UNIVERSIDADE FEDERAL DO MARANHÃO
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE
ÁREA: CIÊNCIA DA COMPUTAÇÃO

UM SISTEMA DE PADRÕES BASEADOS EM AGENTES PARA A MODELAGEM DE USUÁRIOS E ADAPTAÇÃO DE SISTEMAS

Ismênia Ribeiro de Oliveira

São Luís, MA
2004

UM SISTEMA DE PADRÕES BASEADOS EM AGENTES PARA A MODELAGEM DE USUÁRIOS E ADAPTAÇÃO DE SISTEMAS

Ismênia Ribeiro de Oliveira

Bacharel em Ciência da Computação
Universidade Federal do Maranhão, 2002

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 de Eletricidade na área de Ciência da Computação.

Orientadora: Profª. Drª. Rosario Girardi

São Luís, MA
2004

UM SISTEMA DE PADRÕES BASEADOS EM AGENTES PARA A MODELAGEM DE USUÁRIOS E ADAPTAÇÃO DE SISTEMAS

Ismênia Ribeiro de Oliveira

Dissertação aprovada em / /

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

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

Profª. Drª. Rosana Teresinha Vaccare Braga
Universidade de São Paulo

Aos meus pais.

AGRADECIMENTOS

Agradeço a todos que contribuíram direta ou indiretamente para elaboração desta dissertação, em especial:

Aos meus pais pelo o amor, incentivo, dedicação e apoio.

Às minhas queridas irmãs pelo apoio e incentivo.

Ao meu esposo, Edenilson, pelo apoio, dedicação e compreensão.

À Professora Rosario, pela orientação segura e presente, pelos ensinamentos e dedicação imprescindíveis para a realização deste trabalho.

Aos meus companheiros de curso, em especial, Carla, Ivo, Steferson, Alisson e Geovane, pela amizade no decorrer do curso.

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

RESUMO

Este trabalho propõe um sistema de padrões baseados em agentes para o projeto de sistemas multiagente adaptativos e adaptáveis que utilizam técnicas de modelagem de usuários.

O sistema de padrões é composto por um padrão conceitual, um padrão arquitetural e cinco padrões de projeto.

Os problemas, as soluções e os conceitos, comuns no projeto da modelagem de usuários, de sistemas adaptativos e no desenvolvimento de agentes de software, são analisados e descritos na forma de padrões.

A metodologia para a extração dos padrões é baseada na análise dos conceitos e técnicas para o desenvolvimento de sistemas adaptativos e adaptáveis, das técnicas de modelagem de usuário e dos conceitos e técnicas para o projeto de sistemas multiagente e agentes de software.

Uma avaliação preliminar, do sistema de padrões proposto, é feita por meio de um estudo de caso no domínio do acesso à informação. O estudo consiste em identificar e analisar a reutilização do sistema de padrões no desenvolvimento de aplicações multiagente para o acesso à informação personalizada.

Palavras - chave: Modelagem de usuários, Sistemas adaptativos, Sistemas multiagente, Padrões de software.

Fonte: Ribeiro, Ismênia. **Um Sistema de Padrões baseados em Agentes para a Modelagem de Usuários e Adaptação de Sistemas.** 2004. 123 p.. Dissertação (Mestrado em Engenharia de Eletricidade), Universidade Federal do Maranhão, São Luís

ABSTRACT

This work proposes an agent-based pattern system for design of adaptive multi-agent systems using user-modeling techniques.

The pattern system is composed of a conceptual, an architectural, and design patterns.

The patterns describe recurrent problems and design solutions for user modeling and development of adaptive multi-agent systems. The pattern mining is based on the analysis of: concepts and techniques for development of adaptive and adaptable systems; user modeling techniques; and multi-agent design techniques.

An initial validation of the pattern system has been conducted through the development of a case study on the domain of information access, where patterns in the system have been reused on the development of a multi-agent system for personalized information access.

Keywords: user modeling, adaptive systems, multiagent systems, software patterns

Source: Ribeiro, Ismênia. **An Agent-based Pattern System for User Modeling and Adaptation of Systems.** 2004. 123 p. Thesis (Graduation in Eletrical Engineering), Universidade Federal do Maranhão, São Luís**.**

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABARFI Arquitetura baseada em Agentes para a Recuperação e Filtragem de Informação

AIP Protocolos de Interações de Agentes (do inglês *Agent Interaction Protocols*)

AMS Sistema de Gerenciamento de Agente (do inglês *Agent Manager System*)

APSUMAS Sistema de Padrões Baseados em Agentes para a Modelagem de Usuários e Adaptação de Sistemas (do inglês *Agent-based Pattern System for User Modeling and Adaptation of Systems*)

AUML Linguagem de Modelagem Unificada para Agente (do Inglês *Agent Unified Modeling Language)*

DDEMAS Projeto de Domínio para Sistemas Multiagente (do inglês Domain *Design for Multi-Agent Systems)*

FIPA Fundação para Agentes Físicos Inteligentes (do inglês *Foundation for Intelligent Physical Agents*)

JADE Ambiente de Desenvolvimento de Agente Java (do inglês *Java Agent Development Environment*)

JESS Interface de comando do Sistema Especialista Java (do inglês *Java Expert System Shell*)

MaAE Engenharia de Aplicação Multiagente (do inglês *Multi-agent Application Engineering*)

ONTODD Ontologia Genérica de Projeto

ONTODM Ontologia Genérica para a Construção de Modelos de Domínio

ONTODUM Ontologia Genérica para a Análise de Domínio e Usuário na Engenharia de Domínio Multiagente

PLoP Conferência sobre Linguagens de Padrões em Programação (do inglês *Pattern Language of Programs*)

RETSINA Estrutura de Tarefas Reutilizável baseada em Agentes de Rede Inteligentes (do inglês *Reusable Task Structure-based Intelligent Network Agents*)

SMA Sistema Multiagente

UML Linguagem de Modelagem Unificada (do Inglês *Unified Modeling Language)*

1. INTRODUÇÃO

1.1 Motivação

A maioria dos sistemas de software não tem a habilidade para satisfazer as necessidades heterogêneas dos seus usuários. Cada usuário tem níveis de conhecimento, necessidades, habilidades e preferências bastante variadas. Nesse contexto, surge a necessidade de construir sistemas que se adaptem a cada tipo de usuário ou grupo de usuários com características comuns.

Por causa de suas vantagens para abordar a complexidade do software, a prática do desenvolvimento de software orientado a agentes está sendo bastante utilizada para o desenvolvimento de aplicações que se adaptam a diferentes tipos de usuários.

A reutilização de padrões no desenvolvimento de sistemas multiagente possibilita uma otimização no tempo e no custo do desenvolvimento e um aumento na qualidade do software. Porém, essa prática ainda está em estágio inicial se comparada com a reutilização de padrões orientados a objeto, devido principalmente à carência de padrões documentados para o desenvolvimento de sistemas multiagente. Isso leva os desenvolvedores a solucionar os mesmos problemas de projeto sem se beneficiar de experiências passadas adquiridas durante a resolução desses problemas, resultando em duplicação de esforços e inconsistências de projeto. Portanto, uma abordagem efetiva e formal é construir sistemas multiagente por meio da reutilização de padrões e sistemas de padrões.

Apesar do uso de padrões individuais ser útil para resolver problemas específicos de projeto, há um benefício superior na utilização de sistemas de padrões, pois cada padrão ocupa uma posição em uma rede de padrões relacionados, na qual cada padrão contribui para a realização dos padrões que o precedem na rede e é complementado pelos padrões que o sucedem [64].

Sistemas de padrões estão sendo criados para guiar os desenvolvedores na documentação e no desenvolvimento de projetos de sistemas. Em particular, há um crescimento no uso de padrões para captar práticas de projeto comuns em sistemas baseados em agentes [64].

A partir de uma análise sobre as diversas técnicas para modelagem de usuário e adaptação de sistemas, são propostas soluções recorrentes para problemas recorrentes para o projeto de sistemas adaptativos/adaptáveis, baseados em agentes, que utilizam técnicas de modelagem de usuários, na forma de um sistema de padrões. O sistema de padrões proposto capta práticas de projeto, comuns no desenvolvimento de sistemas adaptativos/adaptáveis.

1.2 Objetivo do trabalho

Este trabalho tem como objetivos principais especificar um sistema de padrões de projeto, baseados em agentes para sistemas adaptativos, que utilizam técnicas de modelagem de usuários e avaliá-los por meio do desenvolvimento de um estudo de caso que reutiliza os padrões propostos no projeto de desenvolvimento de aplicações multiagente para o acesso à informação personalizada.

1.3 Estrutura da Dissertação

Esta dissertação está estruturada em sete capítulos.

O capítulo 2 descreve os principais conceitos relacionados a padrões de software.

O capítulo 3 aborda, de maneira geral, os principais conceitos sobre o paradigma computacional de agentes e sobre o projeto de sistemas multiagente. Neste capítulo também são descritas e analisadas arquiteturas de agentes genéricos.

O capítulo 4 apresenta os principais conceitos e técnicas utilizadas no desenvolvimento de sistemas adaptativos e da modelagem de usuários, explorando a compreensão dos problemas e dos conceitos comuns que são relevantes para a extração de padrões neste domínio.

O capítulo 5 descreve o sistema de padrões orientados a agentes, para o projeto de sistemas adaptativos e da modelagem de usuários.

No capítulo 6 é apresentado um estudo de caso, na área do acesso à informação, desenvolvido para avaliar o sistema de padrões proposto.

No capítulo 7 são apresentadas as considerações finais do trabalho, destacando os resultados obtidos e os trabalhos futuros que poderão ser desenvolvidos a partir desta pesquisa.

2. PADRÕES DE SOFTWARE

Qualidade, baixo custo e rapidez no desenvolvimento, constituem um desafio para o desenvolvimento de software. Uma forma de tornar esses objetivos viáveis é aumentar a reutilização. Ao invés de começar a resolver um problema a partir do zero, o desenvolvedor adapta e melhora, ao seu problema, soluções que funcionaram no passado. Pode-se fazer a reutilização de uma classe, de um componente (estrutura de dados, módulo, unidade funcional), de um projeto, de uma interface, de uma idéia, de um código fonte, de uma documentação, de um padrão. A reutilização de código produz menores resultados de produtividade e confiabilidade que a reutilização de projetos, modelos, componentes, padrões e frameworks.

A reutilização de uma idéia, processo ou artefato se dá quando são utilizados em um contexto diferente para o qual foram originalmente projetados. Se o contexto e as condições se mantêm, simplesmente estaríamos utilizando-os de novo.

As técnicas de reuso de software têm contribuído de forma significativa para reduzir o tempo e o custo do desenvolvimento de software. Uma abordagem que está sendo bastante disseminada é o uso de padrões de software para promover reutilização.

Padrões de software são desejáveis para o desenvolvimento de software baseado na reutilização. Em particular, os padrões de software são desejáveis para o desenvolvimento de sistemas multiagente, pois eles constituem-se de soluções melhoradas para problemas recorrentes e podem ser aplicados durante o desenvolvimento de aplicações multiagente, em problemas conceituais, arquiteturais ou de projeto.

Um padrão apresenta 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 [27].

2.1 Categoria de Padrões

Os padrões abrangem diferentes tipos de abstração e podem ser classificados em diversas categorias [2] [12] [14]:

- **Padrão conceitual:** aquele cuja forma é descrita pelos principais termos e conceitos do domínio da aplicação.
- **Padrão arquitetural:** apresenta o esquema ou organização estrutural fundamental de sistemas de software ou hardware; ele provê um conjunto de subsistemas predefinidos, especifica suas responsabilidades e inclui regras e guias para organizar os relacionamentos entre eles.
- **Padrão de projeto:** refina os componentes de uma arquitetura de software e descreve soluções para problemas de projeto de software.
- **Padrão de programação:** descreve soluções de programação particulares em uma determinada linguagem ou regras gerais de estilo de programação.

Em particular, neste trabalho são abordados padrões arquiteturais e de projeto especializados para estabelecer soluções para a modelagem de usuário e adaptação de sistemas.

2.2 Coletâneas de Padrões

As diferentes categorias de padrões podem ser agrupadas em coletâneas que têm por objetivo agrupar os padrões segundo algum critério. As coletâneas podem ser divididas em:

- **Coleções de padrões:** é uma coletânea qualquer com padrões que não possuem nenhum vínculo entre si e em geral, não possuem nenhuma padronização no formato de apresentação [2].
- **Catálogos de padrões:** é uma coletânea de padrões relacionados, talvez relacionados apenas fracamente ou informalmente e pode oferecer um esquema de classificação e recuperação [2].

- **Sistemas de padrões:** é um conjunto coeso de padrões co-relacionados, que trabalham juntos para apoiar a construção e evolução de arquiteturas completas [2] [12].

- **Linguagem de padrões:** é uma coleção de padrões que trabalham juntos para resolver um problema complexo dentro de uma solução ordenada de acordo com uma meta pré-definida [2]. Coplien [14] define uma linguagem padrões como:

  "Uma linguagem de padrões define uma coleção de padrões e regras para combiná-los em um estilo arquitetural. A linguagem de padrões descreve frameworks ou famílias de sistemas relacionados".

Os padrões propostos, descritos no capítulo 5, estão agrupados em um sistema de padrões.

2.3 Atributos de um Padrão

Um padrão é descrito por vários atributos que podem variar de acordo com os critérios de descrição adotados. O sistema de padrões, apresentado no capítulo 5, é descrito pelos seguintes atributos:

- **Nome:** todo padrão deve ter um nome significativo. Pode ser uma única palavra ou frase curta que se refira ao padrão e ao conhecimento ou estrutura descritos por ele [2].

- **Contexto:** estabelece pré-condições dentro das quais o problema e sua solução costumam ocorrer e para as quais a solução é desejável, o que reflete a aplicabilidade do padrão [2].

- **Problema:** estabelece o problema a ser resolvido pelo padrão, descreve a intenção e objetivos do padrão perante o contexto e forças específicas [2].

- **Forças:** devem mostrar quais são os fatores que, frente ao contexto especificado e ao problema estabelecido, influenciam a solução proposta e como ela é implementada. Também pode apresentar

limitações e restrições da solução, desvantagens em utilizar o padrão ou o porquê da solução não utilizar outras estratégias [2].

- **Solução:** são instruções que descrevem como o problema é resolvido, podendo para isso utilizar texto, diagramas e figuras [2].

- **Padrões relacionados:** são relacionamentos estáticos e dinâmicos entre este padrão e os outros dentro da mesma linguagem de padrões ou sistema de padrões. Eles também têm um contexto inicial ou resultante, que é compatível com o contexto inicial ou resultante de outros padrões. Tais padrões podem ser padrões predecessores, cuja aplicação conduz a este padrão; padrões sucessores cuja aplicação segue deste padrão; padrões alternativos que descrevem uma solução diferente para o mesmo problema, mas sob diferentes forças e restrições; e padrões co-dependentes, que podem (ou devem) ser aplicados simultaneamente com este padrão [2].

- **Usos conhecidos:** são exemplos bem sucedidos do uso dos padrões em sistemas reais [2].

2.4 Técnica para a Identificação de Padrões

Uma das dificuldades encontradas na formação de um padrão é justamente a sua derivação, ou seja: “como um padrão é descoberto?”. Para responder essa pergunta, Buschmann [12] apresenta algumas regras de derivação de novos padrões, também conhecida como “extração de padrões" (*pattern mining*). As regras de derivação descritas por Buschmann são:

1. Encontre pelo menos três exemplos nos quais um problema é resolvido efetivamente usando a mesma solução;

2. Extraia a solução, o problema e as influências;

3. Declare a solução como candidata a padrão;

4. Execute um "writer's workshop" [1] para melhorar a descrição do candidato a padrão e para compartilhá-lo com outros;

5. Aplique o candidato a padrão em um outro projeto de desenvolvimento de software;

6. Declare o candidato a padrão como padrão se sua aplicação for bem sucedida. Caso contrário tente procurar uma solução melhor.

As regras de derivação descritas por Buschmann foram utilizadas como referência para a extração dos padrões que compõem o sistema APSUMAS, descrito no capítulo 5.

Os padrões que compõem o APSUMAS foram identificados da seguinte forma:

- Foi feita uma análise sobre os problemas existentes nos processos de modelagem de usuários e adaptação de sistemas em diversas áreas de aplicação, por meio de um levantamento dos principais conceitos e técnicas relacionados ao desenvolvimento de sistemas adaptativos e da modelagem de usuários; e uma análise sobre arquiteturas genéricas de agentes de software;

- Essas análises foram utilizadas para a extração dos atributos que descrevem cada padrão;

- Os padrões extraídos foram submetidos aos "writer's workshops" das conferências PLoP [2]: SugarLoafPLoP [52] e VikingPLoP [29];

---

[1] Writer's workshop é uma reunião de trabalho de autores de padrões, na qual cada autor analisa os aspectos positivos e negativos e sugere melhoras na descrição dos padrões dos outros autores.

[2] Essas Conferências ocorrem anualmente em diversos locais no mundo todo: a mais tradicional é a PLoP realizada nos Estados Unidos anualmente a partir de 1994, mas existem outras como a EuroPLoP (a partir de 1995), a ChiliPLoP (a partir de 1998) e a KoalaPLoP (a partir de 2000). No Brasil, já foram realizadas três edições da SugarLoafPLoP-Conferência Latino-Americana em Linguagens de Padrões para Programação.

- Após a extração dos padrões, os possíveis relacionamentos entre os mesmos foram identificados para que pudessem ser organizados em um sistema de padrões.

- Os padrões foram analisados através da reutilização dos mesmos no projeto de desenvolvimento de uma aplicação multiagente para o acesso a informação personalizada.

2.5 Desenvolvimento de Sistemas usando Padrões

O uso de padrões não veio criar um novo método ou critério de desenvolvimento de software, mas sim facilitar e complementar a análise, o projeto e a implementação do software. Alguns passos para o desenvolvimento de um sistema usando padrões de software são [12]:

1. Utilize seu método preferido para o processo de desenvolvimento de software em cada fase de desenvolvimento;

2. Utilize um sistema de padrões adequado para guiar o projeto e implementar soluções para problemas específicos, isto é, sempre que encontrar um padrão que resolva um problema de projeto presente no sistema, utilize os passos de implementação associados a esse padrão. Se esses se referirem a outros padrões, aplique-os recursivamente;

3. Se o sistema de padrões não incluir um padrão para seu problema de projeto, tente encontrar um padrão em outras fontes conhecidas;

4. Se nenhum padrão estiver disponível, aplique as diretrizes de análise e projeto do método que você está usando. Essas diretrizes fornecem pelo menos algum apoio útil para resolver o problema de projeto em mãos.

2.6 Padrões para o Desenvolvimento baseado em Agentes

Coletâneas de padrões estão sendo criadas para guiar os desenvolvedores na documentação e no desenvolvimento de projetos de sistemas.

Em particular, há um crescimento no uso de padrões para capturar práticas de projeto comuns em sistemas baseados em agentes [64].

O desenvolvimento de componentes reutilizáveis para o desenvolvimento de sistemas multiagente, tais como padrões de projeto, vem sendo objeto de várias pesquisas. Entre eles podemos citar:

- Weiss [65] descreve um sistema de padrões baseado em agentes para o comércio eletrônico.
- Aridor e Lange [3] descrevem um conjunto de padrões para o projeto de sistemas baseados em agentes móveis.
- Deugo, Weiss e Kendall [15] identificam um conjunto de padrões que trata da coordenação entre agentes.

No sistema de padrões APSUMAS, apresentado no capítulo 5, são descritos padrões orientados a agentes, específicos para o projeto de sistemas adaptativos/adaptáveis. Da mesma forma, o sistema de padrões proposto por Weiss [65], descreve padrões orientados a agentes, específicos para o domínio do comércio eletrônico. Porém, no APSUMAS, também são descritos padrões baseados em agentes que podem ser utilizados em aplicações multiagente em geral, como nos padrões propostos por Aridor [3] e por Deugo [15], cujos padrões podem ser aplicados em diversos domínios, como por exemplo, no domínio do comércio eletrônico, no domínio dos sistemas adaptativos/adaptáveis, entre outros.

2.7 Frameworks e Padrões de Projeto

Um framework não é nada mais que um conjunto de classes cooperantes que constroem uma arquitetura reutilizável para uma aplicação de software específica. Quando usamos um framework, reutilizamos o corpo principal e acrescentamos ou reescrevemos o código de suas classes. Qualquer mudança substancial no projeto de um framework reduzirá seus benefícios consideravelmente, uma vez que a principal contribuição de um framework para uma aplicação é a arquitetura que ele define. Portanto, é necessário projetar frameworks flexíveis e

extensíveis. Os padrões ajudam a tornar a arquitetura do framework adequada a muitas aplicações diferentes, sem a necessidade de reformulação [27]:

Podemos identificar as seguintes diferenças entre frameworks e padrões arquiteturais e de projeto [27]:

- Os frameworks modelam um domínio em particular. Os padrões arquiteturais e de projeto geralmente podem ser utilizados em vários domínios.

- Os padrões arquiteturais e de projeto podem ser usados tanto no projeto, como na documentação de um framework. Os padrões podem ser vistos como descrições abstratas de um framework que facilitam o reuso da arquitetura de software. Os frameworks são realizações concretas de padrões que facilitam o reuso de projeto e código fonte.

- Os padrões arquiteturais e de projeto representam micro-arquiteturas para a construção de projetos mais complexos. Os frameworks são macro-arquiteturas que geralmente contêm vários padrões.

- Um framework é um software executável enquanto que um padrão de projeto representa conhecimento e experiência a respeito de um software. Os frameworks são de natureza concreta e os padrões são de natureza abstrata.

- Os padrões arquiteturais e de projeto estão descritos independentemente de uma linguagem de programação, enquanto que os frameworks estão implementados em uma linguagem de programação em particular.

3. PROJETO DE SOFTWARE BASEADO EM AGENTES

Um sistema multiagente (SMA) pode ser caracterizado como um grupo de entidades (agentes) que atuam em conjunto no sentido de resolver problemas que estão além das suas capacidades individuais. Os agentes são entidades autônomas e tipicamente heterogêneas (caracterizadas por diferentes capacidades).

Quando comparados com sistemas orientados a objetos, os SMA apresentam algumas vantagens importantes, tais como [8]:

- Maior eficiência na resolução de problemas por meio da exploração de paralelismo;
- Maior flexibilidade na resolução de problemas, permitindo que agentes com diferentes capacidades cooperem dinamicamente;
- Maior robustez, pois em caso de falha de um ou vários agentes, outros agentes podem assumir as suas responsabilidades, de forma dinâmica.

Estas características tornam o desenvolvimento de software baseado em agentes bastante atrativo para a realização de sistemas de elevada complexidade, permitindo assim facilitar tanto a modelagem de sistemas, caracterizados por interações complexas e dinâmicas, como a definição de arquiteturas confiáveis e tolerantes a falhas.

A maioria do desenvolvimento de software baseado em agentes tem sido feito de maneira independente e intuitiva, isso tem produzido problemas como [35]:

- Falta de acordo sobre a definição de agentes;
- Duplicação de esforços, pois há pouca reutilização de arquiteturas, componentes e agentes;
- Dificuldade de identificar e especificar abstrações de software para o desenvolvimento de agentes;

- Falta de um vocabulário comum para o desenvolvimento de agentes;
- Incapacidade de satisfazer às exigências da indústria do software, pois os agentes devem integrar-se com o software e hardware existentes e também devem demonstrar segurança e escalabilidade.

Todos esses problemas limitam e dificultam a construção de software baseado em agentes. Com o intuito de facilitar e popularizar o desenvolvimento de tais softwares, vários padrões estão sendo propostos para serem usados nas diversas fases do seu processo de desenvolvimento. Para tal, estão sendo utilizados os vários conceitos e experiências já adquiridas no desenvolvimento de software baseado em agentes.

3.1 Conceitos sobre Agentes

Existem diversas definições para agentes. Cada autor tem sua própria definição e concepção do que é um agente. Geralmente, estas definições estão associadas a diferentes pontos de vista e dependem muito das características e funcionalidades apresentadas pelo agente em questão. Uma das definições da literatura é que “um agente é uma entidade computacional que está situado em algum ambiente e que é capaz de realizar ações autônomas neste ambiente visando atingir seus objetivos” [66].

Pode-se, ainda, definir um agente como sendo uma entidade que percebe seu ambiente por meio de sensores e atua neste ambiente por meio de executores (**Figura 1**). Por exemplo, nos agentes humanos, os olhos e os ouvidos são sensores, as mãos e a boca são executores. A **Figura 1** demonstra um agente genérico [54].

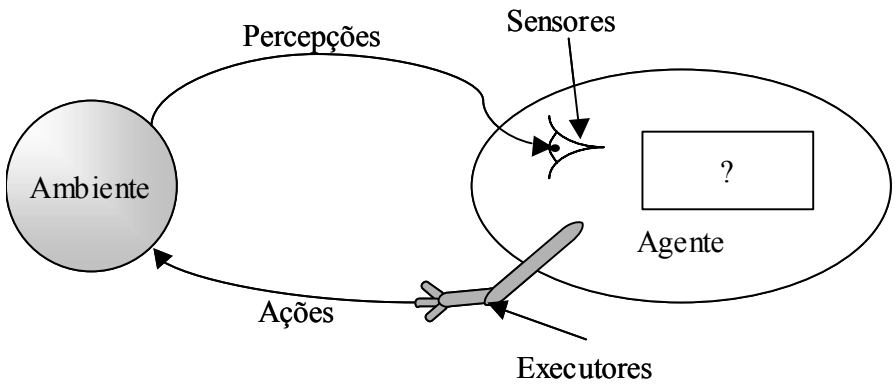

Figura 1 – Agente genérico [54]

Segundo Wooldridge e Jennings [66], a construção e a especificação da estrutura e funcionamento de um agente genérico pode ser realizada segundo três tipos de arquiteturas:

- **Arquitetura deliberativa.** Esta arquitetura segue a abordagem clássica da Inteligência Artificial, na qual os agentes contêm um modelo simbólico do mundo, explicitamente representado e cujas decisões (ações) são tomadas via raciocínio lógico, baseadas na combinação de padrões e manipulações simbólicas [54].

- **Arquitetura reativa.** A arquitetura reativa não inclui nenhum tipo de modelo central e simbólico do mundo e não utiliza raciocínio complexo e simbólico. Baseia-se na proposta de que um agente pode desenvolver inteligência a partir de interações com seu ambiente, não necessitando de um modelo pré-estabelecido[54].

- **Arquitetura híbrida.** Esta arquitetura mistura componentes das arquiteturas deliberativas e reativas com o objetivo de torná-la mais adequada e funcional para a construção de agentes. Ela propõe um subsistema deliberativo que planeja e toma decisões, da maneira proposta pela Inteligência Artificial Simbólica e um reativo capaz de reagir a eventos que ocorrem no ambiente sem se ocupar de raciocínios complexos [54].

Os padrões Deliberativo e Reativo, apresentados no capítulo 5, descrevem as estruturas de agentes organizados segundo as arquiteturas deliberativa e reativa, respectivamente.

