View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repositório Institucional da UFSC

ESTUDO DE UMA METODOLOGIA ORIENTADA A AGENTES – UM PROTÓTIPO PARA UM AMBIENTE VIRTUAL

MARCELO ANTONIO PEROTTO

ESTUDO DE UMA METODOLOGIA ORIENTADA A AGENTES: UM PROTÓTIPO PARA UM AMBIENTE VIRTUAL

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universida de Federal de Santa Catarina como requisito parcial para a obtenção do título de mestre em Engenharia de Produção.

Orientador : Prof. Marcello Thiry Comicholi da Costa, Dr.

Florianópolis

2002

Marcelo Antonio Perotto

ESTUDO DE UMA METODOLOGIA ORIENTADA A AGENTES: UM PROTÓTIPO PARA UM AMBIENTE VIRTUAL

Esta dissertação foi julgada e aprovada para a obtenção do título de **Mestre em Engenharia da Produção** no Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina.

Florianópolis, 25 de Outubro de 2002.

Prof. Ricardo de Miranda Bárcia, Ph.D.
Coordenador do Programa

BANCA EXAMINADORA

Profª. Elizabeth Sueli Specialski. Drª.	**Prof. Marcello Thiry Comicholi da Costa, Dr.** Orientador
Profª. Alessandra Schweitzer, M.Sc.	**Prof. Joao Bosco da Motta Alves, Dr.**

A minha esposa Viviane, que me incentivou durante todo o processo.
Ao meu filho, Vitor Hugo.

AGRADECIMENTOS

À Universidade Federal de Santa Catarina.
Ao orientador, professor Marcello Thiry.
À minha Tutora, Professora Alessandra Schweitzer pelo apoio e dedicação.
À Família Andrade pelo tempo disponibilizado para a realização desta dissertação.
Aos Professores do Curso, pela sua dedicação e atenção.
A todos os que direta ou indiretamente contribuíram para a realização desta pesquisa.

Resumo

PEROTTO, Marcelo Antonio. **Estudo de uma Metodologia Orientada a Agentes: Um Protótipo para um Ambiente Virtual**. Florianópolis, 2002. 120 f. Dissertação (Mestrado em Engenharia de Produção) - Programa de Pós-graduação em Engenharia de Produção, UFSC, 2002.

Pesquisa que aborda o estudo de uma metodologia orientada a agentes e a utilização de uma ferramenta para a especificação de sistemas orientados a agentes tendo como base a especificação de agentes de interface.

Neste trabalho são analisadas metodologias para a modelagem de sistemas baseados em agente, onde a Metodologia MaSE (Multiagent System Engineering) demonstrou ser a mais indicada para a modelagem de um sistema colaborativo, dirigindo o analista por todas as fases de análise e projeto de sistemas MultiAgentes. Foi utilizado o Framework agentTool, que é uma ferramenta que dá suporte a metodologia MaSE, através de um ambiente acionado por menus e diagramas manipuláveis graficamente em janelas. A utilização do Agenttool como ferramenta para projetar e modelar agentes demonstrou-se eficaz, permitindo ao projetista passar rapidamente da fase de análise para a fase de implementação.

palavras-chave : Inteligência Artificial, Agentes Inteligentes, Agentes de Interface, Metodologia Orientada a Agentes, MaSE.

Abstract

PEROTTO, Marcelo Antonio. **Estudo de uma Metodologia Orientada a Agentes: Um Protótipo para um Ambiente Virtual**. Florianópolis, 2002. 120 f. Dissertação (Mestrado em Engenharia de Produção) - Programa de Pós-graduação em Engenharia de Produção, UFSC, 2002.

This research proposes to investigate an agent oriented methodology and the utilization of a tool to project and design agent oriented systems. The model for this investigation is an interface agent system specification.

Agent modeling systems methodologies are analyzed in this work, where the MaSE Methodology (Multiagent System Engineering) demonstrated to be indicated for the modeling of a collaborative system, driving the analyst for all phases of analysis and project of MultiAgents Systems. Is presented the Agenttool Framework, which it is a tool that offer support to the MaSE methodology, by an environment set in motion through menus and adaptable diagrams graphically in windows. The use of the Agenttool as tool to project and shape agents demonstrated to be efficient, allowing the designer to pass quickly belongs of analysis phase to the implementation phase.

Key words: Artificial Intelligence, Intelligent Agents, Interface Agents, Agent Oriented Methodology, MaSE.

SUMÁRIO

Resumo vi
Abstract vii
Lista de Figuras x
Lista de Quadros xii
1 INTRODUÇÃO 1
1.1 Justificativa 2
1.2 Objetivos 3
1.3 Objetivos específicos 4
1.4 Estrutura do Trabalho 4
2 INTERFACES 6
2.1 Histórico 6
2.2 Interação humano-computador 10
2.3 Usabilidade 12
2.4 Dificuldades do projeto de interfaces 13
2.4.1 A complexidade inerente às tarefas e aplicações 13
2.4.2 A variedade de aspectos e requisitos diferentes 13
2.4.3 Teoria e métodos não são suficientes para resolver o problema 14
2.4.4 Dificuldade de se fazer um projeto iterativo 14
2.5 Dificuldades de implementação de interfaces 14
2.5.1 A necessidade de se fazer uma implementação iterativa 14
2.5.2 Programação reativa 14
2.5.3 Multiprocessamento 15
2.5.4 Programação em tempo real 15
2.5.5 Necessidade de robustez 15
2.5.6 Dificuldade de sistematizar testes 15
2.5.7 Complexidade das ferramentas 16
2.5.8 Dificuldade de modularização 16
3 INTELIGÊNCIA ARTIFICIAL 17
3.1 Histórico da Inteligência Artificial 17
3.2 Inteligência Artificial Distribuída 19
4 AGENTES INTELIGENTES 22
4.1 Definição 22
4.2 Agentes Autônomos 23
4.3 Agentes de Interface 28
5 MODELAGEM VISUAL PARA APLICAÇÕES BASEADAS EM AGENTES ... 31
5.1 Metodologias para o Desenvolvimento de Sistemas Orientados a agentes .. 31
5.1.1 Metodologia Gaia 32
5.1.2 Metodologia MaSE (Multiagent System Engineering) 38
5.2 Avaliação da Metodologia 46
6 FRAMEWORKS PARA O DESENVOLVIMENTO DE AGENTES 48
6.1 AgentTool 48
6.2 Architecture type-based Development Environment (ADE) 48
6.3 Ascape 48
6.4 Bee-gent 49
6.5 Bond Distributed Object System 49

6.6 DECAF Agent Framework 49
6.7 FIPA-OS 50
6.8 Gypsy 50
6.9 Hive 50
6.10 JADE 50
6.11 JAFMAS 51
6.12 JATLite 51
6.13 JATLiteBean 51
6.14 JIAC 52
6.15 Open Agent Architecture 52
6.16 SOMA (Secure and Open Mobile Agent 52
6.17 Zeus 53
6.18 AgentTool 53
6.18.1 AgentTool e a metodologia MaSE 54
6.18.2 Análise e Projeto com o AgentTool 54
7 ESPECIFICAÇÃO DO SISTEMA PROPOSTO 65
7.1 Objetivos do Sistema 65
7.2 Diagramas do sistema 68
7.2.1 Hierarquia de objetivos 68
7.2.2 Casos de uso 70
7.2.3 Diagrama de Papéis 78
7.2.4 Diagrama de Modelos de Agentes 80
7.2.5 A Arquitetura dos Agentes 80
7.2.6 Diálogos entre agentes 82
7.2.7 Painel de Desenvolvimento 92
8 CONCLUSÕES E PERSPECTIVAS FUTURAS 94
8.1 Conclusões 94
8.2 Perspectivas Futuras 96
9 FONTES BIBLIOGRÁFICAS 97

Lista de Figuras

Figura 1 –Metodologias de desenvolvimento Orientadas a Agente. 32
Figura 2 – Atributos dos Papéis 34
Figura 3 - Metodologia MaSE e suas etapas 39
Figura 4- Diagrama Hierárquico de Objetivos 41
Figura 5 - Exemplo de diagrama de Casos de Uso 42
Figura 6- Diagrama de Sequência. 42
Figura 7 - Modelagem dos Papéis 43
Figura 8 - Criação de Classes de Agentes 44
Figura 9 - Diagrama de Comunicação de Classes 45
Figura 10 – Interface com o usuário de AgentTool com a hierarquia de objetivos selecionada 55
Figura 11 - Objetivos dos papéis 56
Figura 12 - Descrição dos Casos de Uso 57
Figura 13 - Diagrama de seqüência do Caso de Uso 57
Figura 14 - Diagrama de Papéis 59
Figura 15 - Diagrama de Modelos de Agentes 60
Figura 16 - Componentes de um Agente 61
Figura 17 - Diagrama de estados do componente 61
Figura 18 - Inicializador do diálogo 62
Figura 19 - Resposta do diálogo 63
Figura 20 – Painel de Desenvolvimento, com as instâncias de agentes e seus diálogos. 64
Figura 21 - Interface com o usuário 67
Figura 22- Diagrama de Hierarquia de objetivos 69
Figura 23 – Diagrama de Casos de Usos 71
Figura 24 - Caso de Uso Entra no Atendimento 72
Figura 25 - Caso de Uso Espera Atendimento 73
Figura 26 - Caso de Uso Atender Aluno 74
Figura 27 - Diagramas de Seqüência 75
Figura 28 - Diagrama de seqüência – Entrar 76
Figura 29 - Diagrama de seqüência – Agendamento de Chat 76
Figura 30 - Diagrama de Sequência – Espera de Atendimento 77
Figura 31 - Diagrama de Seqüência - Atender Aluno 77
Figura 32 - Diagrama de Seqüência - Verificar FAQ 78
Figura 33 – Tela para modelagem do Diagrama de Papéis 78
Figura 34 - Diagrama de Papéis 79
Figura 35 - Diagrama de Modelos de Agentes 80
Figura 36 - Arquitetura do Agente Agente_aluno 81
Figura 37 - Descrevendo a Comunicação entre os Agentes 82
Figura 38 - Inicializador do diálogo entre agentes Sair 83
Figura 39 - Resposta do diálogo entre agentes Sair 83
Figura 40 - Diálogo Entrada - Inicialização 84
Figura 41 - Diálogo Entrada – Resposta 85
Figura 42 - Status do Atendimento - inicialização 85
Figura 43 - Status do Atendimento - Resposta 86
Figura 44 - Dados do Usuário – Inicialização 86

Figura 45 - Dados do Usuário - Resposta 87
Figura 46 - Pergunta do Usuário - Inicializador..... 87
Figura 47 - Pergunta do Usuário - Resposta 88
Figura 48 - Resposta do Professor - Inicializador 88
Figura 49 - Resposta do Professor – Resposta 89
Figura 50 - Inicialização do Diálogo - Solicitar Agenda..... 89
Figura 51 - Resposta do Diálogo - Solicitar Agenda 90
Figura 52 - Inicialização do Diálogo - Agendar Chat..... 90
Figura 53 - Resposta do Diálogo - Agendar Chat 91
Figura 54 - Inicialização do Diálogo – FAQ..... 91
Figura 55 - Resposta do Diálogo – FAQ 92
Figura 56 - Painel de Desenvolvimento do Sistema 93

Lista de Quadros

Quadro 1 - Entidades Abstratas e Concretas em GAIA 33
Quadro 2 – Operadores para expressões de sobrevivência 34
Quadro 3 - Esquema para a modelagem dos papéis 35
Quadro 4 - Exemplo de modelagem de papel 36
Quadro 5 - Esquema para a Modelagem das Interações 37
Quadro 6 - Exemplo de modelagem de Interação 37

1 INTRODUÇÃO

A era da informação requer novas estratégias para que as organizações possam reconfigurar o processo de interação com os seus clientes e colaboradores.

O avanço das tecnologias de informação e das telecomunicações torna o acesso a informações cada vez mais desterritorializado e atemporal, acelerando o contato entre governos, empresas e instituições em geral.

Levy (1996) descreve que o movimento de virtualização torna coexistentes os sistemas de proximidades, tornando-nos nômades de estilos e redes de espaços que se transformam diante de nós.

O acesso a tecnologias de informação, deixam as pessoas cada vez mais exigentes a tempo de resposta e a eficácia da comunicação.

Observamos nos últimos anos, uma desordenada marcha de instituições que se atiravam a estas tecnologias, muitas vezes para fincar uma bandeira, sem noção de qual estratégia deveriam utilizar, onde aos poucos, a euforia inicial foi passando e a busca por uma estratégia real de virtualização foi alterando dramaticamente a distância entre organizações virtualizadas das que buscavam apenas o seu lugar no ciberespaço.

Segundo Maes (1994) a “Supervia da informação” irá apresentar uma explosão de novas tarefas e serviços baseados em computadores. Porém, a complexidade deste novo ambiente ira demandar um novo estilo de interação homem-computador, onde o computador tornar-se-á um colaborador ativo e inteligente.

A competição e a conseqüente busca pela audiência dos usuários, levou as organizações a buscarem a interatividade, como por exemplo, a utilização de formulários para preenchimento de informações, a personalização como forma de agregar valor aos serviços e produtos, a troca de informações em tempo real que exigem a inserção de controles cada vez mais complexos para prover a interatividade.

A necessidade de uma nova forma de interação homem-computador, especialmente no que diz respeito a WEB, está levando os projetistas de interface a buscar novas soluções. Dentre as alternativas disponíveis está a de deixar para o computador parte da tarefa de decidir o caminho a ser seguido na comunicação com a interface. Porém, a complexidade envolvida nos múltiplos caminhos que uma ação

na interface pode desencadear, exigirá do algoritmo uma inteligência que possa minimizar a ocorrência de erros. A tarefa de simular esta inteligência na máquina é obtida através de técnicas algorítmicas de inteligência artificial, em especial, a tecnologia de Agentes Inteligentes.

Hermans (1996) comenta que embora a interação do usuário foi transformada pelo advento das interfaces de usuário gráficas (GUIs), para muitos, computadores ainda permanecem difíceis de aprender e utilizar. Enquanto as potencialidades e as aplicações dos computadores melhoram, a interface com o usuário necessita suportar o aumento de complexidade. Enquanto as populações de usuário crescem e se diversificam, a interface do computador precisa aprender hábitos e preferências do usuário e adaptar-se aos indivíduos. Os agentes inteligentes (especificamente os agentes de interface) podem ajudar com estes dois problemas. A tecnologia de agentes inteligentes permite que os sistemas monitorem as ações do usuário, desenvolvam modelos de habilidades do usuário e ajude-os automaticamente quando os problemas se apresentarem. A tecnologia dos agentes inteligentes permite que as interfaces tornem-se mais humanas na sua forma de interação.

Dentro deste contexto, esta dissertação orientou-se em pesquisar as metodologias de projeto de sistemas orientadas a agente, tendo como base a a especificação de um ambiente virtual para a comunicação entre professor e aluno, utilizando-se de agentes para facilitar a interação com a interface.

O trabalho também visa estudar a metodologia de projeto de Sistemas MultiAgente MaSE e a ferramenta AgentTool, que propiciam uma abstração condizente para a complexidade envolvida nos múltiplos caminhos que as ações correspondentes na interface podem desencadear.

1.1 Justificativa

A estratégia de virtualização é um processo evolutivo que permite às organizações disseminar o conhecimento e compartilhar recursos, a fim de ampliar o alcance geográfico ou o tamanho aparente que um concorrente pode oferecer a um cliente, proporcionando a agilidade das comunicações e troca de informações

A complexidade crescente das interfaces, proporcionada pela busca por interatividade como forma de agregar valor aos sites da Internet, têm criado

dificuldades para os usuários. Eles precisam adaptar-se a cada nova página visitada, onde as ferramentas mudam constantemente e onde não compensa perder tempo com treinamento em um ambiente tão dinâmico, e que por vezes, tão pouco utilizado.

O crescimento anual da taxa de interatividade de interfaces (menus, e opções adicionadas) é insustentável e se for mantida a correspondência um-para-um (usuário-interface) chegaremos a ponto de que não poderá ser adicionada mais funcionalidade aos sistemas. Agentes de interface são a saída para este dilema. (LIEBERMAN, 1997)

O desenvolvimento de sistemas orientados a agentes exige que o analista seja dirigido por todas as fases de análise e projeto de sistemas MultiAgentes.

A conceitualização e implementação de sistemas está progressivamente sendo melhor compreendida pelos engenheiros de software (WOOLDRIDGE 2001). Mas, em sistemas onde existe uma rede complexa de protocolos de coordenação e onde a computação é baseada em processos concorrentes, exige-se novos modelos de abstração e ferramentas.

Programadores acostumados com a metodologia orientada a objeto, falham em perceber as diferenças entre um agente e um objeto, dada a tendência a antropomorfizar um objeto. (NeXT, 2001 Apud WOOLDRIGE)

Este trabalho servirá como um guia para a utilização de uma metodologia orientada a agentes, tendo como modelo de análise uma aplicação baseada em Agentes de Interface, suficientemente complexa para justificar a utilização da ferramenta de análise orientada a agentes que será estudada.

1.2 Objetivos

O objetivo geral deste trabalho é o estudo e aplicação de metodologias para desenvolvimento de projetos utilizando agentes inteligentes e especificação de um ambiente virtual, através da utilização de Agentes de Interface para facilitar a interação entre usuários e a interface, reduzindo a complexidade de acionamentos que seriam necessários para recuperar informações neste ambiente. Neste sentido, este trabalho serve como um guia para a metodologia de Sistemas MultiAgente

MaSE e para a ferramenta AgentTool, utilizadas para a Análise e Projeto de Sistemas MultiAgente.

1.3 Objetivos específicos

Estudar e Avaliar algumas arquiteturas disponíveis para a especificação de agentes de interface.

Estudar e Avaliar algumas metodologias para a Análise e Projeto de sistemas baseados em agentes.

Modelar um Sistema utilizando uma das arquiteturas e metodologias estudadas.

Desenvolvimento de um guia para a utilização da metodologia MaSE e do Framework AgentTool.

1.4 Estrutura do Trabalho

O presente trabalho está organizado em oito capítulos, conforme segue.

No capitulo 2 é apresentada a fundamentação teórica relacionada a interfaces, enfatizando a interface homem-máquina, os problemas e estudos efetuados na área, as tecnologias de agentes de interface, conceitos, estudos e aplicações em interfaces com o usuário.

O capítulo 3 descreve o desenvolvimento da Inteligência Artificial e a inteligência artificial distribuída.

No capítulo 4 são apresentados os Agentes Inteligentes, como a evolução da inteligência artificial distribuída, os agentes autônomos e a definição de vários autores para agente e descreve os Agentes de Interface, a sua utilização e categorização.

O Capítulo 5 aborda metodologias para a análise e projeto de sistemas orientados a agentes.

No capítulo 6 são apresentados alguns Frameworks disponíveis atualmente para o desenvolvimento de agentes, destacando o framework AgentTool.

No capítulo 7, é especificada a etapa de análise do sistema de atendimento a alunos através da metodologia MaSE, onde são construídos os diagramas usando a ferramenta AgentTool.

Estão descritos no Capítulo 8, a análise, conclusão e recomendações para trabalhos futuros.

2 INTERFACES

O acesso à informação em um sistema automatizado se dá através da interface , pela velocidade de acesso e pela facilidade de utilização está intimamente ligada a sua complexidade. Neste capítulo é apresentado um breve histórico sobre a interação homem-computador e descrição da importância da interface.

A partir da década passada, a área da interação homem-computador tem saído do âmbito dos pesquisadores para se tornar parte da vida diária de muitas pessoas. Atualmente, a interface com o usuário é o primeiro item a ser questionado quando se discute um novo software aplicativo.

Para o usuário, a interface é a representação do sistema como um todo. O usuário entende que a comunicação com o sistema é tão importante quanto a computação efetuada pelo mesmo. (Hix 1993)

2.1 Histórico

Para que a relação homem-máquina ocorra, é indispensável o uso das interfaces e da interatividade. Sem estes dois fundamentos, é impossível haver qualquer tipo de relação homem-máquina dentro de qualquer tipo de aplicativos ou sistemas operacionais.

As máquinas fazem parte da vida do ser humano há muito tempo, desde que os homens das cavernas passaram a caminhar e sobreviver pela Terra. Neste caso deve-se atribuir ao termo máquina um novo significado: tudo que ajuda ao homem a desempenhar algum tipo de função com maior facilidade.

Mais tarde, com o aparecimento dos primeiros artesãos e depois com a Revolução Industrial, a máquina passou a fazer parte da vida do ser humano com seu sentido real, ou seja, foi criada para auxiliar o homem sendo uma extensão dele próprio, executando tarefas que o homem não podia, trazendo então inúmeros benefícios à ele.

Durante a Revolução Industrial já haviam discussões à respeito da exclusão que a máquina proporcionava junto a todos aqueles benefícios à vida do ser humano. Ainda naquela época muitos homens perderam seus empregos por causa das máquinas e os que ficavam tinham maior contato com elas do que com as pessoas que com que trabalhavam. Hoje não é diferente, só que as máquinas que fazem

parte da nossa realidade são os computadores. Desde a criação do ENIAC (*Eletronic Numeral Integrator and Computer*), os computadores não pararam de evoluir e de fazer parte da vida humana.

A espécie humana depende tanto dos computadores, que muitas empresas, hospitais, indústrias e mesmo pessoas comuns não conseguiriam viver sem o auxílio desta máquina. Inúmeras pesquisas vêm acompanhando a relação que ocorre entre o ser humano (Homem) e os computadores (máquinas), tanto que a HCI (*Human Computer Interaction*) foi instituída para estudar a relação de interação homem-computador (HIX 1993). E esta está tão presente que os computadores futuros serão projetados para atender às necessidades do usuário, reconhecê-las e entender sua linguagem verbal e não-verbal, já que a nossa linguagem usual não é adequada para o relacionamento homem-máquina.

Os dois significados empregados ao termo máquina são pertinentes, pois o computador é uma máquina que ajuda ao homem a desempenhar algum tipo de função com maior facilidade (lembrando a definição de interface, que será discutida mais à frente, justamente pelo computador ser uma interface) e foi criado para desempenhar tarefas que o homem não poderia realizar (EVOLUTION NETWORK, 2001)

Conceituando interfaces, Lévy (1993) descreve que para que dois elementos funcionem em conjunto, é necessária uma conexão, e a ponte onde é feita essa conexão chama-se interface.

As interfaces são usadas constantemente e estão presentes em tudo, tornando-se de fundamental importância. Muitas vezes, sua existência nem é lembrada, mas durante a utilização de um sistema, são necessárias várias interfaces só para entrar nele, desde mouse até o próprio Windows.

Na informática há diferentes tipos de interface, distribuídas em camadas, variando desde as interfaces visíveis ao usuário, (que permitem a eles a comunicação com os programas) até as invisíveis, porém necessárias, como as interfaces dos softwares, que conectam dispositivos e componentes dentro dos computadores.

A interface contribui para definir todo o modo de captura da informação oferecida aos usuários. Elas abrem, fecham e orientam os domínios de significação e de utilização possíveis de uma mídia.

Segundo Lévy (1993), tudo aquilo que é tradução, transformação e passagem, é da ordem da interface: "Se todo processo é interfaceamento, e, portanto tradução, é porque nenhuma mensagem se transmite tal qual em um meio condutor neutro, mas antes deve ultrapassar a descontinuidade que a metamorfoseia..."

Construir interfaces é o que fazem a maior parte dos engenheiros, programadores e consultores. Para projetar uma interface é necessário analisar detalhadamente a própria interface. Tal análise pode se dar através da especificação de requisitos, módulos de qualidade e perfil dos usuários.

Para uma interface ser compreensível, agradável e controlável é necessário que esta tenha sido bem projetada. quando isso ocorre os usuários se sentem satisfeitos e responsáveis pelas ações.

Segundo (GUIMARÃES, 1997), a maioria das aplicações utilizam-se das interfaces gráficas GUI (Grafical User Interface) e algumas já estão evoluindo o conceito, deixando de ser voltadas à aplicação para serem voltadas ao objeto, utilizando as OUI (Object User Interface). Paralelamente, com a crescente utilização da Internet, surgiram as WUI (Web User Interface) amplamente defendidas dentro da World Wide Web.

A tecnologia é rápida, mas confusa e nem toda interface facilita o acesso. Um exemplo disso é a interface de um aparelho celular, tão cheio de características e funções especiais, que se tornam por demais complicadas.

Por este motivo, as futuras interfaces estão sendo projetadas com o objetivo de atender a todas as necessidades dos usuários, sem que elas ocupem espaço (físico ou de memória, por exemplo) desnecessário ou possuam finalidades que não sejam proveitosas aos usuários. A grande idéia é que elas possam se adequar a todos os requisitos do usuário, para facilitar as relações e a interatividade com o computador. (LÉVY, 1993)

Deve-se pensar em como interfacear o controle com o comando, como proporcionar a melhor integração do homem com a máquina. Para esta finalidade utilizavam-se com freqüência anunciadores de alarmes, sinaleiros chaves seletoras

botoeiras, etc..., que permitiam comandar ou visualizar estados definidos com ligado e desligado, alto ou baixo, temperatura elevada ou normal, mas não permitia visualizar os valores de alto, quanto alto, ou normal quão normal. Surgiram então os displays e chaves digitais (*thumbweel switchs*) que no caso dos *displays* permitia visualizar os valores das variáveis do processo, bem como mudar parâmetros pré-definidos como, por exemplo, temporizações através das chaves digitais. (MARTIN, 1999)

No entanto, este tipo de interface trazia dois problemas claros, o primeiro a dimensão da superfície do painel, que por muitas vezes necessitava de ser ampliada, somente para alojar tantos botões, ou informações que eram necessárias.

Com o desenvolvimento das IHM's (Interfaces Homem-Máquina) com visores alfanuméricos, teclados de funções e comunicação via serial com o dispositivo de controle, o qual muitas vezes era um CP (Controlador Programável), estas traziam consigo os seguintes benefícios:

- Economia de fiação e acessórios, pois a comunicação com o CP seria serial com um ou dois pares de fio trançados, economizando vários pontos de entrada ou saída do CP, e a fiação deste com os sinaleiros e botões.
- Redução da mão-de-obra para montagem, pois ao invés de vários dispositivos, agora seria montado apenas a IHM.
- Diminuição das dimensões físicas do painel.
- Aumento da capacidade de comando e controle, pois a IHM pode ajudar em algumas funções o CP, com, por exemplo, massa de memória para armazenar dados, entre outros.
- Maior flexibilidade frente a alterações no campo.
- Operação amigável.
- Fácil programação e manutenção.

A evolução seguinte foi a utilização de interfaces gráficas ao invés de alfanuméricas.

Quando utilizadas, as interfaces gráficas, em alguns casos mais simples substituem os chamados Sistemas Supervisórios (sistemas utilizados para aquisição e gerenciamento de dados provenientes dos Controladores Programáveis).

Quando usadas em Sistemas de Controle, integradas a Sistemas Supervisórios, além das funções das IHM's alfanuméricas já citadas, executam também funções de visualização que aliviam o Sistema Supervisório para que a performance das funções de supervisão, alarme, tendências, controle estatístico de processo entre outras possa ser elevada. (MARTIN, 1999).

Tradicionalmente, os aplicativos e sistemas operacionais sempre foram complicados de utilizar. Os primeiros computadores digitais eram operados através da conexão/desconexão de cabos em um painel de controle, o que tornava a entrada de dados um processo trabalhoso e demorado.

Os primeiros PC's, com o sistema operacional mais popular na época (MS-DOS), também possuíam uma interface baseada em texto, em vista da pouca quantidade de memória e poder de processamento disponível (MILLER, 2002) .

Caudill (2000), observa que o principal entrave a difusão do computador era a dificuldade de uso. As pessoas tinham que decorar mnemônicos e combinações de teclas para escrever uma simples carta usando o editor de texto *Wordstar.*

Segundo Torres (1998), no final da década de 70, o PARC (Palo Alto Research Center) era um centro de pesquisas financiado pela Xerox. Nesta época, vários pesquisadores estavam desenvolvendo sistemas capazes de melhorar a relação homem/máquina, a fim de desenvolver computadores mais fáceis de utilizar. A principal novidade era a interface gráfica ao invés da interface texto. Os comandos não necessitavam mais ser entrados via teclado sob a forma de textos, mas sim, ícones eram selecionados e reconhecidos de forma muito mais rápida.

Steven-Jobs co-fundador da *Apple Computer* levou a idéia da interface gráfica para computadores pessoais e criou o primeiro microcomputador com sistema operacional baseado em interface gráfica, o projeto Lisa e paralelamente o projeto do Macintosh, o primeiro microcomputador pessoal com interface gráfica a ter sucesso comercial (TORRES, 1998)

2.2 Interação humano-computador

CYBIS (2000) define Interação Humano-Computador (IHC) como o processo de comunicação entre dois sistemas cognitivos que fazem tratamento simbólico de informações. De um lado, o ser humano com sua estrutura de tratamento cognitivo

baseado em representações, e do outro o computador, que trata os sinais produzidos pelos programadores de modo a que usuários interpretem e manipulem a sua interface.

Segundo (Adersen, 1993 Apud CYBIS) programar no sentido semiótico do termo, é tentar dizer algo as pessoas através do computador. Na comunicação com o usuário, os símbolos propostos pelo programador se realizam como sinais. O projetista de sistemas informatizados se comunica como emissor ou receptor com o usuário através do computador.

