



Universidade Federal de Pernambuco
Graduação em Ciência da Computação
Centro de Informática

Implementação de Padrões de Projeto Orientados a
Agentes Utilizando a Plataforma JADE
Trabalho de Graduação

Aluna: Mariana Pinto Xavier

Orientador: Jaelson Freire Brelaz de Castro

Co-orientadora: Carla Taciana Lima Lourenço Silva



Universidade Federal de Pernambuco

Recife, agosto de 2005

Mariana Pinto Xavier

Implementação de Padrões de Projeto Orientados a
Agentes Utilizando a Plataforma JADE

*Dissertação apresentada à Coordenação
do curso de Ciências da Computação do
Centro de Informática, como parte dos
requisitos para a conclusão da disciplina
de Trabalho de Graduação.*

Orientador: Jaelson Freire Brelaz de Castro
Co-orientadora: Carla Taciana Lima
Lourenço Silva

Recife, agosto de 2005

ASSINATURAS

Este trabalho de graduação é resultado dos esforços da aluna Mariana Pinto Xavier, sob orientação do professor Jaelson Freire Brelaz de Castro e co-orientação da doutoranda Carla Taciana Lima Lourenço Silva, na participação do projeto “Implementação de Padrões de Projeto Orientados a Agentes Utilizando a Plataforma JADE”, conduzido pelo Centro de Informática da Universidade Federal de Pernambuco. Todos abaixo estão de acordo com o conteúdo deste documento e os resultados desse trabalho de graduação.

Mariana Pinto Xavier

Carla Taciana Lima Lourenço Silva

Jaelson Freire Brelaz de Castro

*De tudo, ficaram três coisas:
A certeza de que estamos sempre começando
A certeza de que precisamos continua
A certeza de que seremos interrompidos antes de terminar
Portanto, devemos fazer:
Da interrupção, um novo caminho
Da queda um passo de dança
Do medo, uma escada
Do sonho, uma ponte
Da procura, um encontro*

Fernando Pessoa

AGRADECIMENTOS

Agradeço a minha família pelo apoio e compreensão, em especial a minha filha Marina pela sua presença em minha vida e por sua maneira singular de me fazer compreender a simplicidade do mundo.

Ao meu orientador Prof. Jaelson Freire pela confiança em mim depositada.

À doutoranda Carla Taciana pela presteza e disponibilidade, estando sempre me apoiando em todas as etapas desse trabalho.

Aos meus amigos do CIn pelo incentivo e pela amizade. Em particular agradeço a Bárbara pelas conversas construtivas que tanto contribuíram para esse trabalho e a Daniel pelo carinho e amizade.

Ao controlador de voo, Ciro pela atenção e pelas explicações que tanto me ajudaram no entendimento do sistema de controle de tráfego aéreo.

RESUMO

O desenvolvimento de sistemas multi-agentes é ainda bastante recente. Entretanto, esse tipo de sistema vem demonstrando uma grande capacidade em resolver problemas complexos e distribuídos. Assim, diversos trabalhos têm apresentado conceitualizações, formalizações, protocolos, técnicas e métodos para aplicação desse tipo de abordagem. A proposição desses trabalhos torna-se bastante importante para a consolidação desse novo paradigma.

Dessa maneira, esse trabalho de graduação se dispõe a estudar o desenvolvimento de sistemas multi-agente. Estudaremos a metodologia de desenvolvimento Tropos e a plataforma de desenvolvimento orientado a agentes, JADE. Ao final desse trabalho, apresentaremos um estudo de caso utilizando a metodologia Tropos e a plataforma JADE no desenvolvimento de um sistema de software multi-agentes para uma aplicação no sistema de controle de tráfego aéreo.

Palavras-chave: Engenharia de Software Orientada a Agentes, Sistemas Multi-Agentes, Tropos, JADE.

ABSTRACT

The development of multi-agent systems still is very recent. However, this kind of system has been demonstrating a great capacity in solving complexes and distributed problems. In this way, many works have been presented the contexts, formal ways, protocols, techniques and methods to apply this kind of approach. The proposal of these works becomes very important to the consolidation of this new paradigm.