Os agentes podem ser identificados por possuírem as seguintes características principais[66]:

- **Autonomia:** um agente pode funcionar sem intervenção humana e executar ações pela sua própria iniciativa, segundo o conhecimento que ele possui do seu ambiente e sua racionalidade;
- **Habilidade social:** um agente interage com outros agentes (humanos ou computacionais) por meio de protocolos de interação para atingirem seus objetivos, ou ainda para auxiliarem outros agentes;
- **Reatividade:** um agente deve ser capaz de perceber mudanças em seu ambiente e atuar de acordo com estas mudanças;
- **Pró-atividade:** os agentes não respondem apenas ao ambiente, mas perseguem um objetivo, ou seja, buscam alcançar uma meta;
- **Capacidade de aprendizagem:** um agente é capaz de aprender a partir das ações realizadas, considerando sucessos e fracassos, de forma a melhorar seu desempenho.

3.2 O Projeto de Sistemas Multiagente

A arquitetura de um sistema multiagente representa os tipos de agentes no sistema, a forma como eles interagem e a estrutura interna destes agentes. Essas arquiteturas podem ser analisadas considerando dois níveis de abstração [22] [59]:

- O *nível global* que trata a questão da coordenação e cooperação dos diversos agentes da sociedade. Esses mecanismos podem ser representados pela configuração dos componentes que constituem o sistema e pelas conexões que coordenam as atividades entre os componentes.

- O *nível detalhado* que trata da estrutura interna do agente. Mostra como ele é implementado em relação às suas propriedades, à sua estrutura e como os módulos que o compõem interagem, garantindo sua funcionalidade. Trata do projeto de cada agente da sociedade, segundo a abordagem reativa, deliberativa ou híbrida.

O sistema de padrões, apresentado no capítulo 5, descreve padrões para o projeto global e para o projeto detalhado de sistemas adaptativos baseados em agentes.

3.3 AUML – Uma Linguagem de Especificação para o Desenvolvimento de SMA

A AUML (Linguagem de Modelagem Unificada para Agentes) surgiu tendo em vista que a UML [7] (Linguagem de Modelagem Unificada) possui limitações para modelagem de agentes e sistemas baseados em agentes. Dado que a UML tem tido uma larga aceitação na modelagem de software orientado a objetos, nada mais coerente que tentar expandir a UML para dá suporte à modelagem de sistemas baseados em agentes, sendo que um agente pode ser considerado como a extensão de um objeto. Logo, essa extensão levou em consideração as exigências e as distinções da tecnologia de agentes, surgindo então a AUML [48]. Odell, Parunak e Bauder [48] abordam um subconjunto da AUML, no qual são especificados Protocolos de Interações de Agentes (AIP) e outras noções baseadas em agentes.

De forma geral, as técnicas para representação dos AIP's foram divididas em três níveis de apresentação [48]:

- **Nível 1:** representação dos protocolos de interações globais do sistema.
- **Nível 2:** representação das interações entre os agentes.
- **Nível 3:** representação do processamento interno do agente.

O nível 1 especifica uma solução que pode ser reutilizável para organizar os sistemas. Os protocolos de interação de agentes (AIP's) oferecem soluções reutilizáveis que podem ser aplicadas em vários tipos de seqüências de mensagens

que são encontradas entre agentes. Na AUML há uma técnica que expressa soluções de protocolos para o reuso: os *pacotes* [48].

Os pacotes são mecanismos de propósito geral, utilizados para organizar elementos do modelo em grupos. Um pacote pode estar inserido dentro de um outro pacote. Os pacotes são utilizados para tratar uma grande parte do sistema.

A **Figura 2** mostra dois pacotes. O pacote *comprador* expressa um protocolo simples entre um corretor e um varejista. O varejista recebe uma proposta de compra do corretor que responde com uma contraproposta. Para certos produtos, o varejista pode solicitar, previamente, a um atacadista as disponibilidades e os custos dos produtos solicitados pelo comprador. Baseado nas informações recebidas do atacadista, o varejista pode fornecer uma proposta mais precisa. Toda essa operação poderia ser colocada dentro do pacote *comprador*, entretanto, há situações em que não há necessidade de protocolos adicionais envolvendo o atacadista. Portanto, dois pacotes podem ser definidos: o pacote *comprador* e o pacote *fornecedor.* Quando um cenário particular requer um protocolo com o atacadista, o pacote *fornecedor* pode ficar aninhado ao pacote *comprador*, caso contrário ele pode ficar como um pacote distinto.

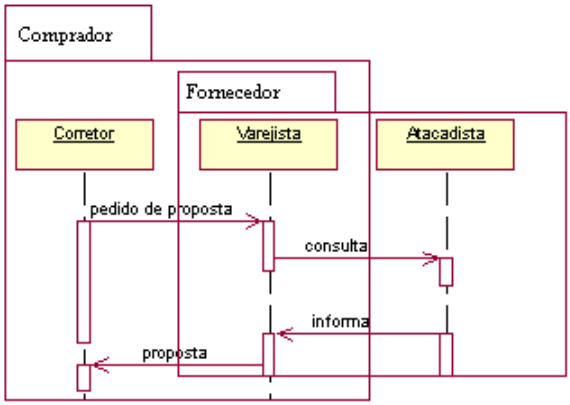

Figura 2 - Utilização de pacotes para expressar protocolos aninhados [48]

O nível 2 utiliza diagramas de seqüência, de colaboração, de atividades e de estado que mostram como os agentes interagem entre si por meio da troca de mensagens.

No diagrama de seqüência, a sintaxe utilizada para representar um agente (ou conjunto de agentes) é a seguinte: *nome do agente/papel: classe*, na

qual *classe* se refere à categoria que o agente pertence; o *papel* identifica a "função específica" que um agente executa durante sua existência. Por exemplo, João/empregado: pessoa - João é o agente que desempenha o papel de empregado e é instância da classe pessoa. Outra forma de representar um agente, de acordo com a sintaxe é: *Nome do agente/papel*, por exemplo, João/empregado; ou simplesmente *Papel*, ou seja, Empregado. A **Figura 3** mostra que as interações entre os agentes utilizam um AC (ato de comunicação) para representar a comunicação entre o *agente 1* e o *agente 2*, ao invés de uma "mensagem" típica da orientação a objetos, como utilizada na UML .

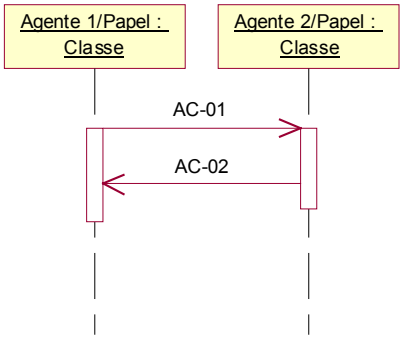

Figura 3 - Exemplo de um diagrama de seqüência entre agentes [48]

Uma outra extensão a UML se dá no suporte à interação concorrente. A **Figura 4** mostra três formas de representar a interação múltipla:

- Todas as ACs são enviadas ao mesmo tempo, e de forma concorrente;
- Dependendo da decisão, nenhuma ou várias ACs são enviadas. Caso mais de uma seja enviada, elas o fazem de forma concorrente;
- Ou-exclusivo - apenas uma AC é enviada.

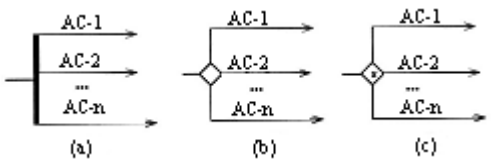

Figura 4 - Extensões que dão suporte à interação concorrente [48]

A **Figura 5** mostra formas equivalentes de representar os três tipos de interações concorrentes. As barras verticais paralelas em seqüência indicam os

agentes recebedores dos atos de comunicação (AC). Um mesmo agente pode ser ativado por vários ACs e possuir instâncias do mesmo papel, ou instâncias de diferentes papéis, ou ainda diferentes agentes recebedores podem ser ativados.

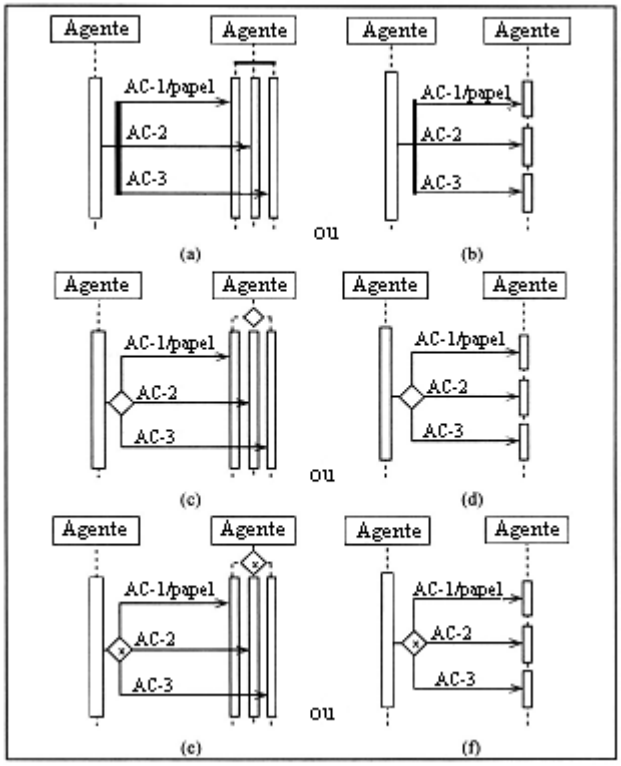

Figura 5 - Exemplos de formas de representação de interações concorrentes [48]

O diagrama de colaboração mostra a informação equivalente ao diagrama de seqüência, ou seja, mostra a ordem cronológica das comunicações entre os agentes, porém possui uma representação gráfica diferenciada (**Figura 6**). São características dos diagramas de colaboração:

- Os agentes (retângulos) podem estar dispostos em qualquer lugar no diagrama;
- A seqüência das interações é numerada;
- Pode fornecer maior clareza de entendimento;
- É semanticamente equivalente ao diagrama de seqüência.

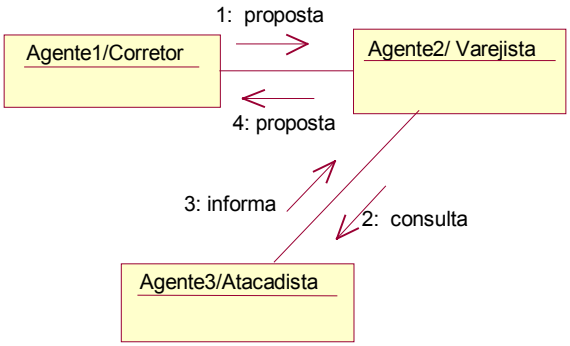

Figura 6 - Exemplo de um diagrama de colaboração entre os agentes [48]

O diagrama de atividades fornece uma visão baseada no processamento das informações, no qual os agentes são representados pelas divisões verticais, as operações são representadas por retângulos de bordas arredondadas e os eventos por setas.

O exemplo da **Figura 7** mostra o protocolo utilizado no processamento de uma compra eletrônica. O agente *Cliente* faz um pedido. Em seguida, o agente *Intermediador* recebe o pedido e o processa. Em seguida o agente *RC* (Rede de Comércio) recebe o evento com o pedido e o aceita. O agente *RC* aguarda então a aceitação da quota pelo agente *Distribuidor*. Somente após a aceitação recebida é que o processamento do pedido é prosseguido. Após o pedido ter sido processado, os agentes requisitantes recebem eventos de notificação que disparam operações para conclusão do mesmo.

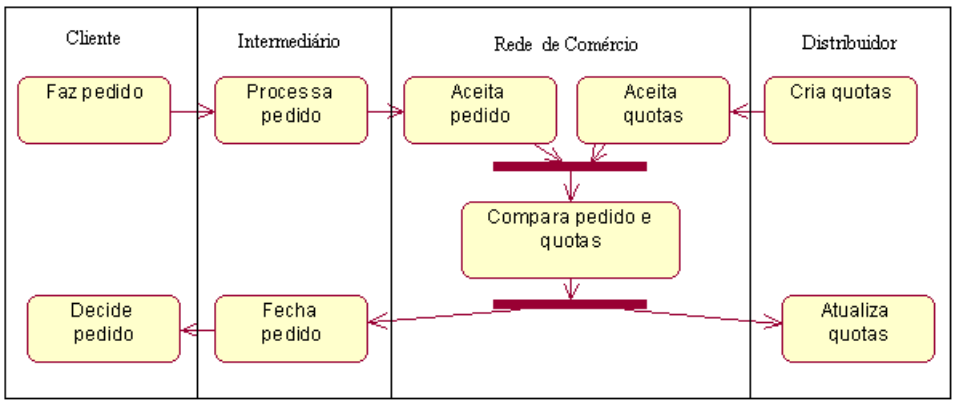

Figura 7 - Exemplo de um diagrama de atividades [48]

O diagrama de estados (**Figura 8**) não fornece uma visão centrada no processo de interação entre agentes; serve apenas como uma visão auxiliar, pois é focada nos estados permissíveis e fornece um mecanismo de verificação da

integridade das mudanças de estado por meio de transições válidas. **Figura 8** mostra um processo de pedido de compra realizado entre um agente A (cliente) e um agente B (fornecedor). O diagrama de estado indica os estados e as transições válidas no protocolo de pedido de compra.

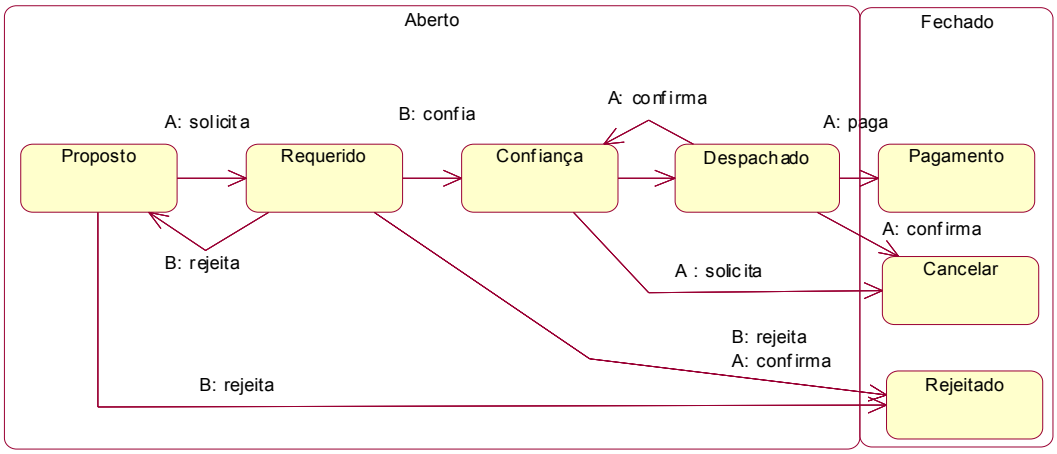

Figura 8 - Exemplo de um diagrama de estados [48]

No nível 3 é abordado o comportamento interno dos agentes. Para especificar esse comportamento, são utilizados os diagramas de estado e de atividades do nível 2.

A AUML foi utilizada para a para a modelagem das estruturas dos padrões propostos no capítulo 5. A escolha da AUML, se deu em virtude dessa linguagem de modelagem abordar o projeto de sistemas multiagente, em diferentes níveis de abstração (nível global e nível detalhado, como descrito na seção 3.2).

3.4 Arquiteturas de Agentes Genéricos

No processo de identificação de padrões descrito na seção 2.4, foram analisadas as arquiteturas dos agentes genéricos das ferramentas Zeus [67], *AgentBuilder* [1], JADE [34], bem como a proposta de uma arquitetura em camadas elaborada por Kendall [36]. Nas subseções seguintes descreve-se cada uma dessas arquiteturas.

3.4.1 Arquitetura Agente em Camadas

Kendall [36] propõe um padrão arquitetural para organizar o comportamento de agentes de software. Na descrição da solução do padrão, a

arquitetura do agente é decomposta em sete camadas, de acordo com o tipo de padrão comportamental utilizado em cada camada, como pode ser visualizado na **Figura 9**. As camadas podem ser identificadas a partir das características gerais de agentes, que são: autonomia, mobilidade, inteligência, cooperação e reatividade. Dependendo do tipo de agente que será desenvolvido, o número de camadas pode variar.

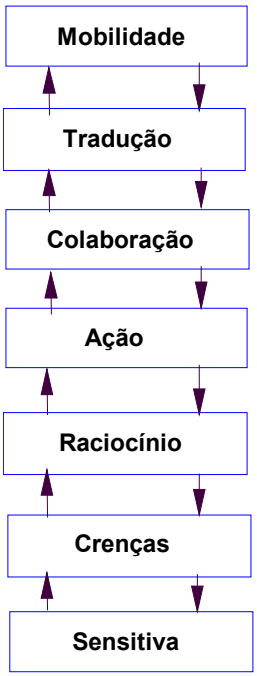

Figura 9 - Arquitetura em camadas para o desenvolvimento de agentes

Na camada *Sensitiva*, o agente capta o seu ambiente por meio de sensores. Na camada *Crenças*, o agente mantém modelos de seu ambiente e de si mesmo. Baseado nestes modelos, o agente determina o que fazer na camada *Raciocínio*. A camada *Ação* cumpre um plano selecionado pela camada *Raciocínio*. Na camada *Colaboração*, o agente determina sua abordagem para cooperar ou trabalhar com outros agentes. Na camada *Tradução*, o agente formula mensagens para outros agentes, traduzindo-as para outras semânticas. A camada *Mobilidade* permite a migração real e virtual entre agentes.

Cada uma das camadas, descrita acima, é composta por um conjunto de padrões de projeto orientados a agentes, independentes do domínio da aplicação, que fornecem uma solução para desenvolver características próprias de agentes em cada camada.

3.4.2 O Agente Genérico da Ferramenta AgentBuilder

Na especificação da arquitetura de um agente, na *AgentBuilder*, é definido o modelo mental de um agente que é caracterizado pelos seguintes conceitos [1]:

- **Estado -** representa o atual estado do ambiente interno e externo do agente. Esse estado é atualizado conforme o ambiente percebido pelo agente. O agente pode ter informação de estado sobre o ambiente externo, sobre outros agentes, sobre as interações com outros agentes e sobre si próprio.

- **Capacidades -** define as ações que o agente pode executar quando condições pré-definidas são satisfeitas. A arquitetura do agente na *AgentBuilder* define ações em duas categorias principais: ações privadas e ações comunicativas. As ações privadas afetam ou interagem com o ambiente do agente e não dependem de interações com outros agentes. As ações comunicativas são as que enviam mensagens para outros agentes.

- **Negociação -** é um acordo feito com outro agente para que uma ação seja executada em um determinado instante. Nesse modelo, não há garantia de que uma determinada ação seja executada por um agente; ele tentará executá-la de acordo com as suas capacidades.

- **Regras de comportamento -** podem ser vistas como sentenças *quando – se - então*. O *quando* da regra enfoca novos eventos ocorrendo no ambiente de um agente, como novas mensagens recebidas de outros agentes. O *se* compara o modelo mental atual com condições que são requeridas para uma regra ser aplicável. O *então* define as ações do agente executadas em resposta a um evento.

- **Intenções -** são similares a negociações, em que um agente executa ações em nome de outro agente. No entanto, uma negociação é um acordo para executar uma ação simples, já uma intenção é um acordo para executar quaisquer ações que são necessárias para atingir um

estado desejado do ambiente. Para atingir a meta especificada por uma intenção, o agente deve estar apto a construir planos para executar uma meta, a monitorar o sucesso das ações executadas e a construir planos alternativos caso o original falhe.

O conjunto dessas características forma a Arquitetura do *Agente Inteligente Reticular,* ou seja, a arquitetura do agente genérico da ferramenta *AgentBuilder*, ilustrada na **Figura 10**. Nesta arquitetura, um interpretador monitora, continuamente, as mensagens que chegam, atualiza o modelo mental do agente e executa ações. No começo do processo, o agente é inicializado com crenças, intenções, capacidades e regras de comportamento iniciais. Um agente não trivial requer pelo menos uma regra de comportamento, os outros elementos são opcionais. O modelo mental contém as crenças, intenções, capacidades e regras atuais do agente [1].

O ciclo de execução do agente consiste nos seguintes passos:

1. Processamento de novas mensagens;
2. Determinação de quais regras são aplicáveis à situação corrente;
3. Execução das ações especificadas por essas regras;
4. Atualização do modelo mental, de acordo com estas regras;
5. Planejamento.

O processamento de novas mensagens requer identificação de quem envia a mensagem e da autenticidade do mesmo; então, a mensagem é analisada e colocada como parte do modelo mental. O próximo passo é determinar quais regras casam com a situação corrente. Padrões de comparação comparam os elementos do modelo mental com os padrões condicionais, localizados nas regras comportamentais, para determinar quais regras são satisfeitas. Uma regra é escolhida para ser executada quando todas as condições são satisfeitas; a regra é colocada na agenda do agente para ser executada. A execução da regra consiste na execução de ações privadas e abertas e em mudanças no modelo mental. Depois, o modelo mental do agente é atualizado, adicionando elementos mentais ou

removendo-os, como especificado pela execução das regras de comportamento. O passo final do ciclo requer o desenvolvimento de um plano[3]. O módulo de planejamento deve desenvolver planos que satisfaçam as metas especificadas pelas intenções do agente [1].

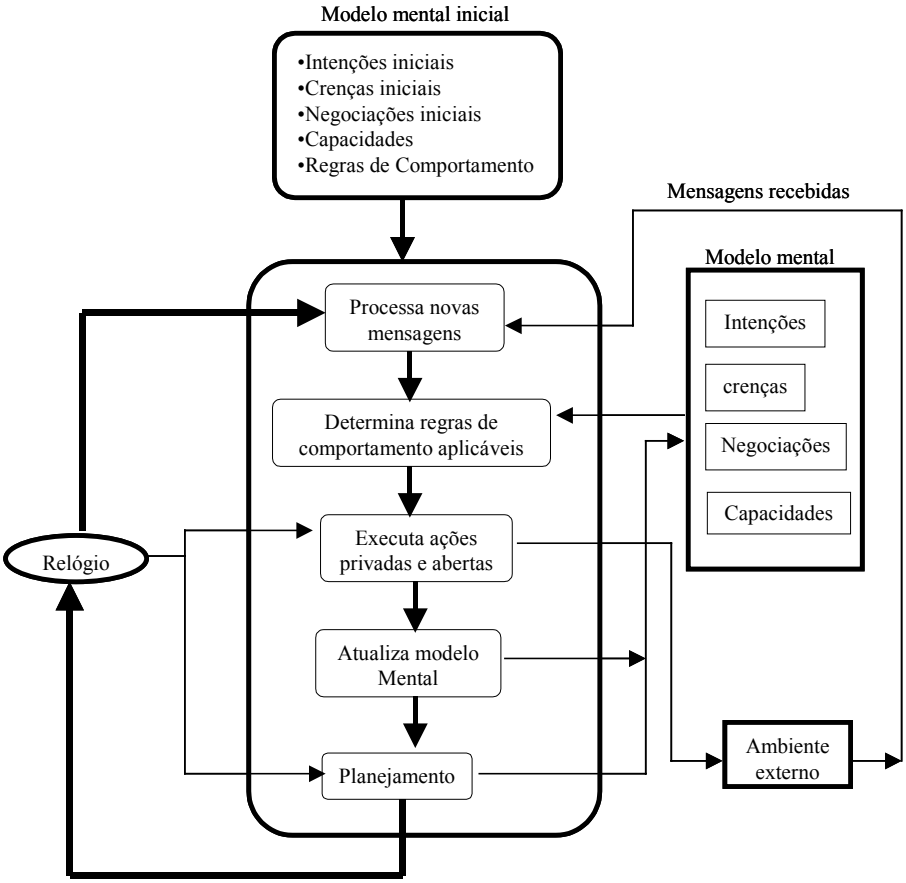

Figura 10 - Arquitetura do agente genérico da ferramenta *AgentBuilder* [1]

3.4.3 O Agente Genérico da Ferramenta Zeus

Como mostrado na **Figura 11**, o agente genérico Zeus inclui os seguintes componentes [17] [67]:

- Uma **Caixa de Correio** que manipula as comunicações entre o agente genérico e outros agentes.

---

[3] Um plano pode ser definido como um conjunto de atividades organizadas de acordo com uma ordem de execução definida. Estas atividades são distribuídas entre agentes capacitados a executá-las.

- Um **Manipulador de Mensagens** que processa a as mensagens que chegam na caixa de correio, despachando-as para os componentes relevantes do agente genérico.

- Um **Mecanismo de Coordenação** que toma decisões a respeito das tarefas dos agentes, e também é responsável pela coordenação das interações com outros agentes.

- Um **Banco de dados de Conhecimento** que descreve as relações do agente genérico com outros agentes na sociedade e suas crenças sobre as capacidades desses agentes.

- Um **Planejador e Escalonador** que planeja as tarefas dos agentes, baseadas nas decisões tomadas pelo mecanismo de coordenação.

- Um **Banco de Dados de Recursos** que mantém uma lista de recursos que pertencem e estão disponíveis ao agente genérico.

- Um **Banco de Dados de Ontologias** que armazena a definição lógica de cada tipo de atributo.

- Um **Banco de Dados de Tarefas/Planos** que fornece uma descrição lógica das tarefas disponíveis nos agentes.

- Um **Monitor de Execução** que é responsável pelo relógio interno do agente genérico. Ele inicia, suspende e monitora as atividades programadas pelo *Planejador/Escalonador*. Também informa ao *Planejador* sobre as condições de término satisfatórias e excepcionais das tarefas que estão sendo monitoradas.

Quando uma mensagem de outro agente é recebida pela *Caixa de Correio* do agente genérico, a mensagem é passada ao *Manipulador de Mensagens* para ser processada. Após a recepção da mensagem, o *Manipulador de Mensagens* interpreta isto como um pedido para alcançar uma determinada meta. Conseqüentemente, remete a mensagem para o *Mecanismo de Coordenação* para determinar que se alcance essa meta e, nesse caso, elabore e coordene um plano de ações apropriado para este objetivo.

O *Mecanismo de Coordenação* procura alcançar a meta e invoca o *Planejador* para construir um plano para atingir esse objetivo. O Planejador então cria um plano, com a descrição das ações contidas no seu *Banco de Dados de Planos* e reserva os recursos, que serão necessários para a execução do plano, em seu *Banco de dados de Recursos*. Porém, o *Planejador* pode achar que alguns recursos que são requeridos pelo plano não estão disponíveis em seu *Banco de dados de Recursos*. No entanto, ele não é capaz de produzir esses recursos e, para isso, chama o *Mecanismo de Coordenação* para buscar ajuda externa a fim de produzir esses recursos.

O *Mecanismo de Coordenação* começa a tentar prover os recursos exigidos. Para fazer isto, consulta no seu *Banco de Dados de Conhecimento,* os nomes dos agentes que ele acredita que possam produzir os recursos exigidos. Não achando nenhum agente com as habilidades apropriadas, o *Mecanismo de Coordenação* usa a *Caixa de Correio* para enviar uma mensagem a um agente que conhece as habilidades de cada agente membro de um determinado grupo (chamado de agente facilitador) e pede uma lista de todos "os agentes ativos" e as suas correspondentes habilidades. Após o recebimento de uma resposta do facilitador, a *Caixa de Correio* remete a mensagem de resposta para o *Mecanismo de Coordenação*.