PREECE (1993), define Interação Humano-Computador como o estudo a compreensão da forma de como as pessoas utilizam os sistemas de computador e de que forma os sistemas podem ser projetados para preencher as necessidades dos usuários. O conhecimento nesta área vem da compreensão da maneira como os usuários interagem com sistemas de computadores com as particularidades de cada ambientes de trabalho. Esta interação é composta de quatro componentes:

- o usuário
- quem faz uma tarefa ou trabalho específico
- um contexto particular
- um sistema de computador

Cada componente destes possui as suas próprias características, cada qual influenciando a natureza da interação entre usuários e sistemas de computador.

O trabalho de um usuário de computador é composto por várias tarefas que depois devem ser quebradas em pequenas unidades (sub-tarefas) através da comunicação com o usuário.

A interface com o usuário possui uma forma específica de diálogo que deve ser projetada para facilitar a interação usuário-computador. Este diálogo deve habilitar o usuário a relatar os detalhes das tarefas para a funcionalidade do sistema.

É na interface com o usuário que a comunicação entre projetista e usuário se realiza. Para que esta comunicação seja eficiente, é necessário que o sistema tenha alto grau de usabilidade, que será discutido no próximo tópico.

2.3 Usabilidade

Usabilidade sob o ponto de vista técnico é a combinação das seguintes características orientadas ao usuário (Shneiderman apud Hix , 1993):

- Facilidade de aprendizagem
- Desempenho em alta velocidade de tarefas do usuário
- Baixa taxa de erros
- Satisfação subjetiva do usuário
- Manter a atenção do usuário

Usabilidade é relacionada com a eficiência e eficácia da interface e da reação do usuário com a interface. A naturalidade da interface com o usuário é também um importante aspecto da usabilidade (Hix 1993).

O objetivo da maioria dos trabalhos em interação homem-computador é, de uma maneira ou de outra, prover ao usuário um alto grau de usabilidade. (Hix 1993).

É um mito que o sucesso na confecção da interface com o usuário será conseguido na primeira tentativa. Reis (2000), defende que um seja adotado um processo de refinamento sucessivo que avalie freqüentemente os resultados a partir das metas pré-definidas no projeto da interface.

Os pesquisadores mostram que a vasta maioria de problemas de usabilidade vem de uma única causa: a equipe de desenvolvimento não conhece (ou não sabe) qual a peça chave da informação. Se eles tivessem conhecido tais informações mais cedo, teriam projetado o produto para acomodá-lo e o problema de usabilidade nunca teria ocorrido. (REIS, 2000)

Os projetos concebidos sem a preocupação com a usabilidade podem prejudicar a interação do usuário com o sistema, seja pela dificuldade de identificação das ações, respostas ambíguas ou comportamento imprevisível.

Os efeitos podem causar demora, prejudicar ou inviabilizar a execução de uma tarefa.

Segundo CYBIS 2000, os problemas de usabilidade têm como variáveis: o tipo de usuário, o tipo de tarefa, equipamentos e ambiente onde as tarefa são realizadas, podendo ser classificados como problemas de barreira, obstáculo e de ruído:

- Barreira: função da interface com o usuário que apresenta problemas, e mesmo com a insistência do usuário em utilizá-la, não apresenta sucesso, levando-o a desistir de sua utilização.

- Obstáculo: Problema na interface do sistema que o usuário aprende a superar depois de seguidas tentativas. Causa perda de desempenho e prejuízos na operação e aprendizado do sistema.

- Ruído: Má impressão causada por um aspecto subjetivo na função da interface. Causa uma diminuição no desempenho do usuário em relação à tarefa.

Os problemas de interface podem manifestar-se em funções amplamente utilizadas e neste caso são classificados como problemas de usabilidade principal. Caso o problema seja em uma função pouco utilizada da interface, o mesmo é classificado como problema de usabilidade secundário.

2.4 Dificuldades do projeto de interfaces

Embora os benefícios da melhora da usabilidade de uma interface sejam indiscutíveis, não estão resolvidos os problemas que levam à dificuldade do projeto de uma Interface. Discute-se algumas dificuldades abaixo: (MYERS, 1993)

2.4.1 A complexidade inerente às tarefas e aplicações

Em geral, o domínio da aplicação a ser criada envolve situações de difícil modelagem - seja porque a tarefa em si é complicada, seja porque a aplicação se propõe a resolver problemas de gama extensa. (MYERS, 1993)

2.4.2 A variedade de aspectos e requisitos diferentes

Além das limitações inerentes a qualquer projeto, interfaces com o usuário envolvem questões como padrões, *design* gráfico, documentação, internacionalização, performance, entre outras. Estas questões associadas contribuem para aumentar a complexidade do desenvolvimento da interface. (MYERS, 1993)

2.4.3 Teoria e métodos não são suficientes para resolver o problema

Embora existam muitas metodologias para a criação de uma interface boa, a maior parte dos estudos feitos a seu respeito revela que a habilidade dos projetistas é o fator primário para a qualidade das interfaces geradas. O fato de existir grande proporção de casos que sejam exceções às regras propostas nos métodos, contribui para a dificuldade de se criar um método abrangente. (MYERS, 1993)

2.4.4 Dificuldade de se fazer um projeto iterativo

Embora se veja como ideal o processo de se refinar ciclicamente uma interface, este processo em si já é difícil de ser executado muitas vezes. As modificações trazem uma piora de usabilidade, e é difícil saber quando a interface está realmente bem-elaborada. Além disso, é difícil obter resultados do uso da interface diretamente dos usuários primários, que muitas vezes não são os seus compradores e nem responsáveis. (MYERS, 1993)

2.5 Dificuldades de implementação de interfaces

Além de serem difíceis de se projetar as interfaces com o usuário, também são difíceis de se implementar. Há estudos que quantificam em 50% o tempo e a quantidade de código relacionado diretamente à interface. Há razões importantes que explicam porque software relacionado à interface está entre os mais difíceis de se implementar. (MYERS, 1993)

2.5.1 A necessidade de se fazer uma implementação iterativa

Implementar a interface iterativamente significa entrelaçar as etapas de projeto e implementação e teste; o impacto direto disto é que o processo de software tradicional se torna inadequado, e o processo utilizado em si, mais difícil se administrar e documentar. (MYERS, 1993)

2.5.2 Programação reativa

Uma das diferenças principais entre a implementação de software para a interface e software de outros tipos é que o software deve ser escrito contemplando

seu controle pelo usuário, e não pela aplicação. Esta diferença tem mostrado resultar em uma maior dificuldade na modularização e organização do código (MYERS, 1993).

2.5.3 Multiprocessamento

Além de existirem diversas tarefas simultâneas que necessitem responder à atenção do usuário, o tempo de execução destas tarefas varia e é importante manter o sistema respondendo instantaneamente. Por este motivo, os mesmos problemas relacionados ao multiprocessamento, sincronização, controle de concorrência e consistência das informações, existem no desenvolvimento de interfaces (MYERS, 1993).

2.5.4 Programação em tempo real

Há limitações importantes do tempo de resposta e do redesenho das informações exibidas pela interface; esta característica embute preocupações que a maior parte dos sistemas não contempla a fundo. (MYERS, 1993)

2.5.5 Necessidade de robustez

Toda entrada do usuário deve ser atendida satisfatoriamente. Isto difere em muito da robustez costumeira que se deseja em um software. As entradas dos usuários podem ser de qualquer natureza, e muitas vezes é difícil prever todos os tipos diferentes de interação que ocorrerão. A necessidade de cancelar e desfazer as operações que o usuário executa acrescenta a esta dificuldade (MYERS, 1993).

2.5.6 Dificuldade de sistematizar testes

Sistemas de manipulação direta à interface são em geral pouco úteis para efetuar seu teste, porque não simulam bem a atividade caótica do usuário, e porque alterações na interface implicam em se ter de fazer mudanças nos testes. Além disso, grande parte da qualidade da interface é não-avaliável por um sistema, por ser resultado de apreciações essencialmente humanas (MYERS, 1993).

2.5.7 Complexidade das ferramentas

Como a linguagem de programação muitas vezes não suporta completamente as necessidades da interface, cria-se uma necessidade por ferramentas que enderecem esta questão. No entanto, estas ferramentas em geral são incompatíveis e requerem longo treinamento por parte dos desenvolvedores (MYERS, 1993).

2.5.8 Dificuldade de modularização

Uma das maiores dificuldades relacionadas à criação de interfaces é a dificuldade de se separar a forma da interface com o conteúdo a ser apresentado, muitas vezes, mudanças na interface implicam em mudanças na funcionalidade da aplicação. Além disso, o uso extensivo de funções *call-back* nas ferramentas tornam o problema ainda maior, porque cada interação com uma parte da interface (um *widget*) requer uma função correspondente (MYERS, 1993).

Todas estas dificuldades apresentadas, estão levando os projetistas de interface a buscar novas soluções para o interação humano-computador, e dentre as alternativas disponíveis, está a de deixar para o computador parte da tarefa de decidir o caminho a ser seguido na comunicação com a interface. A complexidade envolvida nos múltiplos caminhos que uma ação executada na interface pode desencadear, exigirá do algoritmo uma inteligência que possa minimizar a ocorrência de erros. A tarefa de simular esta inteligência na máquina é obtida através de técnicas algorítmicas de inteligência artificial que são descritas no próximo capítulo.

3 INTELIGÊNCIA ARTIFICIAL

Segundo Maes (1994) a “Supervia da informação” irá nos apresentar uma explosão de novas tarefas e serviços baseados em computadores. Porém, a complexidade deste novo ambiente ira demandar um novo estilo de interação homem-computador, onde o computador tornar-se-á um colaborador ativo e inteligente.

Para que o computador deixe de ser passivo na interação com o usuário e passe a colaborar com o mesmo, torna-se necessário o uso de técnicas de inteligência artificial.

A Inteligência Artificial (I.A) é o estudo de como fazer os computadores realizarem tarefas em que, no momento, as pessoas são melhores (RICH, 1988).

Outra definição citada em (FLACH 1991) é de que a Inteligência Artificial é o estudo dos meios pelos quais os computadores possam realizar tarefas que requisitariam inteligência se efetuadas por humanos.

A inteligência artificial é um campo da Ciência da Computação que ainda está em desenvolvimento, mas que já conta com soluções para muitos dos problemas específicos que somente um ser humano treinado poderia resolver.

3.1 Histórico da Inteligência Artificial

Em meados dos anos 50, os cientistas pensavam que com o desenvolvimento dos computadores, as pessoas ficariam envolvidas apenas com atividades recreativas, deixando o trabalho para as máquinas fazerem (Gevarter, 1984).

Na década de 60 muitos avanços foram obtidos com pesquisas em I.A (Inteligência Artificial), dentre eles problemas de jogos e provas de teoremas, solução de problemas gerais que manipulavam expressões lógicas de forma simbólica, pesquisas avançavas para desenvolver meios para a percepção (visão e fala), compreensão da linguagem natural e resolução de problemas em domínios especializados, como diagnose médica e análise química. (GEVARTER, 1984).

Brehm (1997) descreve que durante os anos 80, especialistas em I.A desenvolveram sistemas para resolver problemas da área de otimização de análise em oleodutos nas industrias químicas. Alguns destes sistemas tiveram um sucesso espetacular, com seu retorno medido em horas.

A despeito do sucesso de alguns sistemas especialistas em I.A durante os anos 80, Brehem (1997) comenta que Esther Dyson, editora da Release 1.0 (um boletim de notícias mensal) fez a previsão que a I.A não se tornaria realmente importante comercialmente até que a I.A estivesse envolvida em sistemas estrategicamente importantes.

O tempo tem provado que a predição de Dyson estava correta. Nos anos 90, o campo da Inteligência Artificial, cresceu em importância conforme a sua ênfase se deslocou para a troca de caros especialistas humanos por sistemas especialistas que criam estratégias avançadas. Muito dos sistemas I.A atuais estão conectados a grandes bancos de dados, onde eles manipulam dados, falam com redes, gerenciam ruído e corrupção de dados com estilo e graça, estão implementados em linguagens populares, e rodam em sistemas operacionais padrões. (WINSTON, 1997)

A inteligência artificial mudou suas metas gradativamente ano após ano, do sonho de construir uma inteligência comparável a de um ser humano, até os objetivos atuais de fazer com que os computadores se tornem ferramentas mais eficientes para representar o conhecimento. (BITTENCOURT, 1998)

A inteligência artificial esta dividida em duas linhas de pesquisa principais para a construção de sistemas inteligentes. A linha conexionista e a linha simbólica.

A linha conexionista estuda a modelagem da inteligência nos sistemas através da simulação dos neurônios cerebrais e suas interligações, dando origem a área de pesquisa das redes neuronais artificiais.

A linha simbólica segue a tradição lógica, empregando a manipulação de símbolos sobre fatos especializados para representar o conhecimento. (BITTENCOURT, 1998)

O poder computacional distribuído levou a pesquisa na área de Inteligência Artificial a buscar soluções distribuídas de problemas, pela divisão do problema em partes menores e pela especialização de sistemas para a resolução de problemas específicos. Levando as pesquisas em inteligência artificial para uma nova modalidade demonstrada a seguir.

3.2 Inteligência Artificial Distribuída

O crescimento de redes de computadores e incremento do poder computacional fizeram a Inteligência Artificial Distribuída (IAD) se tornar uma área proeminente.

Segundo Sighn e Huhns (1993), a Inteligência Artificial Distribuída preocupa-se com a maneira pela qual um grupo de agentes computacionais deve coordenar as suas atividades para atingir os seus objetivos.

Sighn e Huhns (1993) descrevem que a inteligência Artificial Distribuída é a tecnologia apropriada para aplicações onde:

- a experiência está distribuída, como em um projeto;
- a informação esta distribuída, como na automação de um escritório;
- dados estão distribuídos, como acontece em sensores distribuídos;
- decisões são distribuídas, como em um controle de manufatura; e

As bases de conhecimento são desenvolvidas de forma independente, mas devem ser interconectadas ou reutilizadas, como na próxima geração da engenharia de conhecimento.

Segundo Bittencourt, (1998), no fim dos anos setenta, o modelo computacional de Von Neumann com concepções centralizadas e de “não reusabilidade“ de componentes é contrabalançada pelos trabalhos que estavam sendo pesquisados na área de redes neuronais, que incluem distribuição e paralelismo, e pelo modelo de quadro negro, onde as fontes de conhecimento e o espaço da solução são estruturadas de forma hierárquica, tendo as características de sistemas distribuídos.

Bittencourt (1998), descreve este novo enfoque de distribuição em I.A, onde o desenvolvimento de redes e sistemas distribuídos deu origem à área de IAD (Inteligência Artificial Distribuída). Nesta área desenvolveram métodos para a resolução distribuída de problemas, que visam:

- Melhorar a adaptabilidade, confiabilidade e autonomia do sistema.
- Reduzir os custos de desenvolvimento e manutenção.
- Aumentar a eficiência e velocidade.
- Permitir a integração de sistemas inteligentes existentes de maneira a aumentar a capacidade de processamento e, principalmente, a eficiência na solução de problemas.
- Permitir a integração dos computadores nas redes de atividades humanas.

Os principais problemas tratados pela IAD são:

- Formular, descrever, decompor e alocar problemas e sintetizar resultados em um grupo de agentes inteligentes.

- Permitir a comunicação e interação entre agentes.

- Assegurar a coerência entre agentes a respeito de suas decisões e ações e evitar interações indesejadas.

- Permitir que agentes individuais raciocinem a respeito das ações, planos e conhecimentos dos outros agentes visando à coordenação e à cooperação entre os agentes.

- Criar metodologias e projetar ambientes para a implementação de sistemas inteligentes distribuídos.

Bittencourt (1998), descreve que a Inteligência Artificial Distribuída - IAD se divide em Solução Distribuída de Problemas (SDP) e Sistemas MultiAgentes (SMA).

A SDP deriva da I.A simbólica, com seu foco principal no problema, e tem o objetivo distribuir os problemas complexos entre computadores em rede na forma de sub-tarefas para agentes de propriedade variável.

A SMA estuda a cooperação entre agentes para a resolução de problemas em sociedade, e tem em seu foco principal o agente com suas propriedades preestabelecidas.

A SMA usa a metáfora de comunidade inteligente, onde o comportamento social é a base para a inteligência do sistema.

Os membros de uma comunidade inteligente podem ser desde extremamente simples (reativos), com um modelo de funcionamento baseado em estímulo-resposta até extremamente complexos (cognitivos), que dispõem de memória e podem planejar as suas ações futuras.

Conforme (MULLER apud GOTTGTROY, 1998), um sistema MultiAgente é percebido como um agente e seu ambiente. A descrição do ambiente é dependente do modelo utilizado para conceber os agentes do sistema. Em quase todos os modelos, o ambiente é descrito pela representação interna do ambiente que o agente é capaz fazer e por sua capacidade de produzir alterações na representação do ambiente

A principal vantagem desta concepção é a queda da complexidade do sistema em conseqüência da funcionalidade específica e independência de cada agente, onde um agente deve ser capaz de realizar tarefas sem intervenção humana ou de outros agentes

4 AGENTES INTELIGENTES

Para o trabalho proposto nesta dissertação, foi escolhido o enfoque de sistemas MultiAgentes da IAD por apresentar as características necessárias de distribuição de tarefas e aprendizagem. Estas características são desejáveis em ambientes complexos tal como interfaces com o usuário, onde cada novo usuário tem um comportamento diferente frente a interface, exigindo uma readaptação do sistema.

Neste capítulo são apresentadas as características dos Agentes Inteligentes, que são os “personagens” dos sistemas MultiAgentes.

4.1 Definição

Hayes-Roth et all. (1994) caracteriza um Agente Inteligente como um sistema versátil e adaptativo que desempenha diversos comportamentos no seu esforço para atingir múltiplos objetivos em um ambiente dinâmico e de incerteza.

Conforme Kay, Hewitt e Bond (apud THIRY 1999), a idéia de se utilizar agentes para executar uma tarefa dada uma seqüência de ações foi originada em meados da década de 50 com John McCarthy e Oliver Selfridge.

Na década de 70, Hewitt (1977) concebeu um modelo de agente auto-suficiente denominado ator. Com alguns conceitos de orientação a objeto, este modelo apresenta alguns estados internos encapsulados e responde a mensagens de outros objetos similares.

Wooldridge & Jennings (1995) demonstrou a definição do termo agente, caracterizando como os agentes tomam um papel ativo, originando ações que afetam seu ambiente, em vez de permitir passivamente que seu ambiente os afete. Duas condições que costumam descrever freqüentemente as ações dos agentes são autonomia e racionalidade. A autonomia significa geralmente que um agente opera sem intervenção ou direção humana direta. De forma geral, a racionalidade é a suposição que um agente irá perseguir os seus objetivos, e não agirá de forma a impedi-los ao menos que suas crenças permitam. Wooldridge & Jennings (1995) descrevem que a racionalidade não é tão facilmente captada, mas é freqüentemente demonstrada com modelos lógicos da Teoria dos Jogos.

Antes de se definir o termo Agente, com base em diversos autores, é necessário demonstrar o significado de Agência, pois é a característica fundamental de todo

agente. Segundo Hermans, (1996), o grau de autonomia e autoridade investidos no agente é chamado de Agência. A Agência pode ser medida, pelo menos qualitativamente, pela natureza da interação entre o agente e outras entidades no sistema em que ele se encontra. No mínimo, um agente deve funcionar assincronamente e o seu Grau de Agência é aumentado se um agente representa um usuário de algum modo. Um agente mais avançado pode interagir com outras entidades como dados, aplicações, ou serviços. Estes agentes mais complexos colaboram e negociam com outros agentes.

Definido o termo Agência, descreve-se a opinião de diversos especialistas na área sobre a definição de Agentes Inteligentes.

Carl Hewit afirmou que a questão “O que é um agente” é tão difícil de responder para a comunidade de computação baseada em agentes como a questão “O que é inteligência” para a comunidade de Inteligência Artificial.

Wooldridge & Jennings (1995) constatam a existência de duas definições para o termo “agente” de acordo com o contexto da agência:

-Noção fraca de Agência: A forma mais usual de utilização do termo agente, especifica um hardware ou um sistema de computador com características de autonomia, habilidade social, reatividade e pró-atividade.

-Noção forte de Agência: Para alguns pesquisadores, a definição de agente tem um sentido mais específico. Além das características citadas acima, o agente apresenta conceitos comumente atribuídos a humanos como: convicções, anseios e intenções.

Hermans (1996) discorrendo sobre a inteligência dos agentes, cita que é difícil definir o que faz exatamente um agente "inteligente", sendo ainda assunto de muitas discussões no campo de Inteligência Artificial, e uma definição exata ainda tem que ser encontrada .

4.2 Agentes Autônomos

Luck & d'Inverno (1995) defendem que a utilização dos termos Agentes e Agentes Autônomos seja baseada no significado dos termos, e propõe três camadas hierárquicas de entidades: objetos, agentes e agentes autônomos usando a linguagem de especificação Z (SPIVEY, 1998).

Em sua hierarquia, o autor baseia-se no fato de que toda a entidade é um objeto e neste conjunto de objetos, alguns destes são agentes e dentro destes, alguns são agentes autônomos.

O autor apresenta um objeto como em entidade com um conjunto de ações e atributos, definindo atributo como uma característica perceptível.

Em sua definição de agente, o autor o cita como um objeto com um propósito associado ao mesmo ou um conjunto de propósitos. Quando este agente possui uma motivação que o leva a agir, e alguns meios de avaliar o ambiente em que se encontra para alterar suas motivações, este agente é considerado um agente autônomo.

Franklin & Graesser(1996), realizaram um estudo para propor uma definição formal para agentes autônomos que o distinguissem claramente de outros programas de computador. Eles verificaram que os estudiosos envolvidos nas pesquisas sobre agentes ofereciam uma variedade ampla de definições, cada um desejando explicar o seu uso da palavra agente.

Estas definições vão das simples até as complexas e exigentes, e são apresentadas a seguir.

a) Agente MuBot

“O termo agente é usado para representar dois conceitos ortogonais. O primeiro é a habilidade do agente para execução autônoma. O segundo é a habilidade do agente para raciocinar orientado ao domínio." (VIRDHAGRISWARAN, IN: FRANKLIN E GRAESSER, 1996).

Franklin & Graesser(1996) identificaram que no conceito de agente de (VIRDHAGRISWARAN, IN: FRANKLIN E GRAESSER, 1996), que a execução autônoma é claramente a parte central da agência.

b) Agente AIMA

“Um agente é qualquer coisa que pode perceber seu ambiente através de sensores e agir sobre este ambiente.” (RUSSEL E NORVIG, IN: FRANKLIN E GRAESSER, 1996).

AIMA é um acrônimo para "Artificial Intelligence: a Modern Approach," um texto bem sucedido de I.A que foi usado em 200 academias e universidades em 1995. Franklin & Graesser diz que os autores estavam interessados em agentes de

software personificando técnicas de IA. Certamente, a definição de AIMA depende fortemente do que interpreta-se como ambiente, e qual é o seu sensoriamento e ação percebida. Define-se o ambiente como qualquer um que provenham entradas e receba saídas, e pegue as entradas recebidas para serem analisadas e produza saída para fazer um acionamento, então todo programa é um agente. Assim, se deseja chegar em um consenso entre agente e programa, deve-se restringir pelo menos algumas das noções do que vem a ser ambiente, sensoreando e agindo.

c) Agente Maes

"Agentes Autônomos são sistemas computacionais que habitam um ambiente complexo e dinâmico, sentem e agem autonomamente neste ambiente, e cumprem um conjunto de metas ou tarefas para qual foram projetados." (MAES IN FRANKLIN E GRAESSER, 1996).

Pattie Maes, é uma das pioneiras nas pesquisas sobre agentes. Ela adiciona um elemento crucial em sua definição de agentes: os agentes devem agir autonomamente para "realizar um conjunto de metas." Também os ambientes são definidos como complexos e dinâmicos.

d) Agente de KidSim

"Permita-nos definir um agente como uma entidade de software persistente dedicado a um propósito específico. o termo 'Persistente ' distingue agentes de sub-rotinas; os agentes têm suas próprias idéias sobre como realizar tarefas, eles possuem seus próprios programas de trabalho. 'O propósito especifico' distingue-os de aplicações multifuncionais; os agentes são tipicamente muito pequenos." (SMITH, CYPHER E SPOHRER IN: FRANKLIN E GRAESSER, 1996) .

KidSim foi um protótipo de um ambiente de programação para crianças desenvolvido por Dave Smith e por Allen Cypher nos laboratórios de pesquisa da Apple.

Franklin & Graesser apontam o requisito explícito de persistência como uma “nova e importante contribuição”. Entretanto, criticam esta definição porque muitos agentes são de "propósito especifico" e isto não é uma característica essencial de agência.

e) Agente Hayes-Roth

“Agentes Inteligentes executam continuamente três funções: percepção de condições dinâmicas no ambiente; as ações para influenciar o ambiente; e o raciocínio para interpretar percepções, resolver problemas, planejar conclusões e determinar ações.” (HAYES E ROTH IN: FRANKLIN E GRAESSER, 1996)

Franklin & Graesser criticam a insistência de Barbara Hayes-Roth do Laboratório de Sistemas de Conhecimento de Stanford em dizer que os agentes raciocinam durante o processo de seleção de ação. Neste caso, se o raciocínio é amplamente interpretado, sua arquitetura de agente permite ações de reflexo como também ações planejadas.

f) O Agente IBM

"Agentes Inteligentes são entidades de software que executa algum conjunto de operações em nome de um usuário ou outro programa com algum grau de independência ou autonomia, e quando o fazem, empregam algum conhecimento ou representação das metas ou desejos do usuário." (FRANKLIN E GRAESSER, 1996)

Nesta definição, Franklin & Graesser observam neste artigo da IBM um agente agindo para outro, com a autoridade concedida pelo outro agente.

Agente Wooldridge - Jennings

"... um hardware ou (mais usualmente) um sistema de computador baseado em software que apresenta as seguintes propriedades:

autonomia: os agentes funcionam sem a intervenção direta de humanos ou outros, e que tenham algum tipo de controle sobre as suas ações e estados internos;

habilidade social: os agentes interagem com outros agentes (e possivelmente humanos) via algum tipo de linguagem para comunicação de agentes;

· reatividade: os agentes percebem seu ambiente, (que pode ser o mundo físico, um usuário através de uma interface gráfica do usuário, uma coleção de outros agentes, a INTERNET, ou talvez todos estes combinados), e respondem na hora oportuna as mudanças que ocorrem neste ambiente;

· pró-atividade: os agentes simplesmente não agem em resposta para seu ambiente, eles são capazes de apresentar comportamentos orientados a objetivos pela tomada de iniciativa." (WOOLDRIDGE AND JENNINGS IN: FRANKLIN E GRAESSER, 1996)

Franklin & Graesser observam que a definição de Wooldridge e Jennings, além de citar autonomia, percepção e atividade, permitem uma grande e delimitada faixa de ambientes. Eles também incorporam o requisito de comunicação.

g) Agente SodaBot

"Agentes de Software são programas que tomam parte de diálogo e/ou negociam e coordenam a transferência de informações." (COEN IN: FRANKLIN E GRAESSER, 1996)

SodaBot é um ambiente de desenvolvimento para agentes de software que está sendo construído no laboratório de Inteligência Artificial do MIT por Michael Coen, , Joshua Kramer, Brenton Phillips, e Joanna Yun.

Franklin & Graesser vêem que o sentido desta definição demonstra que a negociação envolve a percepção e a ação e o dialogo que requer a comunicação.

h) Agente Foner

Franklin & Graesser observam na definição de Foner uma exigência muito maior de um agente. Seus agentes colaboram com seus usuários para melhorar a realização das tarefas dos usuários. Isto requer, além de autonomia, que o dialogo do usuário com o agente, seja fidedigno, e degrade graciosamente em face a "comunicações incorretas" Foner (LENNY FONER APUD FRANKLIN & GRAESSER 1996)

i) Agente de Brustoloni

"Agentes Autônomos são sistemas capazes de ações autônomas e intencionais no mundo real." (BRUSTOLONI APUD FRANKLIN E GRAESSER, 1996)

Franklin & Graesser critica a definição de agentes por Brustoloni, porque diferentemente dos agentes anteriores, Brustoloni define que os agentes devem viver e agir "no mundo real." Esta definição exclui agentes de software e programas em geral. Brustoloni insiste também que seus agentes são "reativos", isto é, podem responder a estímulos externos, de forma assíncrona em um momento oportuno."

Baseado nas definições dos diversos autores citados sobre agentes e adicionando as críticas de Franklin & Graesser, pode-se extrair as características mais importantes para o conceito de agentes::

- Agentes possuem a capacidade de perceber e agir sobre o ambiente.
- Agentes devem dispor de autonomia para agir.

- Agentes podem ser complexos ou dinâmicos.

- Agentes possuem um conjunto de tarefas a realizar.

- Agentes possuem persistência, tendo a sua própria forma de resolver problemas.

- Agentes podem delegar autoridade a outros agentes.

- Agentes realizam tarefas para outros agentes ou usuários.

- Agentes não são apenas reativos, mas podem apresentar comportamentos orientados a objetivos pela tomada de iniciativa.

- Agentes podem comunicar-se com outros agentes.

4.3 Agentes de Interface

Maes (1994) afirma que Agentes de Interface são programas de computador que empregam técnicas de inteligência artificial para prover assistência ativa ao usuário com tarefas baseadas em computadores. Os agentes adquirem experiência ao auxiliar os usuários em suas tarefas e ao compartilhar a experiência de outros agentes.