Thus, this work makes use of the study of the development of multi-agent systems. It will be studied the Tropos development methodology and the agent-oriented development plataform JADE. At the end of this work, it will be presented a case study using the Tropos methodology and the JADE plataform in the development of a multi-agent system to a application in the air traffic control system.

SUMÁRIO

1	Introdução	1
1.1	Motivações.....	2
1.2	Objetivos e Abordagem	2
1.3	Estrutura da Monografia	5
2	Engenharia de Software Orientada a Agentes.....	6
2.1	Introdução	7
2.2	Características dos Agentes	8
2.3	Sistemas Multi-Agentes.....	10
2.4	Áreas de Aplicação.....	12
2.5	Metodologias de Desenvolvimento Orientadas a Agentes	15
2.6	Implementação de Agentes de Software.....	17
2.6.1	Comunicação entre agentes.....	18
2.6.2	Plataformas de Desenvolvimento	21
3	Desenvolvendo SMA com Tropos e JADE	23
3.1	Introdução	24
3.2	A metodologia Tropos	24
3.3	A plataforma JADE	27
3.3.1	Ambiente de execução	27
3.3.2	Biblioteca de classes	27
3.3.3	Pacote de ferramentas de suporte	28
4	Estudo de Caso	33
4.1	Introdução	34
4.2	Sistema de Controle de Tráfego Aéreo.....	34
4.2.1	Componentes do Sistema.....	34
4.2.2	Funcionamento do Sistema	35
4.3	Modelagem com Tropos	37
4.3.1	Requisitos Iniciais	37
4.3.2	Requisitos Finais	39
4.3.3	Projeto Arquitetural	41
4.3.4	Projeto Detalhado	43
4.3.5	Implementação em Jade.....	47

4.4	Considerações Finais	53
5	Conclusão.....	55
	Referências Bibliográficas	56

LISTA DE TABELAS

Tabela 4.1: Modelo de descrição do agente monitor	44
---	----

LISTA DE FIGURAS

Figura 3.1: Remote Monitoring Agent (RMA)	30
Figura 3.2: Directory Facilitador (DF).....	30
Figura 3.3: Dummy Agent	31
Figura 3.4: Sniffer Agent	31
Figura 3.5: Introspector Agent	32
Figura 4.1: Modelagem Organizacional do Sistema de Controle de Tráfego Aéreo.....	38
Figura 4.2: Modelagem dos Requisitos Finais do Sistema	40
Figura 4.3: Modelagem do Projeto Arquitetural do Sistema	42
Figura 4.4: Diagrama de Seqüência.....	45
Figura 4.5: Estrutura do padrão monitor	46
Figura 4.6: Agente Aeroporto em estado Aberto	49
Figura 4.7: Aeroportos de destino do agente aeronave.....	49
Figura 4.8: Troca de mensagens entre os atores.....	50
Figura 4.9: Agente Aeroporto1 em estado Fechado	51
Figura 4.10: Agente Aeroporto2 em estado Aberto.....	51
Figura 4.11: Aeroportos de destino do agente aeronave.....	52
Figura 4.12: Troca de mensagens entre os atores.....	53

1 INTRODUÇÃO

Este capítulo apresenta as principais motivações para realização deste trabalho. Apresentamos aqui, os objetivos e a abordagem utilizada e a estrutura do trabalho.

1.1 MOTIVAÇÕES

O crescimento das aplicações de software tanto em tamanho quanto em complexidade acarretaram em crescentes dificuldades em seu projeto e construção. Dessa maneira, os engenheiros de software foram motivados a estudar esses sistemas obtendo um conhecimento maior a respeito das características de um software complexo.

Nesse contexto, surge o desenvolvimento orientado a agentes como um dos mais promissores paradigmas atuais. Um agente é um sistema computacional capaz de ações flexíveis e autônomas em um ambiente dinâmico, aberto e imprevisível [Jennings03]. A orientação a agente oferece um nível mais alto de abstração na forma de pensar sobre as características e os comportamentos de sistemas de software.

Espera-se que sistemas orientados a agentes sejam mais poderosos, mais flexíveis e mais robustos do que sistemas de software convencionais [Yu01]. Entretanto, atualmente não existem notações diagramáticas, metodologias e ferramentas de projeto para orientação a agentes reconhecidas.