De posse de uma lista de agentes com as suas habilidades, o *Mecanismo de Coordenação* armazena esta informação em seu *Banco de Dados de Conhecimento* e então envia mensagens aos agentes, pedindo para estes que façam um acordo para oferecer os recursos necessários. Novamente, as mensagens de partida são enviadas pela *Caixa de Correio* e as suas respostas retornarão ao *Mecanismo de Coordenação* por meio da *Caixa de Correio* e do *Manipulador de Mensagens*.

Uma vez que todos os agentes envolvidos devolvam suas solicitações para a realização das tarefas ou o prazo final de resposta expire, o *Mecanismo de Coordenação* passa a retornar as solicitações ao *Planejador*, que seleciona os agentes satisfatórios para prover os recursos exigidos. O *Planejador* devolve ao *Mecanismo de Coordenação* uma lista dos agentes “contratantes” para os quais ele deve enviar mensagens confirmando o acordo a ser realizado e outra lista para que

o *Mecanismo de Coordenação* envie mensagens sobre as solicitações que foram rejeitadas.

Porém, antes de enviar a mensagem de confirmação ou rejeição da solicitação, o *Mecanismo de Coordenação* envia uma mensagem ao agente que originalmente pediu que ele alcançasse a meta e informa que ele pode executá-la e o custo de fazer isto. Logo, o *Mecanismo de Coordenação* espera por uma resposta à sua solicitação. Se for uma resposta favorável, ele envia mensagens de confirmação ou rejeição da solicitação para os seus próprios agentes contratantes e informa ao *Planejador* que o plano para atingir a meta deve ser executado. Por outro lado, se uma resposta desfavorável for recebida, as mensagens de rejeição de oferta são enviadas para todas os agentes contratantes e o *Planejador* é orientado a cancelar o plano.

Uma vez que um plano marcado esteja pronto para a execução, o *Monitor de Execução* executa as ações especificadas no plano. Se o plano inteiro é executado com sucesso, os resultados finais são enviados pela *Caixa de Correio* para o agente que solicitou a realização da meta.

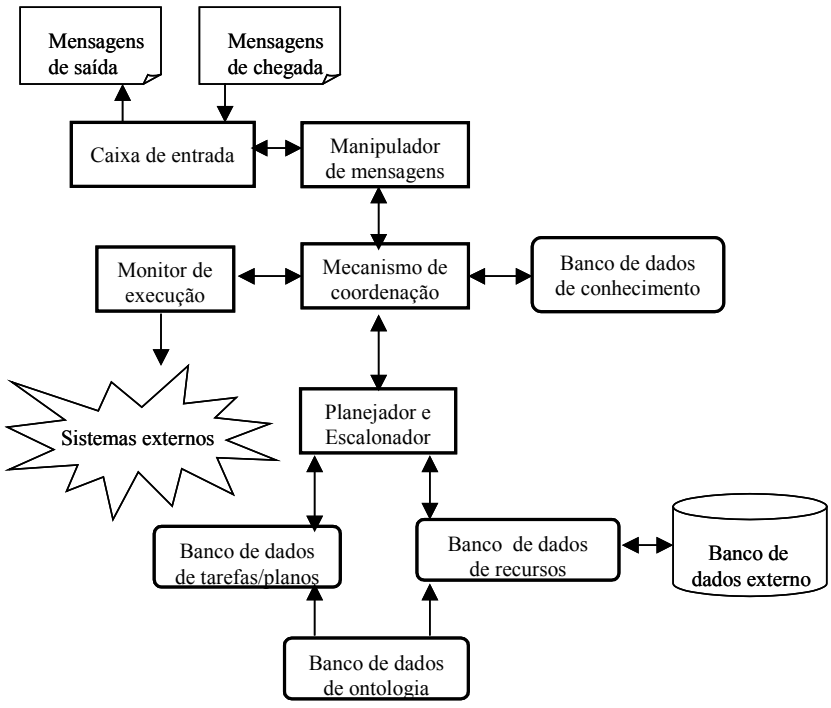

Figura 11 - Arquitetura do agente genérico Zeus [67]

3.4.4 O Agente Genérico da Ferramenta JADE

O ambiente JADE é usado para o desenvolvimento de sistemas e aplicações multiagente em conformidade com os padrões FIPA [25] para agentes inteligentes. Ele inclui dois produtos principais: uma plataforma de agente genérico e um pacote para desenvolvimento de agentes específicos em linguagem Java. O JADE é todo escrito em Java e o programador deve implementar os agentes específicos também nesta linguagem [34].

A Fundação para Agentes Físicos Inteligentes ("Foundation for Intelligent Physical Agents" – FIPA) é uma organização que desenvolve padrões para agentes de software a fim de permitir a interação entre sistemas de agentes heterogêneos. Ela está organizada e estruturada de acordo com dois grupos: aqueles que estão envolvidos na produção e desenvolvimento de padrões e aqueles que estão envolvidos no mecanismo de suporte da FIPA [25].

O agente genérico do ambiente JADE é uma arquitetura reativa e segue a estrutura mostrada na **Figura 12**.

O modelo computacional do agente é multitarefa, no qual tarefas (ou comportamentos) são executadas concorrentemente. Cada funcionalidade (ou serviço) provida por um agente, pode ser implementada com a integração de mais de um comportamento. A definição da ordem de execução das tarefas é feita pelo Gerenciador de Tarefas (**Figura 12)**.

Um agente JADE pode apresentar vários estados, de acordo com o ciclo de vida do agente genérico especificado pela FIPA. Os estados podem ser [34]:

- *Inicial*: o agente foi criado, mas ainda não pode ser registrado no sistema gerenciador de agentes; não possui nome, nem endereço e não pode se comunicar com outros agentes.

- *Ativo*: o agente está registrado no sistema gerenciador de agentes (AMS), tem um nome e um endereço e pode acessar vários serviços JADE.

- *Suspenso*: o agente está em estado de interrupção e nenhum comportamento é executado.
- *Espera*: o agente está bloqueado, esperando por alguma mensagem.
- *Destruído*: o agente morre e seu registro é eliminado.
- *Transitório*: o agente migra para um novo local; o sistema continua recebendo mensagens que serão depois envidas para o agente em seu novo endereço.
- *Cópia*: estado interno usado por JADE durante a clonagem de um agente.
- *Movido*: um agente móvel muda para um novo local.

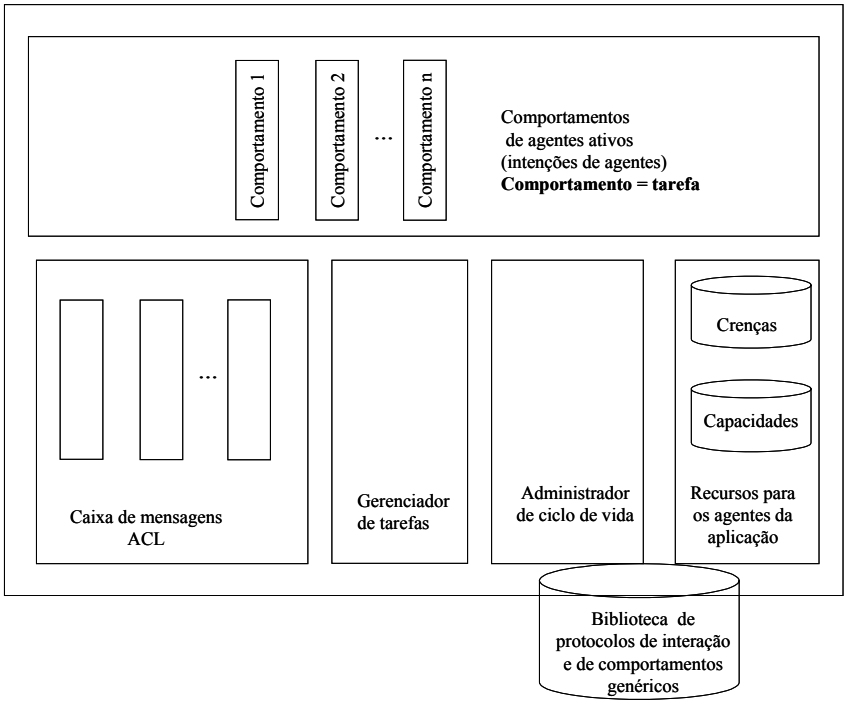

Figura 12 - Arquitetura do agente genérico JADE [34]

O agente JADE provê métodos públicos para realizar transições entre os vários estados possíveis. JADE não habilita agentes com capacidades específicas além daquelas necessárias para comunicação e interação. No entanto, o modelo de agentes permite a integração de programas externos com os agentes de tarefas. A biblioteca existente em JADE inclui uma classe, chamada *JessBehaviour*, que

permite o uso da linguagem *JESS* como mecanismo de raciocínio do agente. O *JESS* é bastante útil no controle da ativação e desativação dos comportamentos em JADE, o que permite a implementação de uma arquitetura formada por agentes reativos e deliberativos (na qual o *JESS* executa os papéis deliberativos e JADE, os reativos). No futuro, os desenvolvedores de JADE prometem o acréscimo de uma ferramenta visual para compor os comportamentos de agentes e oferecer uma arquitetura de mais alto nível.

3.4.5 Análise Comparativa

Uma arquitetura em camadas, como a proposta por Kendall [36], é sem dúvida mais flexível, extensível e de fácil entendimento e manutenção. Porém, a organização de um sistema em camadas pode afetar o desempenho do mesmo. Como por exemplo, para que a camada *Ação* tenha acesso ao conhecimento do agente (camada *Crenças*) é necessário o intermédio da camada *Raciocínio*, como pode ser visualizado na **Figura 9**.

No agente genérico da ferramenta *AgentBuilder*, os processos executados pelo agente não estão distribuídos em hierarquias de camadas, sendo que os módulos que exercem as funcionalidades do agente, até mesmo os que estão em níveis diferentes, comunicam-se diretamente (**Figura 10**). Como na arquitetura em camadas, o conhecimento do agente (denominado modelo mental) da ferramenta *AgentBuilder* está concentrado em um só módulo.

Na arquitetura do agente genérico da ferramenta Zeus, os módulos estão agrupados em níveis. Por exemplo, de acordo com a **Figura 11**, os módulos *Monitor de Execução*, *Mecanismo de Coordenação e Banco de dados de Conhecimento* estão em um mesmo nível e só se comunicam com os níveis adjacentes. Nota-se também que, nesta arquitetura, o conhecimento do agente está distribuído em diferentes níveis, por exemplo, o banco de dados que descreve as relações do agente com outros agentes (Banco de Dados de Conhecimento) está em um nível diferente do banco de dados que fornece descrições lógicas de operadores de tarefas (Banco de Dados de Tarefas/Planos).

A arquitetura do agente genérico da ferramenta JADE não dá suporte direto para o desenvolvimento de agentes do tipo deliberativo (descrito na seção

3.1), sendo necessária a integração de programas externos para a implementação de mecanismos de raciocínio do agente. Essa arquitetura só dá suporte direto apenas para o desenvolvimento de agentes do tipo reativo (descrito na seção 3.1).

Com relação às funcionalidades exercidas pelos agentes, as arquiteturas das ferramentas Zeus, *AgentBuilder* e em camadas, executam basicamente as mesmas tarefas, porém a ordem de execução das mesmas é diferente. Nessas arquiteturas também podemos observar em comum, a existência de mecanismos de raciocínio, como também de módulos nos quais ficam armazenados as crenças, as regras de comportamento, as capacidades e o conhecimento do agente.

Em todas as arquiteturas analisadas existem mecanismos de gerenciamento de mensagens.

Para propormos os padrões Deliberativo e Reativo, que são apresentados no capítulo 5, a principal referência é a arquitetura em camadas, sendo que foram abstraídas as funcionalidades das camadas *Tradução* e *Mobilidade*, uma vez que na nossa proposta, o agente não migrará nem se comunicará com sociedades de agentes com semânticas diferentes da sociedade a qual ele pertence.

Nas arquiteturas das ferramentas *AgentBuilder* e Zeus, os módulos que planejam e que executam as ações têm comunicação direta com os módulos que retêm o conhecimento do agente, tornando o ciclo de processamento do agente mais rápido. Levando essa vantagem em consideração, a base de conhecimento do padrão Deliberativo comunica-se com os módulos *Raciocínio*, *Ação*, *Comunicação* e *Sensores*. Já no padrão Reativo, o conhecimento do agente é reduzido a um conjunto de regras de ação, localizadas na camada *Regras*.

A **Tabela 1** mostra os principais pontos em comum e as diferenças entre as arquiteturas analisadas.

		Arquiteturas de Agentes Genéricos			
		Camadas	*AgentBuilder*	Zeus	JADE
Critérios de Comparação	Organização em camadas	•		•	
	Suporte direto para o desenvolvimento de agentes do tipo deliberativo	•	•	•	
	Mecanismos de Gerenciamento de mensagens	•	•	•	•
	Módulo de conhecimento	•	•	•	
	Conhecimento distribuído		•	•	
	Mecanismos de raciocínio	•	•	•	

Tabela 1 - Tabela comparativa entre arquiteturas de agentes genéricos

4. SISTEMAS ADAPTATIVOS E MODELAGEM DE USUÁRIOS

Este capítulo apresenta os principais conceitos e técnicas utilizadas no desenvolvimento de sistemas adaptativos e da modelagem de usuários, explorando a compreensão de problemas e conceitos comuns que são relevantes para a extração de padrões neste domínio, de acordo com processo de identificação de padrões descrito na seção 2.4.

4.1 Sistemas Adaptativos e Adaptáveis

Os sistemas hipermídia adaptativos têm estreita relação com as tecnologias de modelagem de usuários e grupos de usuários, bancos de dados, programação distribuída na *Web*, métodos colaborativos e interfaces dinâmicas adaptativas. Esses sistemas tentam antecipar as expectativas dos usuários a partir de modelos representando seu perfil. O objetivo geral de tais sistemas é prover aos seus usuários conteúdo atualizado, subjetivamente interessante, com informação pertinente, num tamanho e profundidade adequados ao contexto e em correspondência direta com o *seu perfil*. Este perfil funciona como uma referência para o sistema que busca adaptar seu ambiente a ele [49].

O desenvolvimento de sistemas adaptativos é composto por dois grandes processos: a modelagem de usuários e a geração da adaptação. A **Figura 13** ilustra as tarefas que compõem os processos de modelagem de usuários e de adaptação de um sistema. Na execução da tarefa de modelagem de usuário, o sistema coleciona dados sobre o usuário, processa esses dados e os representa na forma de modelos de usuário; posteriormente, na execução da tarefa de adaptação, o sistema constrói modelos de adaptação baseado nos modelos de usuários e então gera os efeitos de adaptação do sistema.

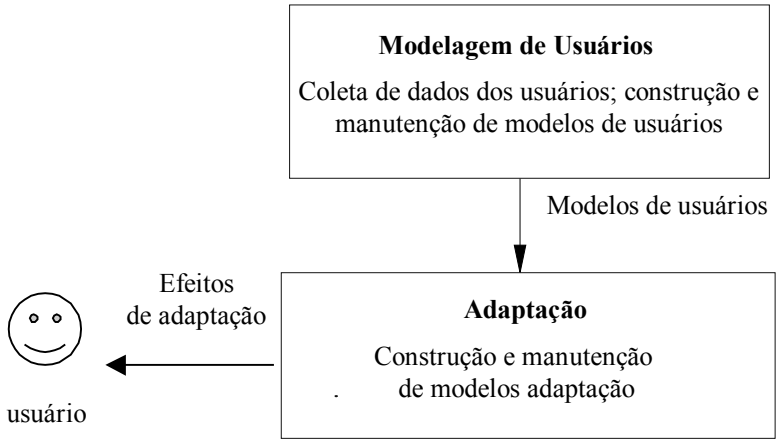

Figura 13 - Os processos de desenvolvimento de sistemas adaptativos

Os sistemas adaptativos surgem com o propósito de melhorar a usabilidade dos sistemas tradicionais. A maior parte dos sistemas adaptativos consegue facilitar as atividades do usuário, fazendo com que o sistema se ajuste a determinadas características deste. O projeto de um sistema adaptativo tenta resolver as seguintes questões relativas à adaptação [44]:

- Qual é a funcionalidade do sistema que é suscetível de adaptação?
- Quais são as características do usuário as quais o sistema se ajusta?
- Quais são as técnicas e os métodos que o sistema utiliza para produzir a adaptação?
- Em que momento durante o funcionamento do sistema se produz a adaptação?

Em função das decisões tomadas em cada uma das perguntas, o autor projeta os componentes do sistema adaptativo. Existem três elementos que, implícita ou explicitamente, estão presentes na maioria dos sistemas adaptativos [44]:

- **Modelo de domínio:** descreve a estrutura do domínio da aplicação em termos de conceitos e relacionamentos entre os conceitos.
- **Modelo de usuário:** armazena as características do usuário que o sistema leva em conta para realizar a adaptação.

- **Modelo de adaptação:** consiste em regras de adaptação que definem o processo de geração da adaptação, que estabelecem como a informação do modelo do usuário influi na adaptação do sistema e também especifica como e quando atualizar a informação armazenada no modelo de usuário.

A decisão de em que momento durante o funcionamento do sistema se produzirá a adaptação dependerá do grau de controle que o usuário tem sobre a adaptação. Os sistemas nos quais o usuário pode iniciar, propor, selecionar e produzir a adaptação ou também deixar que o sistema execute algumas dessas funções, são chamados de *sistemas adaptáveis*, já os sistemas que executam todas essas funções, de forma automática, são chamados de *sistemas adaptativos*.

A adaptação implícita, feita de forma automática e a adaptação explícita, feita com a intervenção direta do usuário, podem coexistir na mesma aplicação. A escolha da forma de adaptação a ser usada deve ser feita cuidadosamente, levando-se em conta as conveniências e exigências dos usuários. O controle do usuário sobre a adaptação pode ser provido em nível geral (os usuários podem habilitar e desabilitar a adaptação totalmente) ou ao nível de tipo de adaptação (usuários podem aprovar ou reprovar que alguns tipos de adaptação ocorram).

4.1.1 Formas de Adaptação

Uma importante questão a responder quando falamos sobre qualquer tipo de sistema adaptativo é: O que pode ser adaptado no sistema? O que pode ser adaptado nos sistemas adaptativos é a *apresentação* e/ou a *navegação* [11] [49].

A maioria das técnicas de *apresentação adaptativa* adapta o conteúdo da interface, acessado por um usuário ou grupo de usuários, ao seu conhecimento, metas e outras características. Por exemplo, para um usuário mais qualificado pode ser provida uma informação mais detalhada e profunda, enquanto que um usuário novato pode receber explicações básicas sobre o assunto. Em sistemas adaptativos, o conteúdo de uma interface pode não ser apenas um texto, mas também pode ser um conjunto de vários itens hipermídia [11].

As técnicas que dão suporte à *navegação adaptativa* ajudam os usuários a encontrar caminhos no espaço de navegação, adaptando a forma de apresentar os *links* às preferências, conhecimento e outras características de um usuário ou grupo de usuários [11].

A **Tabela 2** resume as principais técnicas de apresentação e navegação adaptativa, que são descritas nas próximas subseções.

Formas de adaptação	*Técnicas de apresentação adaptativa*	Texto condicional
		Texto expansível
		Interface variante
		Fragmento variante
		Técnica baseada em *frames*
	Técnicas de navegação adaptativa	Orientação direta
		Classificação adaptativa
		Ocultação
		Anotação adaptativa

Tabela 2 - Técnicas de apresentação e navegação adaptativa

4.1.1.1 Técnicas de Apresentação Adaptativa

As principais técnicas para a adaptação do conteúdo são [11] [49]:

- **Texto condicional:** a informação é dividida em várias porções de texto e cada porção é associada a uma ou mais condições relacionadas ao nível de conhecimento do usuário. Ao apresentar a informação, o sistema mostra apenas as porções de texto que tiveram suas condições satisfeitas. Um exemplo é a ocultação de porções de informação que não são relevantes para o nível de conhecimento do usuário ou a apresentação de uma explicação comparativa se o conceito relacionado já é conhecido.

- **Texto expansível:** é um tipo especial de texto no qual os *links* podem ser expandidos para seus conteúdos ou concentrados novamente em uma palavra-chave. A idéia é apresentar, ao usuário, uma interface na qual todas as informações relevantes estejam expandidas e todas as informações irrelevantes estejam representadas apenas por uma palavra ou frase, de acordo com o modelo do usuário.

- **Interface variante:** consiste em manter duas ou mais interfaces alternativas para cada conceito, descrevendo-as de maneiras diferentes, cada uma delas adaptadas a grupos de usuários com características comuns (denominados estereótipos).

- **Fragmento variante:** uma interface consta de vários conceitos e se dispõe de variantes para cada conceito. Uma interface é instanciada com a combinação desses conceitos, de forma que melhor satisfaça aos conhecimentos do usuário.

- ***Técnica baseada em frames*:** a informação relativa a um conceito é representada sob a forma de um *frame* [4] com vários *slots*. Os *slots* podem conter várias explicações variantes sobre um conceito, ligações para outros *frames* e exemplos. Regras especiais de apresentação são empregadas para decidir quais *slots* de um determinado *frame* devem ser apresentados a um usuário, e em que ordem específica isso deve ocorrer. Como por exemplo, no sistema Hypadapter [33], as regras são empregadas para calcular a prioridade de apresentação de cada *slot*. Em seguida, o subconjunto de *slots* com maior prioridade é apresentado ao usuário [54].

4.1.1.2 Técnicas de Navegação Adaptativa

As principais técnicas que dão suporte à navegação são [11] [49]:

- **Orientação direta:** sua aplicação consiste em decidir, em cada ponto de navegação, qual o melhor nó seguinte a ser visitado, levando em conta o modelo de usuário.

- **Classificação adaptativa:** consiste em classificar todos os *links* partindo de um nó, de acordo com a sua relevância, calculada sobre o

[4] Um *frame* é uma estrutura de representação de conhecimento. Nelas são representados objetos de um domínio. Os atributos dos objetos são representados em *slots*

modelo de usuário. Os *links* são então apresentados em ordem decrescente desta relevância.

- **Ocultação:** consiste em restringir o espaço de navegação, ocultando os *links* para os nós não relevantes ao usuário. A ocultação torna invisíveis ao usuário botões, itens de menu e áreas selecionáveis irrelevantes aos seus objetivos.

- **Anotação adaptativa:** consiste em aumentar a informação presente nos *links,* com alguma forma de anotação ou comentário que pode dizer mais sobre o estado corrente dos nós a que se conectam. Esta informação adicional pode ser oferecida sob a forma de texto ou sob a forma de indicadores visuais, tais como ícones especiais, cores ou tamanho de caracteres.

4.2 A Modelagem de Usuários

O modelo de usuário é uma fonte de conhecimento que contém informações, explícitas ou implicitamente adquiridas, de todos os aspectos relevantes ao usuário para serem utilizados no processo de adaptação de uma aplicação de software. Um componente da modelagem de usuário, em um sistema interativo e adaptativo, formula suposições sobre o usuário baseado na interação com ele; armazena-as em uma estrutura de representação apropriada; infere novas suposições com base nas iniciais; mantém a consistência de um conjunto de suposições e fornece a outros componentes do sistema essas suposições sobre o usuário [38].

O objetivo da análise e modelagem de usuários é identificar quem são os usuários e caracterizá-los, isto é, especificar quais funções exercem, quais capacidades possuem, quais seus interesses e preferências.

A modelagem de usuários é necessária, pois por meio dela se pode sintetizar as características e habilidades de usuários ou grupo de usuários com o objetivo de facilitar e melhorar a interação com o sistema, por exemplo, por meio da adaptação das interfaces.

No campo da modelagem de usuários, várias técnicas têm sido investigadas para abordar esses problemas. Algumas técnicas foram parcialmente tomadas de outras áreas de pesquisa (como por exemplo, sistemas da linguagem natural, representação do conhecimento, sistemas conexionistas e interação humano-computador) e foram adaptadas às necessidades específicas da modelagem de usuários, outras técnicas foram desenvolvidas no próprio campo da modelagem de usuário [38].

A modelagem dos usuários é um processo, ilustrado na **Figura 14**, que geralmente é realizado em três fases: *fase de aquisição*, *fase de representação* e *fase de manutenção* das informações sobre os usuários [55].

Na *fase de aquisição* são utilizadas técnicas para aprender novos fatos sobre o usuário, de forma direta (por exemplo, por meio de questionamentos) ou indireta (observando o comportamento do usuário).

Na *fase de representação* são utilizadas técnicas para a representação formal das informações adquiridas sobre o usuário, ou seja, a construção do modelo do usuário.

Na *fase de manutenção* são utilizadas técnicas para a incorporação de novas informações e atualização das informações existentes no modelo, bem como técnicas para fazer inferências a partir dessas informações.

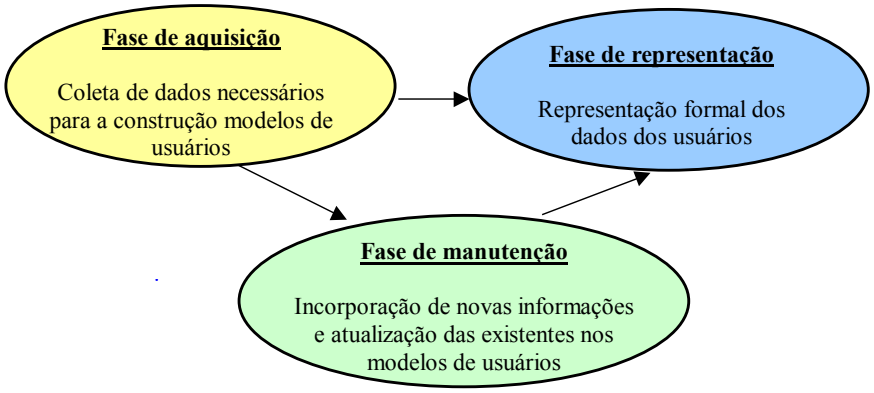

Figura 14 - O processo de modelagem de usuários

Durante a criação de modelos de usuários, surgem algumas questões fundamentais que precisam ser respondidas: “quais informações precisam ser

modeladas?", "como adquirir, representar e manter essas informações?". A **Tabela 3** cita quais informações são necessárias para a construção de modelos de usuários e a **Tabela 4** cita as principais técnicas utilizadas para a aquisição, representação e manutenção desses modelos.

As próximas subseções apresentam as técnicas utilizadas para a aquisição, representação e manutenção de modelos de usuários, bem as informações necessárias para a construção desses modelos.