A metáfora usada em agentes de interface é a de um assistente pessoal que está colaborando com o usuário no mesmo ambiente de trabalho (MAES 1994)

Knapik e Johnson (apud Thiry, 1999) acreditam que os agentes serão um componente padrão de sistemas computacionais por sua capacidade de facilitar a interação com interfaces, traduzir informação em conhecimento e realizar tarefas em nome do usuário.

Agentes de interface estão se tornando cada vez mais atrativos à medida que cresce a complexidade da interface com o usuário. O crescimento anual da taxa de interatividade de interfaces (menus, e opções adicionadas) é insustentável e se for mantida a correspondência um-para-um (usuário-interface) chegaremos a ponto de que não poderá ser adicionada mais funcionalidade aos sistemas. Agentes de interface são a saída para este dilema. (LIEBERMAN, 1997)

Segundo Wexelblat (1997), a utilização de agentes de interfaces não desobriga o projetista de todo o rigor contido em quaisquer outros projetos de interface. A alteração na forma de interagir não deve relegar décadas de pesquisa em ciência cognitiva e pesquisas sobre interfaces. Qualquer nova tecnologia não aliviará o

problema enfrentado no projeto de interação, que continuará a tentar compreender o trabalho efetuado pelo usuário, o contexto no qual o sistema é utilizado, o tipo de pessoas que irão utilizá-lo. Agentes de interface não deveriam agir como desculpa para um projeto de interface pobre.

O melhor exemplo conhecido de agentes de interface são os sistemas de tutoria inteligentes e sistemas de ajuda sensíveis ao contexto. Nestes sistemas o usuário pode operar a interface sem a necessidade do agente, mas, ao ser chamado, o agente pode mostrar sugestões, ou executar ações em cima da interface visualizada, baseado em entradas implicitamente coletadas do usuário. Outros tipos de agentes de interface podem criticar o comportamento do usuário, ou sugerir ao usuário ações de manipulação direta com informações adicionais que o usuário pode considerar muito proveitosas. (LIEBERMAN, 1997).

SÁNCHEZ (1997) divide os agentes de interface em três categorias: Agentes de Informação, Agentes de Tarefas e Agentes Sintéticos.

Os Agentes de Informação auxiliam o usuário na recuperação de informações desorganizadas e muito dinâmicas.

Os Agentes de Tarefas auxiliam o usuário a executar tarefas disponíveis. Eles agem concorrentemente com a aplicação envolvida, observam a atividade do usuário e se oferecem para automatizar certas ações. Dois tipos de Agentes de Tarefas podem ser discriminados: Agentes Pessoais, que assistem a usuários individuais; e Agentes Grupais, que participam das tarefas colaborativas mediadas pelo computador.

Finalmente, os Agentes Sintéticos criam ambientes envolventes para o usuário através da apresentação de personagens animados na interface do usuário. Eles evocam a credibilidade do usuário e promovem a ilusão de autonomia.

A tecnologia de Agentes Inteligentes, especificamente a tipologia de Agentes de Interface dentro um sistema desenvolvido, permite traduzir as requisições dos usuários de uma forma flexível pela abstração da complexidade da interface, deduzindo suas necessidades pela observação de seu comportamento e gerando um perfil de cada usuário de acordo com o seu comportamento, tornando a navegação pelo site mais fácil e rápida para tanto para usuários novatos como experientes. A utilização de Agentes de Interface também personaliza a interface,

eliminando as disrupturas ocasionadas pela interpretação particular dos elementos de interface por cada usuário.

No próximo tópico, serão apresentadas algumas metodologias para o desenvolvimento de sistemas orientados a agentes.

5 MODELAGEM VISUAL PARA APLICAÇÕES BASEADAS EM AGENTES

A conceitualização e implementação de sistemas está progressivamente sendo melhor compreendida pelos engenheiros de software (WOOLDRIDGE 2001). Mas, em sistemas onde existe uma rede complexa de protocolos de coordenação e onde a computação é baseada em processos concorrentes, exige-se novos modelos de abstração e ferramentas.

Programadores acostumados com a metodologia orientada a objeto, falham em perceber as diferenças entre um agente e um objeto, dada a tendência a antropomorfizar um objeto. (NeXT, 2001 Apud WOOLDRIGE)

A Unified Modelling Language – UML é uma tentativa das três principais personalidades no projeto e análise orientadas a objeto (Grady Booch, James Rumbaugh e Ivar Jacobson) em desenvolver uma notação para sistemas orientados a objeto. A OMG (Object Management Group) adotou a UML como padrão em 1997. (FURLAN, 1988).

UML é um padrão de fato para a modelagem orientada a objeto (BOOCH, 2001 Apud WOOLDRIGE) e há um bom número de tentativas de adaptar a notação UML para a modelagem de sistemas baseados em agentes (WOOLDRIDGE 2001).

Entretanto, os tipos de conceitos e notações atualmente suportadas pela UML não são necessariamente as mais adequadas ao desenvolvimento de sistemas orientados a agentes e necessitam de adaptações para representar agentes e a comunicação entre agentes de maneira correta. (FAKE 2001)

5.1 Metodologias para o Desenvolvimento de Sistemas Orientados a agentes

Esta seção descreve metodologias de modelagem através da abordagem top-down e interativa para sistemas baseados em agentes., com a ramificação descrita em (TVEIT 2001) na figura 1.

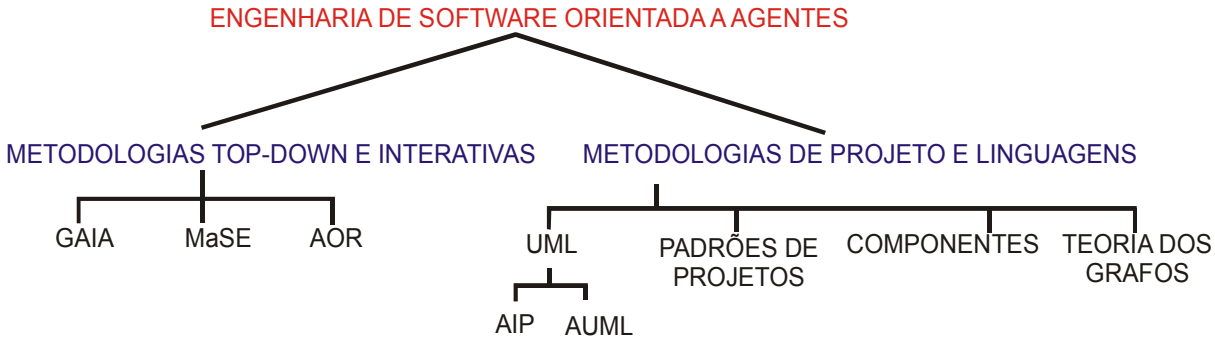

Figura 1 –Metodologias de desenvolvimento Orientadas a Agente.

Fonte: Adaptado de (TVEIT 2001).

5.1.1 Metodologia Gaia

Wooldridge, Jennings e Kinny (WOOLDRIGE, 2000) apresentam a metodologia Gaia para a análise e projeto orientado a agentes. A metodologia Gaia permite a estruturação dos agentes e das sociedades de agentes, mas, requer em sua abordagem que o relacionamento entre agentes seja estático em tempo de execução (TVEIT 2001).

A metodologia Gaia propõe uma abordagem orientada a papéis para a análise e projeto de sistemas baseados em agentes. Após a identificação de papéis-chaves no sistema, um modelo de papéis detalhado é construído. Papéis então são mapeados em um modelo de classe de agentes. (WOOLDRIDGE, 2000)

Segundo (WOOLDRIGE 1999) quando definiu as bases para a metodologia Gaia, o primeiro passo para um projeto orientado a agentes é definir o domínio (escopo do trabalho no âmbito da sociedade dos agentes e de cada agente individualmente) .

5.1.1.1 Conceitos da Metodologia GAIA

Os principais conceitos desta metodologia, estão divididos em duas categorias que são demonstradas no quadro 1 (WOOLDRIGE, 2000) :

- Entidades Abstratas: Usados para conceitualizar o sistema.
- Entidades Concretas: Usados no processo de projeto do sistema

Quadro 1 - Entidades Abstratas e Concretas em GAIA

Entidades Abstratas	Entidades Concretas
Papéis Permissões Responsabilidades Protocolos Atividades Responsabilidades de Sobrevivência Responsabilidades de Segurança	Tipos de Agentes Serviços Conhecimentos

Fonte : Adaptado de (WOOLDRIGE, 2000)

Após a definição das entidades participantes, deve ser definido o papel que cada entidade irá representar no sistema.

5.1.1.2 Modelagem dos papéis

A modelagem dos papéis auxilia a identificação dos papéis-chaves do sistema. Segundo WOOLDRIGE (2000) Um Papel pode ser visto como uma descrição abstrata da funcionalidade esperada de uma entidade

O papel é definido por quatro atributos: protocolos, permissões, responsabilidades e atividades conforme representado na figura 2:

- Protocolos: define a maneira de interação com os outros papéis.

- Permissões: são os direitos associados a um papel, indicando os recursos disponíveis para que o papel possa ser desempenhado.

- Atividades: as Atividades de um papel definem as tarefas por ele executadas sem a interação com outros agentes. Podem ser descritas como um conjunto de ações “privadas”.

Responsabilidades: determinam a funcionalidade e é o atributo chave de um papel. As Responsabilidades estão subdivididas em responsabilidades de sobrevivência e de segurança e são descritas na forma de expressões.

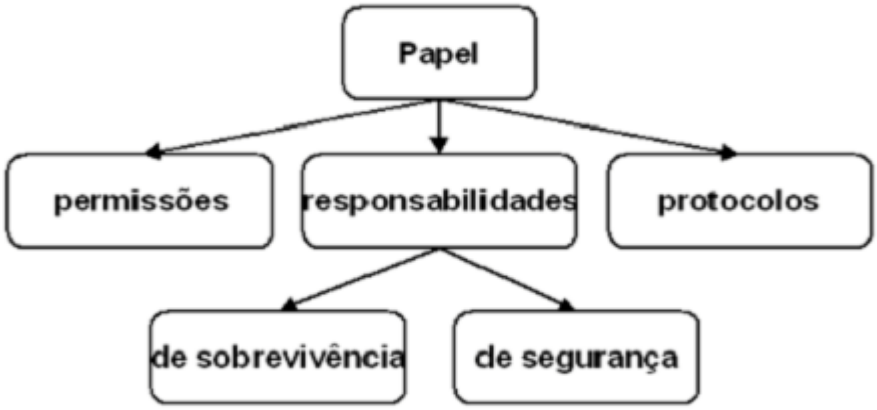

Figura 2 – Atributos dos Papéis

Fonte: Adaptado de Wooldridge (2000)

Segundo (WOOLDRIDGE 2000), as expressões de sobrevivência são similares as expressões de ciclo de vida de FUSION. (COLEMAN, 1994). As expressões de ciclo de vida definem a trajetória potencial de execução através de várias atividades e interações associadas a um papel.

Conforme (WOOLDRIDGE 2000), a forma geral de uma expressão de sobrevivência é:

NOMEDOPAPEL=expressão

Onde o *NOMEDOPAPEL* é o nome do papel cuja as propriedades de sobrevivência estão sendo definidas e *expressão* é a expressão de sobrevivência que define a propriedade de sobrevivência de *NOMEDOPAPEL*.

Na tabela 2, são demonstrados os operadores para construir as expressões de sobrevivência de um papel.

Quadro 2 – Operadores para expressões de sobrevivência

Operadores	Significado
x.y	X seguido de y
x*	X ocorre 0 ou mais vezes
x^{ω}	X ocorre infinitamente
x||y	X e y se sobrepõem
x|y	x ou y ocorre
x+	x ocorre 1 ou mais vezes
[x]	x é opcional

Fonte: Adaptado de Wooldridge (2000)

Para ilustrar uma expressão de sobrevivência, considere as responsabilidades do papel moça-do-cafezinho:

Moça-do-Cafézinho=(Encher.InformarTrabalhadores. VerificarNível. AguardeEsvaziar) $^{\omega}$

A expressão diz que Moça-do-Cafezinho consiste na execução do protocolo encher, seguida do protocolo InformarTrabalhadores, seguido da atividade VerificarNível e do protocolo AguardeEsvaziar. O símbolo após o parêntese indica que a seqüência de execução destes protocolos e atividades é repetida infinitamente.

(WOOLDRIDGE 2000) descreve as Responsabilidades de Segurança como expressões de segurança que relatam a ausência de algumas condições desejáveis, conforme no exemplo abaixo:

EstoquedeCafe>0

Na etapa da análise, o sistema é organizado em dois modelos: A modelagem dos papéis e a modelagem das interações.

5.1.1.3 Modelo para a Modelagem dos Papeis

O modelo para modelagem dos papeis é apresentado no quadro 3.

Quadro 3 - Esquema para a modelagem dos papéis

Modelo do Papel:	***Nome do Papel***
Descrição	*Descrição curta do papel*
Protocolos	*Protocolos dos quais o papel toma parte.*
Permissões	*"Direitos" associados ao papel.*
Responsabilidades Sobrevivência Segurança	 *Responsabilidades de Sobrevivência* *Responsabilidades de Segurança*

Fonte: Adaptado de Wooldridge (2000)

O Exemplo de implementação de uma modelagem de papel está no quadro 4, com o nome do Modelo do Papel, a descrição do Papel, os Protocolos e Atividades implementadas, as permissões de leitura e escrita e as responsabilidades de sobrevivência e de segurança.

Quadro 4 - Exemplo de modelagem de papel

Modelo do Papel: Moça do Cafezinho
Descrição: Este papel envolve a certeza de que a garrafa de café será mantida cheia, e informar aos trabalhadores quando café fresco estiver pronto.
Protocolos e atividades : Encher, InformarTrabalhadores, VerificarNível, AguardarEsvaziar
Permissões: Leitura: Cafeteiro // nome do cafeteiro. EstadoCafé // cheio ou vazio Escrita: NívelCafé // Nível da Garrafa de café.
Responsabilidades Sobrevivência: Moça do Cafezinho = (Encher.InformarTrabalhadores. VerificarNível.AguardarEsvaziar)$^{\omega}$ Segurança: ● NívelCafé ≥ 0

Fonte: Adaptado de WOOLDRIDGE (2000)

5.1.1.4 Modelagem das Interações (Definição do Protocolo)

Segundo WOOLDRIDGE (1999), a modelagem das interações consiste em um conjunto de definições de protocolos, um para cada tipo de interação entre papéis.

No esquema da Modelagem das Interações, os seguintes requisitos são descritos:

- Propósito: resumo da natureza da interação (ex: requisição de informação, Atividade de Agendamento, Definindo Tarefas).

- Inicializador: o(s) papel(is) responsáveis pelo início da interação.

- Interlocutor: o papel com o qual o inicializador interage.

- Entradas: informação usada pelo papel inicializador enquanto habilita o protocolo.

Saídas: Informação fornecida pelo/para a resposta do protocolo durante o curso da interação.

processamento: resumo do processamento efetuado pelo protocolo inicializado durante o curso da interação

Um esquema da Modelagem das Interações com os itens descritos acima é apresentado no quadro 5.

Quadro 5 - Esquema para a Modelagem das Interações

Propósito		
Inicializador	Interlocutor	Entradas
Processamento		Saídas

Fonte : Wooldridge (2000)

No quadro 6 temos um exemplo de modelagem de interação, com o propósito *Encher*, o Inicializador *Moça do Cafezinho*, o Interlocutor *Máquina de Café*, O processamento *Encher a Máquina de Café*, a Entrada *Cafeteiro* e a Saída *NivelCafé*

Quadro 6 - Exemplo de modelagem de Interação

Encher		
Moça do Cafezinho	Máquina de Café	Cafeteira
Encher a Máquina de Café		NívelCafé

Fonte : Adaptado de Wooldridge (2000)

No exemplo da tabela 6, é definido o protocolo Encher, que toma parte do Papel Moça do Cafezinho. Este define que o protocolo Encher é inicializado pelo Papel Moça do Cafezinho, e envolve o papel Máquina de Café. Este envolvimento é dado pelo Papel Moça do Cafezinho colocando café na máquina chamada Cafeteira, e o resultado da Máquina de Café sendo informado ao valor de NívelCafé.

Segundo Zambonelli (2000), nos últimos anos várias tentativas foram feitas para desenvolver técnicas e metodologias de modelagem orientadas a agentes. Entretanto, nenhuma destas técnicas pode ainda ser considerada como uma metodologia completa para a análise e projeto de sistemas MultiAgentes. A metodologia Gaia é uma das poucas tentativas que promove a amarração entre a análise e o projeto de sistemas MultiAgente e que negocia com os níveis micro (intra-agente) e macro (inter-agente) da análise e projeto. Entretanto, Gaia como é atualmente apresentada, não é uma metodologia generalista para todos os tipos de sistemas MultiAgente. Ao invés disto, pretende suportar o desenvolvimento da resolução de problemas distribuídos nos quais os componentes que constituem o sistema são conhecidos em tempo de projeto (sistemas fechados) e nos quais todos

os agentes são cooperadores em busca do objetivo global. Por estas razões, Gaia não é apropriada para a análise e projeto de aplicações Internet, onde a abertura e auto-interação são fatores chaves.

5.1.2 Metodologia MaSE (Multiagent System Engineering)

(Wood e DeLoach 2001a) apresentam a Metodologia MaSE (Multiagent Systems Engineering Methodology) (MaSE). Esta metodologia MaSE é similar a Gaia em sua generalidade e domínio da aplicação, porém, MaSE vai ainda além, através do seu suporte a criação automática de código para agentes através de suas ferramentas. O objetivo principal da metodologia MaSE é levar o projetista da especificação inicial do sistema de agentes até a implementação do mesmo.

A metodologia MASE é composta por duas fases principais: Análise e Projeto.

DELOACH (2001) descreve a Etapa da Análise subdividida em três passos: Levantamento de Objetivos, Caso de Uso (Use Cases) e refinamento de Papéis.

No passo chamado de Levantamento de Objetivos, são fornecidos os requisitos do usuário e retornados os objetivos finais do sistema. Após a definição dos objetivos do sistema, define-se os Casos de Usos e diagramas de seqüência . Concluído o passo de Casos de Uso, inicia-se o passo de identificação dos papéis, que define o conjunto inicial de papéis do sistema e caminhos de comunicação.

Através do levantamento dos objetivos do sistema e da identificação dos papéis encontrados nos Casos de Uso, o conjunto inicial de papéis é refinado e estendido, sendo definidas então as tarefas que devem acompanhar cada objetivo.

Na Fase do projeto, os modelos da análise são transformados em construções utilizáveis para a implementação de um sistema MultiAgente.

A fase do Projeto é composta por 4 passos: Criação das Classes de Agentes, Construção dos Diálogos, Montagem das Classes de Agentes e Projeto do Sistema.

A figura 3 apresenta a metodologia MASE e suas etapas.

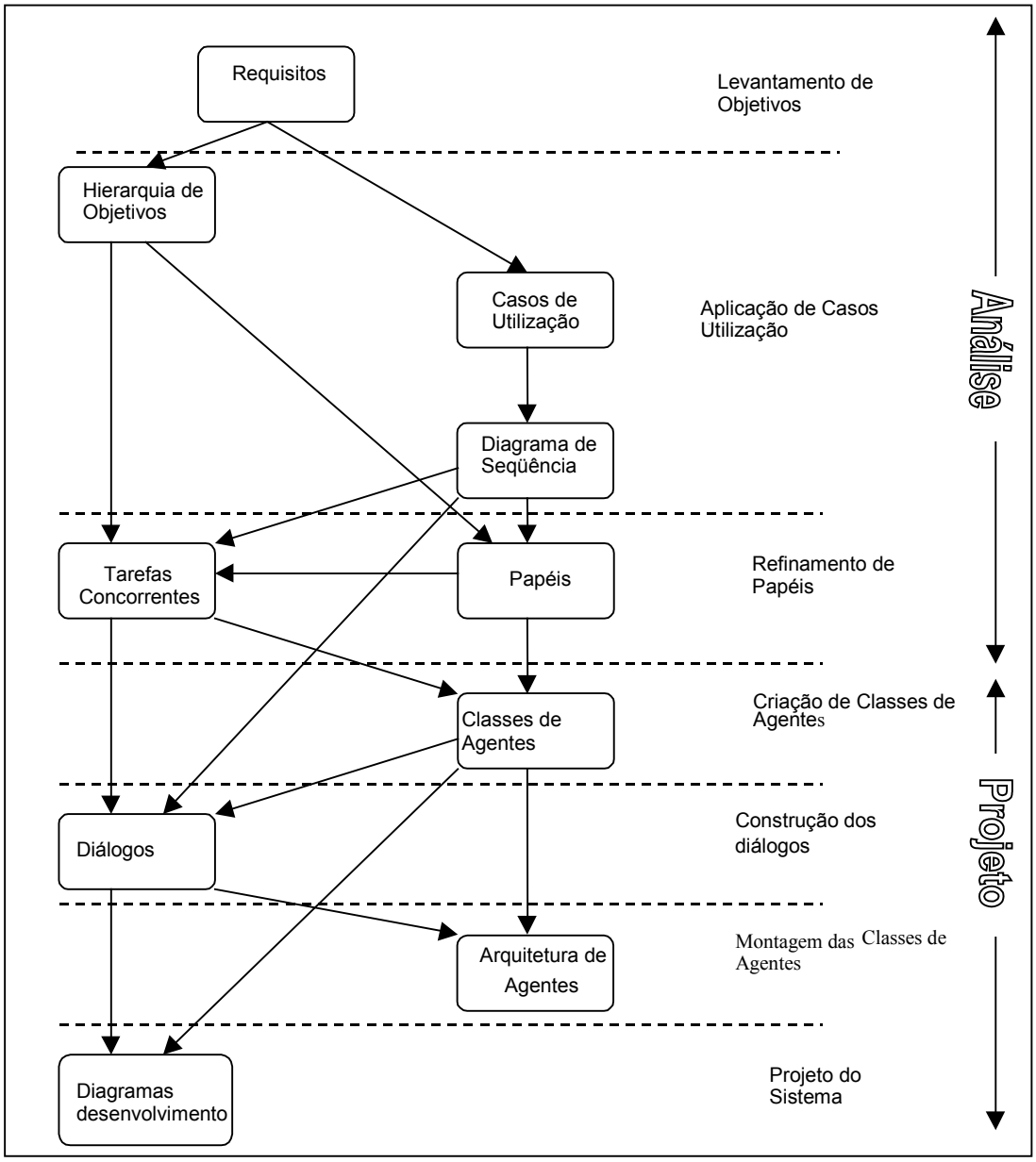

Figura 3 - Metodologia MaSE e suas etapas

Fonte: Adaptado de Deloach (2001)

Como apresentado na figura 3 a metodologia MASE é composta pelas seguintes etapas principais:

5.1.2.1 Etapa de Levantamento de objetivos

O primeiro passo da metodologia MaSE é o levantamento de objetivos do sistema, através de sua especificação, transformando-o em um conjunto estruturado de objetivos, demonstrado em um Diagrama de Hierarquia de Objetivos.

As duas etapas desta tarefa consistem em: identificar objetivos e estruturar objetivos.

A identificação de objetivos pode ser obtida pelo refinamento da essência do conjunto de requisitos. Exemplo: O sistema precisa verificar se houve um acesso não permitido. Isto pode ser estruturado posteriormente em várias etapas hierárquicas, como verificação do tipo de invasão, notificação dos administradores do sistema e providências a serem tomadas após cada tipo de invasão. Com isso cria-se um Diagrama Hierárquico de Objetivos, como observado na figura 4.

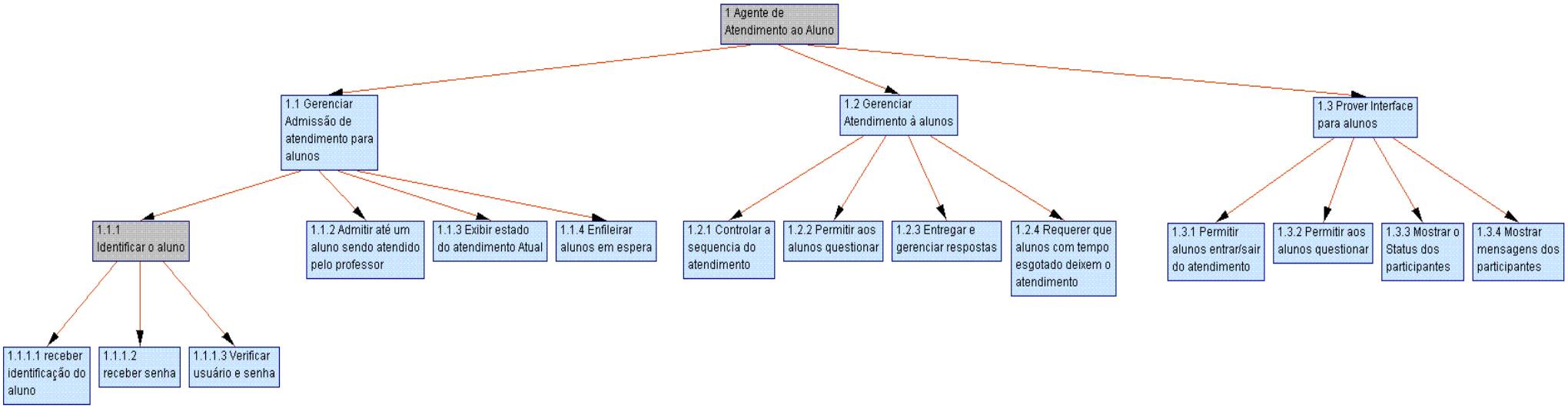

Figura 4- Diagrama Hierárquico de Objetivos

Fonte :Adaptado de Deloach (2001)

5.1.2.2 Etapa de Casos de Uso

Segundo (DELOACH 2001b), a aplicação de Casos de Uso é um passo crucial para a tradução de objetivos em papéis associados a tarefas. O analista monta os Casos de Uso pelos requisitos do sistema e dos usuários. Este diagrama é composto por seqüências narrativas de eventos que definem o comportamento desejado pelo sistema, conforme demonstrado na figura 5.

Entra no Atendimento
Um aluno necessitando de atendimento do professor irá inicializar a aplicação. O aluno deverá identificar-se no sistema através de uma identificação de usuário e senha. Caso seja autenticado corretamente, o aluno poderá entrar na espera de atendimento.

Figura 5 - Exemplo de diagrama de Casos de Uso

Fonte : Adaptado de Deloach (2001)

Os diagramas de Casos de Uso são posteriormente reestruturados em Diagramas de Seqüências (figura 6) , quebrando a seqüência dos eventos em papéis que devem ter um conjunto mínimo de conversação.

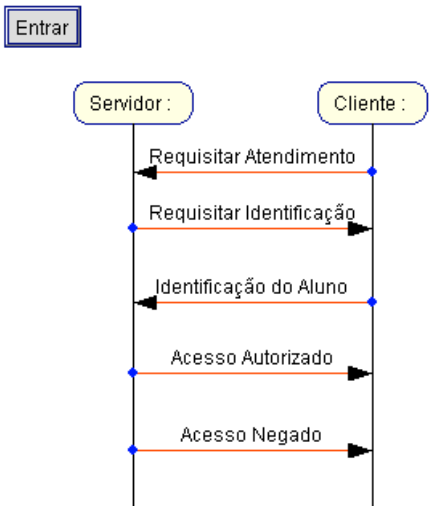

Figura 6- Diagrama de Seqüência.

Fonte : Adaptado de Wood e Deloach (2001A)

5.1.2.3 Etapa do Refinamento dos Papéis

O terceiro passo da metodologia MASE é a conferência de que todos os papéis necessários que foram identificados durante as etapas anteriores, prosseguindo com o desenvolvimento das tarefas que definem o comportamento dos um papeis identificados e dos padrões de comunicação.

Uma vez definidos os papéis, tarefas são criadas e descritas para cada papel conforme a figura 7.

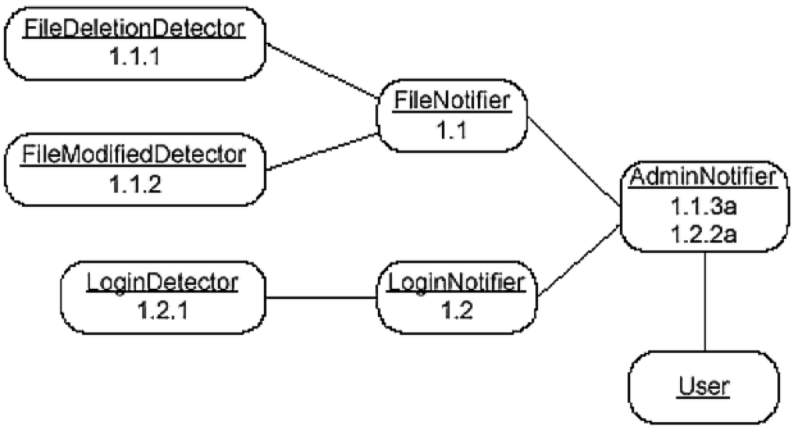

Figura 7 - Modelagem dos Papéis

Fonte: Deloach (2001b).

5.1.2.4 Criação de Classes de Agentes

No processo de Criação de Classes de Agentes, as classes de agentes são identificadas nos papéis e documentadas em um Diagrama de Classes de Agentes, sendo definido um mapeamento um-para-um entre os papeis conforme demonstrado na figura 8.