1.2 OBJETIVOS E ABORDAGEM

Primeiramente, temos como objetivo usufruir as facilidades oferecidas pela engenharia de software ao realizarmos a construção de um sistema usando uma metodologia de desenvolvimento. Em particular, vamos usar o recente paradigma de agentes, uma metodologia orientada a agentes e por fim uma plataforma de implementação específica para codificar os agentes que compõem o sistema.

Durante a realização deste trabalho, esperamos obter conhecimento das áreas de Sistemas Multi-Agentes (SMA) e Engenharia de Software no que diz respeito ao uso do paradigma de agentes para construir software.

O foco desse trabalho se encontra na fase de projeto detalhado da metodologia Tropos. Durante essa fase os agentes são categorizados segundo um dos padrões existentes de acordo com sua estrutura.

Os padrões são classificados em duas categorias:

Padrões entre pares (peers), onde os agentes comunicam-se um com o outro diretamente sem a necessidade de um agente negociador. São eles:

- Padrão **Booking**: envolve um cliente e vários provedores de serviços. O cliente emite um pedido para reservar algum recurso de um provedor de serviço. O provedor pode aceitar o pedido, negar ou colocar o cliente em uma lista de espera, até que o recurso se torne disponível.
- Padrão **Subscription**: envolve um agente do tipo páginas amarelas e vários provedores de serviços. Os provedores anunciam seus serviços solicitando ao agente páginas amarelas através de um pedido de *subscribe*. Quando um provedor não necessitar mais do serviço pode solicitar o término do mesmo através de um pedido de *unsubscribed*.
- Padrão **Call-For-Proposals**: envolve um iniciador e vários participantes. O iniciador emite uma chamada para propostas para um serviço a todos os participantes. Os participante respondem à chamada e o iniciador então escolhe um dos participantes para prover o serviço.
- Padrão **Bidding**: envolve um iniciador e vários participantes. O iniciador inicia o processo de licitação e recebe as propostas dos participantes. A cada iteração o iniciador publica a oferta atual.

Padrões com mediadores, onde existe a figura de um agente negociador o qual mediará a comunicação entre os agentes. São eles:

- Padrão **Monitor**: nesse padrão os agentes solicitam inscrição ao agente monitor para receber notificação de mudança de estado de algum assunto de interesse deles próprios. O monitor aceita oferecer o serviço e envia alertas aos agentes inscritos quando necessário.

- Padrão **Broker**: o broker é um agente intermediário entre provedores e clientes. Os clientes acessam o serviço dos provedores enviando um pedido ao broker, o broker retorna ao cliente o resultado do serviço (o cliente não tem acesso ao provedor).
- Padrão **Matchmaker**: o agente matchmaker age de maneira semelhante ao Broker. Os clientes acessam o serviço dos provedores inicialmente enviando uma solicitação ao matchmaker. O Matchmaker, por sua vez envia ao cliente o provedor do serviço escolhido.
- Padrão **Mediador**: um agente mediador é o interventor entre os agentes. O iniciador se comunica com o mediador ao invés de interagir diretamente com os outros agentes. O mediador coordena a comunicação entre o iniciador e os outros agentes.
- Padrão **Embassy**: possibilita a comunicação entre agentes locais e estrangeiros fazendo a comunicação entre os mesmos.
- Padrão **Wrapper**: age como tradutor entre sistemas multi-agentes e outros sistemas garantindo que os protocolos de comunicação são respeitados.

Neste trabalho visamos a implementação de um dos padrões de projeto com mediadores orientados a agentes utilizando a plataforma JADE. Portanto, os principais objetivos deste trabalho são:

- Investigar o paradigma de Engenharia de Software Orientada a Agentes e a proposta da metodologia Tropos;
- Investigar o uso de ferramentas e linguagens de modelagem apropriadas ao desenvolvimento de sistemas multi-agentes;
- Investigar as plataformas de desenvolvimento de agentes JADE;
- Investigar os padrões de projeto orientados a agente e
- Implementar um padrão de projeto orientado a agente na plataforma JADE.