Informações utilizadas na construção de modelos de usuários	Informações pessoais
	Capacidades e habilidades do usuário
	Interesses e preferências do usuário
	Metas e planos do usuário
	Dados de uso do usuário
	Conhecimento do usuário

Tabela 3 - Informações utilizadas na construção de modelos de usuários

Técnicas de aquisição de informações dos usuários	*Aquisição direta*	Informações fornecidas diretamente pelo usuário
		Uso de estereótipos
	Aquisição indireta	Regras de aquisição
		Reconhecimento de planos do usuário
		Aprendizagem de máquina
		Mineração de uso
Técnicas de representação de modelos de usuários	Representação baseada em lógica	
	Representação probabilística	
	Representação por meio de estereótipos	
	Representação baseada em ontologias	
Técnicas de manutenção de modelos de usuários	Lógica fuzzy	
	Teoria de probabilidades	
	Algoritmos genéticos	

Tabela 4 - Técnicas para a aquisição, representação e manutenção de modelos de usuários

4.2.1 Informações Utilizadas na Construção de Modelos de Usuários

No processo de modelagem de usuários, a coleta de informações é o passo inicial. O conteúdo do modelo do usuário varia de acordo com a aplicação para o qual está sendo construído. Portanto, para definir quais informações precisam ser modeladas, deve-se levar em consideração os recursos computacionais e a necessidade de informação do sistema.

Nas subseções seguintes serão descritas as categorias de informações que têm sido a base para adaptação de muitos sistemas.

4.2.1.1 Informações Pessoais

São informações fornecidas diretamente pelo usuário, como [37]:

- Nome, endereço, número do telefone;
- Dados geográficos (código de área, cidade, estado, país);
- Características do usuário (idade, sexo, escolaridade,estado civil);
- Dados que indicam o estilo de vida do usuário.

4.2.1.2 Conhecimento do Usuário

Suposições sobre o conhecimento que o usuário tem sobre conceitos, relacionamentos entre conceitos, fatos e regras considerando, o domínio da aplicação têm sempre estado entre a maioria dos recursos utilizados para a personalização de sistemas [37].

Muitas abordagens para a construção de sistemas hipermídia adaptativos têm levado o conhecimento do usuário em consideração. O ideal é ajustar a apresentação do material hipermídia, de forma que os usuários não fiquem entediados com explicações desnecessárias, nem se sintam confusos com detalhes que eles não podem entender.

4.2.1.3 Capacidades e Habilidades do Usuário

Além dos sistemas tratarem com o conhecimento do usuário no domínio da aplicação (conhecimento sobre “o que”), o conhecimento do usuário sobre “o como” também exerce um importante papel para a adaptação dos sistemas às necessidades dos usuários. É possível que o usuário saiba como fazer alguma coisa, mas ele pode não estar apto a fazê-la, ou por falta de permissão ou por causa de alguma incapacidade física. É importante que o sistema recomende ou disponibilize apenas aquilo que os usuários estejam aptos a fazer naquele momento [37].

4.2.1.4 Interesses e Preferências do Usuário

Os interesses dos usuários em uma mesma aplicação variam consideravelmente e as informações e/ou os produtos oferecidos a um grupo particular de usuários pode não ser de interesse para outro grupo, podendo até ter efeitos contrários. Um exemplo é um anúncio de um carro apresentado a diferentes públicos, no qual são utilizados enfoques diferentes e até mesmo conflitantes para chamar a atenção dos usuários. Assim, dependendo dos interesses de cada grupo, o anúncio dá enfoque à velocidade, à segurança, à família ou à amizade [37].

4.2.1.5 Metas e Planos do Usuário

Quando interagem com o sistema, os usuários geralmente têm uma meta específica em mente. A meta pode ser, por exemplo, encontrar uma informação ou comprar um produto. Sistemas que ajudam os usuários a atingir suas metas são muito úteis, pois facilitam e aumentam a velocidade da interação, uma vez que o sistema tem expectativas sobre as próximas ações do usuário e pode interpretá-las de uma forma mais flexível [37].

4.2.1.6 Dados de Uso

Dados de uso podem ser diretamente observados e gravados, ou adquiridos pela análise de dados observáveis, ou seja, pela forma como o usuário interage com o sistema. Tecnicamente, a possibilidade dessa observação varia muito de acordo com a aplicação. Os sistemas para *Web* que usam *Java applets* [5] têm um controle maior da interação, podendo colher e gravar dados de uso por meio dos movimentos e cliques do mouse [11].

4.2.2 Técnicas de Aquisição de Modelos de Usuários

Algumas informações dos usuários podem ser observadas diretamente pelo sistema, enquanto que outras podem necessitar de um ou mais passos para

---

[5] *Applets* são pequenos programas construídos na linguagem de programação Java. Os *Applets* podem ser executados dentro de um navegador no cliente quando a página é acessada. Estes pequenos programas podem executar tarefas de controle de acessos.

adquiri-las. Alguns dados podem ser diretamente fornecidos pelos usuários, porém a maior parte deles pode ser inferida a partir de observações feitas quando o usuário interage com o sistema.

Nas próximas subseções iremos descrever técnicas baseadas na entrada explícita de informações pelo usuário (aquisição direta) e técnicas para a aquisição implícita (aquisição indireta).

4.2.2.1 Técnicas de Aquisição Direta

Nas técnicas de aquisição direta, o usuário fornece, de forma direta, as informações necessárias para a construção dos modelos de usuários. Dentre as técnicas de aquisição explícita, podemos destacar:

- **Informações fornecidas diretamente pelo usuário:** Uma técnica de aquisição explícita óbvia é deixar que o próprio usuário forneça os dados necessários. A provisão de dados pelo usuário pode ser feita por meio de questionários e/ou formulários feitos pelo sistema, geralmente na fase inicial do uso do sistema. Há um risco em fazer questionários e afirmações explícitas para que o próprio usuário faça sua auto-avaliação, pois muitas vezes eles não têm consciência das coisas que gostam ou de suas próprias capacidades. Alguns sistemas apresentam consultas controladas, testes, exercícios que visam uma avaliação mais objetiva do usuário [37].

- **Estereótipos:** A classificação dos usuários em estereótipos, ou seja, em grupos de usuários com características semelhantes, é uma técnica bastante utilizada na aquisição direta. O sistema poderá exibir um formulário para o usuário preencher e partir dessas informações classificá-lo em estereótipos pré-definidos pelo sistema e posteriormente podem ser feitas predições baseadas nesses estereótipos. A efetividade da estereotipagem depende da qualidade do estereótipo. Por exemplo, o número de diferentes estereótipos conhecidos pelo sistema, a precisão da distribuição das características dos usuários em estereótipos e a qualidade das inferências que são feitas sobre o membro do estereótipo. Estes estereótipos, por sua vez,

dependem da qualidade da informação sobre a população de usuários e da dimensão que subgrupos com diferentes características relevantes à aplicação podem ser empiricamente distinguidos [37] [11].

4.2.2.2 Técnicas de Aquisição Indireta

Nas técnicas de aquisição indireta, as informações utilizadas para a construção de modelos de usuários são adquiridas por meio da observação do comportamento do usuário durante a interação com o sistema. Dentre as técnicas de aquisição indireta, podemos citar:

- **Regras de aquisição:** É um meio freqüentemente usado para a geração de suposições sobre o usuário. Por exemplo, regras de inferência que são executadas quando uma nova informação sobre o usuário está disponível. Na maioria dos casos, as regras de aquisição referem-se a ações observadas do usuário, ou à simples interpretação do seu comportamento [37]. As regras de aquisição podem ser específicas para um domínio de aplicação particular, ou podem ser independentes do domínio. Sharma [57] usa heurísticas para definir interesses do usuário, como por exemplo: “usuário salvando A, indica interesse em A” ou “usuário demorando mais tempo que o usual no documento A, mostra interesse em A”. Regras de aquisição que são completamente específicas do domínio são os meios mais populares para a aquisição do modelo de usuário, já que são fáceis de implementar.

- **As técnicas de reconhecimento de planos do usuário:** Estão baseadas no raciocínio sobre as metas que o usuário pode ter e a seqüência de ações (plano) para que possa alcançá-las. O processo de reconhecimento de planos do usuário consiste de uma base de conhecimento de tarefas que representa possíveis ações do usuário, o relacionamento entre essas ações e de um mecanismo que identifica o plano atual do usuário e as metas associadas a esses planos, obtidos por meio da observação das interações entre o usuário e o sistema. O reconhecimento de planos é especialmente promissor para aplicações

com um número pequeno de possíveis metas e um número pequeno de possíveis formas de atingir essas metas. Por exemplo, em sistemas de informação, os usuários têm metas específicas, como por exemplo, encontrar uma informação. Se o sistema identifica tais metas, ele pode provê atalhos para esse tipo de rotina, ajudando assim, os usuários na execução de suas tarefas [37] [42].

- **Aprendizagem de Máquina:** Esta técnica utiliza algoritmos da área de aprendizagem de máquina, como algoritmos genéticos, para inferir o estado de conhecimento do usuário, tendo como referência básica o seu comportamento [60]. As técnicas baseadas em aprendizagem de máquina podem inferir conhecimentos do usuário que não estão presentes em um modelo de domínio preestabelecido. Uma importante limitação da aplicação de técnicas de aprendizagem de máquinas, em sistemas que utilizam as tarefas da modelagem de usuário, é que os algoritmos de aprendizagem não constroem o modelo de usuário com um grau de confiança aceitável até que se tenha um grande número de dados de treinamento [63].

- **Técnicas de mineração de uso:** A mineração de uso, utilizada em aplicações para *Web*, focaliza-se em técnicas que possam prever o comportamento do usuário enquanto ele interage com a *Web* [39]. A mineração de uso lida com os dados secundários provenientes da interação do usuário com a *Web*. Os dados de uso da *Web* incluem dados provenientes de *logs* [6] de servidores *web*, *logs* de servidores proxy[7], logs de navegadores *web*, perfis de usuário, *cookies* [8], seções

[6] *Log* é um registro de atividades gerado por programas de computador.

[7] O serviço de *proxy* na *Web* consiste em manter, em uma área de acesso rápido, informações já acessadas. Um servidor *proxy* atua como intermediário entre um cliente e outro servidor. Quando o usuário consulta uma página, o provedor, transparentemente, armazena uma cópia de seu conteúdo em disco, de forma que quando outros usuários acessarem a mesma página, o acesso torna-se local ao provedor e os dados são transferidos entre o servidor *proxy* e o usuário.

ou transações de usuários, pasta favoritos, consultas do usuário, cliques de mouse e qualquer outro dado gerado pela interação do usuário com a *Web*. As aplicações da mineração de uso da *Web* podem ser classificadas em: aprendizado de perfis de usuários, modelagem em interfaces adaptativas (personalização) e aprendizado de padrões de navegação de usuário. A mineração de uso da *Web* despertou interesse especial no comércio eletrônico, principalmente pela sua necessidade de aprender acerca do comportamento dos clientes, perfis de compra, preferências e padrões de navegação [4].

4.2.3 Formas de Representação de Modelos de Usuários

Uma vez que o modelo de usuário foi adquirido, ele precisa ser representado de uma forma adequada para sua utilização posterior [37].

Nas subseções seguintes são descritas as abordagens mais comuns para a representação dos modelos dos usuários.

4.2.3.1 Representação baseada em Lógica

A representação do conteúdo do modelo de usuários pode ser feita por meio de técnicas baseadas em lógica. A representação baseada em lógica utiliza formalismos lógicos, como cálculo proposicional, cálculo de predicado e lógica modal.

Em geral, o modelo do usuário representa as suposições que o sistema faz a respeito do usuário, considerando um determinado estado do sistema. A representação do modelo do usuário, baseada em lógica, descreve estas suposições por meio do formalismo das expressões da representação do conhecimento baseado em lógica. O modelo do usuário então é considerado como um conjunto de expressões que simbolizam as suposições, consideradas

---

[8] *Cookies* são informações geradas por algumas páginas *Web*. Os *cookies* são gravados no computador do usuário quando este navega pelas páginas. Os *cookies* servem para guardar informações como configurações e preferências de um usuário

verdadeiras, que o sistema tem a respeito do usuário. Desta forma o modelo do usuário baseado em lógica é uma base de conhecimento baseada em lógica [50].

O formalismo da representação baseada em lógica também provê mecanismos de inferência que permitem fazer conclusões implícitas, a partir do conhecimento representado explicitamente. Por exemplo, um modelo de usuário baseado em lógica pode conter expressões que representam crenças do usuário que dizem que a impressora (chamada "lj1"), do escritório dele, é do tipo *laserjet* e que todas as *laserjets* são impressoras a laser. Uma conclusão que pode ser extraída a partir do conteúdo desse modelo do usuário é que o usuário acredita que "lj1" é uma impressora a laser. O processo para inferir conteúdos implícitos, a partir de uma base de conhecimento, é denominado dedução. Então, um modelo de usuário baseado em lógica é uma base de conhecimento dedutiva [50].

Um exemplo de sistema que utiliza cálculo de predicado para representar modelos de usuário é o ANATOM-TUTOR [6], um sistema tutorial de anatomia. Um exemplo de uma regra contida no modelo de usuário é [50]:

> *"any ganglion which receives nerve-impulses from a parasympathic object is itself parasympathic"* [6].

Para a representação da regra acima, utilizando cálculo de predicado, (**Figura 15)**, Beaumont [6] declara que as relações '*belongs-to-class*' e '*sends-nerve-impulses-to*' são usadas [50]:

$$
\begin{aligned}
\forall\, x, y \quad & \text{belongs-to-class } (x, \text{ ganglion}) \\
& \wedge \text{ belongs-to-class } (y, \text{ parasympathic-object}) \\
& \wedge \text{ sends-nerve-impulses-to } (y, x) \\
\rightarrow \quad & \text{belongs-to-class } (x, \text{ parasympathic-object})
\end{aligned}
$$

Figura 15 – Cálculo de predicado representando um modelo de usuário [50]

Um aspecto importante do ANATOM-TUTOR é que regras do domínio são parte do modelo do usuário e não do modelo do domínio do sistema.

Uma das falhas das abordagens baseadas em lógica para a representação do modelo de usuário é a sua habilidade limitada em lidar com incertezas e com as mudanças no modelo do usuário. Em diversas aplicações, as

características relevantes dos usuários mudam com o tempo, assim uma representação do modelo do usuário dever estar apta a lidar com essas mudanças.

4.2.3.2 Representação Probabilística

Para lidar com a incerteza que pode ser inerente à modelagem de usuário, muitos sistemas têm usado métodos numéricos para validar o conteúdo do modelo de usuários. Um exemplo é o Hydrive [45], um sistema que modela a competência de um estudante para localizar os defeitos de um sistema hidráulico de uma aeronave. O Hydrive emprega redes Bayesianas.

Uma rede Bayesiana é um grafo orientado, no qual todas as variáveis são associadas a distribuições de probabilidade discretas. Redes Bayesianas representam distribuições de probabilidade em problemas de tomada de decisão com grande número de variáveis [43].

A probabilidade de distribuição de nós da rede usada no Hydrive representa, explicitamente, variáveis como “conhecimento em eletrônica” ou “conhecimento estratégico“, que são organizados em diferentes níveis de abstração. Tais nós armazenam as probabilidades de uma seqüência observada de atividades do usuário pertencendo a um número fixo de categorias de ações. Para inserir uma informação em um modelo de usuário, o Hydrive cria um nó de rede e o adiciona na rede com a probabilidade do conjunto observado de categoria de ações. Este é o processo de aquisição. As inferências secundárias são feitas por meio da propagação de probabilidades, dos nós de ações interpretados, para os nós de níveis mais altos, como “conhecimento estratégico”. Possíveis inferências são determinadas principalmente por meio do conhecimento do domínio predefinido, representado na estrutura da rede Bayesiana e em probabilidades condicionais associadas com os *links* da rede [37].

As redes Bayesianas, redes neurais, lógica fuzzy, modelos de Markov e redes de Petri estão entre os métodos probabilísticos mais populares para representar a incerteza em modelos de usuários em diferentes domínios.

4.2.3.3 Representação baseada em Estereótipos

Um estereótipo é uma representação das características comuns a um grupo de usuários no domínio da aplicação do sistema, ou seja, os estereótipos são modelos de usuários grupais.

Os estereótipos têm sido utilizados tanto como um método de representação de informações, como um método de aquisição de novas informações para um modelo de usuário [55]. Eles são normalmente utilizados para representar o modelo inicial de um usuário. Os elementos essenciais que compõem um estereótipo são [37]:

- Um *corpo*, que contém as informações sobre um grupo de usuários ao qual o estereótipo é aplicado;
- Um *conjunto de condições de ativação* que representam características-chave que permitem identificar um usuário individual como pertencente a um determinado estereótipo.

As condições de ativação de estereótipos disponíveis são avaliadas e se as condições são satisfeitas, para um determinado usuário, os conteúdos do estereótipo correspondente são integrados como suposições no modelo individual do usuário. Por exemplo, o estereótipo ‘pai’ é aplicado a um determinado modelo individual de usuário se a condição de ativação “o usuário comprou pelo menos dois livros sobre crianças”, for verdadeira, ou seja, as informações contidas no estereótipo ‘pai’ são adicionadas a esse modelo individual [37].

A **Figura 16** mostra a representação, por meio de estereótipos, de um modelo de usuário genérico em um sistema de informação na área educacional. Os estereótipos são categorizados em uma estrutura hierárquica de quatro dimensões diferentes: *nível educacional e profissional*, *linguagem nativa, área acadêmica* e *nível de experiência em informática*. Quando as condições de ativação do estereótipo são satisfeitas, o estereótipo é adicionado ao modelo individual do usuário, sendo que o usuário pode pertencer a diferentes estereótipos, quando categorizado em diferentes dimensões [68].

A grande vantagem do uso de estereótipos é que, com poucas informações sobre o usuário, é possível inferir muitas outras a seu respeito. Por outro lado, a principal desvantagem do uso de estereótipos é a impossibilidade de inclusão de inconsistências no modelo do usuário, pois as informações contidas em um estereótipo se aplicam somente à parte dos indivíduos classificados e não a todos [55].

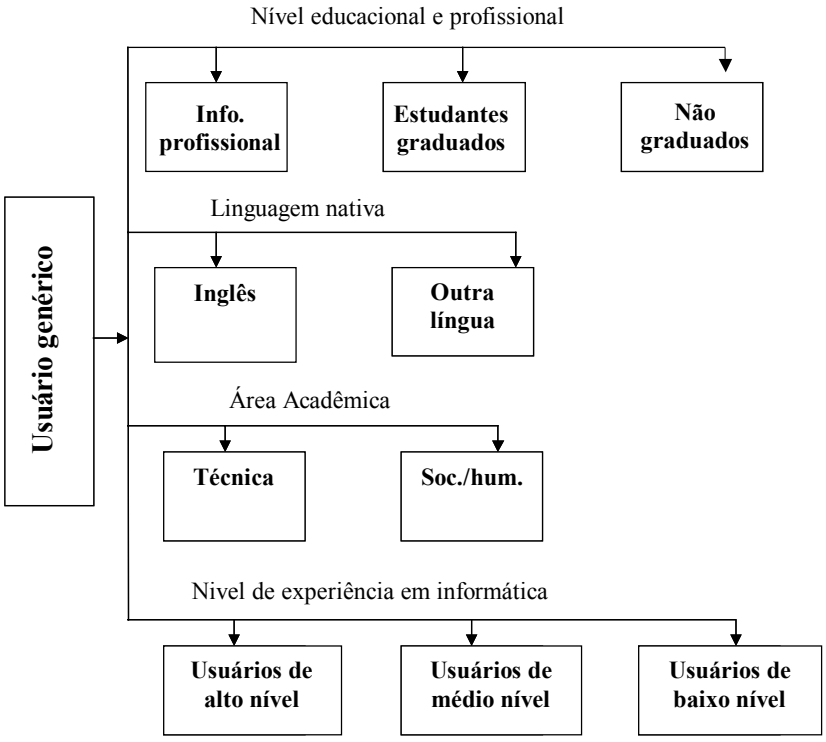

Figura 16 - Uso de estereótipos para representar modelos de usuário [68]

4.2.3.4 Representação Vetorial

O *modelo espaço vetorial* é uma técnica de representação de documentos não estruturados muito utilizada na literatura de recuperação de informação, podendo também ser utilizada para a representação de modelos de usuários. Nesse modelo estatístico, a informação é representada como um vetor num espaço n-dimensional, no qual cada dimensão representa um termo extraído da informação em questão. Um termo pode tanto ser uma única palavra encontrada no texto, como alguma expressão composta de várias palavras. Um fator muito importante para validação desse modelo é a forma como esses termos são extraídos do documento e como é feita a atribuição de pesos (indexação) para cada um deles. Essa

atribuição de pesos deve refletir a importância de cada termo no contexto do documento e também da coleção de documentos, quando existir uma.

No Amalthaea [46], um sistema multiagente personalizado que tenta descobrir informações de várias fontes distribuídas que podem ser de interesse do seu usuário, os modelos de usuários são representados por meio do modelo espaço vetorial. O modelo do usuário no Amalthaea é representado por um vetor de palavras-chave com um peso associado a cada palavra.

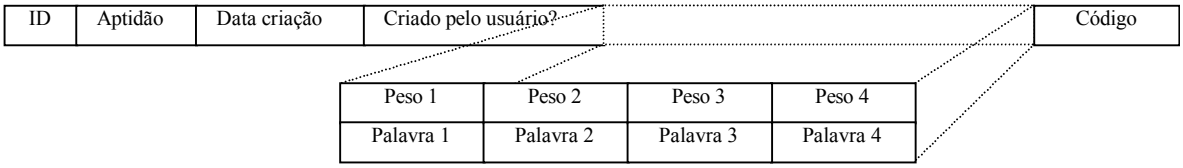

Figura 17 - A técnica *modelo espaço vetorial* representando modelo de usuários

4.2.3.5 Representação baseada em Ontologias

As ontologias são estruturas de representação de conhecimento adequadas para serem utilizadas na representação de modelos de domínio e modelos de usuário, devido à características tais como, notação formal, que providencia uma terminologia clara e não ambígua; flexibilidade, que permite posteriores extensões; e adaptabilidade em diferentes níveis de generalidade/especificidade, facilitando a reutilização [28].

Faria e Ribeiro [19] [20] propõem uma ontologia genérica para a análise de domínio e usuário na Engenharia de Domínio Multiagente, a ONTODUM. A ONTODUM representa o conhecimento de técnicas da análise de requisitos, da análise de domínio e da modelagem de usuários, para a captura e especificação dos requisitos genéricos de uma família de aplicações segundo o paradigma computacional de agentes. Ela guia a captura e a especificação do conhecimento dos conceitos do domínio, das tarefas a serem realizadas no domínio e dos perfis dos usuários que atuam no domínio.

A **Figura 18** ilustra o processo de construção da ONTODUM, que foi inspirado no método proposto para a construção de ontologias [26]. O processo consiste em duas fases: a definição e o projeto da ontologia. Na fase de definição é utilizado o conhecimento de técnicas da análise de requisitos, da análise de domínio

e da modelagem de usuários para gerar uma rede semântica com a representação desses conceitos. Na fase de projeto, a ONTODUM (**Figura 20**) é criada através do mapeamento da rede semântica a uma ontologia baseada em *frames*, representada por uma hierarquia de meta-classes [19].

A **Figura 19** mostra a parte da rede semântica representando as tarefas e os produtos da modelagem de usuários. De acordo com a rede semântica, a *modelagem de usuários* constrói o *modelo de usuários*, através das subtarefas *aquisição do modelo de usuário, representação do modelo de usuário* e *manutenção do modelo de usuário*. *A aquisição do modelo de usuário* usa *técnicas de aquisição*. As *técnicas de aquisição* são classificadas em *técnicas de aquisição explícitas* e *técnicas de aquisição implícitas*. As *técnicas de aquisição explícitas* usam *questionários*, que definem *estereótipos* e classificam os *usuários*. As *técnicas de aquisição implícitas* são baseadas em *técnicas de aprendizagem de máquina*. O *modelo de usuário* é o produto gerado, que modela um *usuário,* um tipo de *entidade externa*. *Entidade externa* é um tipo de *conceito do domínio.* O *modelo de usuário* representa o *usuário* com as suas *preferências*, *habilidades físicas, habilidades cognitivas, objetivos particulares* e *informação pessoal* [19].

Figura 18 - Processo de construção da ONTODUM [19]

A fase de projeto da ontologia foi feita utilizando o Protégé [51], uma ferramenta construída pelo departamento de Informática Médica da Universidade de Standford. A ferramenta Protégé é um ambiente de edição de bases de conhecimento que busca a interoperabilidade com outros sistemas de representação do conhecimento. O Protégé é uma ferramenta é extensível e de aquisição de conhecimento, fácil de configurar e usar.

Uma ontologia no Protégé consiste em:

- Classes são conceitos do domínio abordado que constituem uma hierarquia taxonômica;
- *Slots* que descrevem propriedades de classes e instâncias;

- Facetas que descrevem propriedades de *slots* e permitem a especificação de restrições (*constraints*) nos valores dos *slots*;

No projeto da ontologia, os conceitos e os relacionamentos da rede semântica são mapeados a ONTODUM. Os nós são mapeados para meta-classes. Os nós relacionados por um link do tipo "*a kind of*" são mapeados em uma hierarquia de subclasses e superclasses. Outros *links* são mapeados para *slots* das correspondentes meta-classes. Cada *slot* é associado com facetas apropriadas, como tipo e cardinalidade [19].

Como exemplo de aplicação da ONTODUM, foram construídos modelos de usuários da área jurídica. Foi usada a aquisição explícita, na qual primeiramente é aplicado um questionário para colher informações do usuário. Após o recolhimento das informações, o usuário é inserido nos estereótipos: *criminal*, *cível* e *trabalhista*, que classifica a entidade externa *advogado* em: *advogado criminalista*, *advogado cível* e *advogado trabalhista*. São criadas instâncias da habilidade cognitiva (*expressão oral*, *expressão escrita*, *discernimento para interpretar as normas*, *prudência para apreciar os fatos e capacidade de argumentação*), da preferência (*conhecimento legal trabalhista*, *conhecimento legal cível*, *conhecimento constitucional* e *conhecimento legal criminal*), do objetivo do usuário (*manter seu conhecimento legal atualizado*) e entidade externa (*advogado criminalista*, *advogado cível* e *advogado trabalhista*). Foram criados quatro modelos de usuário para o acesso à informação jurídica: *modelo de usuário do advogado*, *modelo de usuário do advogado criminalista*, *modelo de usuário do advogado cível* e *o modelo de usuário do advogado trabalhista* [19].