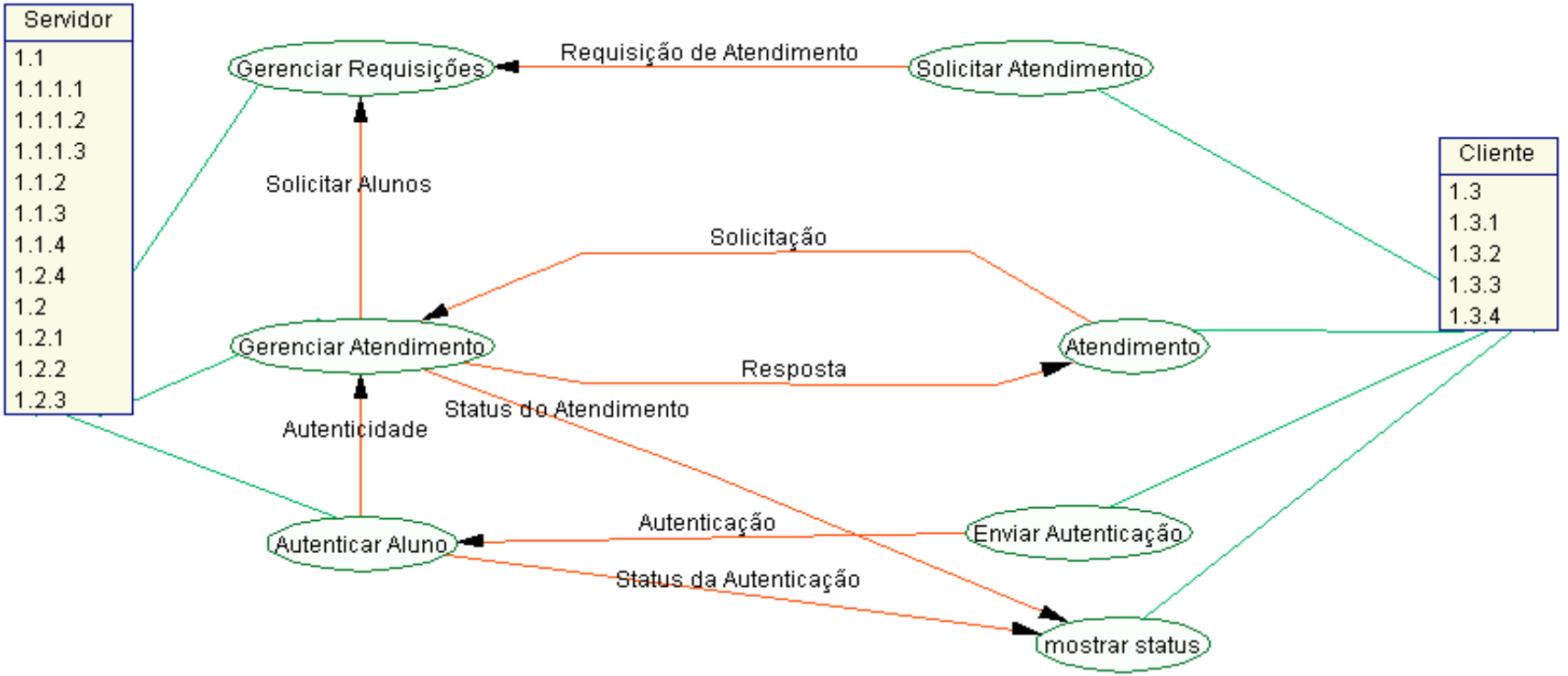

Figura 8 - Criação de Classes de Agentes

Fonte : Wood e Deloach (2001A)

5.1.2.5 Construção do Diálogo

O diálogo MaSE define o protocolo de coordenação entre dois agentes, tendo um inicializador e um reportador. A figura 9 apresenta um exemplo. O inicializador inicia um diálogo pelo envio da primeira mensagem. As mensagens são trocas binárias entre os agentes individuais.

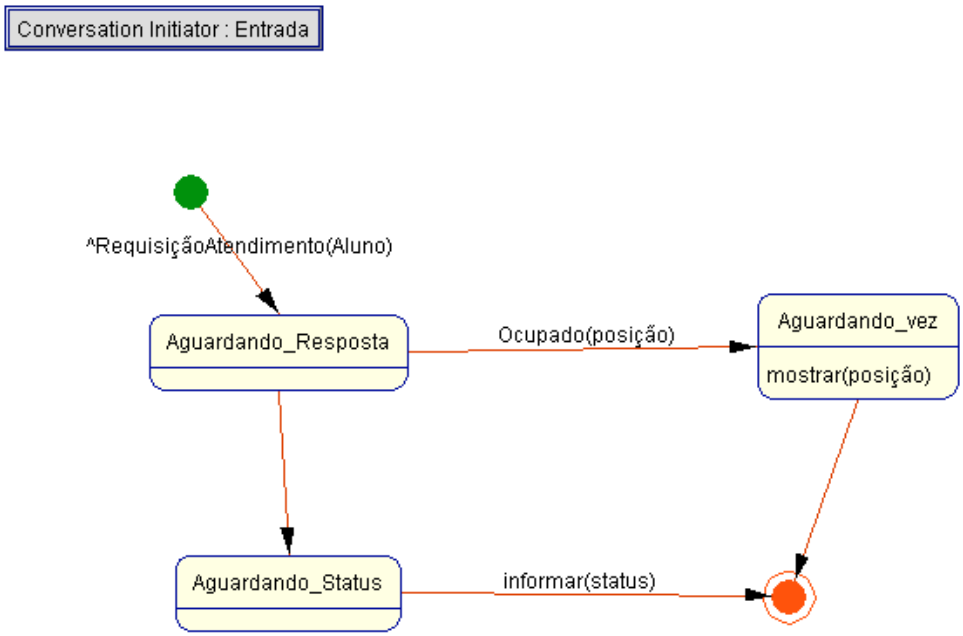

Figura 9 - Diagrama de Comunicação de Classes

Fonte: Deloach (2001b)

5.1.2.6 Montagem dos Agentes

Nesta etapa, as classes internas de um agente são criadas. O projetista pode nesta etapa entrar em detalhes de construção de baixo nível.

5.1.2.7 Desenvolvimento do Sistema

O passo final da metodologia MaSE implica na definição do sistema a ser implementado, pois já foram definidos os diagramas para mostrar os números, tipos e localizações dos agentes no sistema. Nesta etapa, o framework de comunicação e linguagem de programação são utilizados para o processo de desenvolvimento do sistema.

5.2 Avaliação da Metodologia

O objetivo das metodologias de engenharia de software é desenvolver um software de qualidade a um custo baixo dentro do orçamento e do prazo. Para isto, é necessário utilizar uma metodologia que promova a melhor abordagem para resolver o problema.

DELOACH (1998) visualiza MaSE como uma abstração adicional para o paradigma orientado a objeto, onde os agentes estão em um nível de abstração ainda mais alto do que objetos típicos. Ao invés de simples objetos, com métodos que podem ser invocados por outros objetos, os agentes coordenam as suas ações via conversação para compor objetivos individuais e da comunidade

Segundo (CAIRES et all,2002) a abordagem das metodologias de desenvolvimento orientadas a agentes, são indicadas para o desenvolvimento de sistemas complexos e distribuídos, fornecendo métodos, técnicas e ferramentas para resolver os problemas de um grande número de partes com muitas interações. Esta abordagem resolve problemas que geralmente só poderiam ser decompostos em partes menores.

(CAIRES et all,2002) defende que não é apropriado usar as metodologias orientadas a objeto para o desenvolvimento de aplicações orientadas a agente, porque o conceito de "agente" difere do conceito de "objeto", sendo que a diferença é mais significante pelo alto nível de abstração de um agente. Isto não quer dizer que as metodologias orientadas a objeto devam ser ignoradas. Entretanto apenas especificar comportamentos de objetos em termos de interação através de protocolos não faz do mesmo um agente.

No projeto desenvolvido, optou-se pela metodologia MaSE, orientada à agentes, porque segundo DELOACH (1998), a mesma compreende todo o ciclo de análise e projeto de um sistema orientado a agentes, facilitando a sua documentação. MaSE utiliza modelos gráficos para a representação dos tipos de agentes e da interface entre os agentes, criando novas camadas de abstração. MaSE também auxilia no levantamento inicial de requisitos dos sistema, análise, projeto e implementação de sistemas. Através de MaSE, um sistema MultiAgente pode ser implementado a partir de diferentes formas a partir de um mesmo projeto. MaSE também permite ao

analista efetuar modificações em etapas que refletem-se nas etapas posteriores de forma transparente.

.

6 FRAMEWORKS PARA O DESENVOLVIMENTO DE AGENTES

Neste capítulo, são apresentados e avaliados alguns Frameworks para o desenvolvimento de agentes com a implementação dos mesmos em Java.

Optou-se descrever Frameworks que geram agentes em Java, por ser a linguagem que melhor se adapta à um navegador Web, que é a base do ambiente de interfaces do sistema proposto para esta dissertação.

Os seguintes Frameworks pesquisados foram:

6.1 AgentTool

Deloach (2001b) descreve o AgentTool, que foi desenvolvido pelo Air Force Institute of Technology. O objetivo do AgentTool é permitir a projetistas de sistemas baseados em agentes especificar formalmente a estrutura e o comportamento requerido para um sistema MultiAgente e semi-automaticamente sintetizar sistemas MultiAgente. O comportamento de alto nível do sistema é definido graficamente usando a Metodologia de Engenharia de Sistemas MultiAgentes MaSE (Multiagent Systems Engineering Methodology).

6.2 Architecture type-based Development Environment (ADE)

Kupries (1999) apresenta o ADE, desenvolvido na Universidade de Potsdam. Com este framework os agentes de software e sistemas de agentes são modelados com orientação a objeto usando as últimas descobertas da arquitetura de software. Esta abordagem descreve uma arquitetura de software na qual a interação entre os componentes é realizada através de conectores, sendo disponíveis também a implementação independente de plataforma de computador.

6.3 Ascape

Parker (1998) demonstra o framework Ascape, que é uma estrutura para o desenvolvimento e análise de modelos baseados em agentes desenvolvido no *The Brookings Institution*. No framework Ascape, os objetos dos agentes existem dentro dos *scapes*, que são coleções de agentes que funcionam como arrays. Estas coleções, são elas mesmas agentes. Scapes fornecem um contexto para a interação

e conjunto de regras que governam o comportamento do agente. Ascape gerencia visões gráficas e conjunto de estatísticas para scapes e fornecem mecanismos para o controle e alteração dos parâmetros dos modelos de scape.

6.4 Bee-gent

Kawamura et all (1999) descreve o Bee-gent, desenvolvido pelo laboratório de pesquisa de sistemas e softwares da Toshiba Corporation. O Bee-gent é um novo tipo de estrutura para desenvolvimento de software feito 100% em sistemas de agentes. Bee-gent permite aos desenvolvedores construir sistemas abertos e distribuídos para otimizar o uso de aplicações já existentes. A estrutura Bee-gent é composta de dois tipos de agentes. "Agent Wrappers" são utilizados para agentificar aplicações existentes, enquanto "Mediation Agents" suportam a condenação de inter-aplicações pelo gerenciamento de todas as comunicações.

6.5 Bond Distributed Object System

Bölöni (1998) descreve este framework desenvolvido na Purdue University. Este sistema possibilita o desenvolvimento de aplicações distribuídas e utiliza a linguagem KQML para a comunicação entre os objetos. A estrutura de agentes do sistema Bond simplifica a tarefa do desenvolvimento de agentes, permitindo ao programador se concentrar na estratégia especifica de um novo agente. Os Agentes criados com este framework tem a capacidade de serem controlados remotamente e de cooperação com outros agentes. A tarefa do programador é resumida em especificar a agenda, a máquina de estado finita do agente e estratégias associadas a cada estado.

6.6 DECAF Agent Framework

Desenvolvido na *University of Delaware.* Graham (2001) descreve o DECAF (Distributed Environment Centered Agent Framework), que é uma plataforma para o desenvolvimento extremamente rápido de agentes. É constituída por um ambiente para a construção de interfaces, agendamento de agentes internos e monitoramento similares às primitivas do sistema operacional. O desenvolvedor de agentes não

precisa saber sobre nenhuma estrutura e pode focar o desenvolvimento no próprio agente. A arquitetura básica do DECAF é construída usando Java.

6.7 FIPA-OS

Poslad (2000) descreve FIPa-OS, que é desenvolvido pela FIPA, a FIPA é uma organização sem fins lucrativos que auxilia a padronização para a interoperação de agentes heterogêneos de software.

O FIPA-OS é um conjunto de ferramentas para o rápido desenvolvimento de agentes compatíveis com FIPA. FIPA-OS suporta a maioria das especificações experimentais de FIPA e está constantemente sendo melhorado em regime de Projeto de Código Aberto, fazendo-o a escolha ideal para a atividade de desenvolvimento de Agentes compatíveis com FIPA.

6.8 Gypsy

Jazayeri (2000) descreve o projeto Gypsy, desenvolvido na Technical University of Vienna, utiliza Java para a implementação de um ambiente flexível para a experimentação com programação de agentes móveis. É aplicável a recuperação de informações da Internet, Comércio Eletrônico, Computação Móvel e gerenciamento de redes.

6.9 Hive

Minar (1999) descreve HIVE, desenvolvido no The Media Lab, Massachusetts Institute of Technology. Hive é uma plataforma de software em Java para a criação de aplicações distribuídas. Usando Hive, os programadores podem facilmente criar sistemas que conectam e usam dados de toda a Internet.

6.10 JADE

Bellifemine (1999), descreve o Framework JADE, desenvolvido no CSELT S.p.A., University of Parma, JADE (Java Agent DEvelopment Framework) é uma estrutura de software para desenvolver aplicações baseadas em agentes em conformidade com as especificações FIPA para sistemas MultiAgentes interoperáveis. O seu objetivo é simplificar o desenvolvimento padronizado através de um conjunto de

serviços de sistemas e agentes. JADE pode ser considerado um “meio de agentes” para implementar uma plataforma de Agentes e uma estrutura de desenvolvimento. Toda a comunicação entre agentes é realizada através de troca de mensagens , onde FIPA ACL é o protocolo de mensagens utilizado.

6.11 JAFMAS

Chauhan (2000) descreve JAFMAS, desenvolvido na University of Cincinnati, JAFMAS fornece uma estrutura para guiar o desenvolvimento de sistemas MultiAgentes através de um conjunto de classes em Java.. A estrutura é voltada a ajudar a concretizar idéias em agentes tanto para desenvolvedores iniciantes como especialistas. O desenvolvimento é dirigido através da perspectiva de atos-de-fala (speech-act) e suporta a comunicação múltipla e direta, KQML ou outras performativas baseadas em speech-act.

6.12 JATLite

JEON (2000) descreve JATLite, desenvolvido na Stanford University, JATLite é um conjunto de pacotes do Java que facilitam a construção de sistemas MultiAgentes em Java. JATLite fornece a infra-estrutura básica de registro de agentes e facilitador do roteamento de mensagens usando nome e senha, conectando e desconectando da Internet, envio e recepção de mensagens, transferência de arquivos e o acionamento de outros programas ou ações nos vários computadores onde estiver executando. JATLite facilita a construção de agentes que enviam e recebem mensagens utilizando a linguagem padrão de comunicação KQML. As comunicações são realizadas em padrões da Internet, TCP/IP, SMTP e FTP.

6.13 JATLiteBean

Cranefield (1999) descreve JATLiteBean, desenvolvido na University of Otago, JATLiteBean toma a funcionalidade da linguagem KQML do JATLite e empacota-a em um JavaBean. JATLiteBean fornece uma interface fácil para utilizar as características do JATlite, incluindo autenticação, recepção e envio de mensagens KQML. Tem uma extensa arquitetura para o gerenciamento de mensagens e

controle de processos. Também suporta o anuncio automático das capacidades do agente para agentes facilitadores, além de checagem de sintaxe para KQML e configuração genérica para autenticação de arquivos.

6.14 JIAC

Albayrak (1999), descreve a arquitetura JIAC, desenvolvido na Technische Universitat Berlin, Java Intelligent Agents Componentware (JIAC) é uma arquitetura aberta, escalável e universal para agentes. Implementada em Java, oferece agentes móveis baseados em componentes e provê suporte para a criação de aplicações de comércio eletrônico e serviços de telecomunicações distribuídas.

6.15 Open Agent Architecture

Cheyer (2001) descreve esta estrutura desenvolvida pelo Stanford Research Institute (SRI International). Open Agent Architecture (OAA) é uma estrutura para integrar uma comunidade de agentes de software heterogêneos em um ambiente distribuído. Nesta estrutura, um agente é definido como um processo de software que registra os seus serviços de uma forma aceitável, através da linguagem ICL (Interagent Communication Language), e compartilha a funcionalidade comum a todos os agentes OAA

6.16 SOMA (Secure and Open Mobile Agent

Bellavista (1999), apresenta a estrutura SOMA. Desenvolvida na University of Bologna, SOMA é uma estrutura baseada em Agentes Móveis para Java capaz de responder aos requerimentos de escalabilidade, dinamismo, abertura e segurança que são típicos de um cenário Internet. É orientada a dois objetivos principais: Segurança e interoperabilidade. De um lado, SOMA é baseado em um modelo de segurança e têm uma série de mecanismos para construir e reforçar políticas flexíveis de segurança. Por outro lado, SOMA pode interoperar com diferentes componentes de aplicações projetados com diversos estilos de programação, o que garante a interoperabilidade em conformidade com CORBA e MASIF. Alem destas características, SOMA fornece a abstração de localização necessária para atingir a escalabilidade em um cenário global que é configurável e gerenciável

dinamicamente, também através de interfaces beseadas na Web. SOMA suporta também todas as formas de mobilidade.

6.17 Zeus

Nwana (1999) descreve o ambiente ZEUS, Desenvolvido nos laboratórios da British Telecommunications. Zeus é um ambiente colaborativo para a construção de agentes e também uma biblioteca de componentes escrita em Java. Cada agente ZEUS consiste em uma camada de definição, uma camada de organização e uma camada de coordenação. A camada de definição representa o raciocínio dos agentes e habilidades de aprendizado, seus objetivos, recursos, habilidades, crenças, preferências etc. A camada de organização descreve o relacionamento com outros agentes. A camada de coordenação descreve as técnicas de coordenação e de negociação que o agente possui. Os protocolos de comunicação são construídos no topo da camada de coordenação e implementam a comunicação entre os agentes. Acima da camada de definição esta a API.

6.18 AgentTool

Dentro das ferramentas disponíveis atualmente para o desenvolvimento de Sistemas MultiAgentes, Segundo Deloach (2001b), AgentTool é a ferramenta que apresenta adequadamente um ciclo de vida completo e um meio complementar para análise, projeto, e desenvolvimento de Sistemas MultiAgentes heterogêneos, utilizando-se da metodologia Multiagent Systems Engineering (MaSE)

AgentTool utiliza uma estrutura de menus que permitem o acesso a diversas funções do sistema e os diagramas da metodologia MaSE são acessíveis através de painéis. Quando um diagrama é selecionado, o projetista pode manipulá-lo graficamente na janela.

Durante cada fase do desenvolvimento do sistema, os diagramas de análise e projeto ficam disponíveis através do acesso aos painéis pelo menu principal.

A ordem dos painéis segue a ordem da metodologia MaSE. O preenchimento dos dados em um painel oferece subsídios para a continuação do desenvolvimento no painel seguinte e se reflete em todo o restante do projeto.

6.18.1 AgentTool e a metodologia MaSE

A ferramenta agentTool é uma tentativa de fazer uma ferramenta que suporte e reforce a metodologia MaSE. Deloach (2001b), descreve que atualmente o agentTool executa todas as sete etapas de MaSE, assim como suporta a transformação automática de modelos de análise em modelos de projeto. A relação do usuário com a ferramenta agentTool se dá através de menus suspensos que permitem acesso a diversas funções do sistema, incluindo uma verificação persistente da conversação da base de conhecimento e a geração do código. Os diversos diagramas da metodologia MaSE são encontrados através dos painéis tabulares no alto da janela principal. Quando um diagrama MaSE é selecionado, o projetista pode manipulá-lo graficamente na janela. Cada painel tem tipos diferentes de objetos e de texto que podem ser alocados. Ao selecionar um objeto na janela, permite-se que outros diagramas relacionados tornem-se acessíveis. A parte do agentTool que é talvez a mais atraente é a habilidade de trabalhar em partes diferentes do sistema em vários níveis de abstração intercambiavelmente.

6.18.2 Análise e Projeto com o AgentTool

A interface com o usuário do AgentTool é mostrada na figura 10 (WOOD E DELOACH 2001A), onde se pode acessar através de menus tabulares os passos para a análise e o projeto de um sistema na metodologia MaSE.

6.18.2.1 Levantamento dos Objetivos

A Etapa de análise com o AgentTool, começa com a definição da hierarquia de objetivos do sistema proposto.

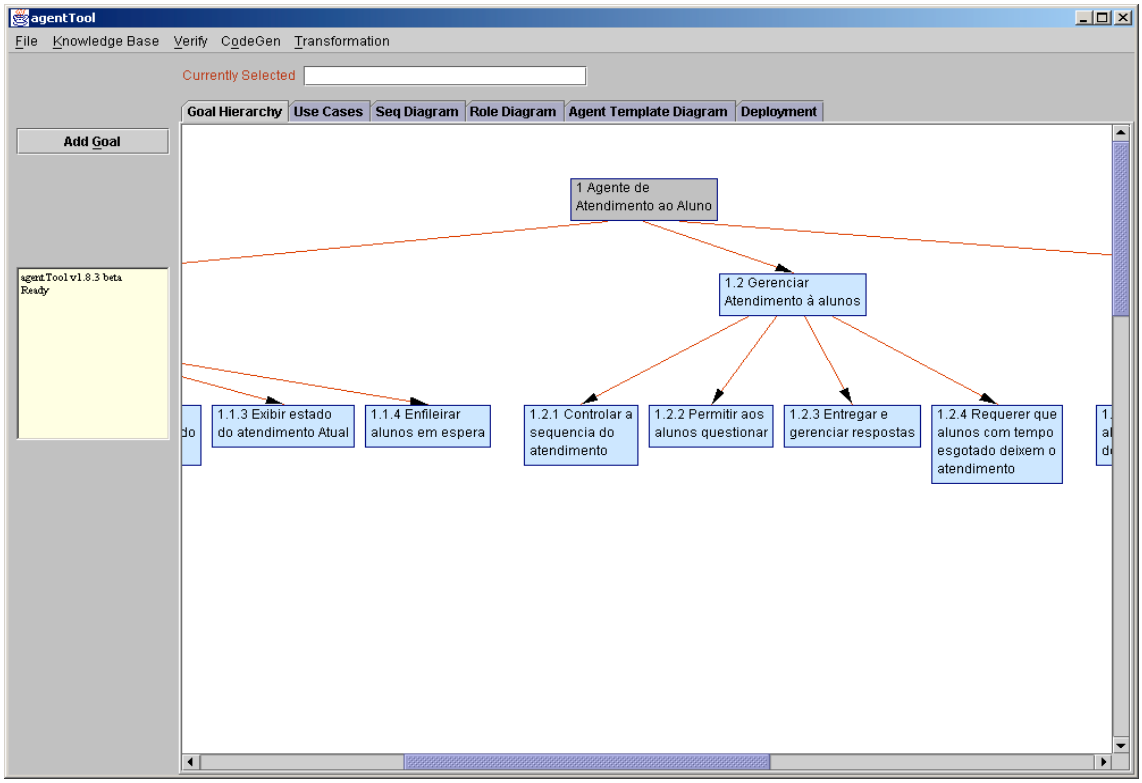

Figura 10 – Interface com o usuário de AgentTool com a hierarquia de objetivos selecionada

Fonte : Wood E Deloach (2001A)

Os objetivos são dispostos em hierarquia, no formato de árvore, e são automaticamente numerados em níveis, onde o objetivo-raiz está na parte superior do diagrama.

Os objetivos deverão ser implementados por um papel e enquanto isso não ocorre, eles permanecerão sinalizados com um contorno em amarelo intenso. Entretanto, certos tipos de objetivos não necessitam ser implementados. São os objetivos particionados que serão implementados através dos objetivos-filhos aos quais estão ligados. Os objetivos particionados são identificados por um retângulo cinza.

Cada um dos objetivos descritos na Hierarquia dos Papéis deve ser posteriormente ligado a um papel no Diagrama dos Papéis conforme demonstrado na figura 11 (WOOD E DELOACH 2001A).

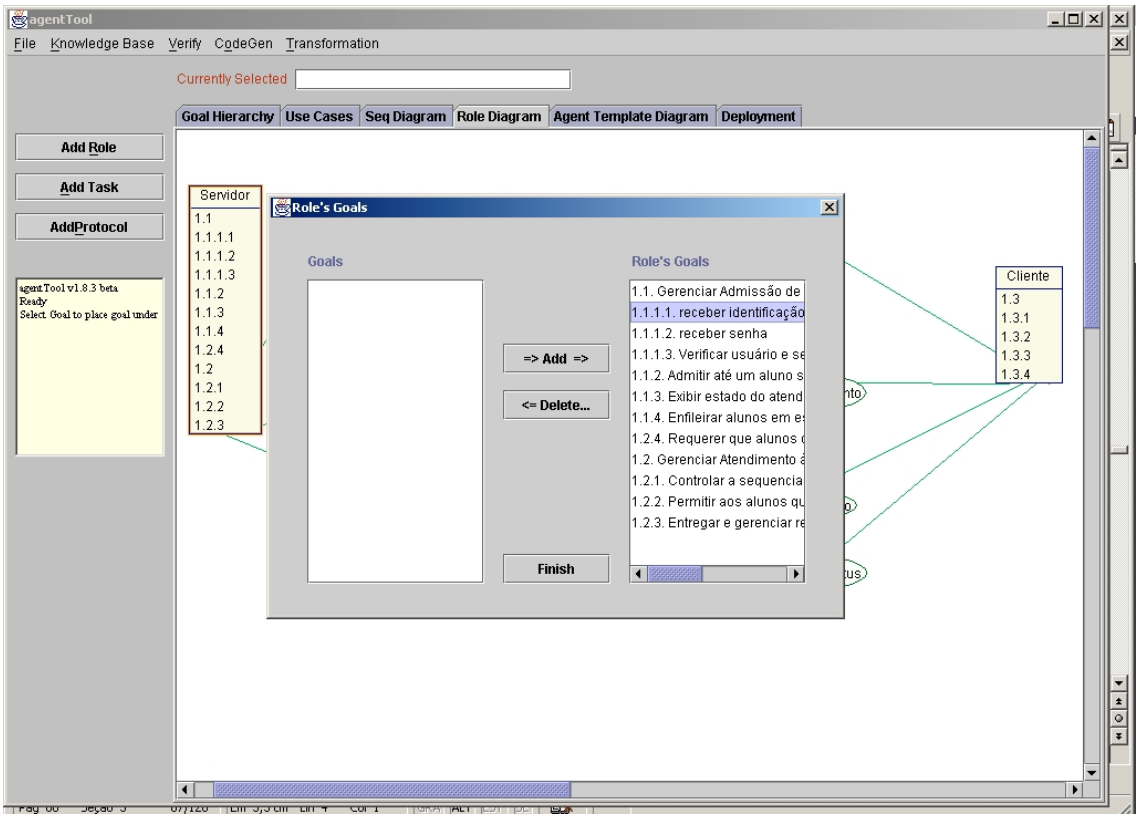

Figura 11 - Objetivos dos papéis

Fonte :Wood e Deloach (2001A).

Segundo Wood (2001), os casos do uso são extraídos dos requisitos do sistema e são descrições narrativas de uma seqüência de eventos que definem o comportamento desejado do sistema. É a descrição de como o usuário (ou de como o editor do documento de requisitos) acha que o sistema deve se comportar em uma determinada situação.

Na figura 12 é demonstrada a edição de casos de uso do AgentTool, onde cada Caso de Uso é uma situação que pode ocorrer no sistema. No painel à direita da figura 12, está a descrição do caso selecionado. Na parte inferior da tela da figura 12 é apresentada a descrição da seqüência de passos para cada situação descrita.