1.3 ESTRUTURA DA MONOGRAFIA

Além desse capítulo introdutório, esse trabalho consiste de mais 4 capítulos, são eles:

Capítulo 2 – Engenharia de Software Orientada a Agentes

Nesse capítulo são mostradas as principais motivações para o uso da abordagem de agentes. Em seguida, são apresentadas algumas definições dos sistemas multi-agentes e suas áreas de aplicação. Por fim, falamos das metodologias de desenvolvimento de orientação a agentes e da implementação dos agentes de software.

Capítulo 3 – Desenvolvendo SMA com Tropos e JADE

Nesse capítulo detalhamos a metodologia Tropos e a plataforma JADE, utilizadas para desenvolver sistemas multi-agentes.

Capítulo 4 – Estudo de Caso

Nesse capítulo é mostrado um estudo de caso desenvolvido de acordo com os estudos realizados no capítulo 3.

Capítulo 5 – Conclusão

Nesse capítulo são mostradas as conclusões obtidas durante o desenvolvimento desse trabalho, assim como possíveis futuros trabalhos.

2 ENGENHARIA DE SOFTWARE ORIENTADA A AGENTES

Esse capítulo aborda a engenharia de software orientada a agentes. Inicialmente daremos as motivações para o uso desse novo paradigma. Adiante daremos uma caracterização de um agente e abordaremos as motivações e importância dos sistemas baseados em agentes para a engenharia de software. Apresentaremos também as principais áreas de aplicação dos softwares orientados a agentes. Ao final mostraremos algumas metodologias de desenvolvimento orientado a agentes e falaremos sobre como se dá a implementação desse tipo de sistema.

2.1 INTRODUÇÃO

A busca pela qualidade motiva o homem desde a antiguidade. Em vários setores essa busca foi responsável pela melhoria de produtos e serviços ao longo do tempo.

Em sistemas de informação não é diferente. Principalmente com a globalização surgiram novas exigências de mercado. Uma alta competitividade e a concorrência internacional tornaram a qualidade uma arma competitiva. Com o passar dos anos a indústria passou a construir softwares cada vez mais complexos e há uma grande preocupação na garantia da conformidade do produto e da satisfação do cliente.

E o que é qualidade em sistemas de informação? Segundo Pressman¹, qualidade de software é garantir a “conformidade a requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software profissionalmente desenvolvido”. Dessa maneira a qualidade de software vai de encontro a diversas dificuldades inerentes a própria peculiaridade do desenvolvimento de software, como a produção específica e não em série ou até mesmo a grande imaturidade que temos hoje na área de software.

Diante disso, vários paradigmas de implementação foram propostos na literatura, a programação estruturada, a programação declarativa, a programação orientada a objetos e a programação baseada em componentes. Entretanto, esses paradigmas não estavam resolvendo os problemas das empresas que produziam softwares cada vez mais complexos e não queriam ter a perda da qualidade dos seus produtos. Neste ínterim, surge o paradigma de orientação a agentes com o objetivo de melhorar o processo de desenvolvimento de software. As técnicas orientadas a agentes representam um novo meio de analisar, projetar e construir sistemas de software complexos. O uso da abordagem orientada a agentes é baseado nos seguintes argumentos [Jennings03a]:

¹ Roger S. Pressman é uma autoridade reconhecida internacionalmente em tecnologias de melhoras de processo de engenharia de software.

- i. O aparato conceitual de sistemas orientados a agentes é adequado para construir soluções de software para sistemas complexos e
- ii. As abordagens orientadas a agentes representam um verdadeiro avanço sobre o atual estado da arte na engenharia de sistemas complexos.

De fato, técnicas orientadas a agentes são adequadas para desenvolver sistemas complexos de software porque:

- As decomposições orientadas a agente são uma maneira efetiva de repartir o espaço do problema de um sistema complexo;
- As abstrações chave presentes no modo de pensar orientado a agentes são um meio natural de modelar sistemas complexos;
- A filosofia orientada a agente para identificar e gerenciar relacionamentos organizacionais é apropriada para lidar com as dependências e interações que existem em um sistema complexo.

Assim, devemos inicialmente caracterizar o que vem a ser um agente, a fim de obter um melhor entendimento desse paradigma.