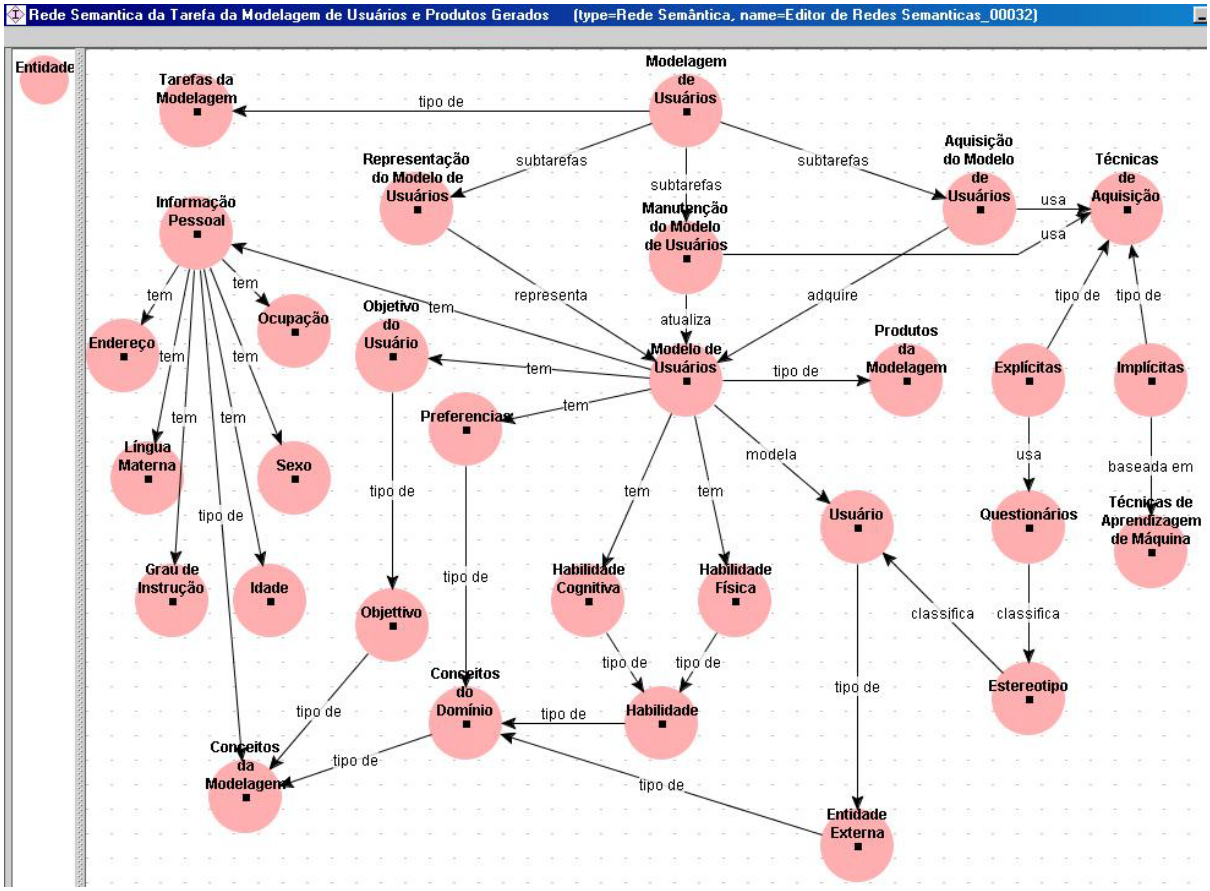

Figura 19 - Rede Semântica da modelagem de usuários [19]

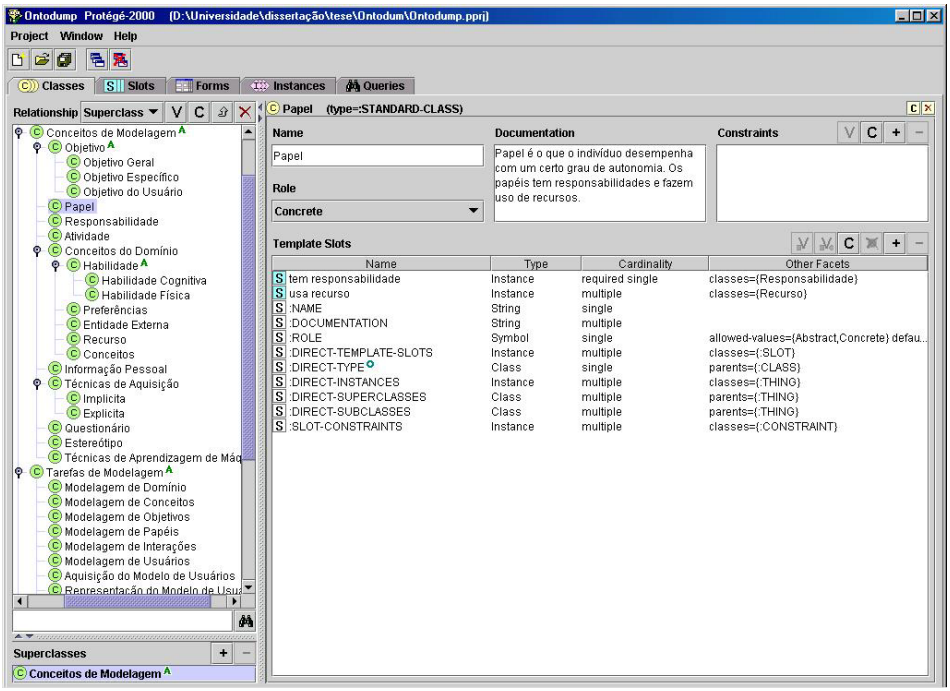

Figura 20 - Hierarquia de meta-classes da ONTODUM e a meta-classe Papel [19]

Na construção do modelo de usuário do advogado, primeiramente é adicionada a instância da entidade externa *usuário advogado*. Em seguida, é

adicionado o objetivo do usuário *manter seu conhecimento legal atualizado* e vinculada à entidade externa *usuário advogado*. São adicionadas instâncias da habilidade cognitiva *expressão oral*, *expressão escrita*, *discernimento para interpretar as normas e prudência para apreciar os fatos e capacidade de argumentação* e vinculada à entidade externa *usuário advogado*. Por último, são adicionadas instâncias da preferência *conhecimento legal trabalhista*, *conhecimento legal cível*, *conhecimento constitucional* e *conhecimento legal criminal* e vinculados à entidade externa *usuário advogado*. A **Figura 21** ilustra o modelo de *usuário advogado* [19].

Na construção do modelo de usuário do advogado criminalista, primeiro é adicionado a instância da entidade externa usuário *advogado criminalista*. Em seguida, é adicionado o objetivo do usuário *manter seu conhecimento legal atualizado* e vinculada à entidade externa *usuário advogado criminalista*. São adicionadas instâncias da habilidade cognitiva *expressão oral*, *expressão escrita*, *discernimento para interpretar as normas e prudência para apreciar os fatos e capacidade de argumentação* e vinculada à entidade externa *usuário advogado criminalista*. Por último, são adicionadas instâncias da preferência *conhecimento constitucional* e *conhecimento legal criminal* e vinculados à entidade externa *usuário advogado criminalista*. A **Figura 23** ilustra o modelo de usuário *advogado criminalista*. O mesmo processo foi executado para a construção dos modelos de usuários *advogado cível* (**Figura 22**) e *advogado trabalhista* (**Figura 24**) [19].

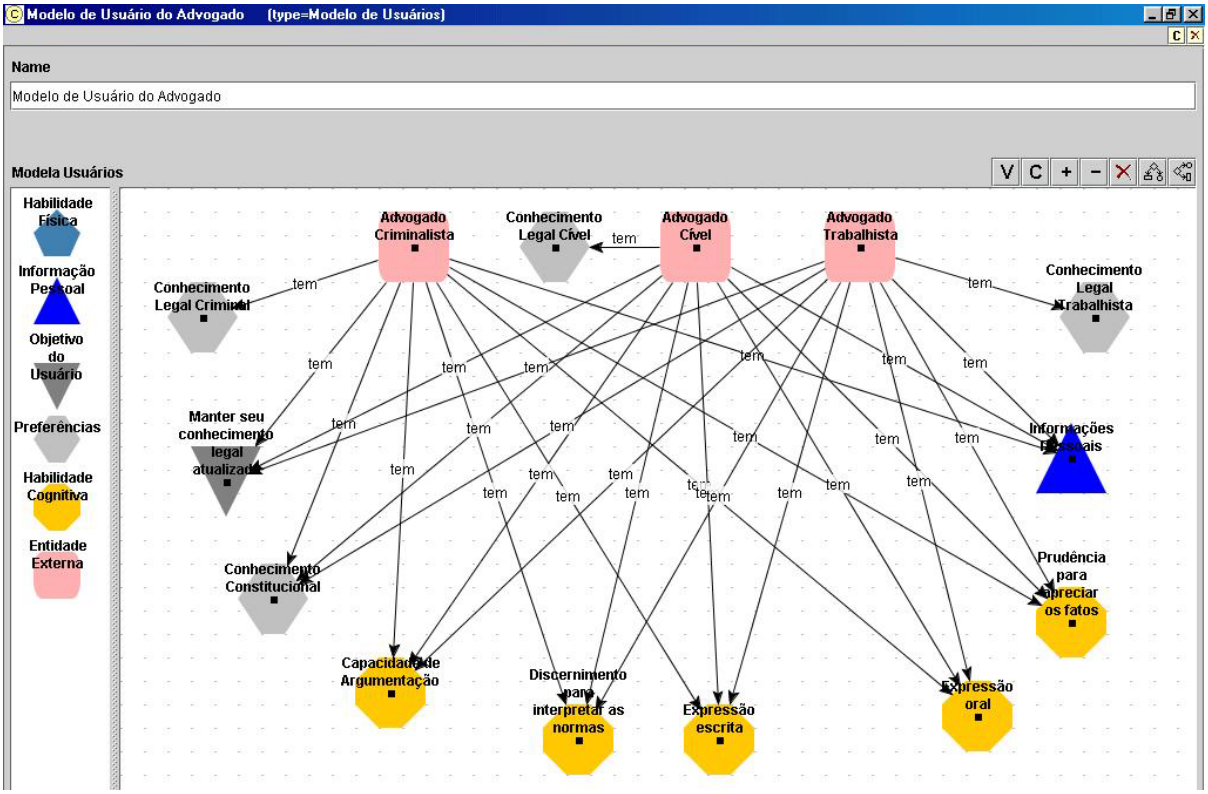

Figura 21 - Modelo de usuário *advogado* [19]

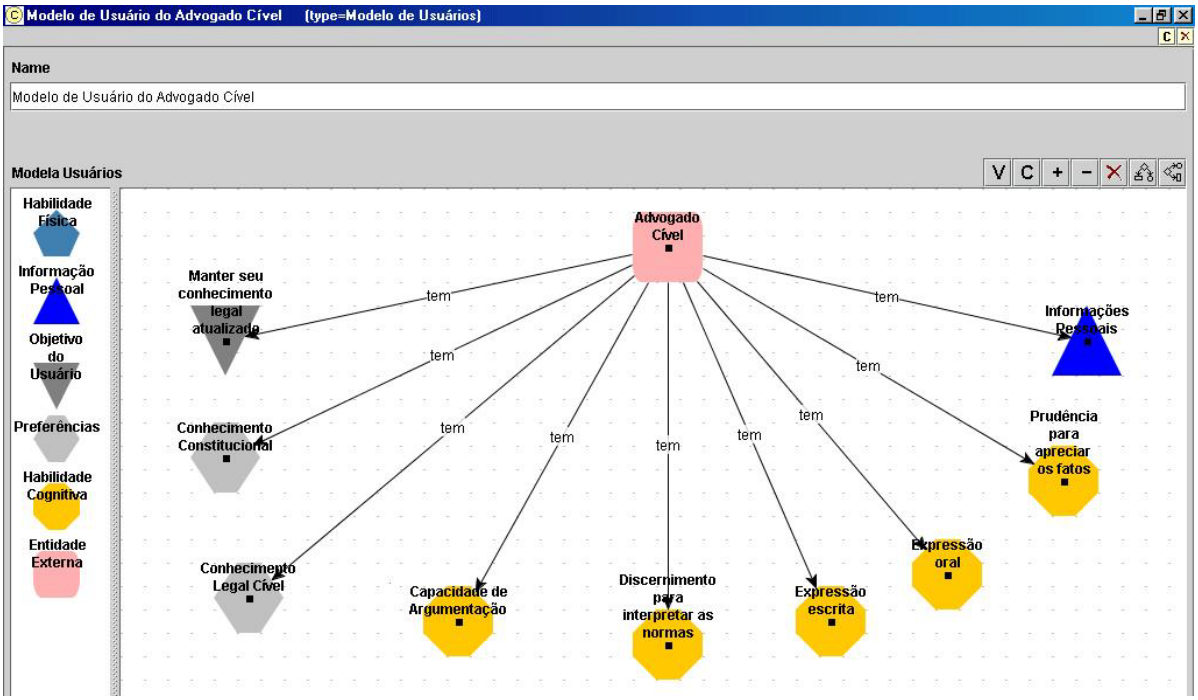

Figura 22 - Modelo de usuário *advogado cível* [19]

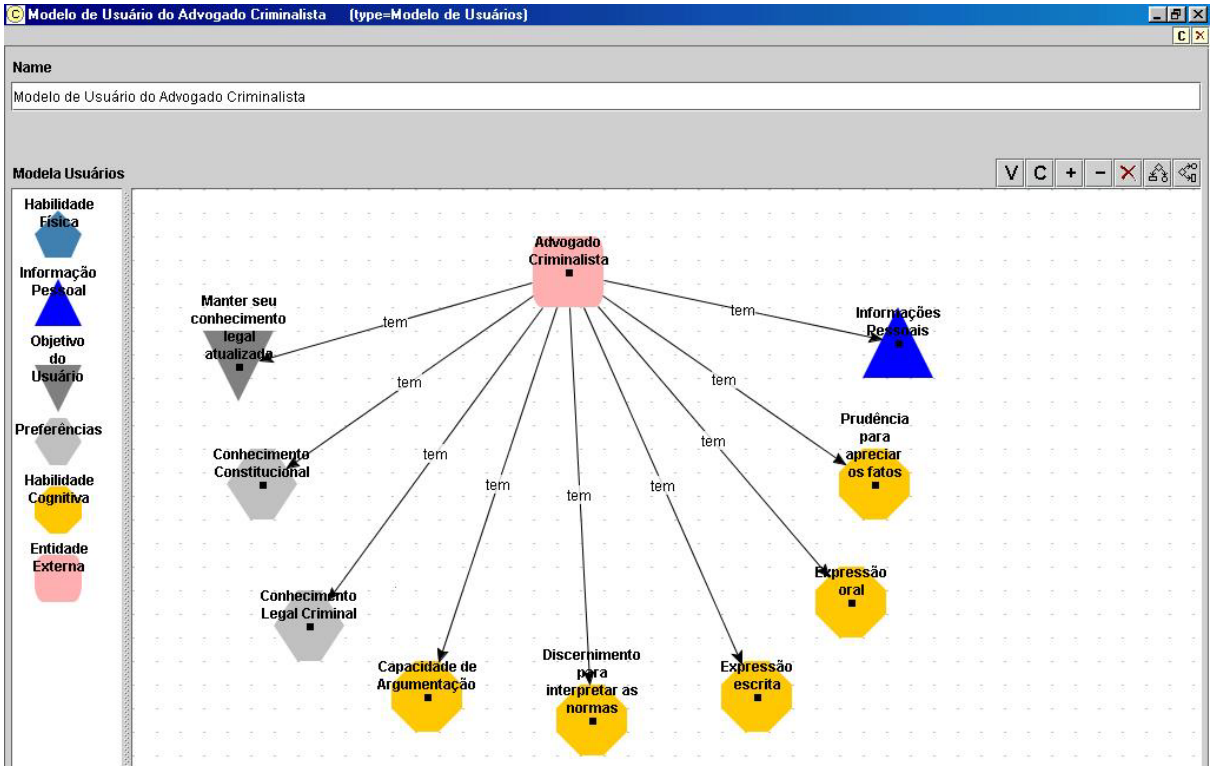

Figura 23 - Modelo de usuário *advogado criminalista* [19]

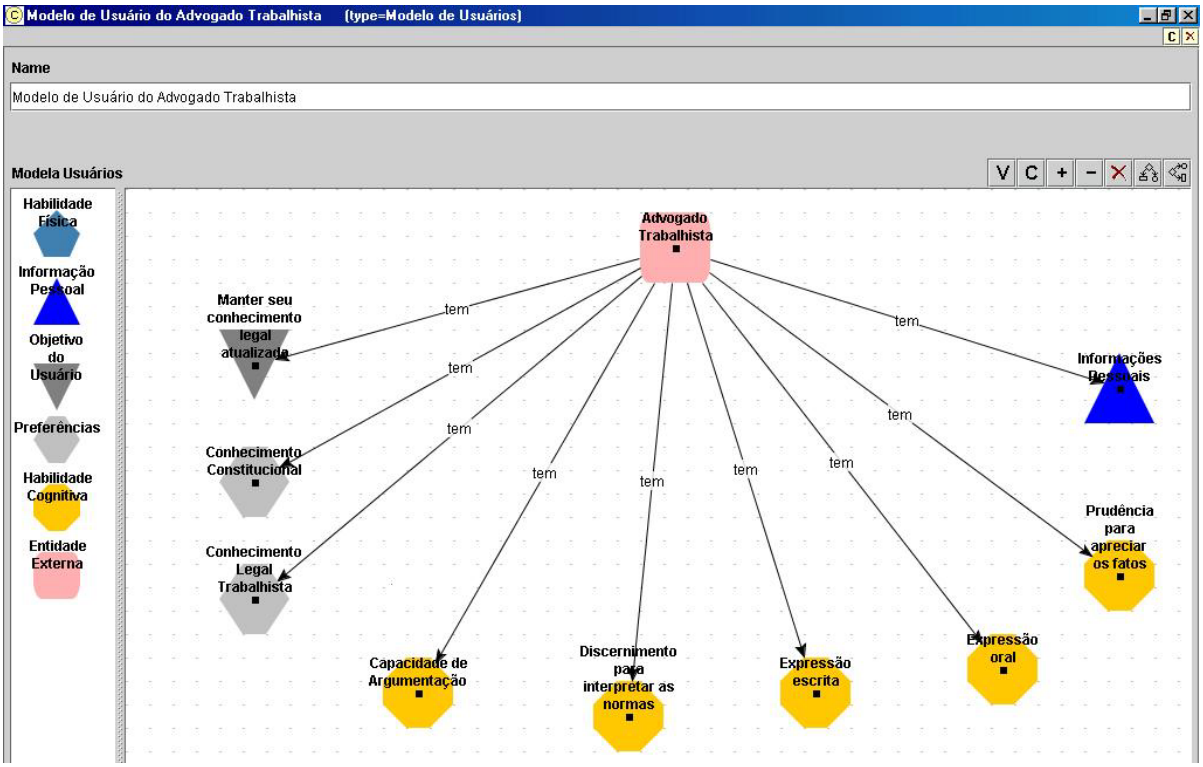

Figura 24 - Modelo de usuário *advogado trabalhista* [19]

4.2.4 Técnicas para realizar a Manutenção dos Modelos de Usuários

Todo o processo de modelagem de usuários necessita manter dinamicamente as informações sobre os usuários em um modelo. Normalmente, para realizar essa tarefa, existe um módulo de manutenção responsável por integrar novas informações no modelo de usuário e atualiza as já existentes,ou seja, é responsável por manter a consistência desse modelo.

Existem várias técnicas computacionais que podem ser utilizadas para realizar a manutenção das informações mantidas pelo processo de modelagem de usuários, dentre elas podemos destacar: lógica fuzzy (teoria de conjuntos difusos), teoria de probabilidades e algoritmos genéticos.

5. APSUMAS - UM SISTEMA DE PADRÕES BASEADOS EM AGENTES PARA A MODELAGEM DE USUÁRIOS E ADAPTAÇÃO DE SISTEMAS

Este capítulo descreve o APSUMAS (*Agent-based Pattern System for User Modeling and Adaptation of Systems*), um sistema de padrões baseados em agentes para o projeto de sistemas adaptativos/adaptáveis que utilizam a modelagem de usuários para oferecer serviços personalizados a usuários e/ou grupos de usuários.

A metodologia utilizada para a identificação dos padrões é apresentada na seção 2.4.

A **Figura 25** mostra, em uma rede semântica, o relacionamento entre os padrões que compõem o APSUMAS. O padrão conceitual *Sociedade multiagente para sistemas adaptativos/adaptáveis* descreve os principais conceitos sobre modelagem de usuários e sistemas adaptativos/adaptáveis e justifica a escolha da abordagem de agentes para o desenvolvimento de tais sistemas. A organização dos agentes, no sistema multiagente, é descrita pelo padrão arquitetural *Camadas multiagente para sistemas adaptativos/adaptáveis* que distribui os agentes em camadas conforme as tarefas que serão executadas pelos mesmos. O detalhamento dos agentes que compõem as camadas é descrito pelos padrões de projeto *Interface, Modelagem e Adaptação*. Os padrões de projeto Interface, Modelagem e Adaptação, por sua vez, podem ser derivados do padrão *Deliberativo* ou do padrão *Reativo* [53].

Nos padrões de projeto do APSUMAS, a especificação de protocolos de interação entre agentes, bem como a definição das classes e métodos que compõem a estrutura interna dos agentes, não são incluídos na solução dos padrões. Esse detalhamento será feito como trabalho futuro.

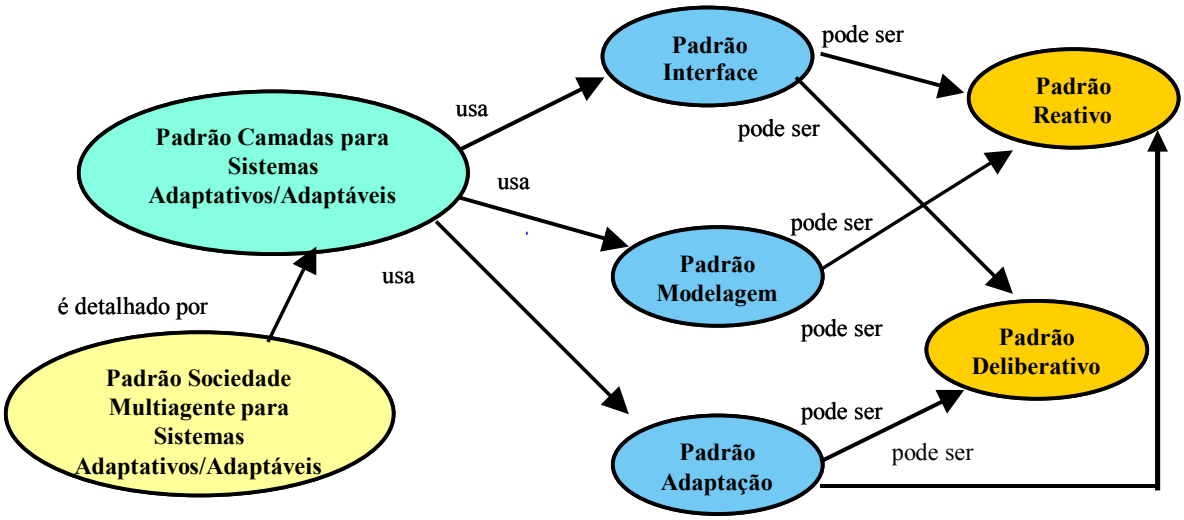

Figura 25 - Rede semântica dos padrões propostos

A **Tabela 5** apresenta, de forma sucinta, os padrões propostos, classificando-os segundo o tipo de padrão (como apresentado na seção 2.1) e segundo a dependência ou independência do domínio da aplicação. Os padrões dependentes do domínio são específicos para o projeto de aplicações adaptativas baseadas em agentes, já os independentes do domínio podem ser utilizadas no projeto de desenvolvimento de aplicações multiagente em geral.

Padrões baseados em Agentes				
Classificação		**Problema**	**Solução**	**Padrão**
Padrões Dependentes de Domínio	**Padrão Conceitual**	Como construir e organizar sistemas que modelam características e preferências de usuários de forma a adaptar-se às suas necessidades?	Projetar e implementar sistemas que modelam características e preferências de usuários e produzem adaptação por meio da abordagem de agentes.	Sociedade multiagente para Sistemas Adaptativos/adaptáveis
	Padrão Arquitetural	Como estruturar uma comunidade de agentes para construir modelos de usuários e se adaptar a diferentes tipos de usuários ou grupos de usuários?	O padrão agrupa os agentes em três camadas, de acordo com as principais funcionalidades exercidas por sistemas adaptativos/adaptáveis: camada de interface; camada de modelagem e camada de adaptação,	Camadas Multiagente para Sistemas Adaptativos/Adaptáveis
	Padrões de projeto	Como gerenciar a interação entre uma aplicação de software e seus usuários de forma personalizada?	A solução envolve a criação de um agente que servirá como interface entre o usuário e a aplicação.	Interface
		Como criar e manter modelos de usuários em sistemas adaptativos?	A solução envolve a criação de um agente responsável por construir e manter modelos de usuários.	Modelagem
		Como construir modelos de adaptação para que a aplicação possa adaptar-se às necessidades dos seus usuários?	A solução envolve a criação de um agente cujo principal papel é suprir modelos de adaptação de acordo com os modelos de usuários.	Adaptação
Padrões Independentes de Domínio		Como projetar um agente para que possa raciocinar sobre um problema de forma que o possibilite atingir metas pró-ativamente?	O agente deve ser do tipo deliberativo, ou seja, ele deve possuir modelos simbólicos internos de do ambiente no qual ele está inserido e de si mesmo e raciocinar sobre esses modelos de forma que o possibilite criar um plano de ações para atingir suas metas.	Deliberativo
		Como projetar um agente para apenas reagir a estímulos do ambiente no qual está inserido?	O agente deve ser do tipo reativo, ou seja, ele não tem um modelo interno do ambiente no qual está inserido, o agente deve agir usando um comportamento estímulo/resposta.	Reativo

Tabela 5 - Padrões baseados em agentes para o desenvolvimento de sistemas adaptativos/adaptáveis

5.1 Padrão Sociedade Multiagente para Sistemas Adaptativos/Adaptáveis

Contexto

Esse padrão é aplicável quando há necessidade de projetar sistemas que ofereçam serviços personalizados aos seus usuários, levando em consideração as diferentes características e necessidades desses usuários, ou seja, os chamados sistemas adaptativos e/ou adaptáveis.

Os sistemas nos quais o usuário pode iniciar, propor, selecionar e produzir a adaptação ou também deixar que o sistema execute algumas dessas funções, são chamados de *sistemas adaptáveis*, já os sistemas que executam todas essas funções, de forma automática, são chamados de *sistemas adaptativos.*

Os sistemas adaptativos têm estreita relação com a tecnologia de modelagem de usuários, pois os modelos de usuários contêm informações de todos aspectos relevantes aos usuários que são utilizados no processo de adaptação de uma aplicação de software.

As principais áreas de aplicação dos sistemas adaptativos e adaptáveis são:

- Comércio eletrônico [62];
- Recuperação de informação [46] [47];
- Interfaces adaptativas [41];
- Turismo [24];
- Educação [6] [10] [55].

Problema

Como construir e organizar sistemas que modelam características e preferências de usuários de forma a adaptar-se às suas necessidades?

Forças

As características de autonomia, habilidade social, reatividade, pró-atividade e capacidade de aprendizagem, próprias dos agentes, provêm mecanismos apropriados para que os sistemas adaptativos/adaptáveis atinjam seus objetivos. Freqüentemente, os agentes estão aptos a adaptar-se aos seus ambientes e exibem algum grau de inteligência, apesar dessas características não serem obrigatórias.

As características dos agentes podem ser associadas aos principais objetivos dos sistemas adaptativos/adaptáveis, refletindo assim, as razões para a utilização da tecnologia de agentes em tais sistemas, como pode ser visto na **Tabela 6**.