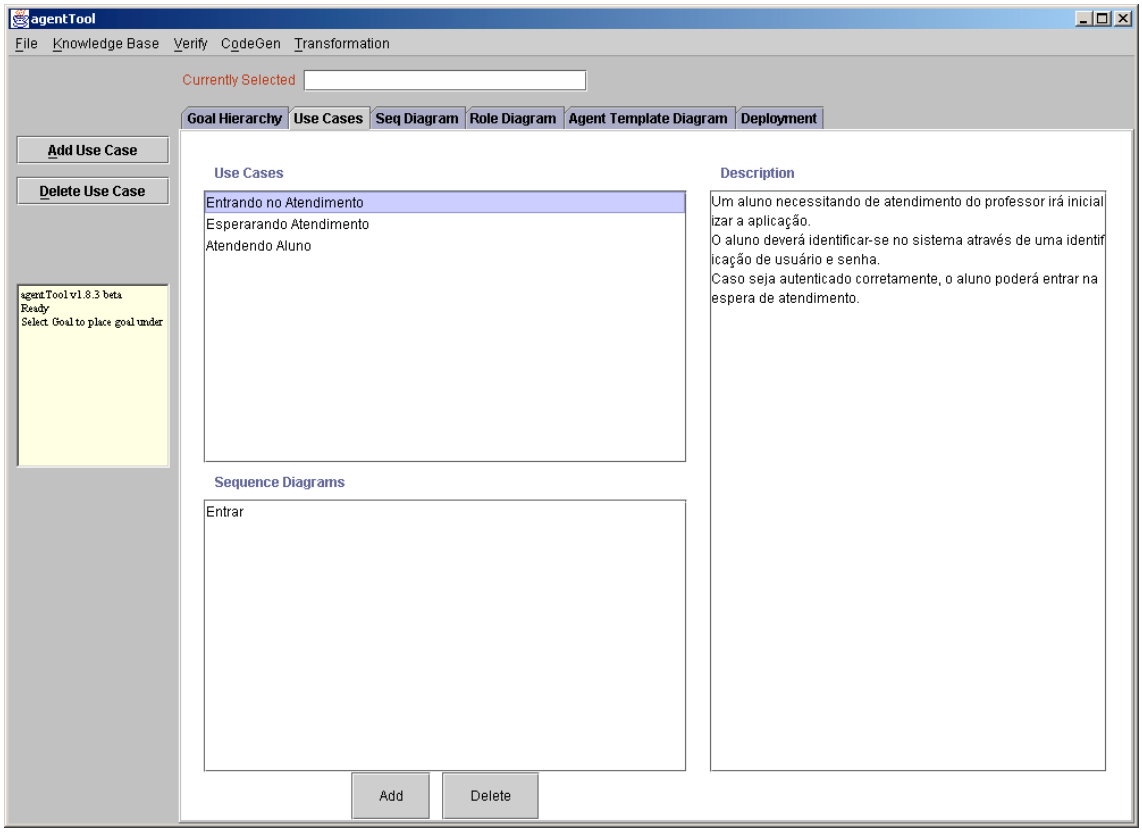

Figura 12 - Descrição dos Casos de Uso

Fonte : Wood e Deloach (2001A)

Após a definição dos casos de uso, pode-se ligá-los a diagramas de seqüência, que contém a definição dos subconjunto dos papéis e das mensagens trocadas, conforme demonstrado na figura 13.

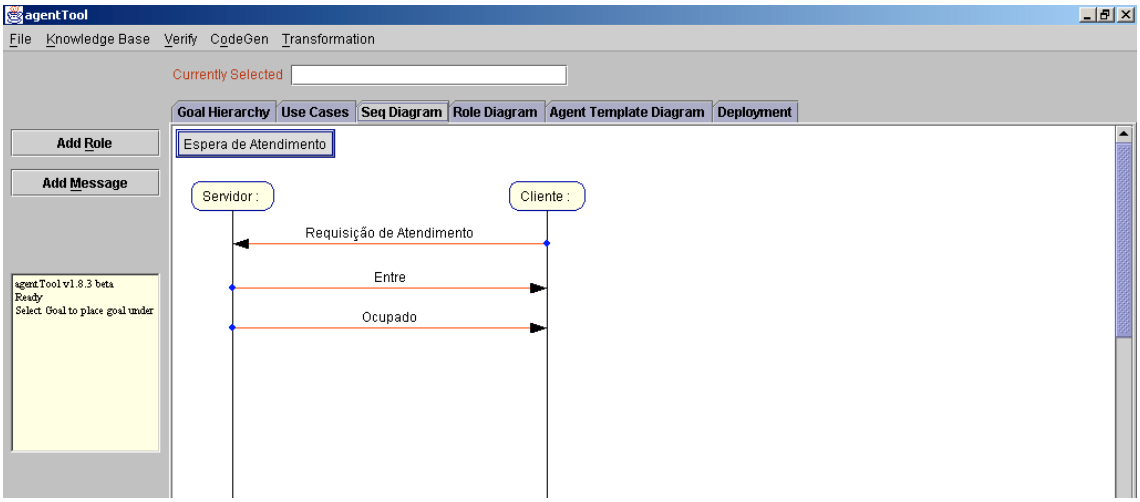

Figura 13 - Diagrama de seqüência do Caso de Uso

Fonte: Wood e Deloach (2001A)

6.18.2.2 Transformando objetivos em papéis e aplicando casos de uso

Wood (2001), descreve a segunda etapa de MaSE, que consiste em transformar os objetivos estruturados do diagrama de hierarquia de objetivos (figura 10) em um formato mais útil para construir sistemas MultiAgente: papéis. Os papéis são os blocos de edifício usados para definir as classes dos agentes e capturar objetivos do sistema durante a fase do projeto. Nesta fase garante-se que os objetivos do sistema estão explicados, assegurando-se de que cada objetivo esteja associado a um papel, e que cada papel está sendo executado por uma classe do agente. Um papel é uma descrição abstrata para a função esperada de uma entidade e encapsula os objetivos do sistema nos quais está delineada a sua responsabilidade. A transformação geral de objetivos em papeis é de uma-para-uma; cada papel tem um objetivo. Entretanto, há muitas situações excepcionais onde é útil combinar objetivos. Os objetivos similares ou relacionados podem ser combinados em únicos papéis para conveniência ou eficiência. Depois que os papéis são criados, as tarefas estão associadas com cada papel. Cada objetivo associado com um papel pode ter uma tarefa expressando em detalhes como o objetivo deve ser realizado.

Os papéis definidos nos diagramas de seqüência serão listados aqui, e permanecerão com um contorno amarelo até que tenham sido mapeados para pelo menos um agente. As tarefas são destacadas através de uma figura oval. Um exemplo de diagrama de papéis no agentTool é demonstrado na figura 14.

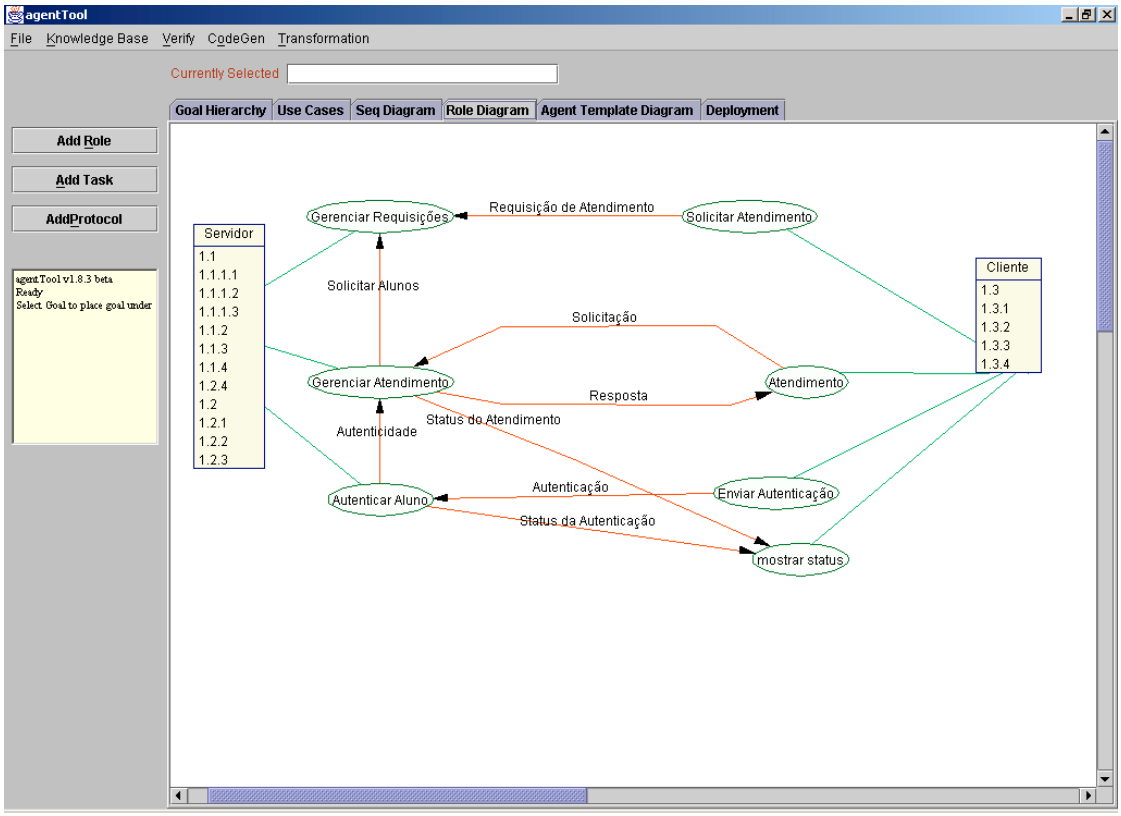

Figura 14 - Diagrama de Papéis

Fonte: Wood e Deloach (2001A)

O botão “Add Role” adiciona um novo Papel, “add Task” uma nova tarefa relacionada a um papel e o botão “addProtocol”, define uma mensagem que será passada entre as tarefas de cada papel.

6.18.2.3 Criando as Classes de Agentes

Deloach (2001b) demonstra que o primeiro passo no processo de construir agentes é o painel do diagrama do modelo de agentes. Este painel (mostrado na figura 15) permite que os agentes e diálogos entre agentes sejam declarados. Um agente é adicionado através do botão "addAgent". Para adicionar diálogos entre agentes dois agentes dever ser selecionados. Um agente executará alguns dos papéis definidos anteriormente. Agentes diferentes podem definir o mesmo papel.

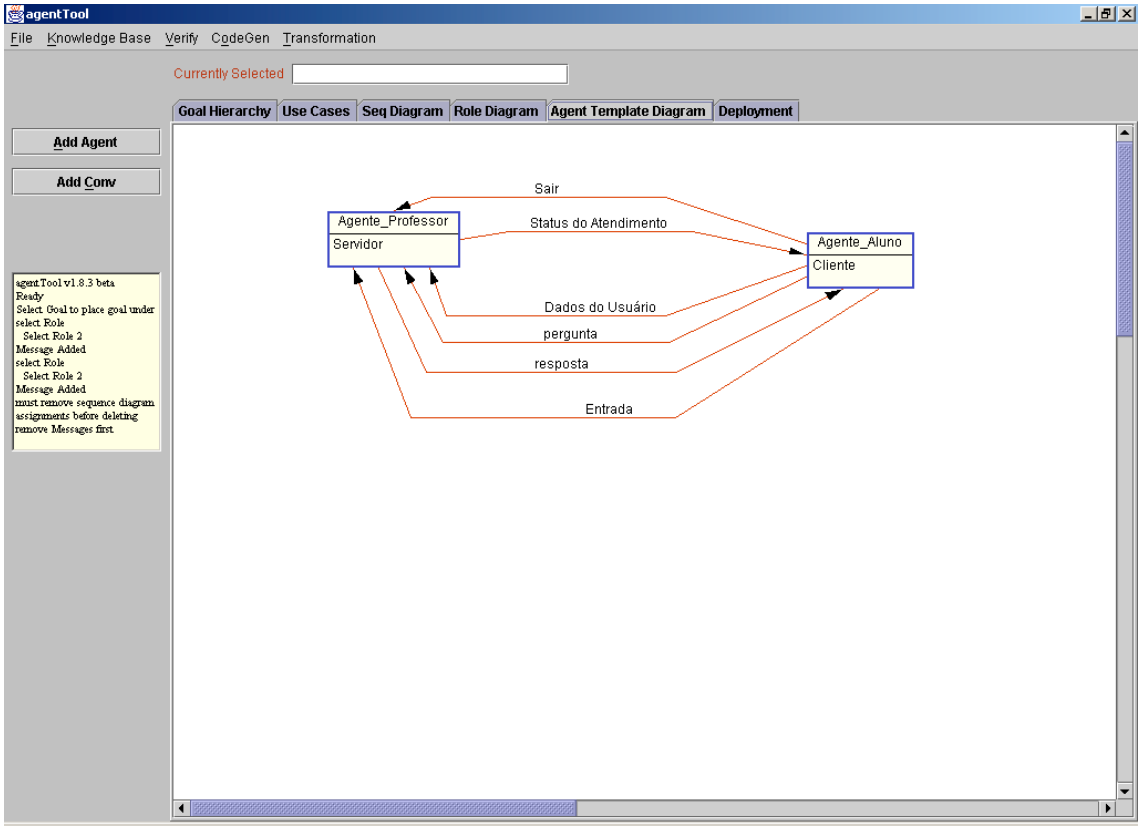

Figura 15 - Diagrama de Modelos de Agentes

Fonte: Deloach (2001b)

Segundo Wood (2001), a Arquitetura dos Agentes compreende em um conjunto de componentes e suas conexões, onde cada agente contem uma definição interna que mostra como o agente irá desempenhar os seus papéis.

Para cada Agente, pode-se definir os Componentes que o compõem, conforme demonstrado na figura 16.

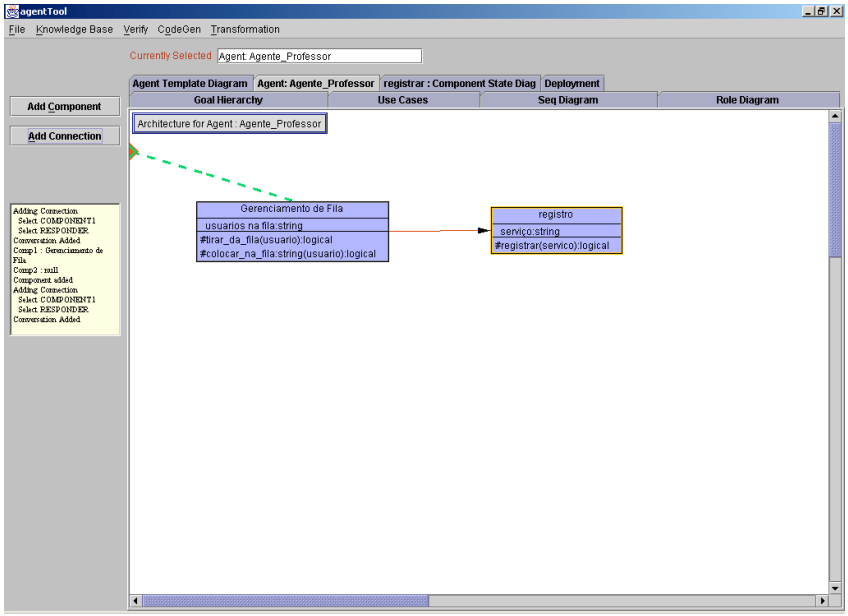

Figura 16 - Componentes de um Agente

Fonte: DELOACH (2001b)

Um componente é subdividido em diversas partes, Seu nome, seus atributos e seus métodos. Cada componente pode ter um diagrama de estado associado conforme demonstrado na figura 17.

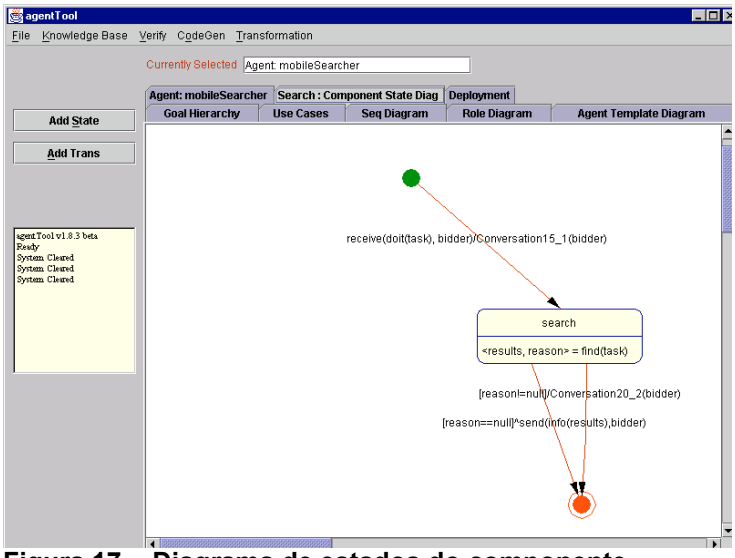

Figura 17 - Diagrama de estados do componente

Fonte: Deloach (2001b)

6.18.2.4 Construindo o Diálogo entre Agentes e Construindo as Classes de Agentes

A construção do diálogo é o próximo passo. E segundo (DELOACH, 2001b), este passo pode ocorrer paralelamente com a etapa anterior na montagem dos agentes.

Wood (2001) relata que um diálogo define o protocolo de coordenação entre dois agentes, composto por dois diagramas de classes de comunicação onde um par de máquinas de estados finitos definem o estado do diálogo de duas classes de agentes.

Na figura 18 é demonstrado o inicializador, que sempre inicia o diálogo enviando a primeira mensagem.

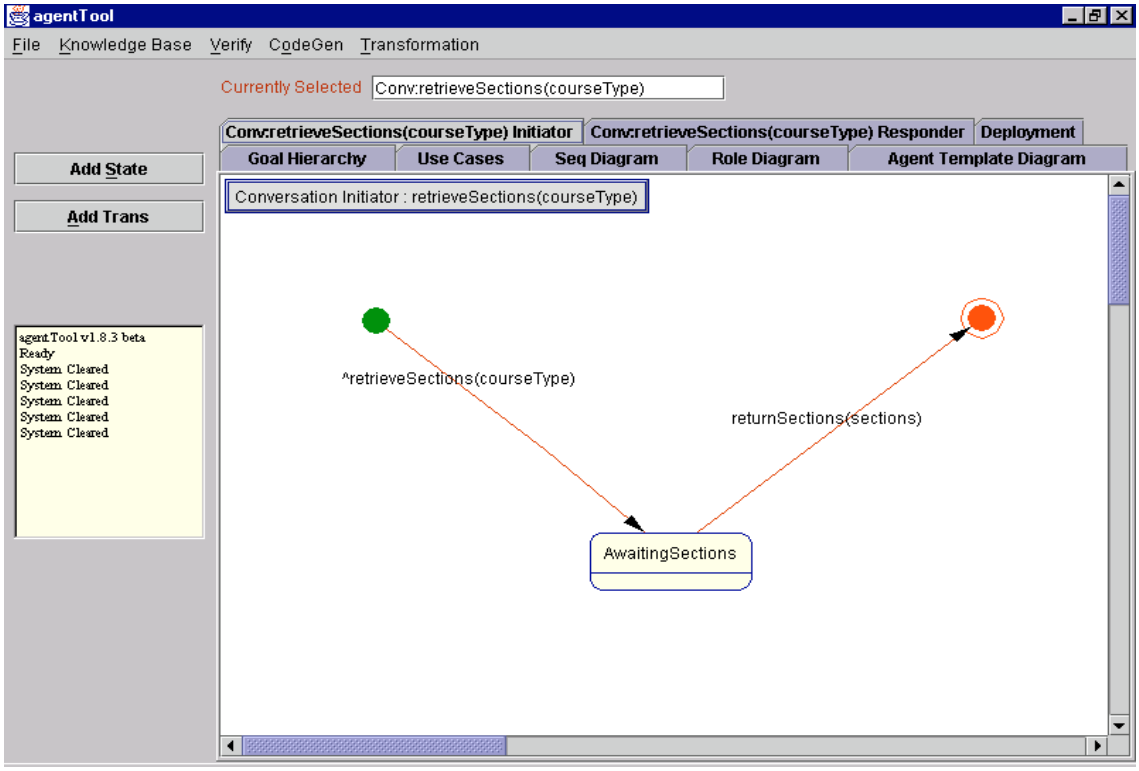

Figura 18 - Inicializador do diálogo

Fonte: Deloach (2001b)

Wood (2001) afirma que quando um agente recebe a mensagem, ele compara-a com os diálogos ativos. Se forem idênticos, o agente encaminha o diálogo apropriado para o novo estado e realiza as ações especificadas.

Na figura 19 é mostrado ao protocolo de Resposta, que é enviado ao agente inicializador, onde pode também ser observado o circulo verde e vermelho, que definem o estado inicial e o estado final respectivamente.

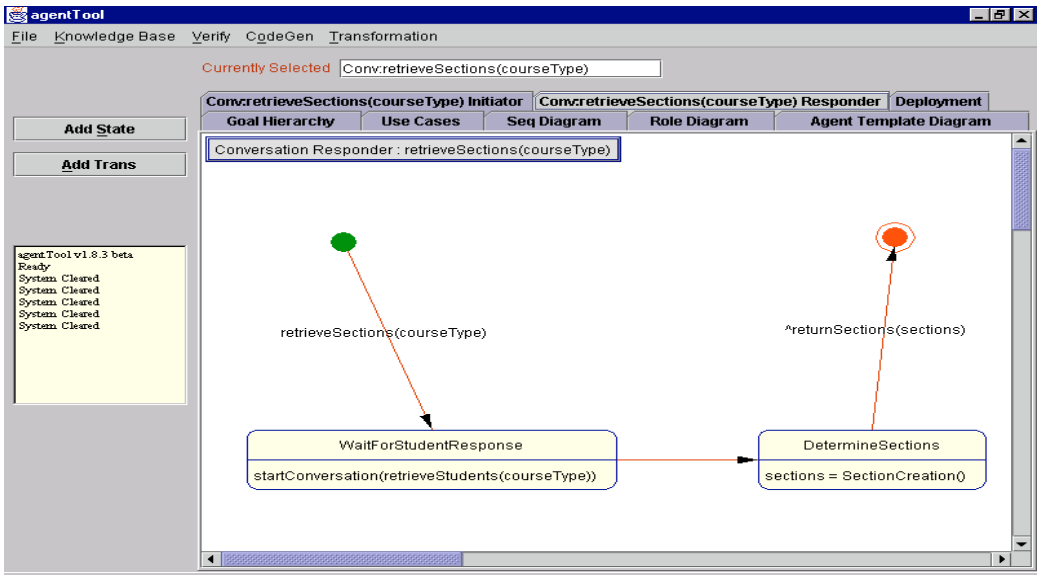

Figura 19 - Resposta do diálogo

Fonte: Deloach (2001b)

6.18.2.5 Desenvolvimento do Sistema

Segundo DELOACH (2001b), o painel de desenvolvimento (figura 20) é o primeiro passo no desenvolvimento do sistema de agentes, pois até este ponto, apenas foi especificada a funcionalidade abstrata desejada para o sistema.

Nesta etapa são criadas as instâncias de agentes a partir dos modelos de agentes e seus diálogos, que são conectadas em sistemas.

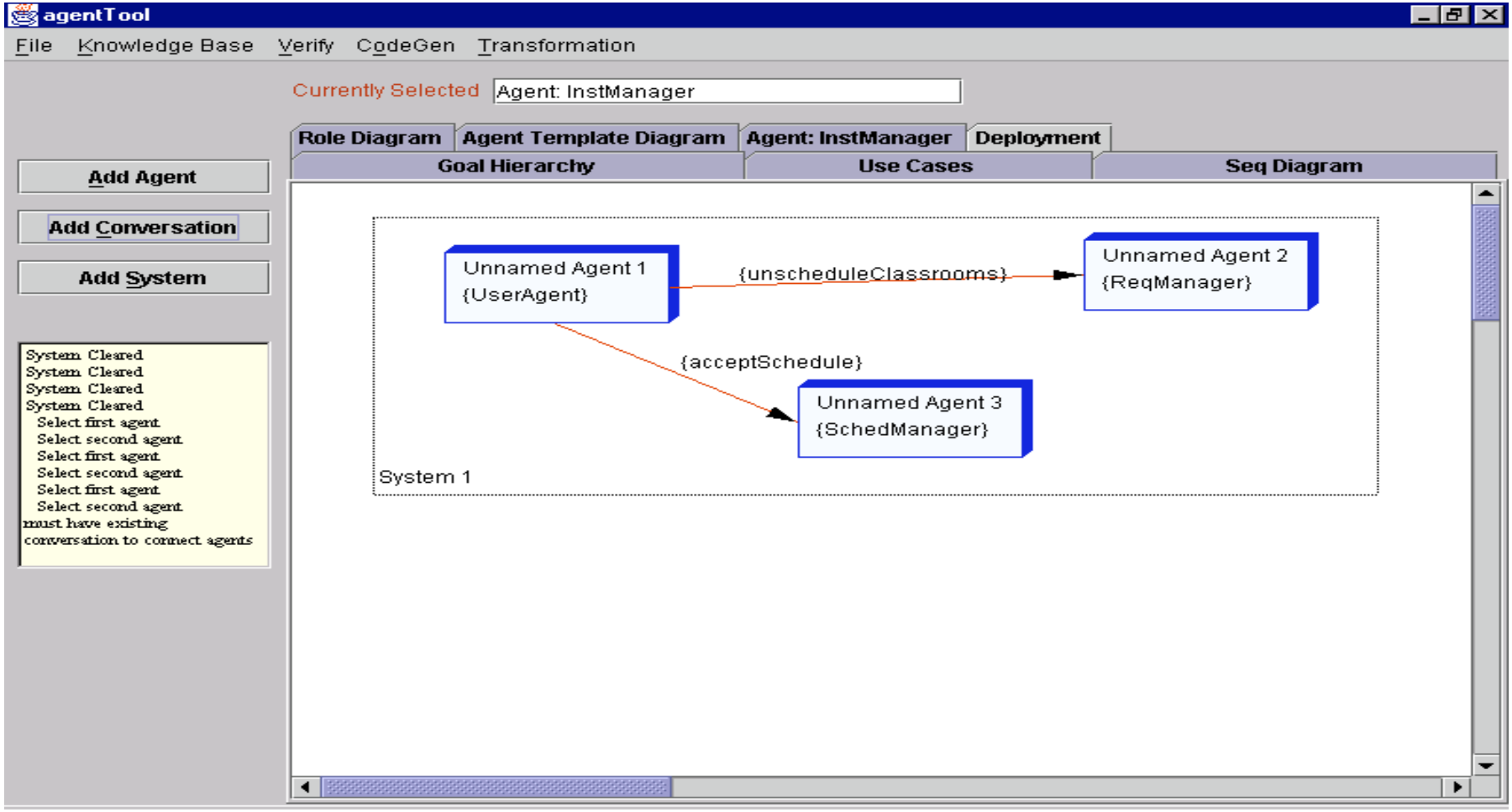

Figura 20 – Painel de Desenvolvimento, com as instâncias de agentes e seus diálogos.

Fonte: Deloach (2001b)

7 ESPECIFICAÇÃO DO SISTEMA PROPOSTO

Em cima de uma proposta Orientada a Agentes, foi modelado um Ambiente Virtual Institucional para o gerenciamento da comunicação entre professor e um grupo de alunos, com controle de atendimento individualizado.

O sistema está projetado para implementação futura no site das disciplinas ministradas pelo Prof. Marcelo Perotto no Centro Universitário Campos de Andrade, Uniandrade, localizada em Curitiba-PR

O sistema não será implementado neste trabalho, tendo sido efetuada tão somente a análise usando a metodologia MaSE e a especificação das classes de agentes através do AgentTool.

7.1 Objetivos do Sistema

O sistema deve gerenciar o atendimento on-line através de uma página na Internet. Os alunos serão atendidos, um de cada vez em um sistema de troca de mensagens.

Os alunos devem ser identificados através de um logon e uma senha, sendo validados pelo sistema.

É estipulado um tempo máximo para cada aluno, e os outros alunos ficam em uma fila de espera por ordem de chegada.

O sistema deve verificar também se, dentro do diálogo, ocorrem questões que já foram levantadas em outros diálogos e então, mostra a resposta dada anteriormente para a questão na tela do aluno.

O gerenciamento da fila de atendimento deve ser feito pelos agentes que ficam constantemente monitorando o estado do atendimento e repassando a posição do atendimento a cada aluno.

O aluno sendo atendido deve ter o seu diálogo estruturado e armazenado. O professor pode sinalizar partes do diálogo que considera interessantes para que sejam inseridas na lista das questões freqüentemente perguntadas.

O aluno que estiver digitando o diálogo, pode receber em janela anexa, as informações sobre faqs (questões freqüentemente perguntadas) através de um mecanismo que busca de palavras-chaves do diálogo em um banco de dados, evitando assim, a repetição de respostas por parte do professor. O sistema irá se

encarregar de forma inteligente da resposta. O faq pode ser ainda atualizado com novos detalhes provenientes de questões novamente abordadas.

Os alunos que estão em espera devem antever o tempo previsto de seu atendimento através do Status mostrado a todos os participantes da fila de atendimento.

O sistema deve permitir ainda o agendamento de atendimentos conforme a agenda do professor.

Lista dos Objetivos do Sistema

1. prover atendimento aos alunos

1.1 Admissão de atendimento para alunos

1.1.1 Identificar o aluno

1.1.1.1 Receber identificação do aluno

1.1.1.2 Receber senha

1.1.1.3 Verificar usuário e senha

1.1.2 Admitir até um aluno sendo atendido pelo professor

1.1.3 Exibir estado do atendimento Atual

1.1.4 Enfileirar alunos em espera

1.2 Gerenciar Atendimento à alunos

1.2.1 Controlar a seqüência do atendimento

1.2.2 Permitir aos alunos questionar

1.2.2.1 verificar se questionamento está contido em faqs

1.2.3 entregar e gerenciar respostas

1.2.3.1 Mostrar resultado de Faqs

1.2.4 Requerer que alunos com tempo esgotado deixem o atendimento

1.3 Prover interface para alunos

1.3.1 Permitir alunos entrar/sair do atendimento

1.3.2 Permitir aos alunos questionar

1.3.3 Mostrar o Status dos participantes

1.3.4 Mostrar mensagens dos Participantes

1.4 Agendar o atendimento

1.4.1 Verificar Agenda de Atendimentos

1.4.2 Agendar data de Atendimento

Os Objetivos do sistema levantados através dos requisitos básicos do sistema descritos acima serão utilizados nas etapas posteriores da análise.

Um modelo para a Interface de atendimento para o usuário é apresentada na figura 21.