2.2 CARACTERÍSTICAS DOS AGENTES

Existem diversos conceitos do que é um agente, entretanto o que melhor se adequou ao que nós propomos tratar nessa monografia é [Wooldridge01a]:

“Um agente é um sistema computacional encapsulado que está situado em algum ambiente e é capaz de ação flexível autônoma neste ambiente, a fim de alcançar seus objetivos de projeto.”

Dessa maneira, temos o agente como um sistema capaz de não somente agir com o ambiente, mas também com outros agentes, os quais, assim como ele, objetivam alcançar suas metas. Um agente pode ainda ter algumas características que o tornem mais ou menos útil para uma determinada tarefa, tais como:

- Autonomia: capacidade de agir sem intervenção externa direta. O agente tem algum grau de controle sobre seu estado interno e suas ações são baseadas em suas próprias experiências.
- Interatividade: capacidade de se comunicar com o ambiente e com outros agentes.
- Adaptabilidade: capacidade de responder a outros agentes e/ou a seu ambiente em alguma proporção. As formas mais avançadas de adaptação permitem que um agente modifique seu comportamento baseado em sua experiência.
- Sociabilidade: capacidade de interagir de forma amistosa ou prazerosa.
- Mobilidade: capacidade de se transportar de um ambiente para outro.
- Representatividade: capacidade de agir em benefício de alguém ou algo, isto é, age em interesse, como um representante, ou em benefício de alguma entidade.
- Pró-atividade: capacidade de orientar-se à meta, ter propósito. O agente não reage simplesmente ao ambiente.
- Inteligência: capacidade de possuir o estado formalizado por conhecimento (isto é, crenças, metas, planos, afirmações). A interação com os outros agentes é através de linguagem simbólica.
- Racionalidade: capacidade de escolher uma ação baseando-se em metas internas e no conhecimento de que uma ação particular o deixará mais próximo de suas metas.
- Imprevisibilidade: capacidade de agir de formas não completamente previsíveis, mesmo se todas as condições iniciais são conhecidas. Capacidade de comportamento não-determinístico.
- Continuidade temporal: capacidade de ser um processo que executa continuamente.
- Caráter: capacidade de possuir personalidade e estado emocional críveis.
- Transparência e responsabilidade: capacidade de ser transparente quando necessário e ainda prover um registro de atividades sob demanda.

- Coordenação: capacidade de executar alguma atividade em um ambiente compartilhado por outros agentes. As atividades são frequentemente coordenadas através de planos, fluxos de trabalho ou algum outro mecanismo de gerência de processo.
- Cooperação: capacidade de cooperar com outros agentes a fim de atingir um objetivo comum; são agentes não adversários que obtêm sucesso ou falham juntos. (Colaboração é outro termo usado como sinônimo de cooperação)
- Competição: capacidade de coordenar com outros agentes exceto no caso em que o sucesso de um agente implica na falha de outros (oposto do cooperativo).
- Robustez: capacidade de lidar com erros e dados incompletos de forma robusta.
- Confiabilidade: capacidade de ser confiável.

É consenso dentro da comunidade orientada a agentes que um agente não é útil se não tiver pelo menos as três primeiras características acima, ou seja, autonomia, interatividade e adaptabilidade.

2.3 SISTEMAS MULTI-AGENTES

Os agentes podem ser úteis como entidades isoladas às quais são delegadas tarefas particulares, ou podem existir em ambientes contendo outros agentes. Nesse último caso, os agentes têm que cooperar, negociar e coordenar ações. Existem sistemas nos quais um único agente é apropriado (como em sistemas especialistas) ou, na maioria das vezes, podemos precisar de mais de um agente, teremos assim um sistema multi-agente.

Um sistema multi-agente pode ser definido como uma coleção de agentes autônomos que se interrelacionam de acordo com uma organização, interagindo com o objetivo de resolver tarefas, o que individualmente não seria possível. Segundo Jennings [Jennings96b], SMA também se refere à subárea da Inteligência Artificial Distribuída (IAD) que investiga o comportamento de um conjunto de agentes autônomos

objetivando a solução de um problema que está além das capacidades de um único agente.