Objetivos dos sistemas adaptativos/adaptáveis	Características dos agentes
Melhorar a eficiência e eficácia da interação (facilitar e acelerar interações)	**Reatividade** – por meio da interação com o ambiente, os agentes reagem rapidamente a mudanças no ambiente e apresentam ao usuário o que ele quer ver, de forma a refletir o seu perfil.
Tornar os sistemas complexos mais usáveis	**Habilidade social**- os agentes cooperam e interagem com outros agentes para resolver tarefas complexas tornando os sistemas mais simples do ponto de vista do usuário
Ajudar o usuário a encontrar informação	**Autonomia** e **pró-atividade**– o usuário pode delegar tarefas (objetivos) para que os agentes de software as executem autonomamente.
Prever o comportamento futuro do usuário	**Capacidade de aprendizagem**- os agentes podem aprender com o comportamento do usuário e fazer inferências sobre o seu comportamento esperado.

Tabela 6 - Objetivos dos sistemas adaptativos e as características dos agentes

Solução

Modelar a aplicação como uma sociedade de agentes. Seriam usados agentes especializados para cada usuário ou grupos de usuários, como agente de interface, agente de modelagem e agente de adaptação.

O processo de modelagem de usuários e o processo de personalização de aplicações pode ser dividido em três tarefas maiores que podem ser executadas por diferentes componentes do sistema [49]:

- *A aquisição* consiste na coleta de informações sobre usuários, como características, preferências, necessidades.

- *A representação* consiste na representação dos conteúdos dos modelos de usuários e modelos de adaptação (ver seção 4.1), apropriadamente, em um sistema formal, permitindo o seu acesso e posterior processamento, bem como fazer a manutenção desses modelos.

- *A produção* consiste na geração da adaptação do conteúdo e da apresentação da aplicação, baseados nos modelos de usuários, nos modelos de adaptação e nos modelos do domínio(ver seção 4.1).

As tarefas de aquisição e produção podem ser atribuídas ao agente de interface, a tarefa de representação de modelos de usuários pode ser atribuída ao agente de modelagem, já o agente de adaptação pode ser responsável pela representação de modelos de adaptação, como pode ser visualizado na **Figura 26**.

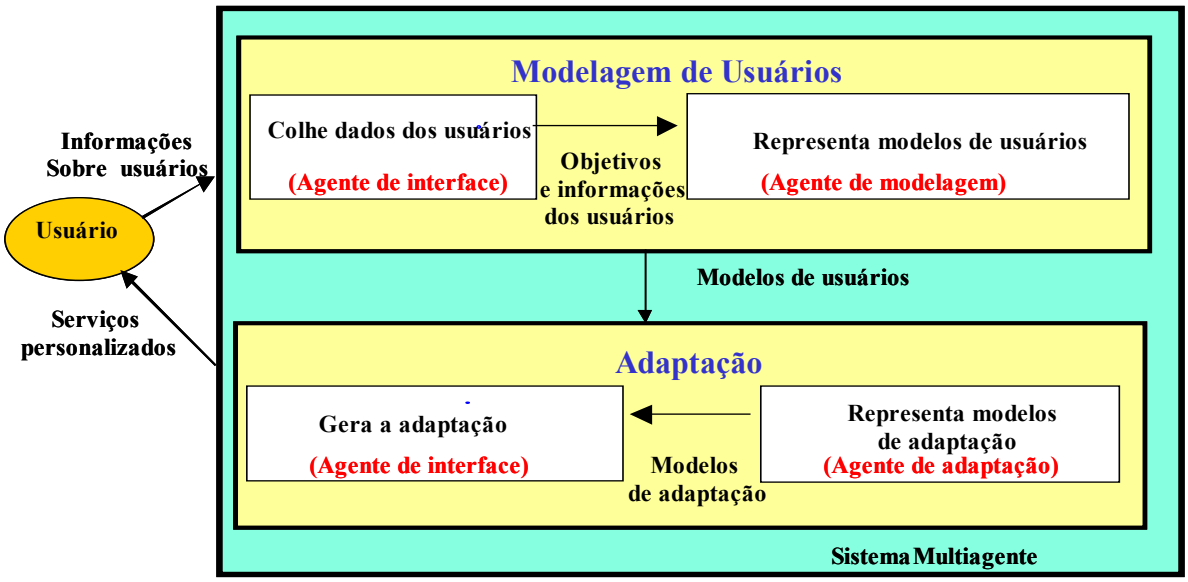

Figura 26 - O processo de modelagem de usuários e adaptação de sistemas utilizando agentes

Durante o processo de modelagem de usuários, de acordo com a **Figura 26**, o agente de interface colhe dados sobre o usuário; esses dados são processados e representados em um modelo de usuário pelo agente de modelagem. Na execução do processo de adaptação, o agente de adaptação produz modelos de

adaptação baseados no modelo de usuário, esses modelos são utilizados pelo agente de interface para personalizar a aplicação.

Padrões relacionados

A solução arquitetural, descrita neste sistema de padrões, para estruturar os agentes dos sistemas adaptativos/adaptáveis é descrita no padrão Camadas multiagente para sistemas adaptativos/adaptáveis.

Usos conhecidos

Baldassin [5], Bylund [13], Diniz [16], Moukas [46], propõem o uso de agentes para oferecer serviços personalizados aos usuários, distribuindo tarefas entre esses agentes.

5.2 Padrão Camadas para Sistemas Adaptativos/Adaptáveis

Contexto

Nos sistemas adaptativos/adaptáveis, baseados na tecnologia de agentes, é preciso projetar soluções para organizar os agentes de maneira que possam satisfazer as necessidades e preferências de seus usuários.

Problema

Como estruturar um sistema multiagente apto a adaptar-se às necessidades de diferentes tipos de usuários ou grupos de usuários?

Forças

A organização dos agentes em camadas, segundo a similaridade de responsabilidades atribuídas aos componentes do sistema, tem todas as vantagens (flexibilidade, facilidade de manutenção, de entendimento e de extensão) descritas pelo padrão *Camadas* [29] [59].

Solução

A solução (**Figura 27**), baseada no uso de técnicas de adaptação e modelagem de usuários, envolve a definição de um critério para a divisão das responsabilidades entre agentes distribuídos em camadas, de acordo com o padrão *Camadas* descrito por Silva Júnior [29] [59], uma extensão do padrão *Camada* proposto por Buschmann [12]. Os sistemas adaptativos/adaptáveis que usam a modelagem de usuários são compostos basicamente por uma camada de interface, uma camada de modelagem e uma camada de adaptação, sendo que podem ser adicionadas novas camadas de acordo com o tipo de serviço que o sistema adaptativo/adaptável irá prover. A **Figura 28** ilustra as principais interações entre os agentes das camadas em um diagrama de colaboração [48] .

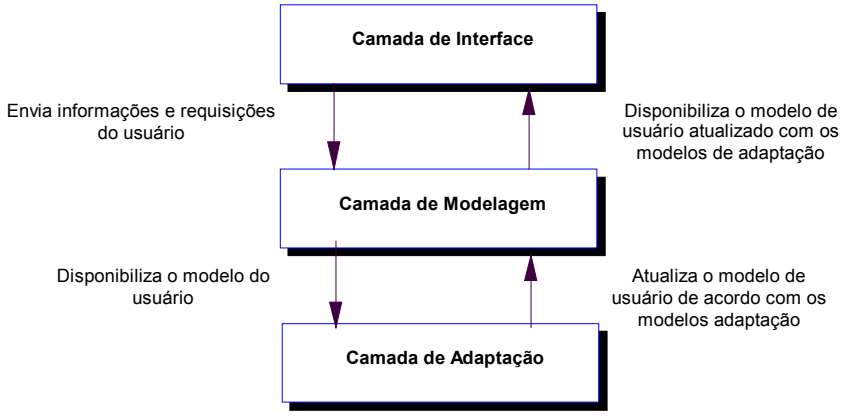

Figura 27 - Estrutura do padrão Camadas para sistemas adaptativos/adaptáveis

Um grupo de agentes está associado a cada camada. Cada camada é estruturada como segue.

Camada de interface

A camada de interface consiste em um grupo de agentes de interface em que cada agente está associado a um usuário ou grupo de usuários com necessidades similares para auxiliá-lo(s) em suas tarefas. As principais responsabilidades dos agentes dessa camada são:

- Apresentar uma interface inicial, caso o usuário ainda não possua um modelo de usuário, ou personalizada, baseando-se no modelo de usuário corrente;
- Receber requisições dos usuários;
- Colher informações sobre usuários, tais como características dos usuários, comportamento durante o uso do computador ou do sistema, por meio do monitoramento do uso do computador ou requisitando essas informações diretamente ao usuário;
- Caso seja adotado um sistema de retroalimentação, fazer perguntas a respeito dos resultados para possíveis melhoras na eficácia do sistema.
- Apresentar informações aos usuários de forma personalizada (respostas e explicações).

Camada de modelagem

A camada de modelagem consiste em um grupo de agentes de modelagem. Cada agente de modelagem provê serviços para um agente de interface. As principais responsabilidades dos agentes dessa camada são:

- Processar a informação provida pelo usuário, por meio de um agente de interface e atualizar e/ou criar modelos de usuários e disponibilizá-los para o agente de adaptação correspondente;
- Formular metas e/ou inferir novas suposições sobre usuários ou grupo de usuários com base em informações prévias contidas nos modelos de usuários;

Camada de adaptação

A camada de adaptação consiste em um conjunto de agentes de adaptação. Cada agente de adaptação provê serviços para o agente de modelagem. As principais responsabilidades dos agentes dessa camada são as seguintes:

- Construir, representar e manter modelos de adaptação (ver seção 4.1), de acordo com os modelos de usuários;
- Atualizar os modelos de usuários de acordo com os modelos de adaptação.

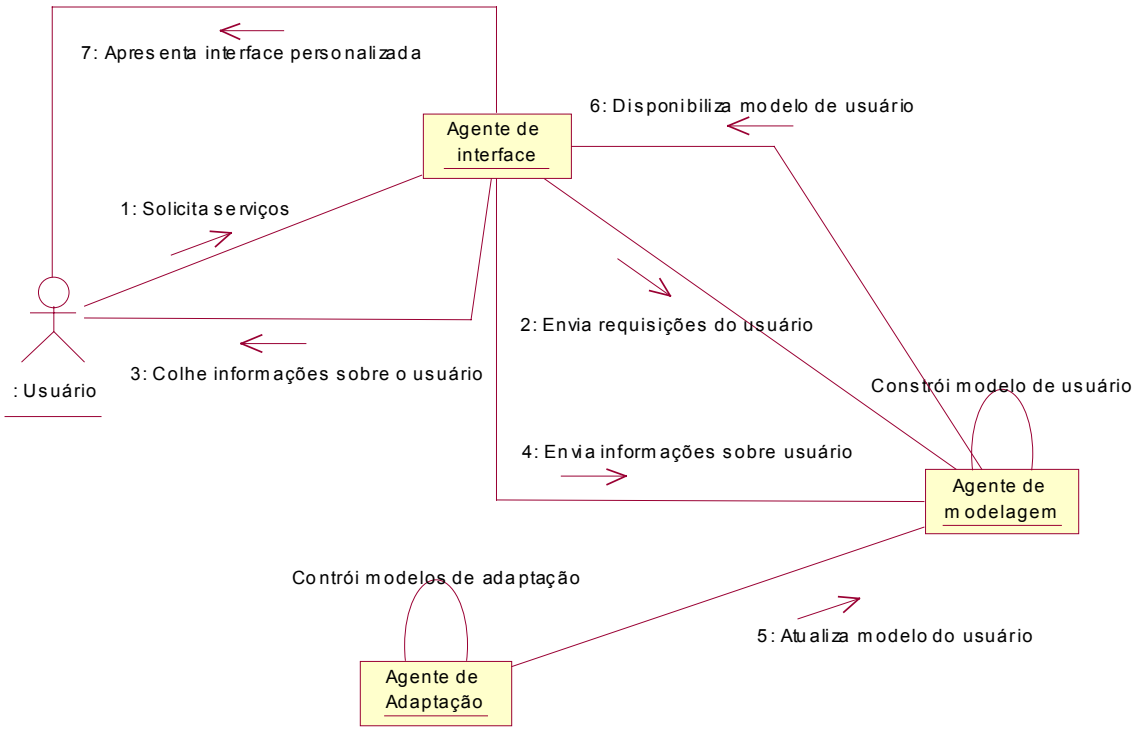

Figura 28 - Diagrama colaboração dos agentes do padrão Camadas para Sistemas Adaptativos/Adaptáveis

Padrões relacionados

- Para o projeto dos agentes da camada de interface, neste sistema de padrões, utiliza-se o padrão Interface.
- Para o projeto dos agentes da camada de modelagem, neste sistema de padrões, utiliza-se o padrão Modelagem.
- Para o projeto dos agentes da camada de adaptação, neste sistema de padrões, utiliza-se o padrão Adaptação.

Usos conhecidos

A organização em camadas é típica das arquiteturas multiagente para filtragem e recuperação de informação, tais como as arquiteturas RETSINA [58], Amalthaea [46] e ABARFI [16], bem como as arquiteturas de sistemas *Web* adaptativos, como a arquitetura genérica descrita por Sharma [57]. Este padrão é inspirado nas soluções arquiteturais propostas por tais sistemas.

5.3 Padrão Interface

Contexto

Nos sistemas adaptativos/adaptáveis, baseados na tecnologia de agentes, é preciso projetar soluções para a interação do sistema com os usuários.

Problema

Como gerenciar a interação entre um usuário ou grupo de usuários e uma aplicação de software, de forma personalizada?

Forças

- Em sistemas multiagente adaptativos/adaptáveis é necessário um agente para interagir com o usuário, de forma que os outros agentes da aplicação tornem-se transparentes aos usuários.
- Interfaces adaptativas melhoram a interação do usuário com o sistema.
- Adquirir informações sobre os usuários do sistema é fundamental para a construção de modelos de usuários e modelos de adaptação.

Solução

A solução envolve a criação de um agente que servirá como interface entre o usuário e a aplicação. O agente de interface está associado a um usuário ou grupo de usuários com necessidades similares.

As principais responsabilidades do agente de interface são:

- Apresentar uma interface inicial, caso o usuário ainda não possua um modelo de usuário, ou personalizada, baseando-se no modelo de usuário corrente;
- Receber requisições dos usuários;

- Colher informações sobre usuários, comunicando-se diretamente com ele e/ou observando o seu comportamento;
- Caso seja adotado um sistema de retroalimentação, fazer perguntas a respeito dos resultados para possíveis melhoras na eficácia do sistema;
- Fornecer as informações colhidas dos usuários (dados e requisições) para o agente de modelagem;
- Apresentar informações aos usuários de forma personalizada (respostas e explicações).

A **Figura 29** mostra as principais interações do agente de interface durante a execução de suas responsabilidades.

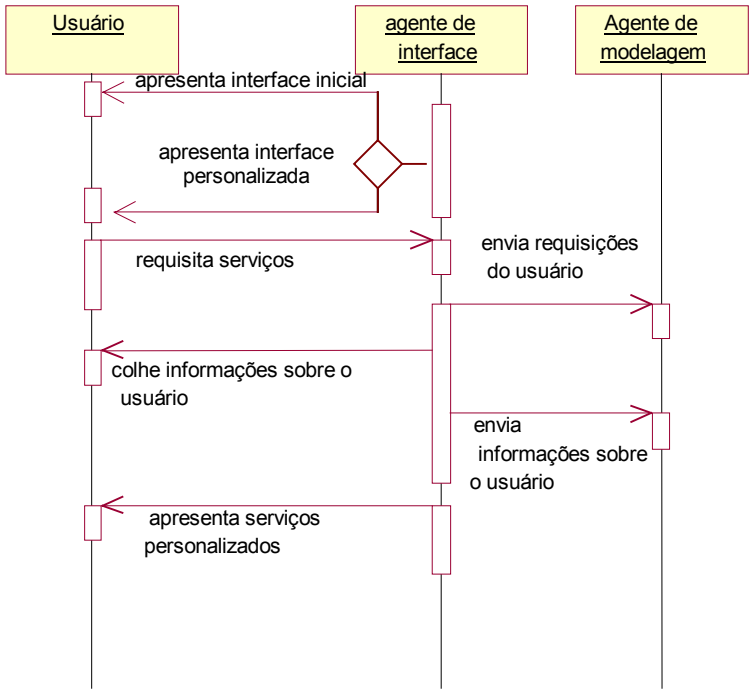

Figura 29 – Diagrama de interações do padrão Interface

No processo de modelagem de usuários e de personalização de aplicações, o agente de interface é responsável pelas tarefas: *aquisição,* que consiste na identificação de informações sobre usuários como, características,

necessidades e preferências; e *geração* da adaptação da interface do sistema, que consiste em personalizar a interface de acordo com os modelos de usuários.

A aquisição pode ser feita de forma direta (explícita) e/ou de forma indireta (implícita), de acordo com o descrito na seção 4.2.2.

Uma técnica de *aquisição explícita* óbvia é deixar que o próprio usuário forneça as informações necessárias para a aplicação. A provisão de informações pelo usuário pode ser feita por meio de questionários feitos pelo sistema.

A classificação em estereótipos é também uma técnica bastante utilizada na *aquisição direta*. O sistema poderá exibir um formulário para o usuário preencher e partir dessas informações classificá-los em estereótipos pré-definidos pelo sistema e posteriormente, podem ser feitas predições baseadas nesses estereótipos.

Dentre as técnicas de *aquisição indireta*, podemos citar: regras de aquisição, reconhecimento de planos do usuário, aprendizagem de máquina e mineração de uso.

Para a escolha das informações a serem adquiridas sobre os usuários, deve-se levar em conta o tipo da aplicação e o objetivo da mesma. Por exemplo, as informações que são relevantes para o comércio eletrônico são diferentes das relevantes para um sistema tutor inteligente. Dentre as categorias de informações sobre os usuários (ver seção 4.2.1) que têm sido a base para adaptação de sistemas, podemos citar: *informações pessoais*, *conhecimento*, *capacidades*, *habilidades*, *interesses*, *preferências*, *metas*, *planos* e *dados de uso*.

Na execução da tarefa de geração, o agente de interface baseia-se nos modelos de usuários que foram construídos e atualizados com base nas informações dos usuários e nos modelos de adaptação.

Padrões relacionados

O agente de interface pode ser definido como um agente do tipo reativo ou deliberativo. O agente reativo tem suas características descritas, neste sistema de padrões, pelo padrão Reativo. O agente deliberativo tem suas características descritas, neste sistema de padrões, pelo padrão Deliberativo.

As informações dos usuários, adquiridas pelo agente de interface, são representadas e mantidas pelo agente de modelagem, que é descrito, neste sistema de padrões, pelo padrão Modelagem.

Kendall [36] propõe um padrão alternativo, chamado *Agente de interface,* que difere do padrão proposto neste trabalho principalmente no seguinte aspecto: o padrão de interface definido por Kendall é utilizado não só para colher informações dos usuários e apresentar uma interface personalizada, mas também é responsável por construir e manter modelos de usuários.

5.4 Padrão Modelagem

Contexto

Nos sistemas adaptativos/adaptáveis, baseados na tecnologia de agentes, é preciso projetar soluções para representar os usuários por meio da construção de modelos de usuários que especificam suas características, suas preferências e os seus interesses.

Problema

Como criar e manter modelos de usuários que serão utilizados como referência para que a aplicação ofereça serviços personalizados aos seus usuários?

Forças

Os usuários diferem em seus níveis de conhecimento, necessidades e preferências. O objetivo da adaptabilidade é oferecer serviços sob medida aos usuários, por exemplo, adaptando o estilo de apresentação durante a interação com o usuário. Qualquer abordagem para oferecer serviços personalizados, envolve a criação e manutenção de modelos de usuários que devem estar aptos a refletir as mudanças nas características desses usuários no decorrer do tempo.

A modelagem de usuários é necessária no projeto de sistemas adaptativos, pois por meio dela é possível sintetizar informações relevantes sobre os usuários que o sistema leva em consideração para oferecer serviços personalizados.

Solução

A solução envolve a criação de um agente de modelagem responsável por construir e manter modelos de usuários com base nas informações fornecidas por um agente de interface, por meio da interação com o usuário.

O agente de modelagem recebe informações sobre os usuários por meio do agente de interface; faz uma representação formal dessas informações; faz inferências de novas hipóteses sobre os usuários, com base nas informações armazenadas previamente nos modelos de usuários.

Algumas das técnicas mais comuns para a representação de modelos de usuários, de acordo com o descrito na seção 4.2.3, são: redes bayesianas, representação baseada em lógica, estereótipos, lógica fuzzy, redes neurais, redes de Petri, representação vetorial e ontologias.

Além da representação dos modelos de usuários, o agente de modelagem é responsável pela manutenção da consistência dos modelos, verificando as novas informações fornecidas pelo agente de interface e/ou pelo agente de adaptação e comparando-as com informações anteriores. Dentre as técnicas mais comuns para fazer a manutenção de modelos de usuários podemos destacar a lógica fuzzy (teoria de conjuntos difusos), a teoria de probabilidades e os algoritmos genéticos, de acordo com o descrito na seção 4.2.4.

Padrões relacionados

- O agente de modelagem pode ser definido como um agente do tipo deliberativo ou reativo. O agente reativo tem suas características descritas, neste sistema de padrões, pelo padrão Reativo. O agente deliberativo tem suas características descritas, neste sistema de padrões, pelo padrão Deliberativo.

- Os modelos de usuários são utilizados pelo agente de adaptação para construir modelos de adaptação para cada usuário ou grupo de usuários. As características do agente de adaptação são descritas, neste sistema de padrões, pelo padrão Adaptação.

Usos Conhecidos

Este padrão está sendo utilizado, no contexto dos projetos de pesquisa MaAE e Jurídicas [32], para o desenvolvimento de agentes de modelagem que utilizam técnicas de aquisição implícitas, baseadas em algoritmos genéticos [61] na construção de modelos de usuários.

5.5 Padrão Adaptação

Contexto

Nos sistemas adaptativos/adaptáveis, baseados na tecnologia de agentes, é preciso projetar soluções para a construção de modelos de adaptação que definem o processo de geração da adaptação, de acordo com os modelos de usuários.

Problema

Como construir modelos de adaptação para que a aplicação possa adaptar-se às necessidades de seus diferentes usuários ou grupo de usuários?

Forças

- A adaptação da aplicação segundo as características de seus usuários diminui a sobrecarga de informação.
- A adaptação também diminui a sobrecarga cognitiva, ou seja, o excesso de informação, pois os efeitos de adaptação fazem com que os usuários alcancem mais rapidamente seus objetivos, ao encontrar a informação ajustada às suas características e interesses.

Solução

A solução envolve a criação de um agente de adaptação. O principal papel do agente de adaptação é construir, manter e representar modelos de adaptação de acordo com os modelos de usuários.

Um modelo de adaptação consiste em regras de adaptação que definem o processo de geração da adaptação, estabelecem como a informação do modelo do usuário influi na adaptação do sistema, como também especifica como e quando atualizar a informação armazenada no modelo de usuário.

O agente de adaptação utiliza técnicas de *apresentação* e *navegação* adaptativas (ver seção 4.1.1) para construir os modelos de adaptação e seleciona o

tipo de adaptação que melhor se adequar às preferências e objetivos de cada usuário ou grupo de usuários, de acordo com o conteúdo dos modelos de usuários.

As técnicas de *apresentação adaptativa* adaptam o conteúdo da interface acessada por um usuário ou grupo de usuários, ao seu conhecimento, metas e outras características. Por exemplo, para um usuário mais qualificado, pode ser provida uma informação mais detalhada e profunda, enquanto que um usuário novato pode receber explicações adicionais. Em sistemas adaptativos, o conteúdo de uma interface pode não ser apenas um texto, mas também pode ser um conjunto de vários itens hipermídia [11].

As técnicas que dão suporte à *navegação adaptativa* ajudam os usuários a encontrar caminhos no espaço de navegação, adaptando a forma de apresentar os *links* às preferências, ao conhecimento e à outras características de um usuário ou grupo de usuários [11].

O agente de adaptação atualiza os modelos de usuários com base nos modelos de adaptação e o agente de interface personaliza a apresentação da aplicação de acordo com o modelo de usuário.

Padrões relacionados

O agente de adaptação pode ser definido como um agente do tipo deliberativo ou reativo. O agente reativo tem suas características descritas, neste sistema de padrões, pelo padrão Reativo. O agente deliberativo tem suas características descritas, neste sistema de padrões, pelo padrão Deliberativo.

Usos Conhecidos

Bylund [13] propõe o uso de agentes de adaptação para prover modelos de adaptação em sistemas multiagente adaptativos.

5.6 Padrão Deliberativo

Contexto

Em sistemas multiagente, nos quais existem tarefas complexas, é preciso projetar agentes que estejam aptos a executar essas tarefas, exibindo comportamento direcionado a metas, tomando iniciativas e decisões por meio de mecanismos de raciocínio e/ou adaptando-se às mudanças no ambiente.

Problema

Como projetar um agente para que possa raciocinar sobre um problema de forma que o possibilite atingir metas pró-ativamente dentro do contexto no qual está inserido?

Forças

- Para executar tarefas complexas e tomar decisões, é necessário que os agentes possuam conhecimento sobre o ambiente no qual estão inseridos e sobre si próprio, e que sejam capazes de manipular esse conhecimento para atingir seus objetivos.

- Agentes que são dotados de conhecimento e de mecanismos de raciocínio são capazes de raciocinar sobre ações passadas e planejar ações futuras. Esses agentes apresentam comportamento inteligente estando isolados ou inseridos em sociedades.

Solução

O agente deve ser do tipo deliberativo (ver seção 3.1), ou seja, ele deve possuir modelos simbólicos internos de raciocínio do ambiente no qual ele está inserido e de si mesmo. Ele raciocina sobre esses modelos para criar um plano que lhe permite atingir suas metas. O padrão Deliberativo é estruturado por cinco módulos organizados verticalmente e horizontalmente: módulo *Comunicação*, módulo *Ação*, módulo *Raciocínio*, módulo *Sensores* e módulo *Conhecimento*. A estrutura do padrão Deliberativo está representada na

Figura 30.

1- Recebe mensagens e percebe mudanças no ambiente
2- Registra percepções e mensagens recebidas
3- Percepções ou mensagens recebidas
4- Consulta capacidades do agente
5- Plano de ações
6- Ações não apoiadas pelo agente
7- Registra e consulta possíveis colaborações com outros agentes
8- Envia mensagens para outros agentes
9- Atua no ambiente
10- Registra ações realizadas

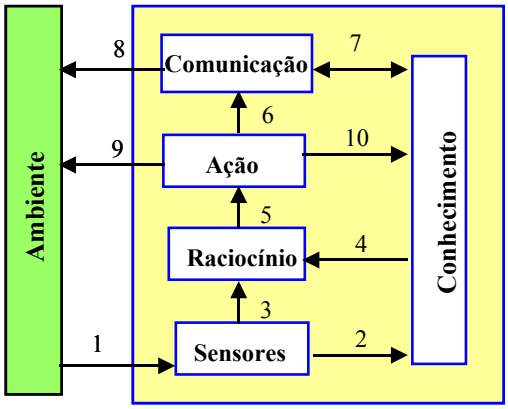

Figura 30 - Estrutura do padrão Deliberativo

Por meio do *módulo Sensores*, o agente percebe o ambiente e atualiza seu conhecimento com estas percepções para refletirem o estado corrente do ambiente. Percepções são informações que o agente pode receber do mundo real, informações de recursos e interações com outros usuários. O módulo Sensores também recebe mensagens de outros agentes da sociedade.