Figura 21 - Interface com o usuário

7.2 Diagramas do sistema

Os diagramas a seguir, buscam demonstrar a estrutura e o comportamento do sistema.

7.2.1 Hierarquia de objetivos

O diagrama da figura 22, demonstra a Hierarquia de objetivos, utilizado para: Identificar objetivos e Estruturar objetivos.

O diagrama de Hierarquia de Objetivos é construído no AgentTool através de uma ferramenta que permite a estruturação de objetivos em forma de árvore. A captura dos objetivos consiste de duas etapas : identificar os objetivos e estruturá-los. Um analista deve identificar os objetivos pela extração dos requisitos básicos do sistema.

Estes requisitos podem incluir documentos técnicos detalhados, entrevistas com os usuários ou especificações formalizadas.

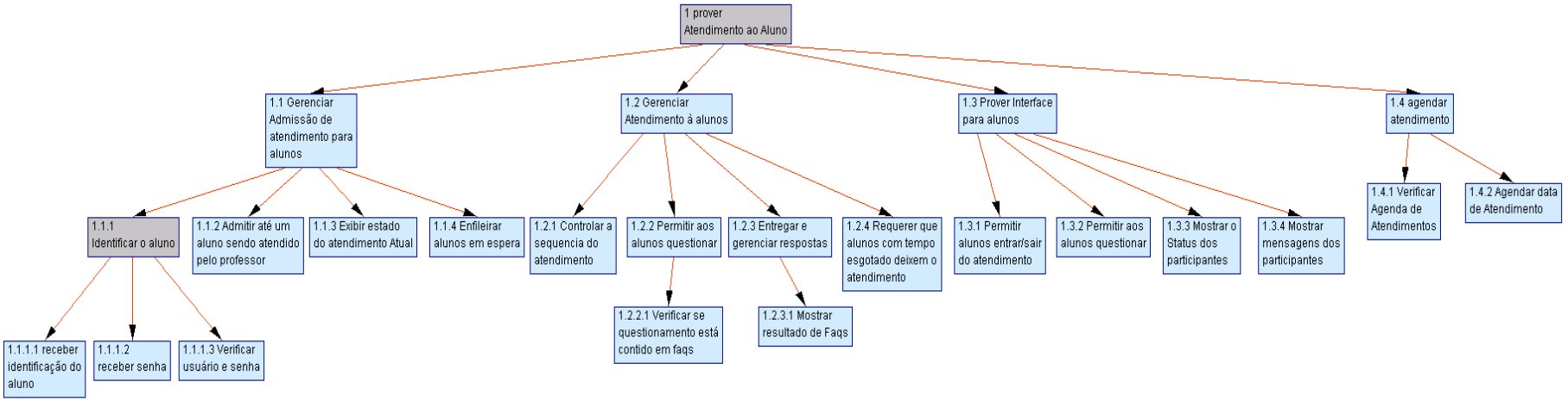

Figura 22- Diagrama de Hierarquia de objetivos

Fonte: Adaptado de Deloach (2001a).

Uma vez que os objetivos foram levantados e explicitados, os mesmos são detalhados em passos e atividades envolvidas em seu funcionamento.

O analista então, identifica os objetivos em um Diagrama Hierárquico, onde o analista organiza os objetivos pela sua importância. Cada nível da hierarquia deve conter objetivos pertencentes ao mesmo escopo.

O analista também deve identificar sub-objetivos necessários para realizar os objetivos-pais. Eventualmente, o analista associa cada objetivo com um papel ou conjunto de classes de agentes responsáveis pelo objetivo.

Na etapa de criação do diagrama de papéis, a ferramenta irá exibir todos os objetivos durante o mapeamento dos objetivos a serem realizados por cada papel. Os objetivos que não estão ligados a nenhum papel serão sinalizados, como forma de dirigir a análise.

7.2.2 Casos de uso

O diagrama da figura 23, é um passo crucial para a tradução de objetivos em papéis e tarefas associadas.

O analista relaciona os casos de uso a partir dos requisitos do sistema e dos usuários. Casos de Uso são descrições narrativas de uma seqüência de eventos que definem o comportamento desejado no sistema.

Os diagramas de Caso de Uso do sistema proposto estão apresentados nas figuras 24, 25 e 26

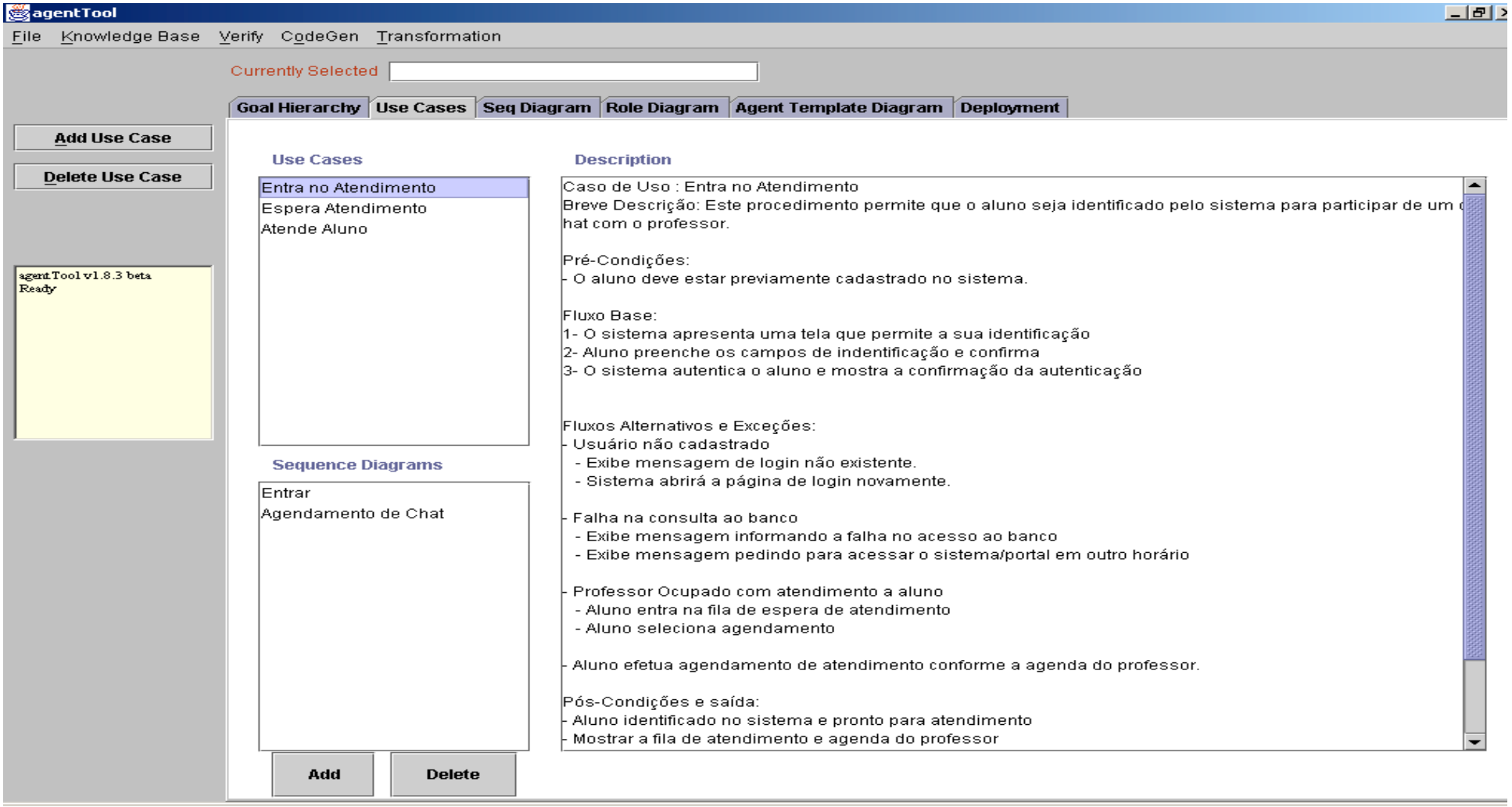

Figura 23 – Diagrama de Casos de Usos

Caso de Uso : Entra no Atendimento
Breve Descrição: Este procedimento permite que o aluno seja identificado pelo sistema para participar de um chat com o professor.
Pré-Condições: O aluno deve estar previamente cadastrado no sistema. **Fluxo Base:** 1- O sistema apresenta uma tela que permite a sua identificação 2- Aluno preenche os campos de identificação e confirma 3- O sistema autentica o aluno e mostra a confirmação da autenticação **Fluxos Alternativos e Exceções:** Usuário não cadastrado • Exibe mensagem de login não existente. • sistema abrirá a página de login novamente. Falha na consulta ao banco • Exibe mensagem informando a falha no acesso ao banco • Exibe mensagem pedindo para acessar o sistema/portal em outro horário Professor Ocupado com atendimento a aluno • Aluno entra na fila de espera de atendimento • Aluno seleciona agendamento Aluno efetua agendamento de atendimento conforme a agenda do professor. **Pós-Condições e saída:** • Aluno identificado no sistema e pronto para atendimento • Mostrar a fila de atendimento e agenda do professor **Regras de Negócio:** • A fila de atendimento deve ser mostrada somente se o professor estiver on-line.
Diagramas de seqüência
Entrar Agendamento de Chat

Figura 24 - Caso de Uso Entra no Atendimento

Caso de Uso : Espera Atendimento
Breve Descrição: Este procedimento gerencia uma fila de atendimento, mantendo o aluno informado sobre o status atual do atendimento.
Pré-Condições: • O Aluno deve estar identificado. **Fluxo Base:** 1- Sistema apresenta uma tela com a o aluno sendo atendido e a situação do atendimento. 2- O aluno é atendido pelo professor **Fluxos Alternativos e Exceções:** • O Aluno pode sair da fila de atendimento, encerrando o use case. • O Professor pode encerrar o atendimento, finalizando o use case. **Pós-Condições e saída:** • Aluno entra em atendimento com o professor **Regras de Negócio:** • Se não houver nenhum aluno sendo atendido pelo professor, o aluno será atendido imediatamente. Caso contrário, aguardará por sua vez em uma fila de atendimento.
Diagramas de seqüência
Esperar Atendimento

Figura 25 - Caso de Uso Espera Atendimento

Caso de Uso : Atender Aluno

Breve Descrição: Este sistema gerencia a comunicação individual entre aluno e professor, onde o aluno pode verificar uma lista com as questões freqüentemente perguntada atualizadas automaticamente de acordo com o contexto do diálogo sendo conversado.

Pré-Condições:

- O aluno deve ter saído da fila de atendimento
- O professor deve estar logado.

Fluxo Base:

4- O Sistema apresenta uma tela com informações sobre a fila de atendimento, tempo restante, lista das questões freqüentemente perguntadas relativa ao contexto do diálogo.
5- O aluno digita as suas dúvidas e submete ao professor através do pressionamento da tecla correspondente ao envio da pergunta.
6- O sistema varre a string digitada, formando expressões de busca para em um banco de dados das ocorrências de questionamentos já foram levantados anteriormente e armazenados.
7- O professor responde os questionamentos do aluno, e pode escolher os que considera mais interessantes para incluir no faq, relacionado-os com outros assuntos abordados.
8- Ao se esgotar o tempo do aluno, o mesmo é informado do fim do atendimento, sendo desconectado.

Fluxos Alternativos e Exceções:

- O Aluno pode sair do atendimento, encerrando o use case.
- O Professor pode encerrar o atendimento, finalizando o use case.
- O Professor pode prolongar o atendimento, impedindo a sua finalização por tempo.

Pós-Condições e saída:

O atendimento ao aluno foi efetuado

Regras de Negócio:

- O aluno é atendido até que saia do atendimento ou o seu tempo seja esgotado.

Diagramas de seqüência

Atender Aluno

Verificar FAQ

Figura 26 - Caso de Uso Atender Aluno

Para auxiliar a determinação da comunicação exigida em um sistema MultiAgente, o analista deve reestruturar os Casos de Uso em Diagramas de Seqüência.

O diagrama de seqüência descreve uma seqüência de eventos entre múltiplos papeis e define a comunicação mínima que deve existir entre os mesmos. Os papéis identificados nesta etapa formam um conjunto inicial usado para definir os papéis do sistema da próxima etapa onde o analista irá usar os eventos identificados aqui para ajudar a definir as tarefas e eventualmente os diálogos entre agentes.

Os Diagramas de Seqüência são acessados através da sua seleção na aba “Use Cases” do AgentTool e posterior seleção da aba “Seq diagram” conforme demonstrado na figura 27.

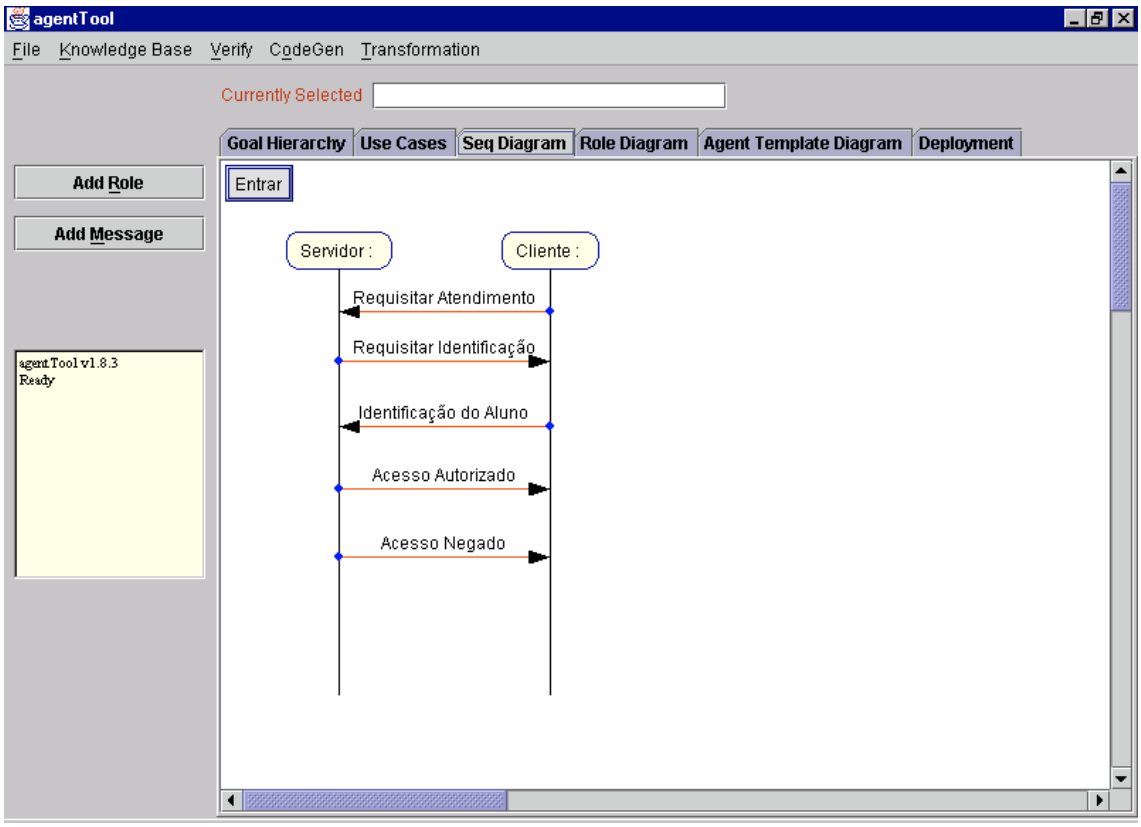

Figura 27 - Diagramas de Seqüência

Na figura 28 é demonstrado o Diagrama de Seqüência entrar, extraído do caso de utilização Entrando no Atendimento. Neste diagrama são identificados dois papéis. O Papel Servidor e o Papel Cliente. Estes papéis serão futuramente as classes de agente a serem implementadas. Entre estes Papéis estão as mensagens

que trafegam entre eles. As mensagem serão refinadas e implementadas como diálogos entre as classes de agentes.

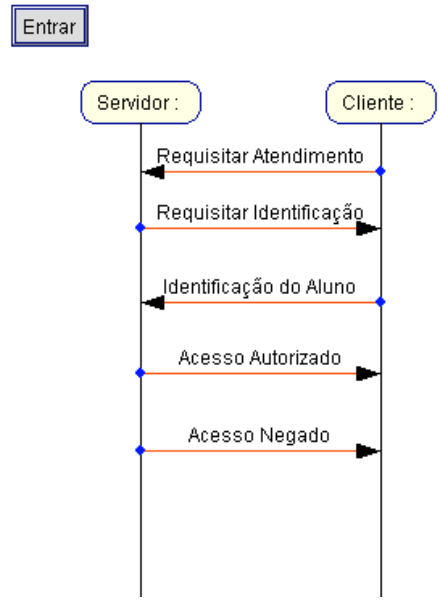

Figura 28 - Diagrama de seqüência – Entrar

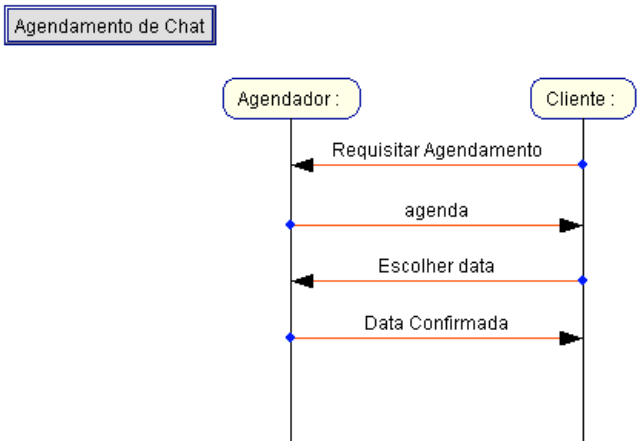

Figura 29 - Diagrama de seqüência – Agendamento de Chat

A figura 29 mostra o Diagrama de Seqüência Agendamento de Chat, extraído do caso de utilização Entrando no Atendimento. Aqui identificamos mais um papel, o Agendado com as mensagens que são trocadas com o papel Cliente.

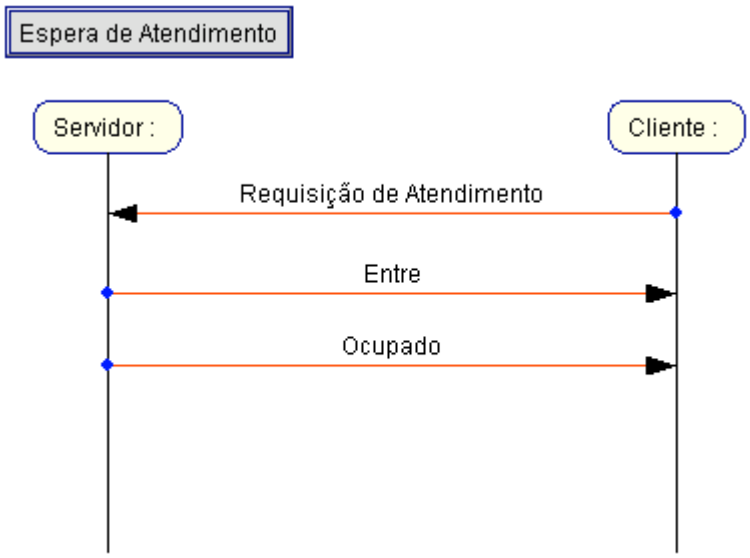

Figura 30 - Diagrama de Seqüência – Espera de Atendimento

A figura 30 mostra o Diagrama de Seqüência Espera de Atendimento, que mostra o gerenciamento da fila de atendimento. Este Diagrama é extraído do caso de uso Esperando Atendimento.

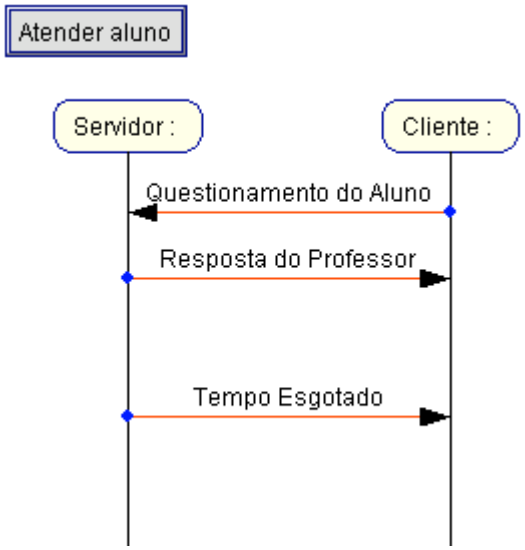

Figura 31 - Diagrama de Seqüência - Atender Aluno

Na figura 31 é demonstrado o Diagrama de Seqüência atender aluno, do Caso de Uso Atendendo Aluno. Este diagrama mostra a comunicação entre o professor e o aluno, Através dos papéis Servidor e Cliente. As mensagens trafegadas são o

conteúdo do diálogo entre os mesmos. O aluno é notificado quando o seu tempo esgotou.

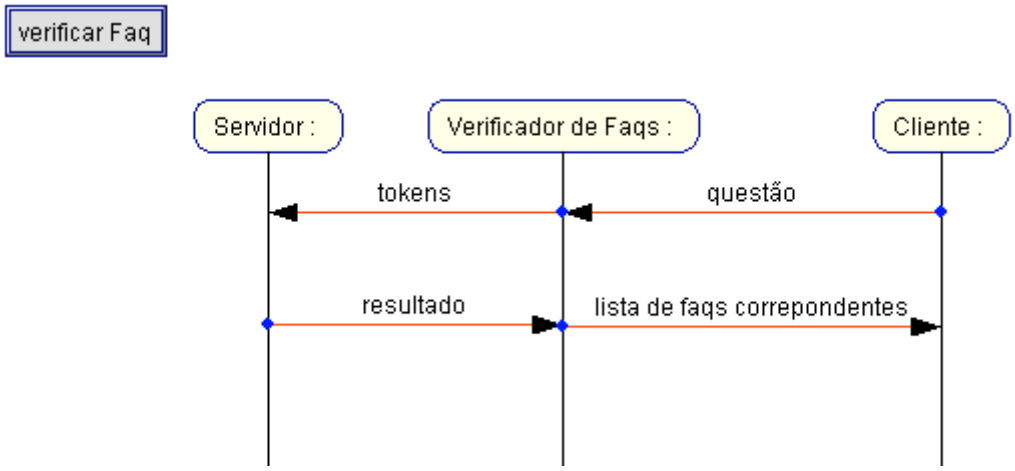

Figura 32 - Diagrama de Seqüência - Verificar FAQ

Na Figura 32 é apresentado o Diagrama de Seqüência Verificar FAQ, que demonstra o monitoramento do diálogo entre os papéis Servidor e Cliente através do papel Verificador de Faqs. Este papel analisa o texto do diálogo do cliente e procura itens relevantes no diálogo para a realização de uma pesquisa no banco de dados do servidor. O conjunto de resultados é disponibilizado na interface do cliente pela mensagem Lista de faqs correspondentes.

7.2.3 Diagrama de Papéis

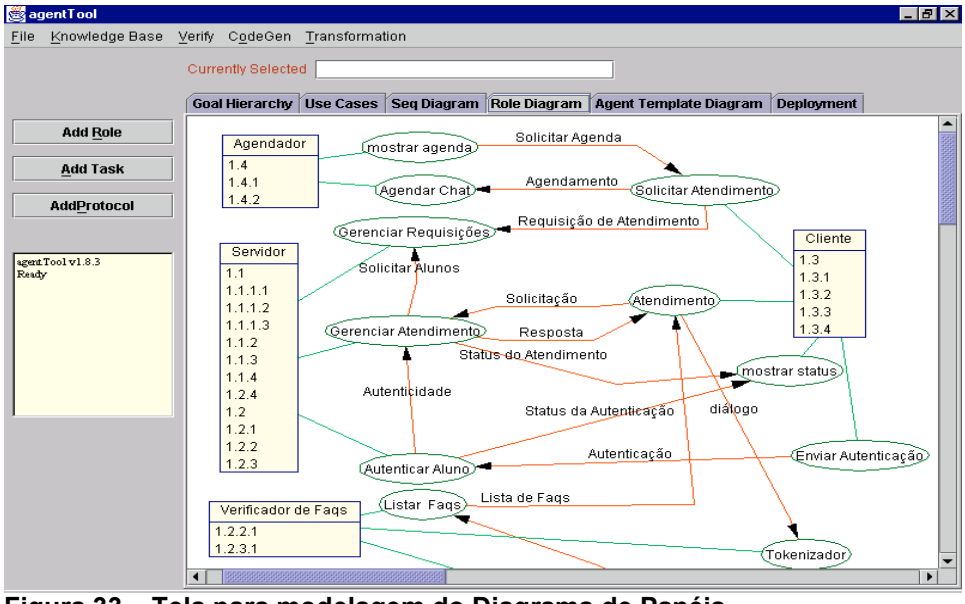

Figura 33 – Tela para modelagem do Diagrama de Papéis

O Diagrama de Papeis da figura 33, serve para verificar se todos os papéis necessários foram identificados e são inseridas as tarefas que definem o comportamento de um papel e os padrões de comunicação.

Os papéis foram identificados nos Diagramas de Seqüência, juntamente com as mensagens entre os papéis que são agora refinadas.

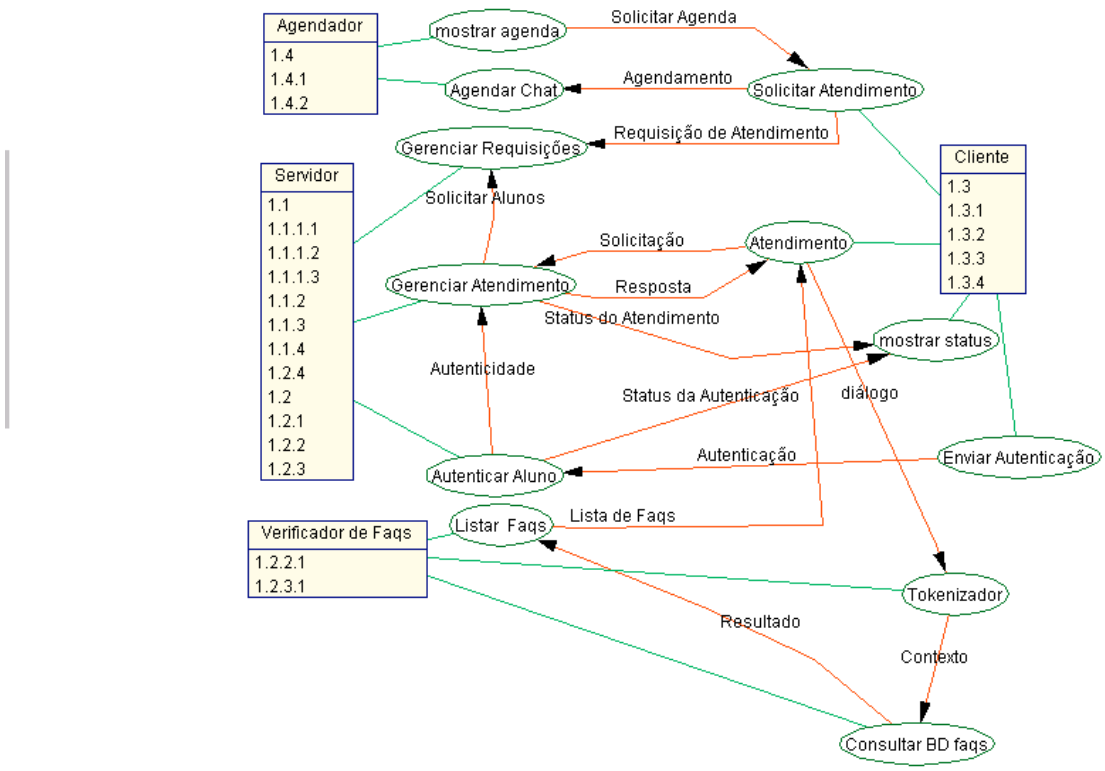

Figura 34 - Diagrama de Papéis

Na figura 34 está o diagrama completo dos papéis identificados no sistema proposto. Os retângulos amarelos demonstram os papéis, juntamente com a numeração interna, que representa os objetivos que devem cumprir vindos do diagrama hierárquico de objetivos.

Os papéis devem ser implementados posteriormente por pelo menos um agente.

Próximos aos papéis estão as elipses que representam as tarefas. Toda a tarefa deve pertencer a um papel.

As linhas que conectam as tarefas são os protocolos. As mensagens podem ser trocadas dentro do mesmo papel ou entre dois papéis diferentes.

