitas ()*

```
        public ontology.MatrizCaracteristicas construirMatrizVisitas() {
            ontology.MatrizCaracteristicas matrizVisitas = new ontology.MatrizCaracteristicas();
            matrizVisitas.contruirMatriz(this.matrizSegmentos.obterLinhas(),this.matrizSegmentos.
obterColunas());
            for(int i = 0; i < (this.sessaoUso.getNumeroSegmentos()); i++) {
                SegmentoUso segmentoUso = this.sessaoUso.getSegmento(i);
                ontology.Posicao posicao =
this.matrizSegmentos.obterPosicaoSegmento(segmentoUso);
                matrizVisitas.incremetarValor(posicao);
                }
                return matrizVisitas;
        }
```

Classe: *LogHandler* Método: *recuperaModelosUsuarios ()*

```
        public List recuperaModelosUsuarios() {

                String modeloUsuarios_URI=uri+"ModeloUsuarios";
                String matrizCaracteristicas_URI=uri+"MatrizCaracteristicas";
                List listaModelos = new ArrayList();
                MatrizCaracteristicas matrizTempo = new MatrizCaracteristicas();
                String queryString = "SELECT ?x "+
                                "WHERE (?x, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
<"+uri+"ModeloUsuarios>)";
                Query query = new Query(queryString) ;
                query.setSource(m);
                QueryExecution qe = new QueryEngine(query) ;
                QueryResults results = qe.exec() ;
                Resource r2 = m.createResource();
                for ( Iterator iter = results ; iter.hasNext(); ) {
                   ModeloUsuarios userModel = new ModeloUsuarios();
                   ResultBinding res = (ResultBinding)iter.next();
```

```
                    Resource r = (Resource)res.get("x");
                    for ( StmtIterator sIter = r.listProperties(); sIter.hasNext(); )
                    {
                       Statement s = (Statement) sIter.next();
                       Resource  subject   = s.getSubject();     // get the subject
                       Property  predicate = s.getPredicate();   // get the predicate
                       RDFNode   object    = s.getObject();      // get the object
                       if (predicate.toString().equals(new String(uri+"mVisitas")))
                                userModel.setMatrizVisitas(montarMatrizVisitas((Resource) object));
                       else if (predicate.toString().equals(new String(uri+"mSequencia")))
                          userModel.setMatrizSequencia(montarMatrizSequencia((Resource) object));

                       else if (predicate.toString().equals(new String(uri+"mTempos")))
                          userModel.setMatrizTempo(montarMatrizTempos((Resource) object));
                    }
                    listaModelos.add(userModel);
                 }
              }

                       results.close();
                       return listaModelos;

        }
```

Classe: KMeans Método: runKMeans ()

```
        public void runKMeans() {

                // Seleciona k modelos randomicamente como grupos iniciais
                for (int i=0; i < k; i++){
                        int numModelo = (int)(Math.random() * this.modelos.size());
                        GrupoUsuarios grupo = new GrupoUsuarios(i);
                        grupo.setMatrizVisitas(((ModeloUsuarios)this.modelos.get(numModelo)).get
MatrizVisitas());
                        grupo.setMatrizSequencias(((ModeloUsuarios)this.modelos.get(numModelo))
.getMatrizSequencia());
                        grupo.setMatrizTempos(((ModeloUsuarios)this.modelos.get(numModelo)).ge
tMatrizTempo());

                        this.grupos.add(i,grupo);
                        System.out.println("Random: "+numModelo);
                }
                do {
                        // Forma os grupos
                        Iterator i = this.modelos.iterator();
```

```
                while (i.hasNext())
                        this.assignToCluster((ModeloUsuarios)(i.next()));
                this.nIterations++;

        }
        // Repete até que todos os modelos tenham sido classificados
        while (atualizaGrupos());

} // end of runKMeans()
```

Para ilustrar a utilização de duas dessas classes pelos agentes, é mostrado o comportamento CriaGrupos () do agente *Minerador*, que utiliza tanto a classe *LogHandler* quanto a classe *KMeans* para a criação dos grupos de usuários.

```
public class RemontaModelos extends TickerBehaviour{

        private Minerador agent;
        private String filename;
        private int numeroGrupos;
        private Codec codec = new SLCodec();
        private Ontology ont = OntowumOntology.getInstance();;
        public RemontaModelos(Minerador a, long duração, fileName, numeroGrupos) {
                super(a,duracao);
                this.agent=a;
                this.fileName=fileName;
                this.numeroGrupos=numeroGrupos
        }

        protected void onTick() {
                //Classe LogHandler para a recuperação dos modelos no arquivo de uso
                LogHandler fileHandler = new LogHandler(filename);
                List listaModelos = fileHandler.remontarModelosUsuarios();
                //Classe Kmeans sendo instanciada passando o numero de grupos a serem criados e a
        lista de modelos que foi recuperado do arquivo de uso
                agent.kMeans = new KMeans(numeroGrupos,listaModelos);
                agent.kMeans.runKMeans();
        }
```