No módulo *Raciocínio*, baseado nos seus conhecimentos, o agente elabora metas e planos para atingir essas metas.

No módulo *Ação,* o agente atua no ambiente, executando as ações do plano gerado pelo módulo Raciocínio. A ação executada é armazenada no módulo Conhecimento para refletir os efeitos das ações no ambiente. O módulo Ação serve o módulo Comunicação. Durante a execução de um plano, um agente pode determinar a necessidade de cooperação com outros agentes, requerendo informações ou pedindo para executar ações. Esta cooperação é apoiada pelo módulo Comunicação.

No módulo *Comunicação*, o agente envia mensagens.

O módulo *Conhecimento* é um repositório no qual o agente armazena informações sobre o estado corrente do ambiente, por meio do histórico das percepções do agente e ações executadas. Este módulo contém recursos e

comportamento do agente e também conhecimento sobre as habilidades dos outros agentes da sociedade.

Os módulos Sensores, Raciocínio, Ação e Comunicação têm acesso direto ao módulo Conhecimento para que possam executar suas respectivas funcionalidades.

Cada módulo pode ser construído pela composição de um grupo de padrões para prover a funcionalidade de cada módulo.

O ciclo de execução, representado na **Figura 31**, do agente consiste em: no módulo Sensores, a percepção é usada para atualizar o conhecimento do agente. No módulo Raciocínio, este conhecimento é usado para determinar, dinamicamente, as metas correntes do agente para as quais um plano de ações foi construído, de acordo com as capacidades e comportamento do agente, armazenadas no módulo Conhecimento. O plano é então executado no módulo Ação. Quando um plano necessita de ações que o agente não está apto a executar porque elas não fazem parte das suas capacidades e comportamentos, o módulo Comunicação determina a necessidade de cooperação com outros agentes. Então, mensagens são enviadas para agentes capazes de ajudá-lo a cumprir o plano; as mensagens de outros agentes são recebidas por meio do módulo Sensores. Depois então, o ciclo completo é repetido. O módulo Conhecimento é atualizado a cada novo ciclo pelos demais módulos.

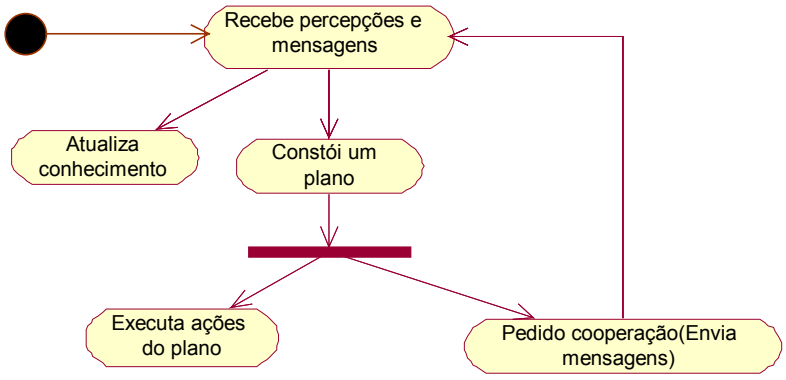

Figura 31 – Diagrama de atividades do padrão Deliberativo

Padrões relacionados

Os agentes deliberativos são amplamente utilizados em arquiteturas multiagente que utilizam raciocínio complexo [66]. Kendall [36] propõe um padrão alternativo, chamado *Agente deliberativo* que difere do padrão proposto neste trabalho, principalmente em dois aspectos. Primeiro, o padrão Agente deliberativo, sugerido por Kendall acrescenta as camadas *Mobilidade* e *Tradução*, funcionalidades das quais fazemos abstração nesse trabalho. Segundo, o padrão Deliberativo propõe a base de conhecimento do agente como módulo que interage verticalmente com todos os outros módulos do agente.

Usos conhecidos

Os ambientes de desenvolvimento de sistemas multiagente, como *AgentBuilder* [1] e Zeus [67], descritos na seção 3.4, usam diferentes arquiteturas genéricas para a construção de agentes deliberativos. Eles provêm funcionalidades semelhantes às providas pelas camadas do padrão Deliberativo. Porém, a forma como os módulos interagem é diferente. No agente genérico da *AgentBuilder*, os módulos interagem diretamente, sem considerar uma hierarquia de camadas. O agente genérico do Zeus é organizado em camadas, no entanto o conhecimento do agente está distribuído em várias camadas.

5.7 Padrão Reativo

Contexto

Em sistemas multiagente, nos quais existem tarefas que demandam respostas rápidas durante a sua execução, é necessário projetar agentes que estejam aptos a reagir a estímulos do seu ambiente, mas que não utilizem raciocínio complexo e que não tenham conhecimento sobre o ambiente no qual estão inseridos.

Problema

Como projetar um agente para apenas reagir a estímulos do ambiente no qual está inserido ou a mensagens de outros agentes, quando ele não tem conhecimento sobre esse ambiente e nem pode aprender a partir dele?

Forças

Em aplicações multiagente que demandam respostas rápidas, é desejável que os agentes se comportem segundo o modo estímulo/resposta, ou seja, os agentes não possuem uma memória sobre ações realizadas no passado e nem previsão de ações que poderão ser executadas no futuro, eles apenas são capazes de acompanhar modificações no ambiente. Porém, os agentes que possuem esse comportamento, podem exibir algum tipo de comportamento inteligente, por meio da interação com outros agentes. Como exemplo, podemos fazer uma analogia entre uma sociedade de agentes e uma colônia de formigas. Uma formiga sozinha não é capaz de realizar tarefas muito inteligentes, mas quando organizadas em colônias, elas procuram alimentos, defendem a colônia, cuidam dos ovos larvas. O mesmo ocorre com agentes reativos, que em grande número, são capazes resolver problemas complexos. Assim, a inteligência pode ser encontrada em um grupo de várias entidades simples.

Solução

O agente deve ser do tipo reativo (ver seção 3.1), ou seja, ele não deve ter um modelo interno do ambiente no qual está inserido, o agente deve agir usando regras do tipo condição-ação, de acordo com o estado corrente do ambiente ao qual

está integrado. A solução do padrão Reativo estrutura um agente em quatro módulos: C*omunicação*, *Ação*, *Regras* e *Sensores*. A estrutura do padrão Reativo está representada na **Figura 32**.

1- Recebe mensagens e percebe mudanças no ambiente

2-Seleciona regra de acordo com as percepções

3- Ações para serem executadas

4- Ações não apoiadas pelo agente

5- Envia mensagens para outros agentes

6- Atua no ambiente

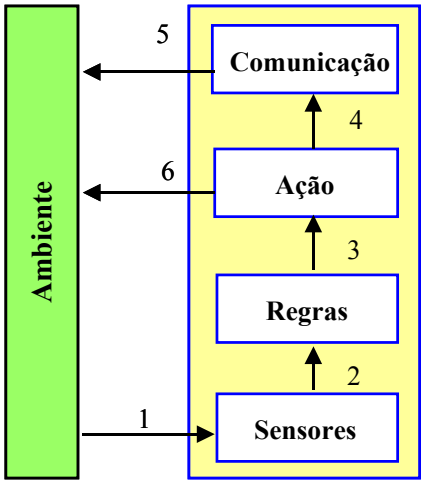

Figura 32 - Estrutura do padrão Reativo

Por meio do módulo S*ensores*, o agente percebe o ambiente. Percepções são informações que o agente pode receber do mundo real, informações de recursos e interações com usuários. O módulo Sensores também recebe mensagens de outros agentes da sociedade. O módulo Sensores serve o módulo Regras.

O módulo *Regras* é composto de um conjunto de regras de condição-ação da forma "**se** <condição> **então** <ação>". Baseado nos estímulos percebidos pelo módulo Sensores, uma regra condição-ação é escolhida. O módulo Regras serve o módulo Ação.

No módulo *Ação*, uma ação é selecionada para ser executada.

Durante a execução de uma ação, o agente determina a necessidade de cooperação com outros agentes para requerer informações ou pedir a execução de uma ação. O módulo Comunicação dá suporte à cooperação com outros agentes.

No módulo *Comunicação,* o agente envia e processa mensagens.

Cada módulo pode ser construído por meio da composição de um grupo de padrões para prover a funcionalidade de cada módulo.

O ciclo de execução do agente, representado na **Figura 33**, consiste em: no módulo Sensores, um estímulo é percebido. No módulo Regras, este estímulo é usado para determinar a ação que o agente irá executar no módulo Ação. Se para a realização da ação selecionada, o agente necessitar cooperar com outros agentes, o módulo Comunicação é ativado. Então, mensagens são enviadas para os agentes capazes de cooperar; as respostas dos outros agentes são recebidas por meio do módulo Sensores, então o ciclo completo se repete.

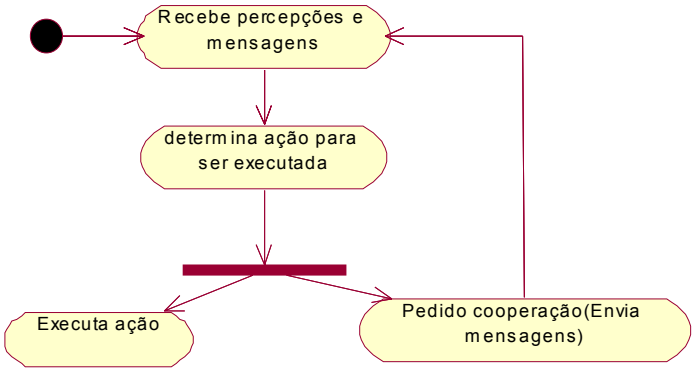

Figura 33 – Diagrama de atividades do padrão Reativo

Padrões relacionados

Kendall [36] propõe um padrão alternativo, chamado *Agente reativo*, para estruturar agentes reativos, similar ao desenvolvido neste trabalho, sendo que o padrão sugerido por Kendall acrescenta as camadas *Mobilidade* e *Tradução*, funcionalidades das quais fazemos abstração atualmente nesse trabalho.

Usos conhecidos

Este tipo de solução para a estruturação dos agentes foi originalmente proposta na arquitetura de classificação de Brooks [9]. Kulkarni [40] criou um sistema de comportamento reativo chamado ReBa que pode disparar diferentes ações em resposta a eventos de dispositivos. Dentre os ambientes de desenvolvimento de software e frameworks multiagente, apresentados na seção 3.4, apenas o JADE [34] provê uma arquitetura genérica para o desenvolvimento de agentes reativos.

6. ESTUDO DE CASO: REUTILIZAÇÃO DO SISTEMA DE PADRÕES APSUMAS NO PROJETO DE DESENVOLVIMENTO DE UMA APLICAÇÃO MULTIAGENTE PARA O ACESSO A INFORMAÇÃO PERSONALIZADA

O estudo de caso, apresentado neste capítulo, tem o objetivo de avaliar o APSUMAS, sistema de padrões baseado em agentes para a modelagem de usuários e adaptação de sistemas, por meio da reutilização dos padrões que compõem o APSUMAS, no projeto de desenvolvimento de uma aplicação para o acesso à informação.

6.1 Metodologia

A metodologia (**Figura 34**) utilizada para o desenvolvimento do estudo de caso esteve baseada na escolha de um domínio de aplicação, o domínio do acesso à informação, representado por meio da ontologia ONTOINFO [56]. Para o desenvolvimento do projeto da aplicação para o acesso à informação, foi utilizada a DDEMAS, uma técnica para o projeto de domínio de aplicações multiagente.

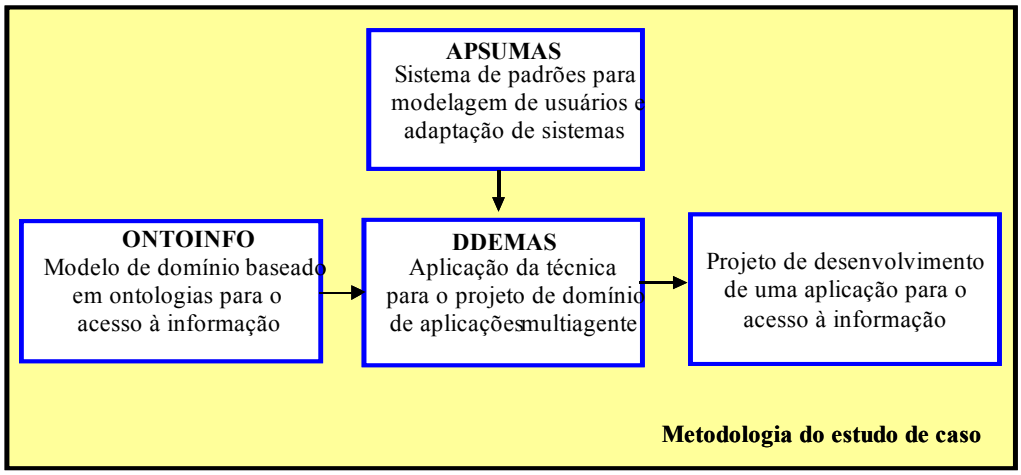

Figura 34 – Metodologia utilizada para o desenvolvimento do estudo de caso

6.2 O Acesso à Informação

O acesso à informação visa prover mecanismos para que seja possível localizar e recuperar informações relevantes aos usuários, da forma mais eficiente

possível. Os principais conceitos e processos envolvidos no acesso à informação são descritos a seguir [31].

A princípio, tem-se uma necessidade de informação, que deve ser expressa em uma consulta ou no modelo de usuário. Uma consulta caracteriza uma necessidade de informação pontual. O modelo de usuário caracteriza uma necessidade de informação em longo prazo.

Se a necessidade de informação for pontual, o sistema deve acessar uma base indexada e recuperar os elementos de informação relevantes, após uma avaliação de similaridade entre a consulta e os elementos de informação recuperados. Se a necessidade de informação for em longo prazo, o sistema deve fazer a filtragem. A filtragem consiste em estar continuamente monitorando fontes de informação à procura de elementos de informação que possam ser relevantes a seus usuários, representados por meio de modelos de usuário, ou seja, na filtragem, a análise de similaridade é feita entre os elementos de informação e o perfil do usuário, representado no modelo de usuário. Este estudo de caso leva em conta apenas necessidades de informação em longo prazo.

Alguns conceitos se destacam no âmbito do acesso à informação, sendo os principais:

- **Necessidade de informação:** é uma carência de informação por parte de um ou mais usuários. Conforme já mencionado, a necessidade de informação pode ser pontual, ou seja, uma necessidade a ser satisfeita imediatamente, ou em longo prazo, isto é, uma necessidade que se prolonga por um certo tempo.

- **Elemento de informação:** é um item que contém informação. Os elementos de informação podem ser dos seguintes tipos: documentos textuais e hipermídia, vídeo, som e imagem.

- **Fonte de informação:** é um repositório de elementos de informação. As fontes de informação podem ser estruturadas ou não estruturadas, podem ainda ser dinâmicas ou estáticas. Um exemplo de fonte de informação desestruturada e dinâmica é a *Web*.

- **Modelo de usuário:** é uma representação das preferências e necessidades do usuário a partir do qual o sistema filtra elementos de informação que possam ser relevantes. O modelo de usuário pode ser capturado de forma implícita ou explicita, de acordo com apresentado na seção 4.2.

- **Consulta:** é a representação da necessidade de informação do usuário expressa em uma linguagem de especificação de consulta.

- ***Surrogate*:** é a forma como as consultas, os elementos de informação e os modelos de usuários são representados internamente no sistema.

- **Filtragem:** é uma modalidade de acesso à informação utilizada no caso de uma necessidade de informação em longo prazo.

- **Recuperação:** é uma modalidade de acesso à informação utilizada no caso de uma necessidade de informação pontual.

- **Indexação:** é o processo de criação de um índice invertido com os *surrogates* dos elementos de informação para agilizar o processo de matching.

- ***Matching*:** é um processo que compara as representações internas dos itens de informação com a representação interna da consulta ou do perfil do usuário e seleciona uma lista de itens de informação que casam parcial ou totalmente com a consulta.

- **Análise de similaridade:** executa uma medida de similaridade ou determina o valor de similaridade que avalia quanto um item de informação satisfaz uma necessidade de informação.

O conhecimento e os conceitos da área do acesso à informação foram representados por meio de uma ontologia, a ONTOINFO [56].

Este estudo de caso leva em consideração apenas a filtragem, a modalidade de acesso à informação em que as características e preferências dos

usuários são utilizadas como referência, para o sistema suprir a necessidade de informação dos usuários de forma personalizada.

De acordo com a ONTOINFO, um dos objetivos do acesso à informação é recuperar apenas informações que estejam de acordo com as preferências e necessidades dos usuários, ou seja, adaptar o sistema ao usuário. Então, os sistemas para o acesso à informação, que utilizam a filtragem de informação, podem ser considerados como sistemas adaptativos. Fazendo um mapeamento entre os atributos **contexto** e **problema,** do padrão conceitual Sociedade multiagente para sistemas adaptativos/adaptáveis (descrito na seção 5.1), e os objetivos do acesso à informação, conclui-se que esse padrão pode ser utilizado para guiar o projeto de sistemas de acesso à informação, pois ele descreve os conceitos fundamentais relacionados aos sistemas adaptativos/adaptáveis, como também oferece diretrizes básicas para a utilização da tecnologia de agentes para projetar tais sistemas.

6.3 A Técnica DDEMAS

A técnica DDEMAS [21] [23], especificada na ONTODD (Ontologia Genérica de Projeto), consiste em três fases, nas quais é especificado o projeto da aplicação multiagente em ordem crescente de detalhamento: *modelagem de agentes e interações*, *projeto global* e *projeto detalhado* (**Figura 35**). A primeira fase aborda a definição dos componentes do sistema (agentes), entidades externas e as relações existentes entre os mesmos. Na segunda, é feito o projeto global do sistema, com os agentes se relacionando, organizados segundo mecanismos de coordenação e cooperação apropriados ao contexto do problema. Na última fase, cada agente é analisado individualmente em termos comportamentais e de conhecimento [23].

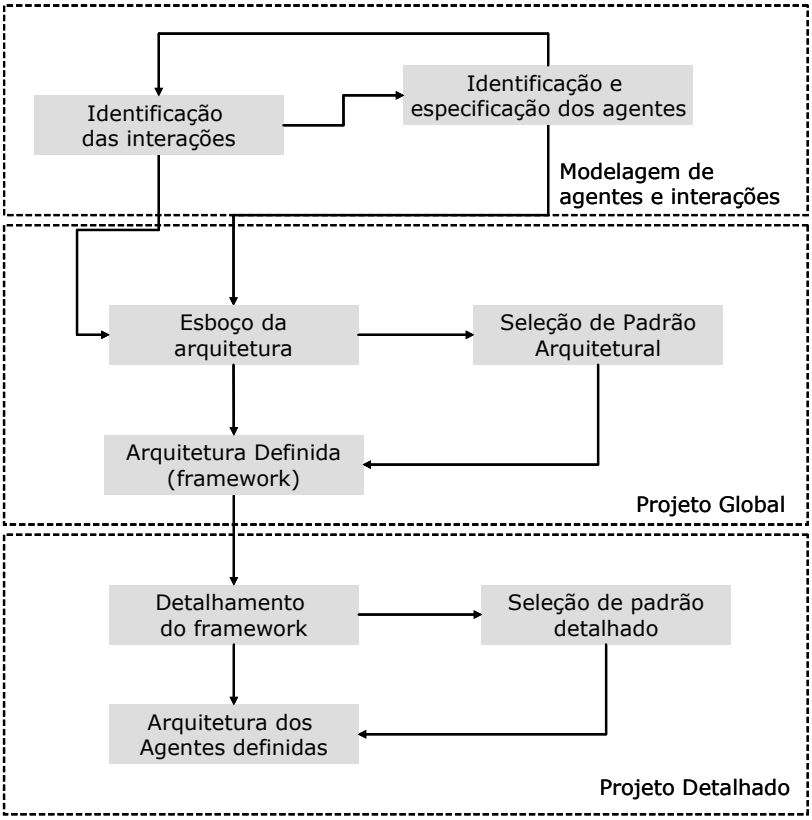

Figura 35 - Fases da técnica DDEMAS [23].

6.4 Aplicação da Técnica DDEMAS

6.4.1 Modelagem de Agentes e Interações

Finalizada a fase de modelagem de agentes e interações da DDEMAS, os agentes e suas respectivas atividades necessárias para a filtragem de informação foram identificadas, como descrito na **Tabela 7**.

Para a filtragem de informação, foi identificada a entidade externa *fonte de informação,* na qual estão armazenados os elementos de informação.

Segundo o modelo de domínio do acesso à informação, o objetivo dos sistemas de acesso à informação personalizada é satisfazer a necessidade de informação contínua dos usuários. Na busca desses objetivos, foram identificadas as relações entre os agentes e entidades externas. A **Figura 36** mostra as interações que são necessárias para satisfazer a necessidade de informação contínua do usuário.

	Nome	Papel	Responsabilidade	Atividades
AGENTES	Construtor de *surrogate*	Construtor de *surrogate*	Construção de *surrogate*	▪ Construir *surrogates* de itens de informação; ▪ Construir os *surrogates* de consulta; ▪ Construir os *surrogates* de modelos de usuários
	Interfaceador do usuário	Interfaceador do usuário	Gerenciamento da interação do usuário com o sistema	▪ Mostrar resultados; ▪ Monitorar o usuário; ▪ Processar consulta; ▪ Fornecer itens de informação filtrados
	Monitor	Monitor	Monitorar as fontes de informação	▪ Detectar mudanças em fontes de informação dinâmicas; ▪ Enviar itens de informação monitorados para o construtor de *surrogates*
	Modelador do usuário	Modelador do usuário	Criação e manutenção de modelos de usuários	▪ Criar e manter modelos de usuários.
	Filtrador	Filtrador	Filtrar informações	▪ Fazer o *matching* entre o item de informação e o *surrogate* do modelo de usuário; ▪ Fazer a análise de similaridade entre os itens de informação e o *surrogate* do modelo de usuário; ▪ Armazenar itens de informação filtrados de acordo com modelos de usuários.

Tabela 7 - Descrição de agentes para filtragem de informação

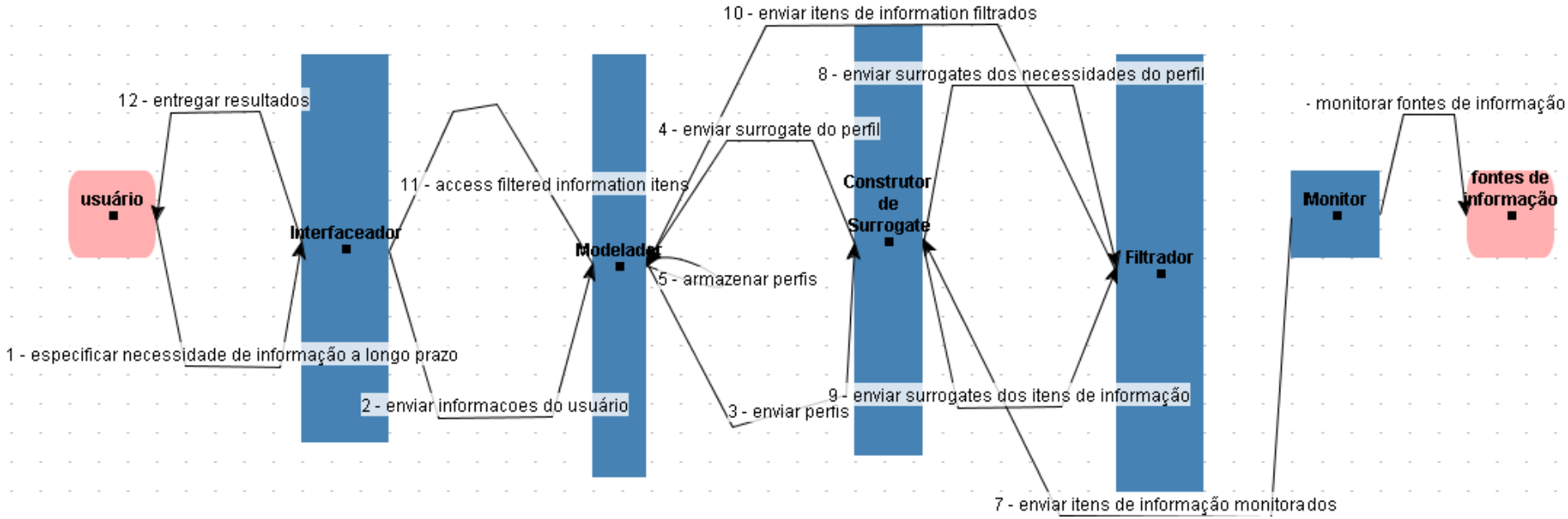

Figura 36 - Modelo de interações para a filtragem de informação [23]

6.4.2 Projeto Global

De posse das relações entre os agentes, o modelo arquitetural é construído (**Figura 37**). A construção do modelo arquitetural é baseada nas interações e relações entre os agentes da sociedade (de acordo com o descrito na seção 3.2).

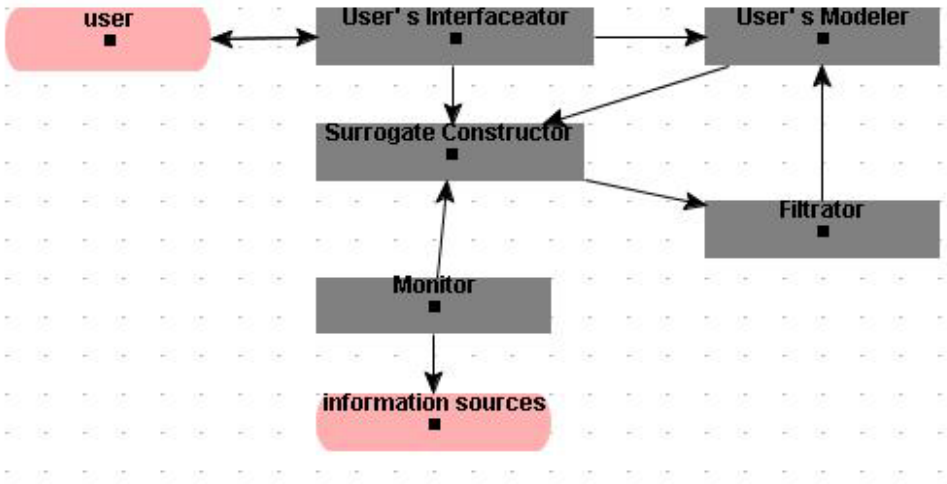

Figura 37 – Modelo arquitetural para a filtragem de informação [23]

Ainda nesta fase, o modelo arquitetural é refinado por meio da seleção de um ou mais padrões arquiteturais, com o objetivo de organizar os agentes que compõem o modelo arquitetural, de acordo com a solução especificada no padrão (ou padrões) selecionado(s).

6.4.2.1 Seleção do Padrão Arquitetural

Para a seleção do padrão arquitetural é feito um mapeamento entre as responsabilidades dos agentes que compõem o modelo arquitetural, e os atributos **problema** e **contexto** do padrão.