7.2.4 Diagrama de Modelos de Agentes

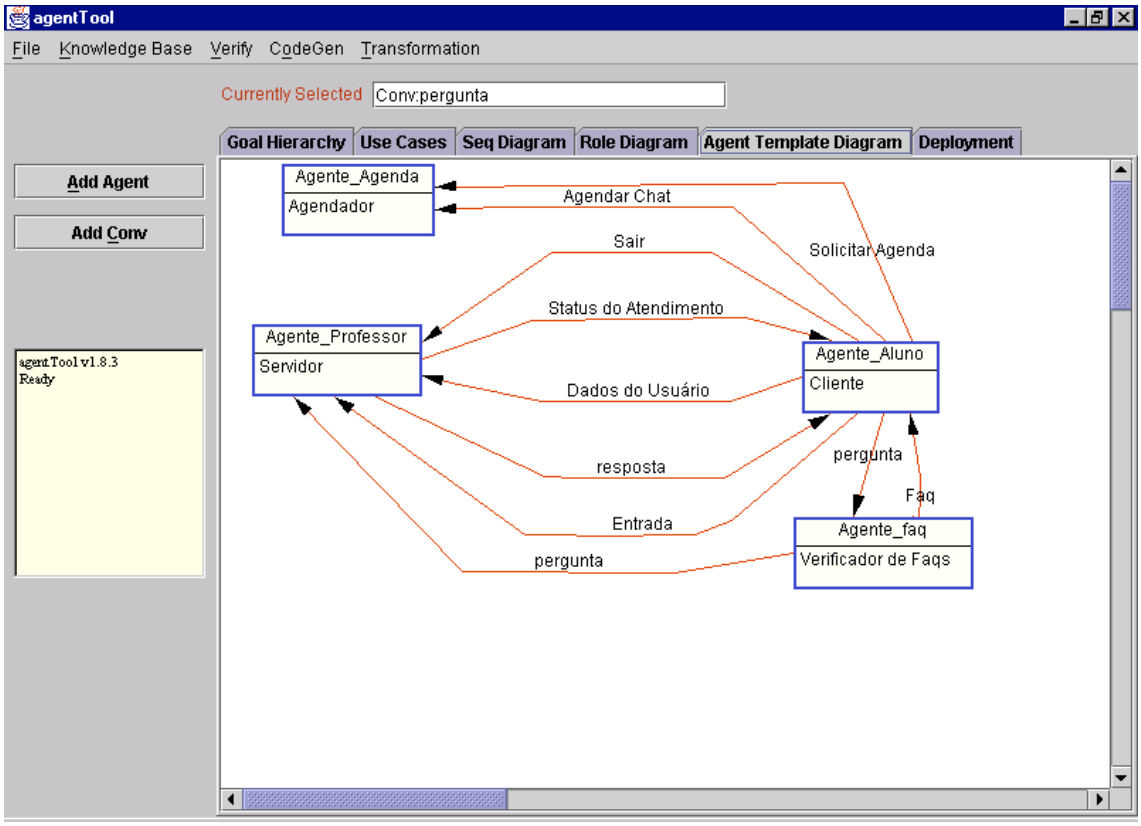

Figura 35 - Diagrama de Modelos de Agentes

A especificação dos agentes é demonstrada na figura 35 através da aba Agent Template Diagram. Este painel também é utilizado para a especificação dos diálogos entre os agentes. Um agente devem implementar alguns dos papéis definidos anteriormente. De modo abstrato, os agentes operam da mesma maneira, mas cada implementação de agente deve gerenciar um tipo diferente de recurso. No modelo proposto, cada agente está implementando um papel.

7.2.5 A Arquitetura dos Agentes

Todo agente contém uma definição interna que descreve como o agente irá desempenhar os seus papéis. Esta definição é chamada de “Arquitetura do Agente”

e é composta por um conjunto de componentes e conexões entre estes componentes. A arquitetura do agente aluno está demonstrada na figura 36.

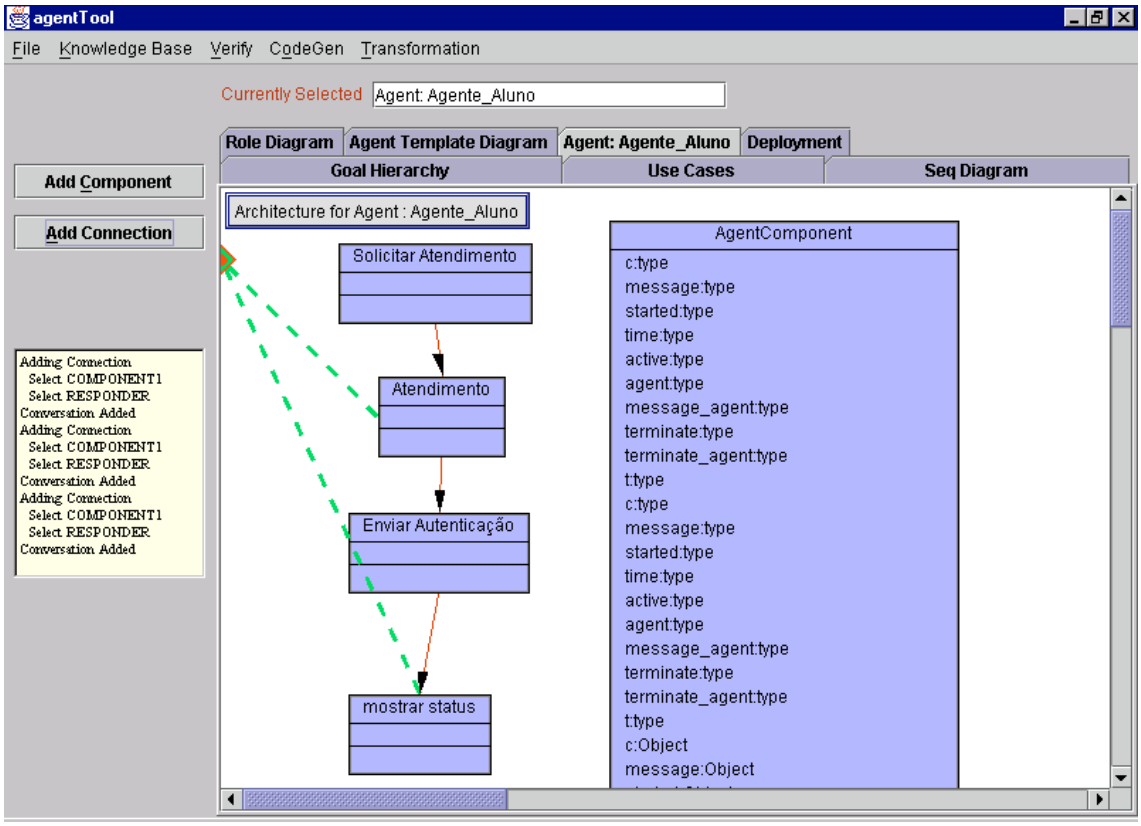

Figura 36 - Arquitetura do Agente Agente_aluno

No painel da arquitetura dos Agentes é possível especificar os atributos que funcionam como variáveis em um programa de computador. Pode ser definido para cada atributo o seu nome, tipo, valor inicial, se são definidos pelo usuário ou em tempo de execução e qual o tipo de conjunto os mesmos representam.

A próxima etapa é a definição dos métodos, que recebem certos parâmetros sobre certas condições ou produzem uma resposta de um determinado tipo.

As conexões entre os componentes são representadas como caminhos para as mensagens entre os componentes. A única informação que o conector contém é

quais componentes são conectados e o sentido da mensagem em um sentido abstrato. Um conector pode permitir tráfico em uma única direção ou em duas vias.

Outro tipo de conexão funciona como uma porta de entrada-saída da arquitetura.

Cada tarefa dentro de um agente pode ter ainda um Diagrama de Estados do Componente.

7.2.6 Diálogos entre agentes

Os agentes se comunicam através de Diálogos entre Agentes.

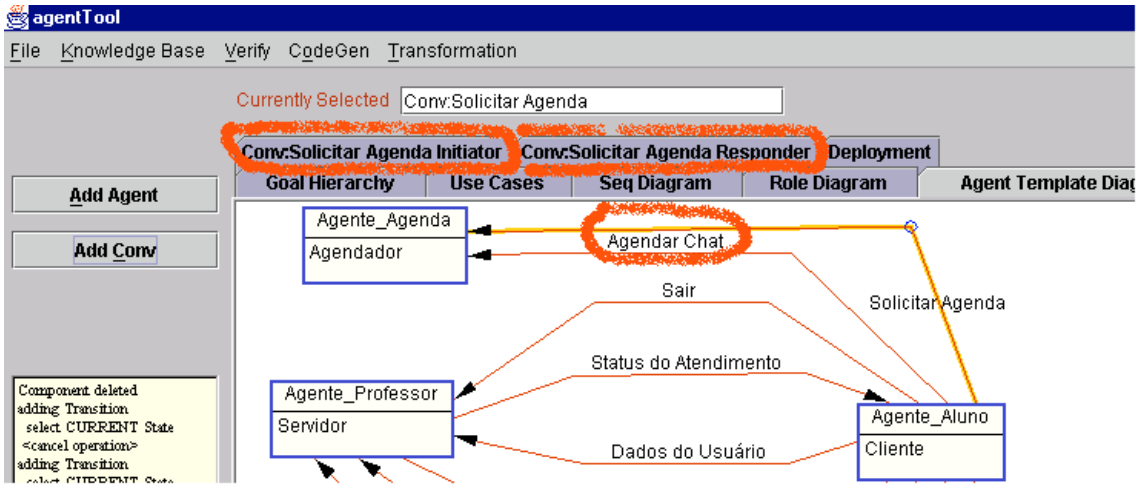

Figura 37 - Descrevendo a Comunicação entre os Agentes

Conforme no exemplo da figura 37, todo diálogo entre agentes é composto por dois diagramas que o descrevem. O Diagrama de estados inicializador e o Diagrama de Estado de Resposta. O agente ao se comunicar estará em um estado inicial e baseado nos dados de entrada irá mudar para o estado apropriado. Nas figuras de número 37 a 38, o protocolo de coordenação entre dois agentes é definido, tendo um inicializador e um reportador. O inicializador inicia um diálogo pelo envio da primeira mensagem. As mensagens são trocas binárias entre os agentes individuais

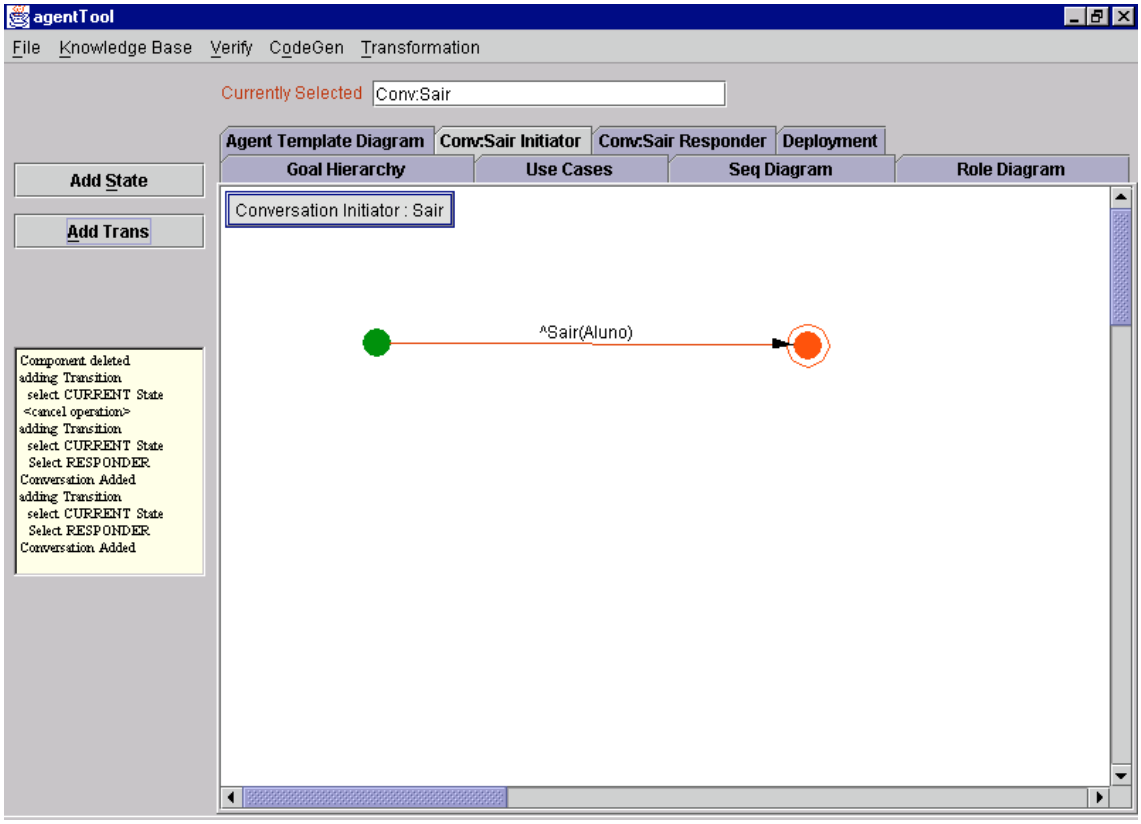

Figura 38 - Inicializador do diálogo entre agentes Sair

Na figura 38 é descrito o diálogo para solicitar a saída do sistema. A mensagem sair será enviada do agente Agente_aluno para Agente_professor. A mensagem a será transportada pelo protocolo sair com o dado Aluno.

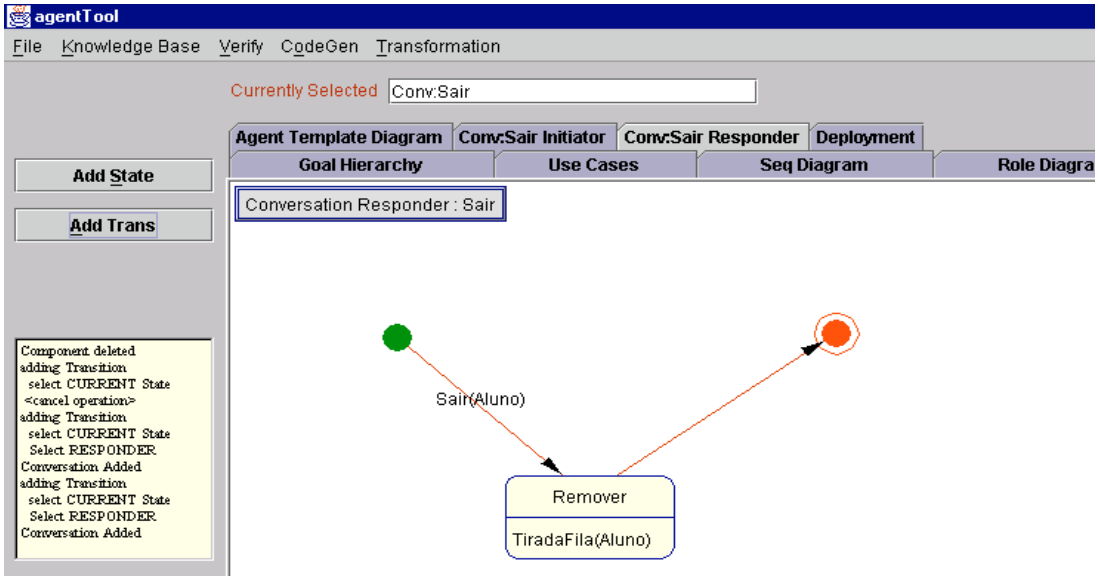

Figura 39 - Resposta do diálogo entre agentes Sair

Na figura 39 é demonstrado a resposta do diálogo entre agentes Sair, onde a ação tiradafila(aluno) é executada. Liberando o professor para atender o próximo aluno.

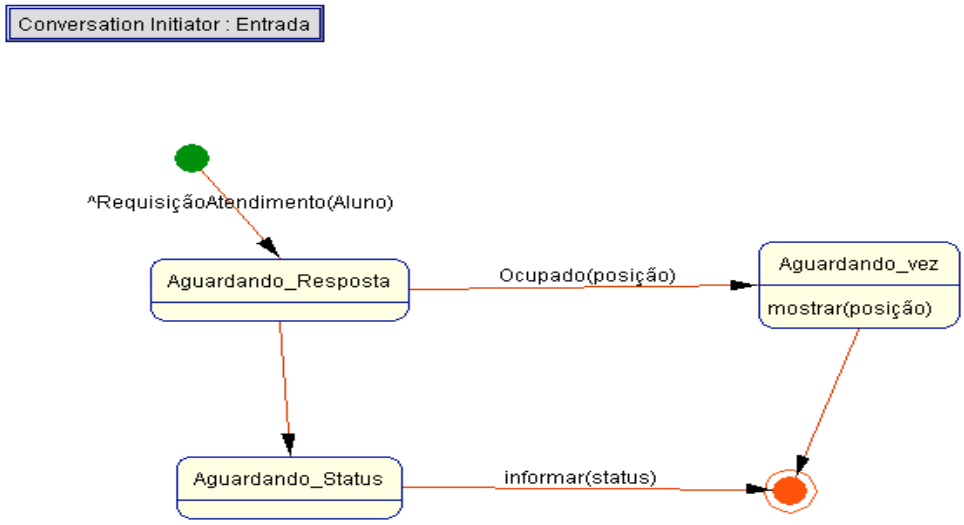

Figura 40 - Diálogo Entrada - Inicialização

Na inicialização do diálogo Entrada, o aluno requisita o atendimento, disparando um processo que responderá se o professor está ocupado ou se o aluno pode ser atendido diretamente.

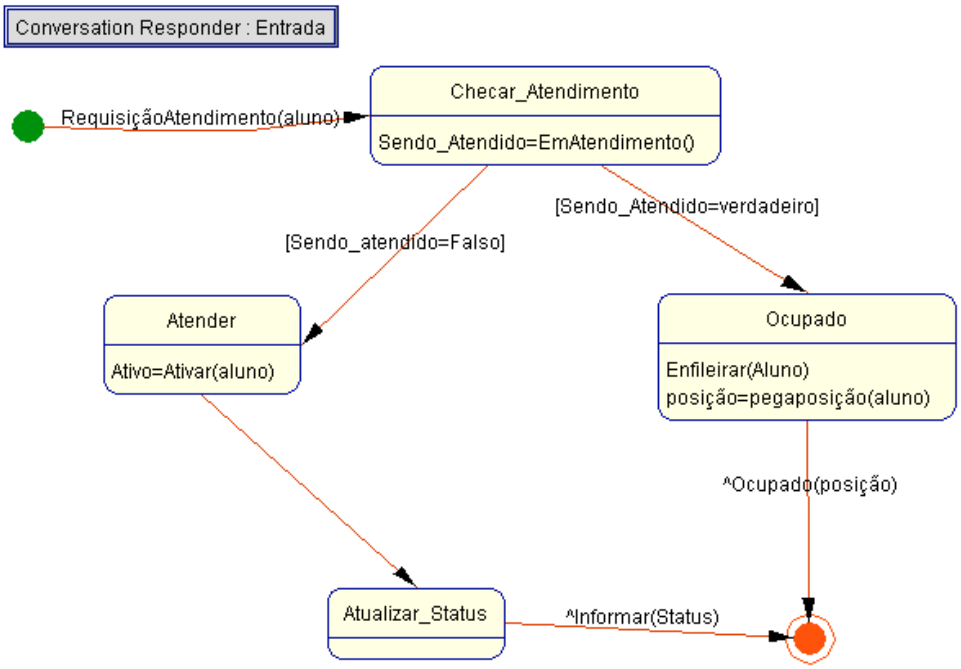

Figura 41 - Diálogo Entrada – Resposta

Na figura 41 é demonstrada a resposta do diálogo Entrada, onde é verificado o flag EmAtendimento. Se o flag estiver ativo, o aluno é colocado em uma fila de atendimento, sendo então informado de sua posição na espera. Se não houver aluno sendo atendido, o aluno atual passa a ser o aluno ativo.

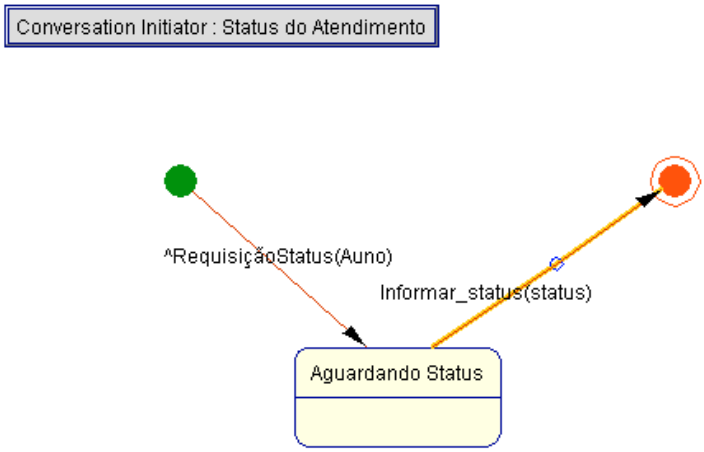

Figura 42 - Status do Atendimento - inicialização

Na figura 42 é demonstrado a informação da requisição de Status de Atendimento. No decorrer da espera, o aluno é informado em períodos de tempo da situação da fila de atendimento.

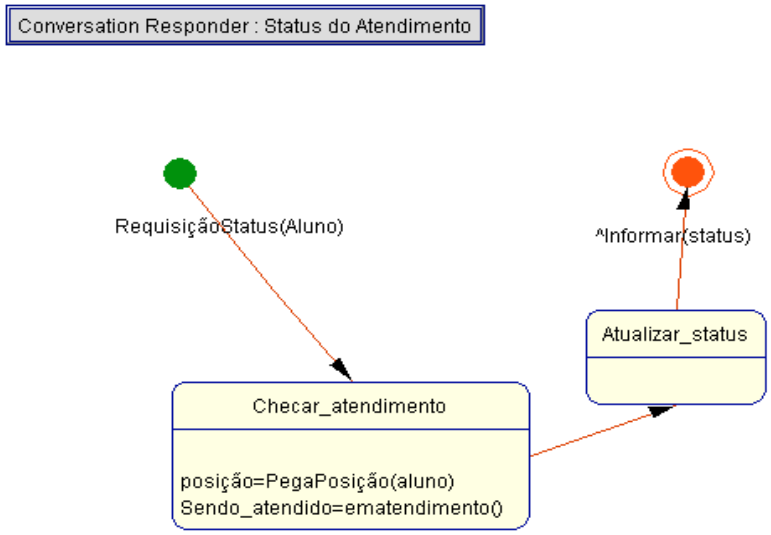

Figura 43 - Status do Atendimento - Resposta

Na figura 43 é mostrada a resposta ao diálogo de solicitação do status de atendimento, onde será recuperada a posição do solicitante e o aluno sendo atendido dentro da fila de atendimento.

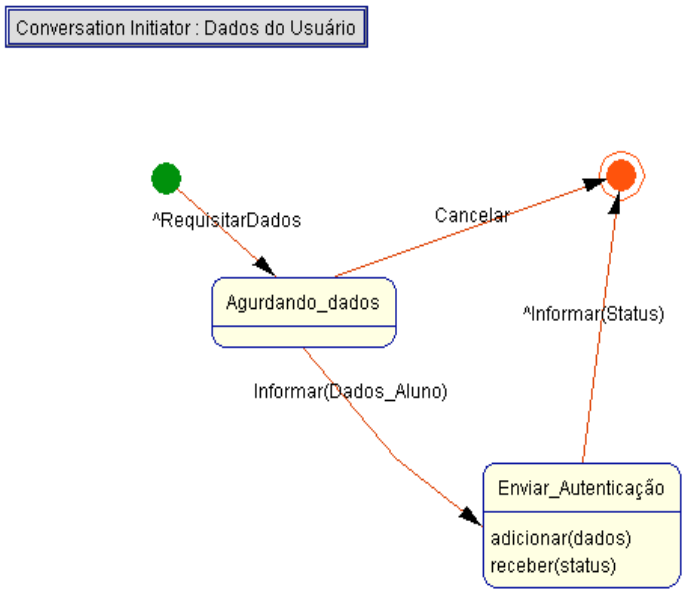

Figura 44 - Dados do Usuário – Inicialização

O diálogo entre agentes demonstrado na figura 44 corresponde a requisição de dados do usuário para a sua identificação e entrada no sistema.

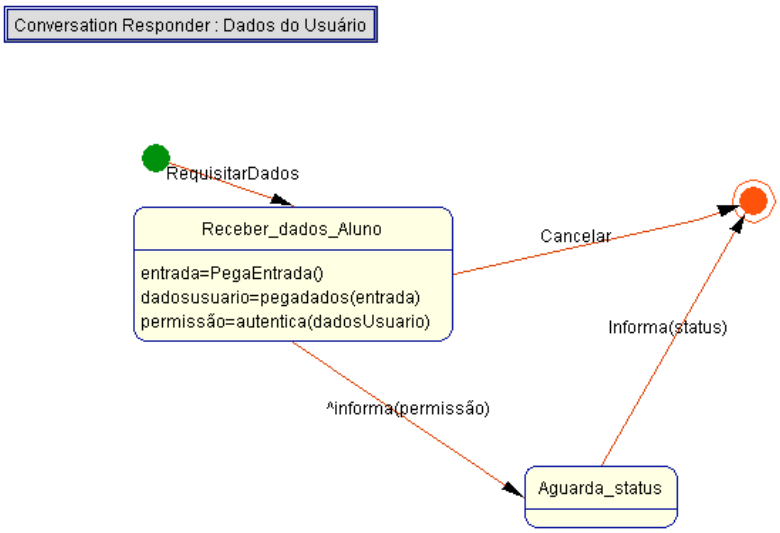

Figura 45 - Dados do Usuário - Resposta

O dialogo demonstrado na figura 46 demonstra o retorno das informações solicitadas, onde os dados do usuário são validados através da consulta em seu cadastro. O usuário receberá a informação sobre a autenticação.

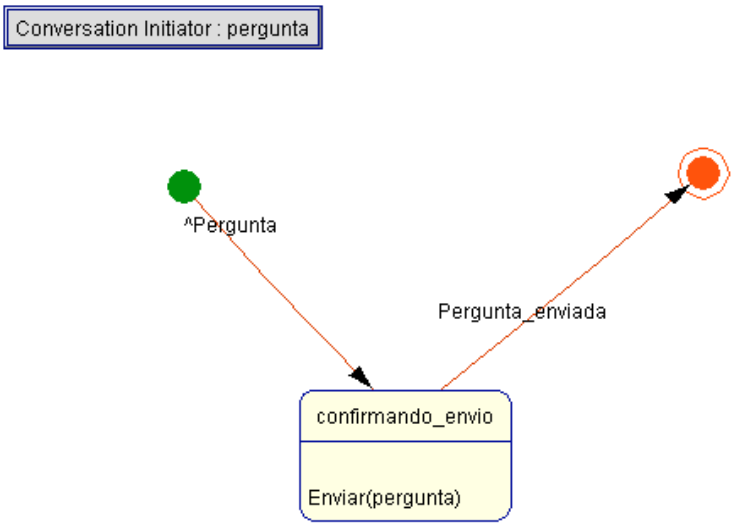

Figura 46 - Pergunta do Usuário - Inicializador

A figura 47 mostra a inicialização de um troca de mensagem entre agentes com os dados referentes a um questionamento de aluno sendo enviado ao agente professor.

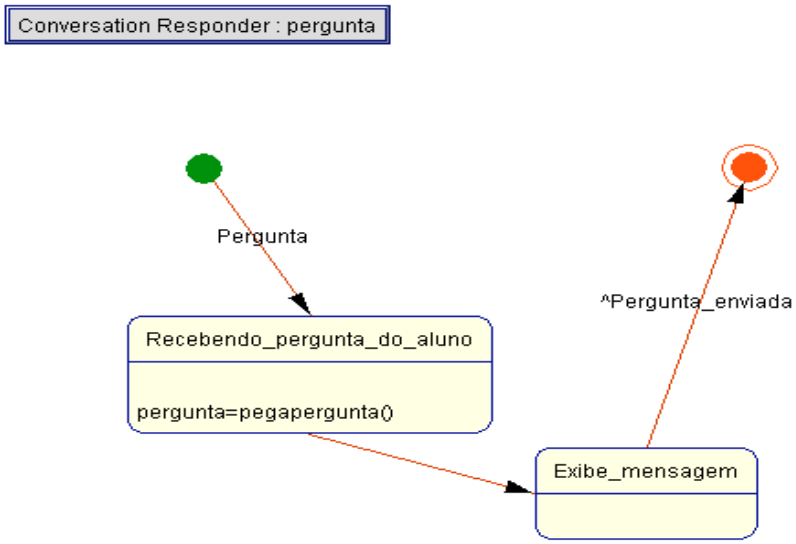

Figura 47 - Pergunta do Usuário - Resposta

A figura 47 mostra o fluxo resposta de mensagem entre os agentes, com a pergunta sendo enviada ao agente professor. Ao agente que inicializou o diálogo é enviada a confirmação de pergunta recebida.

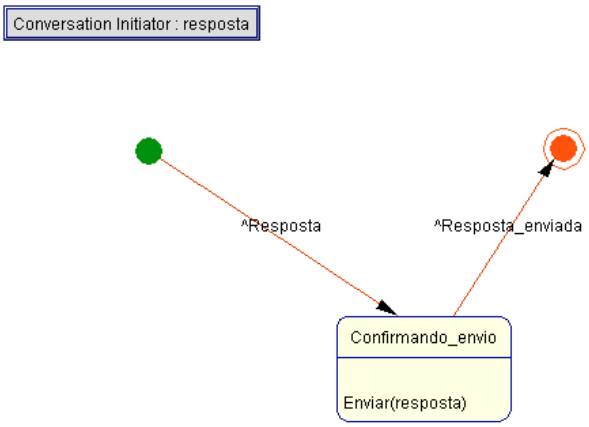

Figura 48 - Resposta do Professor - Inicializador

A mensagem trocada entre agentes demonstrada na figura 48, corresponde à inicialização do diálogo entre o agente professor e o agente aluno, onde o agente professor esta solicitando o envio de uma resposta ao agente aluno.