Podemos distinguir duas principais classes de sistemas multi-agentes [Zambonelli00]:

- **Sistemas de resolução distribuída de problemas**, nos quais os agentes envolvidos são explicitamente projetados para, de maneira cooperativa, atingir um dado objetivo, considerando-se que todos eles são conhecidos à priori e supondo que todos são benevolentes, existindo desta forma confiança mútua em relação as suas interações e
- **Sistemas abertos**, nos quais os agentes não são necessariamente projetados para atingirem um objetivo comum, podendo ingressar e sair do sistema de maneira dinâmica. Nesse caso, a chegada dinâmica de agentes desconhecidos precisa ser levada em consideração, bem como a possível existência de comportamento não benevolente no curso das interações.

Os sistemas multi-agentes encontram-se inseridos nessa segunda classificação. Nesse tipo de sistema, investigar o comportamento de um conjunto de agentes autônomos, possivelmente pré-existentes, que interagem objetivando a resolução de um problema que está além das capacidades de um único indivíduo [O'hare96]. Desta forma, o comportamento global do sistema deriva da interação entre os agentes que fazem parte do sistema [Zambonelli00].

Temos então que as principais características dos sistemas multi-agentes são [Jennings00]:

- Cada agente tem informações ou capacidades incompletas para solucionar um dado problema. Dessa forma, cada agente tem um ponto de vista limitado;
- Não há controle global do sistema;
- O dado é descentralizado e
- A computação é assíncrona.

Dessa maneira podemos justificar a aplicação da tecnologia de agentes na concepção de sistemas quando o problema envolve [Jennings96a]:

- Distribuição intrínseca dos dados, capacidade de resolução de problemas e responsabilidades;
- Necessidade de manter a autonomia das partes, sem a perda da estrutura organizacional;
- Complexidade nas interações, incluindo negociação, compartilhamento de informação e coordenação;
- Impossibilidade de descrição do problema a priori, devido à possibilidade de perturbações em tempo real no ambiente e processos de natureza dinâmica.

Assim, destacamos os principais fatores que tornam os sistemas multi-agentes tão importantes na engenharia de software [Odell99]. Sistema multi-agentes:

- Provêm uma maneira de pensar sobre o fluxo de controle em um sistema altamente distribuído;
- Oferecem um mecanismo que permite um comportamento emergente ao invés de uma arquitetura estática;
- Codificam melhores práticas de como organizar entidades colaborativas concorrentes.

Adiante mostraremos algumas áreas de aplicação dos sistemas multi-agentes.

2.4 ÁREAS DE APLICAÇÃO

A tecnologia de agente está rapidamente se desenvolvendo e sendo aplicada em diversas áreas. Nessa seção nós identificamos as principais áreas onde as abordagens baseadas em agente estão sendo usadas e provemos indicadores para alguns exemplos de sistemas nestas áreas [Agentlink03].

Aplicações industriais

As aplicações industriais de tecnologia de agente estão entre as primeiras que foram desenvolvidas, e hoje os agentes estão sendo aplicados em uma larga escala de sistemas industriais. Abaixo exemplificamos algumas possíveis abordagens:

- **Fabricação:** Sistemas nessa área incluem projeto de configuração de produtos de fabricação, projeto colaborativo, cronograma e controle de operações de fabricação, controle de fabricação de um robô e determinação de seqüências de produção para uma fábrica [Jennings03b].
- **Controle de processo:** Sistemas para monitoração e diagnóstico de faltas em plantas de força nuclear, controle de espaço-nave, controle climático, controle de processamento de bobina de aço, gerência de transporte elétrico e controle de acelerador de partícula.
- **Telecomunicações:** Sistemas baseados em agentes podem ser construídos e incluem controle de rede, transmissão e chaveamento, gerência de serviços e gerência de rede e para prover serviços melhores, mais rápidos e mais confiáveis.
- **Controle de tráfego aéreo:** Agentes são usados para representar tanto os equipamentos de aviação quanto vários sistemas de controle de tráfego aéreo em operação.
- **Sistemas de transporte:** O domínio de gerência de tráfego e transporte é adequado para uma abordagem baseada em agente por causa da sua natureza geograficamente distribuída.

Aplicações comerciais

Enquanto as aplicações industriais tendem a ser altamente complexas, indicando sistemas que operam em áreas de nicho comparativamente pequeno, as aplicações comerciais tendem a ser orientadas muito mais para o mercado.