O padrão Camadas multiagente para sistemas adaptativos/adaptáveis (descrito na seção 5.2) pode ser utilizado para organizar os agentes identificados nas fases anteriores. Esses agentes são organizados segundo a solução descrita pelo padrão.

A organização dos agentes, após a aplicação do padrão, pode ser visualizada na **Figura 38**. Depois de feito o mapeamento, pode-se concluir que as

camadas 1, 2 e 3 correspondem, respectivamente às camadas de *interface*, *modelagem* e *adaptação* descritas na solução do padrão Camadas multiagente para sistemas adaptativos/adaptáveis.

Para o desenvolvimento do projeto da aplicação para o acesso à informação personalizada, foram considerados apenas os agentes que fazem parte da filtragem de informações, ou seja, os agentes que colaboram para satisfazer a necessidade de informação em longo prazo.

De acordo com a solução do padrão aplicado, a divisão em camadas se deu da seguinte forma:

- A camada 1, que chamaremos de camada de interface, é responsável por gerenciar a interface entre o sistema e o usuário; receber consultas e informações dos usuários, entregar os resultados da consulta de acordo com os modelos de usuários.

- A camada 2, que chamaremos de camada de modelagem, é responsável por criar e manter modelos de usuários de acordo com informações fornecidas pela camada de interface.

- A camada 3, que chamaremos de camada de adaptação, é responsável por construir *surrogates* de modelos de usuários, de consulta e de itens de informação; por filtrar a informação de acordo com os modelos de usuários; e por monitorar as fontes de informação. Os modelos de adaptação, no acesso à informação, são os itens de informação filtrados de acordo com os modelos de usuários. A camada de adaptação também atualiza a base de modelos de usuários com os itens de informação filtrados.

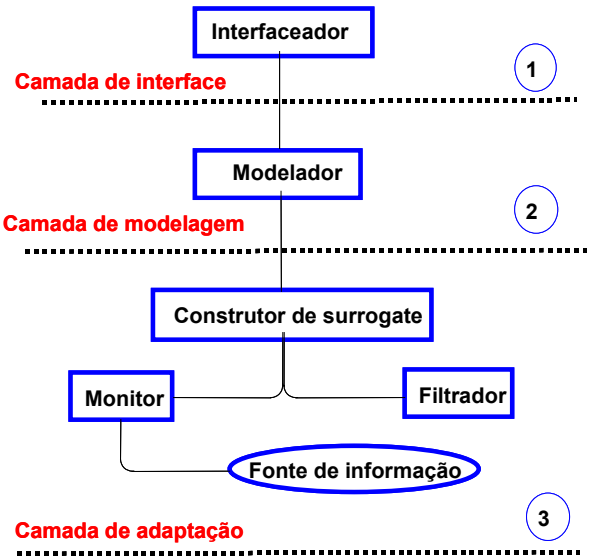

Figura 38 – Modelo arquitetural após a aplicação do padrão Camadas multiagente para sistemas adaptativos/adaptáveis

6.4.3 O Projeto Detalhado

Após a aplicação do padrão arquitetural, teremos a estrutura do sistema definida, iniciando então a especificação de cada um dos agentes que compõem as camadas da arquitetura, ou seja, o projeto detalhado dos agentes da sociedade (de acordo com o descrito na seção 3.2). Para o projeto detalhado de cada agente é feito um mapeamento entre as responsabilidades do mesmo e os atributos **problema** e **contexto** dos padrões de projeto, de forma a selecionar o padrão mais adequado.

Dentre os agentes que compõem a arquitetura, definida após a aplicação do padrão Camadas Multiagente para Sistemas Adaptativos/Adaptáveis, os agentes *interfaceador*, *modelador* e *filtrador* têm suas principais responsabilidades descritas nos padrões *Interface*, *Modelagem* e *Adaptação* (descritos no capítulo 5), respectivamente, como descrito na **Tabela 8**.

Cada agente pode ainda ser refinado, por meio da classificação dos agentes em deliberativos ou reativos (ver seção 3.1). Os padrões *Deliberativo* e *Reativo* podem ser utilizados para descrever o comportamento do agente. A **Tabela 9** relaciona os agentes aos respectivos padrões que melhor descrevem o seu comportamento..

Agente	Responsabilidade do agente	Padrão	Contexto do padrão	Problema do padrão	Descrição da seleção do padrão
Interfaceador	Gerenciamento da interação entre o usuário e o sistema	**Interface**	Nos sistemas adaptativos/adaptáveis, baseados na tecnologia de agentes, é preciso projetar soluções para a interação do sistema com os usuários	Como gerenciar a interação entre um usuário ou grupo de usuários e uma aplicação de software de forma personalizada?	Este agente serve como intermediador entre o usuário e o resto do sistema, recebendo requisições do usuário, interpretando-as e repassando-as ao modelador. Ele também é responsável pela entrega dos resultados aos usuários. Na filtragem de informação, a tarefa de produção da adaptação, especificada na solução do padrão Interface, se resume a entregar os resultados filtrados de acordo com o modelo do usuário. A escolha das técnicas para a aquisição das informações do usuário, fica a critério do projetista. Quanto aos dados que são relevantes para a filtragem, apenas os interesses do usuário são levados em conta.
Modelador	Criação e manutenção de modelos de usuários;	**Modelagem**	Nos sistemas adaptativos/adaptáveis, baseados na tecnologia de agentes, é preciso projetar soluções para representar os usuários por meio da construção de modelos de usuários que especificam suas características, suas preferências e os seus interesses.	Como criar e manter modelos de usuários que serão utilizados como referência para que a aplicação ofereça serviços personalizados aos seus usuários?	Este agente faz a modelagem dos usuários por meio da coleta de informações feita pelo agente *interfaceador* e então repassa a especificação destes modelos de usuários ao agente c*onstrutor de surrogate*. A escolha das técnicas de representação e a manutenção de modelos de usuários, descritas no padrão modelagem, fica a cargo do projetista.
Filtrador	Filtragem de itens de informação	**Adaptação**	Nos sistemas adaptativos/adaptáveis, baseados na tecnologia de agentes, é preciso projetar soluções para a construção de modelos de adaptação que definem o processo de geração da adaptação de acordo com os modelos de usuários.	Como construir modelos de adaptação para que a aplicação possa adaptar-se às necessidades de seus diferentes usuários ou grupo de usuários?	Na solução do padrão adaptação, o agente de adaptação é responsável por construir modelos de adaptação, ou seja, ele seleciona o tipo de adaptação que melhor se adequar às preferências e objetivos de cada usuário. Na filtragem de informação, a construção de modelos de adaptação se restringe a filtrar as informações de acordo com os interesses dos usuários, definidos no modelo de usuários. O agente *filtrador* atualiza os modelos de usuário com os itens de informação filtrados.

Tabela 8 - Detalhamento dos agentes para o acesso à informação ,por meio da aplicação dos padrões Interface, Modelagem e Adaptação

Padrão	Contexto do padrão	Problema do padrão	Agente	Descrição
Deliberativo	Em sistemas multiagente, nos quais existem tarefas complexas, é preciso projetar agentes que estejam aptos a executar essas tarefas, exibindo comportamento direcionado a metas, tomando iniciativas e decisões por meio de raciocínio e/ou adaptando-se às mudanças no ambiente.	Como projetar um agente para que possa raciocinar sobre um problema de forma que o possibilite atingir metas pró-ativamente dentro do contexto no qual está inserido?	Modelador	O agente *modelador* pode ser definido como um agente do tipo deliberativo, pois ele necessitará de mecanismos de raciocínio para inferir necessidades e preferências dos usuários a serem representados no modelo a partir das observações do comportamento do usuário realizadas pelo agente *interfaceador*.
			Construtor de *surrogate*	O agente *construtor de surrogate* pode ser definido como um agente do tipo deliberativo, pois ele pode necessitar de mecanismos de raciocínio para fazer as representações dos modelos de usuários, das consultas e dos itens de informação.
Reativo	Em sistemas multiagente, nos quais existem tarefas que demandam respostas rápidas durante a sua execução, é necessário projetar agentes que estejam aptos a reagir a estímulos do seu ambiente, mas que não utilizem raciocínio complexo e que não tenham conhecimento sobre o ambiente no qual estão inseridos.	Como projetar um agente para apenas reagir a estímulos do ambiente no qual está inserido ou a mensagens de outros agentes quando ele não tem conhecimento sobre esse ambiente e nem pode aprender a partir dele?	Interfaceador	O agente *interfaceador* pode ser definido como um agente do tipo reativo, pois ele não precisará exibir raciocínio complexo, necessitará apenas de um conjunto de regras de condição-ação para colher os dados dos usuários e entregar os resultados das suas requisições, ou seja o agente *interfaceador* reagirá às entradas fornecidas pelos os usuários e atuará no ambiente por meio da entrega de resultados.
			Monitor	O agente *monitor* pode ser definido como um agente do tipo reativo. Ele necessitará apenas de um conjunto de regras de condição-ação para detectar mudanças no ambiente, mais precisamente nas fontes de informação dinâmicas.
			Filtrador	O agente *filtrador* pode ser definido como um agente do tipo reativo, pois ele necessitará apenas de um conjunto de regras de condição-ação para fazer o *matching* entre o item de informação e o *surrogate* do modelo de usuário e para fazer a análise de similaridade entre os itens de informação e o *surrogate* do modelo de usuário.

Tabela 9 - Detalhamento dos agentes para o acesso à informação, por meio da aplicação dos padrões Reativo e Deliberativo

6.5 Avaliação do Estudo de Caso

O estudo de caso desenvolvido contribuiu para a uma primeira avaliação do sistema de padrões proposto, sendo de fundamental importância para a identificação das vantagens na utilização desses padrões no desenvolvimento de aplicações adaptativas orientada a agentes.

A aplicação do APSUMAS facilitou o desenvolvimento do projeto da aplicação para o acesso à informação personalizada, uma vez que o sistema fornece diretrizes para guiar o processo de desenvolvimento, desde a organização dos agentes da sociedade, de acordo com uma arquitetura, até a definição da arquitetura particular de cada agente da aplicação.

7. CONSIDERAÇÕES FINAIS

A atividade de modelagem de usuários em sistemas adaptáveis é bastante complexa, mas de grande importância e aplicabilidade quando se pretende sintetizar características e habilidades de pessoas ou grupo de pessoas com o intuito de oferecer serviços personalizados.

Para lidar com a complexidade e os problemas encontrados na modelagem de usuários, utilizamos a tecnologia de agentes juntamente com os conceitos de padrões, propondo assim, padrões conceituais, arquiteturais e padrões de projeto detalhado orientados a agentes. A tecnologia de agentes e os padrões foram utilizados neste trabalho por dois motivos principais:

- A estruturação de arquiteturas e componentes de software na forma de padrões simplifica o desenvolvimento do software em várias fases do seu ciclo de vida, pois os padrões descrevem soluções bem sucedidas para problemas recorrentes em um formato consistente e de fácil entendimento;

- O paradigma de agentes diminui a complexidade do software, pois os agentes podem tomar decisões sobre suas interações em tempo de execução, podem cooperar com outros agentes para atingir metas e podem acumular conhecimentos por meio de suas experiências.

7.1 Resultados e Contribuições da Pesquisa

As principais contribuições desta pesquisa foram:

- Análise dos problemas existentes nos processos de modelagem de usuários e adaptação de sistemas em diversas áreas de aplicação;

- Análise do estado da arte das técnicas utilizadas na modelagem de usuários e adaptação de sistemas;

- Análise das arquiteturas genéricas de agentes de software propostas nos principais ambientes de desenvolvimento de sistemas multiagente

- Especificação de padrões de projeto baseados em agentes que captam conhecimento, técnicas e problemas comuns na modelagem de usuários e adaptação de sistemas;

- Especificação de padrões de projeto que organizam a arquitetura de agentes do tipo reativo e deliberativo, segundo as características e funcionalidades específicas de cada tipo;

- Organização dos padrões propostos em um sistema de padrões baseados em agentes para a modelagem de usuários e adaptação de sistemas, facilitando assim, sua reutilização na construção de frameworks para o desenvolvimento de aplicações adaptativas baseadas em agentes;

- Avaliação inicial do sistema de padrões proposto por meio de um estudo de caso que reutiliza os padrões no projeto de desenvolvimento de uma aplicação para sistemas de acesso à informação personalizada.

7.2 Trabalhos Futuros

Vários trabalhos futuros podem ser concebidos para o refinamento do sistema de padrões proposto:

- Especificação de protocolos de interação entre os agentes que compõem as camadas do padrão arquitetural Camadas multiagente para sistemas adaptativos/adaptáveis;

- Detalhamento dos módulos que compõem as estruturas dos padrões Deliberativo e Reativo, por meio da extração de novos padrões de projeto que descrevem as funcionalidades de cada módulo;

- Detalhamento das funcionalidades e características dos agentes especificadas nos padrões Interface, Modelagem e Adaptação, por meio da extração de novos padrões de projeto;

- Utilização do sistema de padrões no desenvolvimento de sistemas adaptativos baseados em agentes;
- Construção de frameworks para o desenvolvimento de sistemas adaptativos e sistemas de recomendação, em áreas como, turismo, comércio eletrônico e educação.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] "AgentBuilder User's guide", Reticular Systems, external documentation, disponível em: http://www.agentbuilder.com/, acesso em Janeiro, 2003.

[2] APPLETON, Brad. **Patterns and Software: Essential Concepts and Terminology**, disponível em: http://www.enteract.com/~bradappdocpatterns-intro.html, acesso em 10/10/2002.

[3] ARIDOR, Y, LANGE, D. **Agent Design Patterns Elements of Agent Application**, Proceedings of Autonomous Agents, 1998.

[4] BALBY, Leandro, GIRARDI, Rosario, **Mineração na Web**, Revista Eletrônica de Iniciação Científica da SBC, Vol. III, nº 4, 2003.

[5] BALDASSIN, Alexandro, RIZZO, Ivan, MALTEMPI, Marcus. **Uma Abordagem Baseada em Agentes para Filtragem de Correspondências Eletrônicas**. Revista Eletrônica de Iniciação Científica (REIC), Vol. II, n° 4, 2002.

[6] BEAUMONT, I. **User Modelling in the Interactive Anatomy Tutoring System ANATOM-TUTOR**. User Modelling and User-Adapted Interaction 4(1) 21 -45, 1994.

[7] BOOCH, G., RUMBAUGH, J., JACOBSON, I.. **The Unified Language User Guide,** Addison-Wesley, Reading, MA, 1999.

[8] BRAZIER, Frances, JONKER, Catholijn, TREUR, Jan. **Principles of Component-Based Design of Intelligent Agents**, Data Knowledge Engineering, 41(2002) 1-28.

[9] BROOKS, R. **A Robust Layered Control System for a Mobile Robot**. IEEE Journal of Robotics and Automation, 2 (1): 14-23, 1986.

[10] BRUSILOVSKY, P. **Student as User: Toward an Adaptive Interface for Intelligent Learning Environment**, Proceedings of AI-ED93 World conference on Artificial Intelligence in Education, pp 386-396, 1993.

[11] BRUSILOVSKY, Peter. **Methods and Techniques of Adaptive Hypermedia.** User Modeling and User-Adapted Interaction, 6, pp. 87-129, 1996.

[12] BUSCHMANN, F., MEUNIERS, R., Rohnert, H., Sommerlad, P., Stal, M. **A System of Patterns. Pattern-Oriented Software Architecture**, Wiley, 1996.

[13] BYLUND, Markus, WAERN, Annika. **Adaptation Agents: Proving Uniform Adaptations in Open Service Architectures,** Proceedings of 3rd ERCIM Workshop on UI for All, 1997.

[14] COPLIEN, Jim. **Software Design Patterns: Common Questions and Answers,** disponível em: *ftp://st.cs.uiuc.edu/pub/patterns/papers/PatQandA.ps*, acesso em 23/07/2003.

[15] DEUGO, Dwight; WEISS, Michael; KENDALL, Elizabeth. **Reusable Patterns for Agent Coordination**. Computer Systems Engineering, Royal Melbourne Institute of Technology, Australia, 1998.

[16] DINIZ, Alessandra. **Uma Arquitetura baseada em Agentes para a Recuperação e Filtragem da Informação**, Dissertação (Mestrado em Ciência da Computação) – Curso de Pós-Graduação em Engenharia de Eletricidade, Universidade Federal do Maranhão, São Luís-MA, 2001.

[17] ERICEIRA, Bruno. **Desenvolvimento de Sistemas Multiagente Utilizando a Ferramenta Zeus,** Monografia do Curso de Bacharel em Ciência da Computação, Universidade Federal do Maranhão, São Luís-MA, 2001.

[18] FARIA, Carla, GIRARDI, Rosário. **Especificação de uma Ontologia Genérica para a Análise de Requisitos da Engenharia de Aplicações Multiagente**, Anais do Terceiro Congresso Brasileiro de Computação (CBComp 2003), UNIVALI, Itajaí, SC, Brasil, 2003.

[19] FARIA, Carla, RIBEIRO, Ismênia, GIRARDI, Rosario. **Especificação de uma Ontologia Genérica para a Construção de Modelos de Usuários**, Anais da Terceira Jornada Ibero-americana de Engenharia de Software e Engenharia do Conhecimento (JIISIC 2003), Valdivia, Chile, 2003.

[20] FARIA, Carla, RIBEIRO, Ismênia, GIRARDI, Rosario. **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, 2003.

[21] FERREIRA, Steferson Lima Costa, GIRARDI, Rosario **Especificação de uma Ontologia Genérica para o Projeto de Domínio de Aplicações Multiagente**, Anais do III Workshop de Engenharia de Software (WIS 2003) nas Jornadas

Chilenas de Computação, Chilán, Chile, 03 a 08 de novembro de 2003

[22] FERREIRA, Steferson Lima Costa, GIRARDI, Rosario, **Arquiteturas de Software baseadas em Agentes: do Nível Global ao Detalhado**, Revista Eletrônica de Iniciação Científica da SBC, Junho 2002.

[23] FERREIRA, Steferson Lima Costa, GIRARDI, Rosario. **Uma Técnica para o Projeto de Domínio de Sistemas Multiagente**, Anais do II Workshop de Informática do IMESA, Assis, São Paulo, 13 a 17 de outubro de 2003

[24] FINK, J, KOBSA, A.. **User Modeling in Personalized City Tours**. Artificial Intelligence Review 18(1), 33-74, 2002.

[25] FIPA – Foundation for Intelligent Physical Agents, disponível em: http://www.fipa.org, acesso em junho, 2003.

[26] FRIDMAN, N., MCGUINNESS, D. **Ontology Development 101: A Guide to Creating Your First Ontology**, Stanford Knowledge Systems Laboratory, Technical Report KSL-01-05, March 2001.

[27] GAMMA, Erich, HELM, Richard, JOHNSON, Ralph, VLISSIDES, J. **Padrões de Projeto: Soluções Reutilizáveis de Software orientado a Objetos;** trad. Luiz A. Meirelles Salgado. Bookman, 2000.

[28] GIRARDI, Rosario, FARIA, Carla. **A Generic Ontology for the Specification of Domain Models**, Proceedings of 1st International Workshop on Component Engineering Methodology (WCEM'03) at Second International Conference on Generative Programming and Component Engineering, Erfurt, Germany, 2003.

[29] GIRARDI, Rosario, RIBEIRO, Ismênia, BEZERRA, Geovane. **Towards a System of Patterns for the Design of Agent-based Systems**, Proceedings of The Second Nordic Conference on Pattern Languages of Programs" (VikingPLoP 2003). Bergen, Norway, 2003.

[30] GIRARDI, Rosario. **Agent-Based Application Engineering**, Proceedings of 3rd International Conference on Enterprise Information Systems, Setúbal, 2001.

[31] GIRARDI, Rosario. **Main Approaches to Software Classification and Retrieval**. En Ingeniería del Software y reutilización: Aspectos Dinámicos y Generación Automática. Editores J. L. Barros y A. Domínguez, Universidad de Vigo - Ourense, 1998.

[32] GIRARDI, Rosario. **Reuse in Agent-based Application Development**, Proceedings of 1º International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'2002), ICSE'2002, May 2002.

[33] HOHL, Hubertus; BÖCKER, Heinz-Dieter, GUNZENHÄUSER, Rul. **Hypadapter: An Adaptive Hupertext System for Exploratory Learning an Programming**, User Models and User Adapted Interaction, 6, n. 2-3. Special issue adaptive hypertext and hypermedia, 1996.

[34] JADE, Java Agent Development framework, disponível em : http://jade.cselt.it, acesso em maio, 2003.

[35] KENDALL, Elizabeth**. Patterns of Intelligent and Mobile Agents**, Proceedings of Proceedings of Autonomous Agents'98, 1998.

[36] KENDALL, Elizabeth. **The Layered Agent Pattern Language**, Proceedings of Pattern Languages of Programming (PLOP'97), 1997.

[37] KOBSA, Alfred, KOENEMANN, Jürgen, POHL, Wolfgang**. Personalized hypermedia Presentation Techniques for Improving Online Customer Relationships**. GMD Report 66, 1999.

[38] KOBSA, Alfred**. User Modeling and User-Adapted Interaction**. Conference Companion on Human factors in computing systems, p.415-416, April 24-28, 1994, Boston, Massachusetts, United States.

[39] KOSALA, R., BLOCKEEL, H**., Web Mining Research: a Survey**. SIG KDD Explorations, vol.2, pp. 1-15, 2000.

[40] KULKARNI, A. **A Reactive Behavioral System for the Intelligent Room**. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 2002. disponível em: http://citeseer.nj.nec.com/kulkarni02reactive.html. Acesso em maio, 2003.

[41] LANGLEY, P. **User Modeling in Adaptive Interfaces**, In: J Kay (ed.) User Modeling: Proceedings of the Seventh International Conference Springer 357–370, 1999.

[42] LESH, N., RICH, C., SIDNER, C.. **Using Plan Recognition in Human-computer Collaboration,** Proceedings of the Seventh International Conference on User Modeling, pps 23-32, June 1999.

[43] LIGEIRO, Roberto, DUTRA, Inês, **Redes Bayesianas: o que são, para que servem, algoritmos e exemplos de aplicações**, disponível em: www.cos.ufrj.br/~ines/courses/cos740/ leila/cos740/Bayesianas.pdf, acesso em agosto, 2003.

[44] MEDINA, Nuria, GARCÍA, Lina, RODRÍGUEZ, MªJosé, PARETS-LLORCA, José. **Adaptación al Usuario en Sistemas Hipermedia: El Modelo SEM-HP,** disponível em: http://lsi.ugr.es/~gedes/personales/nuria/Articulos/MedDolmenII.pdf, acesso em julho, 2003.

[45] MISLEVY, RJ, GITOMER, DH. **The Role of Probability-based Inference in an Intelligent Tutoring System**, User Modeling and User-Adapted Interaction 5(3-4) 253-282, 1996

[46] MOUKAS, Alexandros, MAES, Pattie. **Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem,** Proceedings of the Conference on Practical Applications of Agents and Multiagent Technology, 1996.

[47] NICK, Z, THEMIS, P. **Web Search Using a Genetic Algorithm**. IEEE Internet Computing, 5(2) (2001), 18-26.

[48] ODELL, James, PARUNAK, Van, BAUER, Bernhard. **Extending UML for Agents**. Agent-Oriented Information Systems Workshop at American Association for Artificial Intelligence, Austin TX, 2000.

[49] PALAZZO, Luiz Antônio Moro. **Sistemas de Hipermídia Adaptativa,** disponível em: http://ia.ucpel.tche.br/~lpalazzo/sha/sha.htm, acesso em fevereiro, 2003.

[50] POHL, Wolfgang, **Logic-Based Representation and Reasoning for User Modeling Shell Systems**, User Modeling and User-Adapted Interaction 9: 217-282, Kluwer Academic Publishers. Printed in the Netherlands, 1999.

[51] PROTÉGÉ PROJECT, disponível em: http://protege.stanford.edu, acesso em: 05 de maio de 2003.

[52] RIBEIRO, Ismênia, GIRARDI, Rosario. **Padrões Arquiteturais e de Projeto para a Modelagem de Usuários baseada em Agentes**, Proceedings of The Third Latin American Conference on Pattern Languages of Programming - SugarLoafPlop, Porto de Galinhas, Brasil, 2003.

[53] RIBEIRO, Ismênia, GIRARDI, Rosario**. Padrões baseados em Agentes para a Modelagem de Usuários**, XII Seminário de Computação (SEMINCO 2003). Centro de Convenções, Willy Sievert, PROEB, Blumenau, Santa Catarina, Brasil, pp. 85-98, agosto de 2003.

[54] RUSSELL, S, NORVIG, P. **Artificial Intelligence: A Modern Approach**. Prentice-Hall, 1995.

[55] SERRA JR, Gentil Cutrim. **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.

[56] SERRA, Ivo José da Cunha, GIRARDI, Rosário. **Uma Abordagem Gerativa para a Engenharia de Domínio Multiagente,** Anais do V Encontro de Estudantes de Informática do Tocantis (ENCOINFO), Ed. ULBRA, pp. 261-270. Palmas, Tocantis, Brasil. 29 a 30 de outubro de 2003

[57] SHARMA, Amit. **A Generic Architecture for User Modeling Systems and Adaptive Web Services**, Proceedings of Workshop on E-Business & the Intelligent Web. (IJCAI, 2001), August 2001.

[58] SHEHORY, Onn, SYCARA, Katia. **The Retsina Communicator,** In Proceedings of Autonomous Agents, Poster Session, 2000.

[59] SILVA JUNIOR, Geovanne Bezerra da. **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, 2003.

[60] SOBRINHO, Antonio Carlos. **Uma Análise das Aplicações dos Algoritmos Genéticos em Sistemas de Acesso à Informação Personalizada**, 2003 (artigo submetido)

[61] SOBRINHO, Antonio Carlos. **Uma Análise dos Algoritmos Genéticos e suas Aplicações em Sistemas de Acesso à Informação**, Monografia do Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís-MA, 2002.

[62] STRACHAN, L., ANDERSON, J., SNEESBY, M, EVANS, M.. **Pragmatic User Modelling in a Commercial Software System**, Proc. Sixth International

Conference on User Modelling, 189-200, Springer, Vienna, New York, 1997.

[63] WEBB, Geoffrey, PAZZANI, Michael, BILLSUS, Daniel. **Machine Learning for User Modeling**, User Modeling and User-Adapted Interaction 11: 19-29, Kluwer Academic Publishers. Printed in the Netherland, 2001.

[64] WEISS, Michael. **On the Use of Patterns in Agent System Design**. School of Computer Science, Carleton University, Ottawa, Canada.

[65] WEISS, Michael. **Patterns for e-Commerce Agent Architectures: Using Agents as Delegates.** Conference on Pattern Languages of Programming (PLoP-01), 2001

[66] WOOLDRIDGE, Michael, JENNINGS, Nicholas. **Intelligent Agents: Theory and Practice**, Knowledge Engineering Review, October 1994, Rev. January 1995.

[67] ZEUS DOCUMENTATION, disponível em: http://193.113.209.147/projects/agents.htm, acesso em julho, 2002.

[68] ZHANG, Xiangmin**, Discriminant Analysis as a Machine Learning Method for Revision of User Stereotypes of Information Retrieval Systems**, Proceedings of UM'03, 9th International Conference on User Modeling Pennsylvania, USA, 2003.