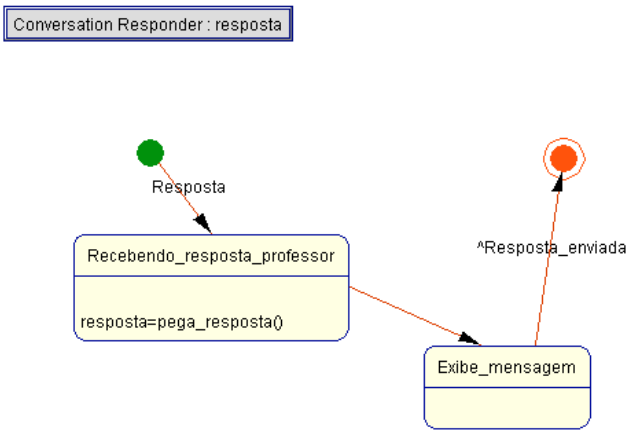

Figura 49 - Resposta do Professor – Resposta

Na figura 49, a resposta do professor é enviada ao agente aluno, sendo confirmado o seu envio ao agente professor.

Figura 50 - Inicialização do Diálogo - Solicitar Agenda

A inicialização do diálogo solicitar agenda é demonstrada na figura 50, onde o Agente Aluno solicita a agenda do professor de um determinado período.

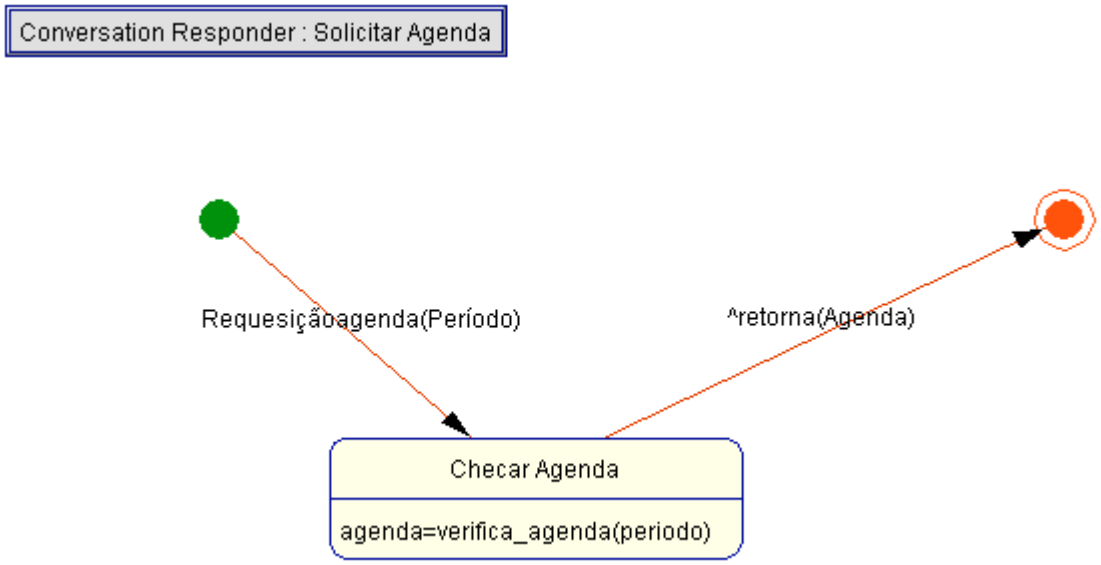

Figura 51 - Resposta do Diálogo - Solicitar Agenda

O agente Professor irá responder à requisição da agenda feita pelo agente aluno no período solicitado conforme demonstrado na figura 51.

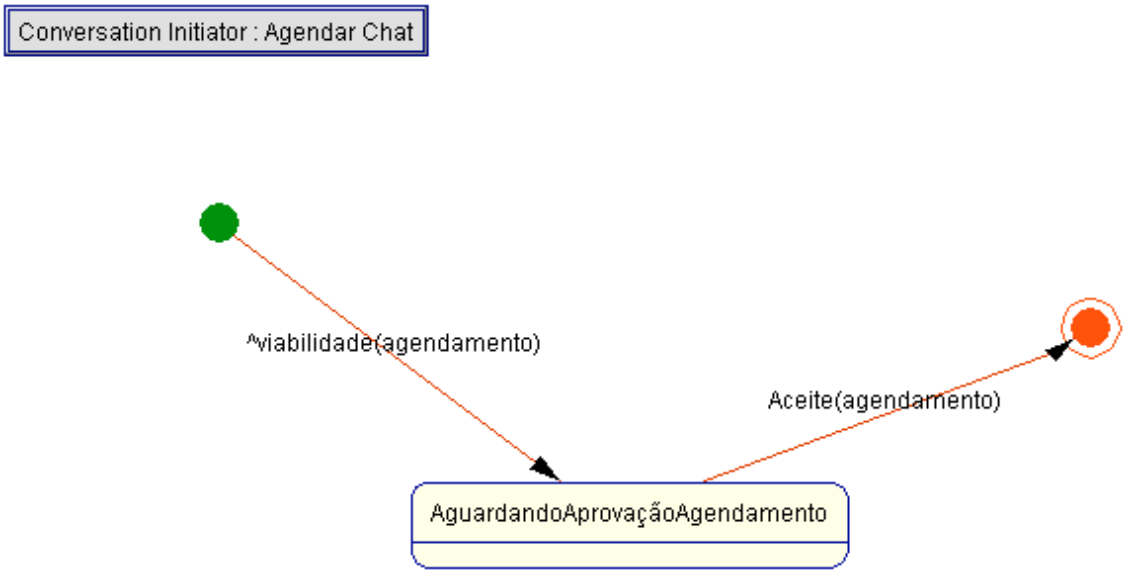

Figura 52 - Inicialização do Diálogo - Agendar Chat

Na figura 52 é visualizado a inicialização do diálogo agendar chat, requisitado pelo agente aluno, com as informações referentes à data e hora de agendamento.

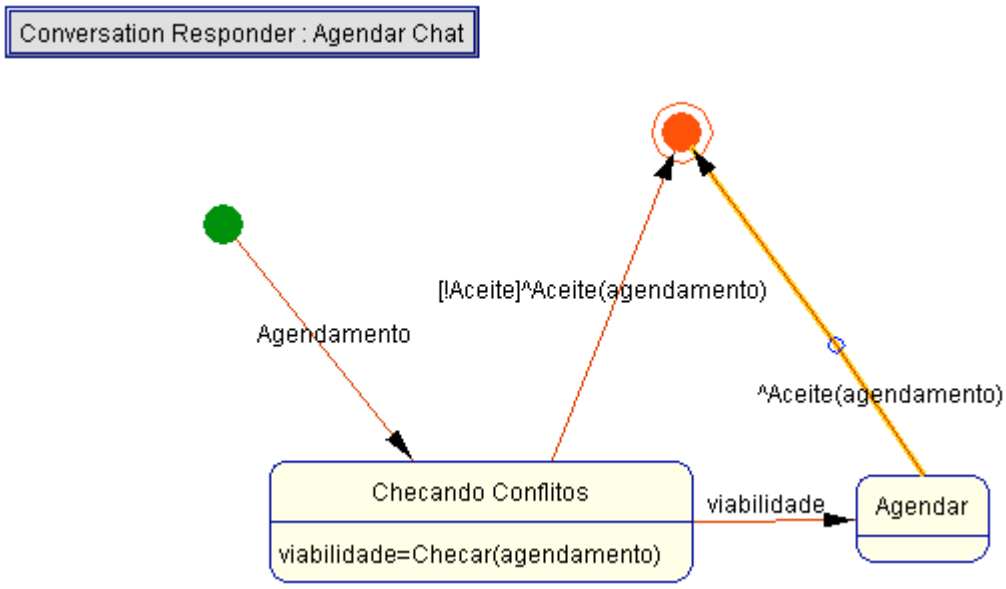

Figura 53 - Resposta do Diálogo - Agendar Chat

Na figura 53 é demonstrada a resposta a solicitação do agendamento, sendo verificados possíveis conflitos com a data solicitada. Caso hajam conflitos, o agendamento não é processado.

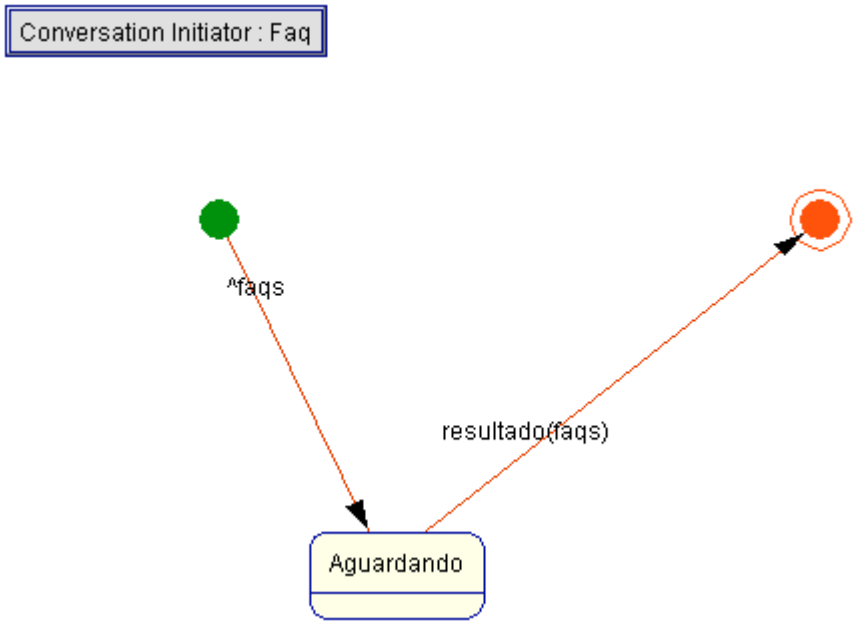

Figura 54 - Inicialização do Diálogo – FAQ

As questões freqüentemente perguntadas são processadas no decorrer da conversação entre o agente aluno e o agente professor. O agente aluno procede com a requisição conforme demonstrado na figura 54.

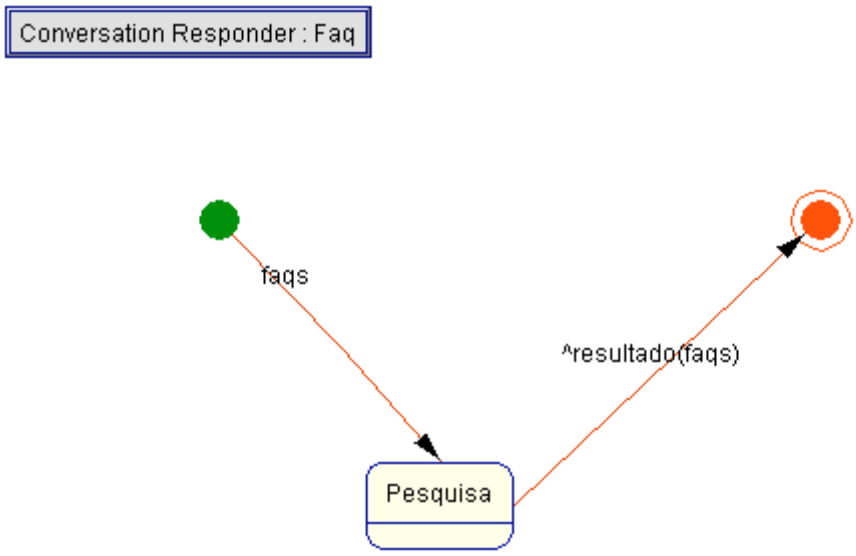

Figura 55 - Resposta do Diálogo – FAQ

O FAQ é enviado ao agente processador, onde uma pesquisa com os termos da pergunta são pesquisados em um banco de dados, conforme exibido na figura 55.

7.2.7 Painel de Desenvolvimento

O painel de desenvolvimento é o primeiro passo do desenvolvimento de sistemas reais. Até o momento, somente foram especificadas as funcionalidades abstratas desejadas no sistema. Nesta seção serão criadas as instâncias de agentes a partir dos modelos e seus diálogos, sendo então conectadas em sistemas. É aqui que o desenvolvimento propriamente dito começará. O Painel de Desenvolvimento com alguns agentes é apresentado na figura 56.

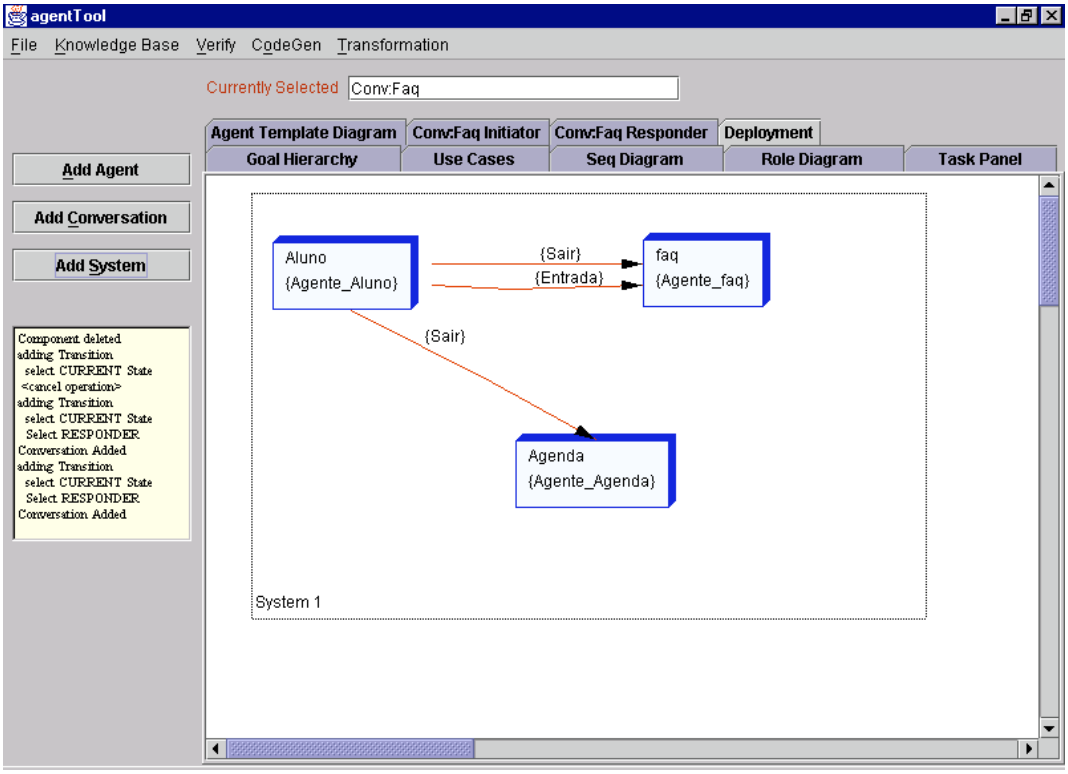

Figura 56 - Painel de Desenvolvimento do Sistema

A ferramenta agentTool possibilita a criação de código em java para *agentMom*, uma infra-estrutura de comunicação simples *socket-based*, e *Carolina*, um framework MultiAgente em desenvolvimento na Universidade de Connecticut.

Esta Dissertação não irá prosseguir até o desenvolvimento completo do agente porque isto não está dentro do escopo pretendido pelo trabalho.

8 CONCLUSÕES E PERSPECTIVAS FUTURAS

8.1 Conclusões

Este trabalho procura contribuir como um guia para a utilização do Framework agentTool e apresenta o estudo de uma Metodologia Orientada a Agentes, tomando como base a análise (sem a implementação) de um sistema de atendimento professor–aluno, detalhando os passos da metodologia MaSE necessários para a especificação do sistema proposto.

A modelagem das classes de agentes do sistema de atendimento de alunos foi efetuada sem dificuldades, pois o framework agenttool auxiliou na identificação dos agentes e na construção do diálogo entre agentes.

A principal técnica utlizada neste trabalho foi a modelagem de sistemas multi-agente usando a Metodologia MaSE, procurando demonstrar que sistemas baseados em agente onde existe uma rede complexa de protocolos de coordenação e onde a computação é baseada em processos concorrentes, exige novos modelos de abstração e ferramentas de conceitualização e implementação.

MaSE demonstrou ser uma metodologia independente da arquitetura e da linguagem de programação, tendo como objetivo principal direcionar o analista em todos os passos do ciclo de vida com ênfase a análise e projeto de sistemas, fornecendo suporte a criação automática do código para agentes através de suas ferramentas. O projetista pode ir facilmente da especificação do sistema até a implementação do mesmo.

Demonstrou-se que o Framework Agenttool facilita a composição dos diagramas de análise da metodologia MaSE, indicando falhas, identificando classes, direcionando a construção de comunicação entre classes, a criação de agentes e a implementação da estrutura interna dos agentes, semi-automatizando a criação de sistemas multiagentes que correspondam as exigências especificadas.

O Framework agentTool cumpre satisfatoriamente com a proposta de uma ferramenta que suporte e reforce a metodologia MaSE, em um ambiente acionado através de menus e diagramas manipuláveis graficamente em janelas.

Cabe ressaltar porém, que o código gerado pelo agentTool não é específico, contendo muita informação desnecessária, o que pode dificultar a codificação. A

construção de Diálogos entre as classes de agentes não emprega protocolos de comunicação padronizados .e amplamente difundidos.

Finalmente, conclue-se que apesar da metodologia MaSE ainda estar em desenvolvimento e os modelos para a análise orientada a agentes estarem em discussão, a tarefa de desenvolver sistemas orientados a agentes usando uma metodologia tambérm orientada a agentes cria novas camadas de abstração que . permitem uma visão global do projeto, diminuindo a complexidade inerente a este tipo de sistema.

9 Perspectivas Futuras

Questionamentos :

- O AgentTool Implementa protocolos de comunicação da própria ferramenta. Poderiam ser implementados protocolos de comunicação padronizados no agentTool.

- A metodologia MaSE criou um padrão próprio para a notação de seus diagramas. A ferramenta deveria buscar uma padronização como a notação ultilizada em UML.

- A ferramenta deveria dar mais suporte para a validação e teste da especificação do que para a geração de código.

Para trabalhos futuros recomenda-se :

- Implementar as classes de agentes desenvolvidas neste trabalho em um ambiente colaborativo para a comparação com outros ambientes que não usam agentes de interface.

- Ampliar a utilização de agentes para distribuir e facilitar a utilização da interface.

- Utilizar outro framework de agentes, visando a comparação entre as ferramentas.

- Sugerir a implementação de uma descrição mais detalhada da fase de projeto da metodologia MaSE, através de uma linguagem de notação.padronizada.

10 FONTES BIBLIOGRÁFICAS

ALBAYRAK, S., WIECZCOREK, D., JIAC - A Toolkit for Telecommunication Applications, Proceedings of the Intelligent Agents for Telecommunication Applications Workshop, Stockholm, Sweden, 1999.

BITTENCOURT, G. Inteligência Artificial: ferramentas e teorias – Florianópolis: Ed. da UFSC, 1998.

BELLAVISTA, P., CORRADI, A., STEFANELLI, C., A Secure and Open Mobile Agent Programming Environment Proceedings of the Fourth International Symposium on Autonomous Decentralized Systems (ISADS '99), Tokyo, Japan, March 21-23, 1999, pp. 238-245, Conference Proceedings, IEEE Computer Society Press.

BELLIFEMINE, F., POGGI, A., RIMASSA, G. JADE: A FIPA-compliant agent framework. Technical report, CSELT S.p.A, 1999. Part of this report has been also published in the Proceedings of the Conference on Practical Applications of Agents and Multi-Agents, 1999, 97-108.

BÖLÖNI, L. Bond objects - a white paper. Technical Report 002, Department of Computer Sciences, a Purdue University, 1998.

BREHM, D. Speakers discuss past, present and future of AI - MIT Tech Talk, out , 1997. Disponível em <http://web.mit.edu/newsoffice/tt/1997/oct01/ai.html> . acessado em 10 Jul 2002.

CAUDILL, S. , Design More Effective User Interfaces, Lotus Advisor Magazine - Out 2000. Disponível em <http://e-businessstrategiesadvisor.com/Articles.nsf/aid/ CAUDS51> acessado em 8 jan 2002

CHAUHAN., D. JAFMAS: A Java-based agent framework for multiagent systems development and implementation, 1997, Dissertação de Mestrado. ECECS Department, University of Cincinnati.

CHEYER, A., MARTIN, D., The Open Agent Architecture, appears in JAAMS Vol. 4 (1/2), pp. 143-148, March 2001.

COLEMAN D.,et al., Object-Oriented Development: The FUSION Method. Prentice Hall International: Hemel Hempstead, England, 1994.

CRANEFIELD, S.J.S., MOREALE, E., MCKINLAY, B. AND PURVIS, M. K., "Automating the Interoperation of Information Processing Tools", Proceedings of the

32nd Hawaii International Conference on System Sciences (HICSS-32). Maui, Hawaii, IEEE , 1999. CD-ROM.

CYBIS , W. Ergonomia de Interfaces Homem-Computador , Disponível em <http://www.labiutil.inf.ufsc.br/apostila/apostila.htm>, acessado em :maio de 2000.

DELOACH, S, WOOD M., Developing Multiagent Systems with agentTool. Lecture Notes in Artificial Inteligence, Springer-Verlag, Berlim, 2001a.

DELOACH, S., Analysis and Design using MaSE and agentTool, Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001). Miami University, Oxford, Ohio, March 31 - April 1, 2001b.

DELOACH, S. Multiagent Systems Engineering: A Methodology and Language for Designing Agent Systems, Agent-Oriented Information Systems '99 (AOIS'99), Seattle WA, 1 May 1998.

PREECE, J., A Guide to Usability: Human Factors in Computing. Wokingham, UK: Addison-Wesley, 1993

EVOLUTION NETWORK. A Relação Homem-Máquina: A Ligação entre o Homem-Máquina, Interface, Interatividade e Realidade Virtual. Disponível em <http://geocities.yahoo.com.br/interaface>. Acessado em: 20 janeiro 2002.

FLACH, P. MEERSMAN, R. (Editors) Future Directions in Artificial Intelligence, Amsterdam: Elsevier Science Publisher, 1991.

FRANKLIN, S., GRAESSER F., Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. , 1996. Disponivel em <http://www.msci.memphis.edu/~franklin/AgentProg.html> acessado em 20 janeiro 2002

FURLAN,J.D.; Modelagem de Objetos através da UML - the Unified Modeling Language, Makron Books do Brasil Ltda, 1998.

GEVARTER, W. Artificial Intelligence, Expert Systems, *Computer Vision and natural Language Processing*. Noyes Publications, Park Ridge, NJ, 1984

GRAHAM, J., Real-Time Scheduling in Distributed Multi Agent Systems Ph.D. Dissertation, University of Delaware, Janeiro, 2001. Disponível em <http://www.eecis.udel.edu/~decaf/Papers/main.pdf> acessado em 8 de mar. 2002.

GOTTGTROY, M.. Data mining agents. Laboratório de Lógica e Inteligência Computacional, Departamento de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, Natal, RN, 1998.

GUIMARÃES, F. Interfaces Homem-Máquina. Bate Byte, Curitiba, n. 69, out./1997,. Disponível em <http://www.pr.gov.br/celepar/ celepar/batebyte/edicoes/1997/ bb69/interfac.htm> acessado em 20 de abril de 2002.

HAYES-ROTH B., LALANDA P., MORIGNOT P., PFLEGER K, AND BALABANOVIC M., "Plans and Behavior in Intelligent Agents", Technical Report KSL-93-43, Knowledge Systems Laboratory, Computer Science Department, Stanford University (April 1993). Disponível em <ftp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-93-43.ps> acessado em : 20 de abril de 2002.

HERMANS, B., Intelligent Software Agents on the Internet: nan Inventory of currently Offered Functionality in the Information Society & a Prediction of (Near-) Future Developments, 1996. Disponível em <http://www.hermans.org/agents> acessado em 25 de abril de 2002.

HIX, D. Developing user interfaces: ensuring usability trougth product & process, New York: WILEY, 1993.

JAZAYERI, M., LUGMAYR W., Gypsy: A Component-based Mobile Agent System. Accepted at the 8th Euromicro Workshop on Parallel and Distributed Processing (PDP2000) (Rhodos, Greece, January 19 -21, 2000).

JEON, H. PETRIE, C. CUTKOSKY, M. R. " JATLite: A Java Agent Infrastructure with Message Routing,", IEEE Internet Computing Mar/Apr 2000.

LÉVY, Pierre. As Tecnologias da Informação: O futuro do pensamento na era da informática. Rio de Janeiro: Editora 34, 1993.

LIEBERMAN, H. Autonomous Interface Agents. CHI 1997: 67-74

LUCK, Michael, D'INVERNO, Mark. *A Formal Framework for Agency and Autonomy* In: ICMAS-9*5,* 1995.

MAES, P. Agents that Reduce Work and Information Overload. Communications of the ACM, Vol. 37, No. 7, Julho 1994.

MARTIN, José López. O Futuro dos controladores programáveis, interfaces homem máquina e filosofias de controle. H&S Automação, 1999. Disponível em <http://www.quasar.com.br/isaba/art_ind.htm> acessado em 20 de outubro de 2001.

MILLER, M. J. Living History: Retracing the Evolution of the PC and PC Magazine (12 de março de 2002). Disponível em <http://www.pcmag.com/article2/ 0,4149,26831,00 .asp> acessado em 08 de novembro de 2002.

MINAR, N. ET ALL., Hive: Distributed Agents for Networking Things., Proceedings of ASA/MA'99, the First International Symposium on Agent Systems and Applications and Third International Symposium on Mobile Agents. 1999.

MYERS, Brad A. Why are Human-Computer Interfaces Difficult to Design and Implement ?, Carnegie Mellon University School of Computer Science Technical Report CMU-CS-93-183, Julho 1993.

NWANA, H. et all, ZEUS: A Tool-Kit for Building Distributed Multi-Agent Systems, In Applied Artifical Intelligence Journal, Vol 13 (1), 1999, p129-186.

NODINE, M., AND CHANDRASEKHARA, D. 1999. Agent communication languages for information-centric agent communities. Disponível on-line em <http://citeseer.nj.nec.com/article/nodine99 agent.html> acessado em outubro de 2001.

PARKER, M. 1998-.Ascape. The Brookings Institution. Disponível em <http://www.brook.edu /es/dynamics/models/ascape> acessado em ouubto de 2001.

POSLAD S., BUCKLE P., HADINGHAM R., The FIPA-OS Agent Platform: Open Source for Open Standards, in: Proc. of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents, UK, 355-368, 2000.

REIS, Christian. Product Usability: Survival Techniques. Disponível em <http://www.acm.org> acesado em novembro de 2001.

RICH, E. Inteligência Artificial. São Paulo: McGraw-Hill, 1988.

SÁNCHEZ, A. 1997. A Taxonomy of Agents. Laboratory of Interactive and Cooperative Technologies, Department of Computer Systems Engineering, Universidad de las Américas - Puebla. Cholula, Puebla, México.

SIGHN, M., and HUHNS M., Declarative Representations of Multiagent Systems, IEE Transactions on Knowledge and Data Engineering, Vol. 4. N. 5, Outubro 1993.

SPIVEY, J. M., Understanding Z: A Specification Language and its Formal Semantics, Cambridge University Press , 1988.

THIRY, M. Uma Arquitetura Baseada em Agentes para Suporte ao Ensino à Distância. Florianópolis: Programa de Pós-Graduação em Engenharia de Produção, 1999. Tese (Doutorado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção, UFSC, Florianópolis.

TORRES, G. Hardware curso completo.Rio de Janeiro: Axcel Books do Brasil Editora Vol. 2. 1998.

TVEIT A. A Survey of Agent-Oriented Software Engineering. Norwegian University of Science and Technology . Maio 2001.

WEXELBLAT, A., and Maes, P., "Issues for Software Agent UI." Submitted to Interactions, ACM Press. Copyright 1997.

WINSTON, P. Rethinking Artificial Intelligence, MIT AI Lab, 1997.

WOOD, M., DELOACH, S., An Overview of the Multiagent Systems Engineering Methodology, In P. Ciancarini and M. Wooldridge, editors, Agent-Oriented Software Engineering - First International Workshop (AOSE), Limerick, Ireland, June 10, 2000. Lecture Notes in Computer Science. Vol. 1957, Springer Verlag, Berlin, 2001.

WOOLDRIDGE M., CIANCARINI P., “Agent-Oriented Software Engineering: The State of the Art,” To appear in the *Handbook of Software Engineering and Knowledge Engineerin*g, World Scientific Publishing Co., 2001.

WOOLDRIDGE, M., JENNINGS, N. Agent Theories, Architectures, and Languages: A Survey , 1994. Disponível em <http://www.ecs.soton.ac.uk/~nrj/download-files/ECAI94-WS.ps> acessado em outubro de 2001.

WOOLDRIDGE, M., JENNINGS, N. Intelligent Agents: Theory and Practice, 1995. Disponível em <http://www.ecs.soton.ac.uk/~nrj/download-files/KE-REVIEW-95.ps> acessado em outubro de 2001.

WOOLDRIDGE , M.; JENINGS N. R.; KINNY. D. A methodology for Agent-Oriented Analysis and Design. Proceedings of the third annual conference on Autonomous Agent , Seattle, USA.May 1 – 5, 1999.

WOOLDRIDGE M. J., JENNINGS N . R.; KINNY. D. The Gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multi-Agent Systems, 3(3):285-312, Setembro 2000.

ZAMBONELLI, F. , et all, "Agent-Oriented Software Engineering for Internet Applications", in Coordination of Internet Agents: Models, Technologies and

Applications, Springer, 2000, to appear. Disponível em <http://citeseer.nj.r com /zambonelli00agentoriented.html> acessado em janeiro de 2001.