- **Gerência de informação:** Mecanismos de busca são realizados por agentes, que agem autonomamente para efetuar buscas na *web* em benefício de algum usuário. De fato, esta é provavelmente uma das áreas

mais ativas para aplicações de agente. Outras áreas incluem um assistente pessoal que aprende sobre os interesses do usuário e com base neles compila um jornal pessoal, um agente assistente para automatizar várias tarefas de usuário em um *desktop* de computador, um agente de procura de *home page*, um assistente de navegação na *web* e um agente especialista de localização.

- Comércio eletrônico: Algumas tomadas de decisões comerciais podem ser colocadas nas mãos de agentes. Um aumento da quantidade de comércio está sendo empreendido por agentes. Entre estas aplicações comerciais encontra-se um agente que descobre os *Compact Discs* mais baratos, um assistente pessoal de compra capaz de buscar lojas *on-line* para disponibilizar o produto e informação sobre o preço, um mercado virtual para comércio eletrônico e vários catálogos interativos baseados em agentes.
- Gerência de processo de negócio: Organizações têm procurado desenvolver vários sistemas de Tecnologia da Informação para dar assistência a vários aspectos da gerência de processos de negócio. Outras aplicações nessa área incluem um sistema de gerência de trabalho heterogêneo e um sistema baseado em agentes móveis para gerência de fluxo de trabalho interorganizacional.

Aplicações de entretenimento

Agentes têm um papel bastante claro em jogos de computador, cinema interativo e aplicações de realidade virtual, pois tais sistemas tendem a ser cheios de caracteres animados semi-autônomos, os quais podem naturalmente ser implementados como agentes.

Aplicações médicas

Existem vários exemplos de problemas cobertos por aplicações de sistemas multi-agentes na área médica, abaixo estão listados alguns deles:

- Resolução de problemas espacialmente distribuídos em diferentes localizações. Conhecido como o problema da programação do paciente, consiste em programar diferentes tarefas para um paciente hospitalizado

(como, por exemplo, um exame de sangue e uma radiografia). Não se trata de um problema trivial coordenar essas tarefas quando as mesmas podem ser executadas em locais diferentes (o que demanda um tempo para deslocamento do paciente) além de existirem restrições médicas para essas tarefas (como, por exemplo, diferenças de horários entre um exame e outro) [Decker98].

- Resolução de problemas os quais envolvam esforços de várias pessoas com atividades e funções diferentes. Em um hospital é bastante comum que o paciente seja atendido por diversos profissionais (como enfermeiras, médicos, atendentes, auxiliares etc) e essas pessoas necessitam de coordenar seus esforços a fim de prover a melhor estada possível ao paciente [Alamillo03] [Godo2003].
- Problemas relacionados a transplantes de órgãos humanos. Os órgãos são retirados num hospital e é necessário que em pouco tempo se localize um receptor compatível em algum outro hospital para que o órgão possa ser transplantado rapidamente [Aldea01] [Vázquez03].
- Informação médica através da internet sendo colhida de diversas fontes e disponibilizada para profissionais da área e pessoas interessadas. Proporcionando assim a troca de informações e conseqüentemente provocando um aumento do conhecimento na área [Baujard98] [Kostkova02] [Moreno2002].

2.5 METODOLOGIAS DE DESENVOLVIMENTO ORIENTADAS A AGENTES

Sob o ponto de vista da engenharia de software, a construção de software de alta qualidade de maneira produtiva é viabilizada por um conjunto de métodos, ferramentas e procedimentos, sendo que o caminho para a evolução no desenvolvimento do produto passa por uma combinação de métodos abrangentes para todas as etapas de desenvolvimento, melhores ferramentas para automatizar estes métodos, blocos de construção mais poderosos para sua implementação, melhores técnicas para a garantia

da qualidade e uma filosofia de coordenação predominante, controle e administração [Pressman95].

Dessa maneira, assim como o paradigma orientado a objetos, o paradigma orientado a agentes necessita de um conjunto de técnicas e métodos que possam garantir a qualidade do processo de produção do software.

Nos últimos anos, foram propostas várias metodologias para desenvolvimento de sistemas multi-agentes. Embora, essas metodologias tenham terminologias e abstrações diferentes, existe um consenso entre as mesmas de que a construção de softwares tradicionais é extremamente diferente da construção de sistemas multi-agentes.

A seguir faremos uma breve descrição de algumas dessas metodologias.

GAIA [Wooldridge00]

Baseada na visão de que um sistema multi-agente se comporta como uma organização computacional que consiste em vários papéis interagindo. Ela permite que um analista vá sistematicamente do estabelecimento de requisitos até um projeto que seja suficientemente detalhado a ponto de ser implementado diretamente.

AUML [Odell01]

Agent UML² (AUML) é uma metodologia de análise e projeto que estende a *Unified Modeling Language* (UML) para representar agentes. Ela sintetiza uma preocupação crescente das metodologias de software baseado em agentes com o aumento da aceitação da UML para o desenvolvimento de software orientado a agentes.

MESSAGE/UML [Caire01]

A metodologia *Methodology for Engineering Systems of Software Agents* (MESSAGE) cobre a análise e o projeto de sistemas multi-agentes usando conceitos bem definidos e uma notação baseada em UML.

² A linguagem UML é uma linguagem de modelagem utilizada para especificar, visualizar, construir e documentar os artefatos de um sistema.

MaSE [DeLoach01]

A metodologia maSE (*Multiagent Systems Engineering*) suporta grande parte do ciclo de vida do desenvolvimento de um sistema multi-agente. MaSE começa da representação textual do sistema e segue de forma estruturada até a sua implementação combinando vários modelos preexistentes em uma única metodologia estruturada.

Tropos [Castro02]

Baseada nos conceitos usados durante a análise de requisitos iniciais e tenta modelar e implementar sistemas multi-agentes de um ponto de vista orientado a requisitos, visando reduzir tanto quanto possível o mau casamento da impedância entre o sistema e seu ambiente.

Essa metodologia será descrita com mais detalhes no capítulo 3.

2.6 IMPLEMENTAÇÃO DE AGENTES DE SOFTWARE

Devido às características inerentes aos sistemas multi-agentes houve uma necessidade em se buscar alternativas que possibilitassem sua implementação. Dessa maneira, começaram a surgir diversos *frameworks*³ que possibilitavam o desenvolvimento de sistemas multi-agentes. Entretanto não havia ainda uma padronização para a implementação de sistemas desse tipo.

Em 1996, surge então a *Foundation for Intelligent Physical Agents* (FIPA) uma organização suíça com o objetivo de produzir especificações de padrões de software para sistemas com agentes heterogêneos. Desde sua fundação, a FIPA teve um papel crucial no desenvolvimento de padrões de agentes e promoveu várias iniciativas e eventos que contribuíram com o desenvolvimento e captação de tecnologia de agente. Em 1997, a FIPA apresentou seu primeiro conjunto de especificações e em 2002 essas especificações foram definidas como um padrão.

³ Estrutura de suporte definida na qual um outro projeto de software pode ser organizado e desenvolvido. Tipicamente, um *framework* pode incluir programas de apoio, bibliotecas de código, linguagens de script e outros softwares.

Em 8 de junho de 2005, a FIPA passou a ser aceita oficialmente pelo *Institute of Electrical and Eletronics Engineers*⁴ (IEEE) como um novo comitê, o *Foudation for Intelligent Physical Agents Standards Committee* (FIPASC). O FIPASC é o décimo primeiro comitê a operar juntamente com o *IEEE Standards Associations* e o *IEEE Computer Society's Standards Activity Board*. Esse comitê passou a ser uma referência para organização de sistemas multi-agentes.

2.6.1 COMUNICAÇÃO ENTRE AGENTES

Dentre as padronizações propostas pela FIPA foi definido um padrão para prover a comunicação entre os agentes, o *FIPA Agent Communication Language* (FIPA-ACL). Uma mensagem FIPA-ACL é formada por um conjunto de parâmetros. Esses parâmetros são divididos em cinco categorias: tipos de atos de comunicação, participantes da comunicação, conteúdo da mensagem, descrição do conteúdo e controle da conversa.

Tipo de atos de comunicação

A semântica do FIPA-ACL é baseada na teoria dos Atos da Fala⁵, onde as mensagens estão associadas a primitivas de comunicação (performativas) que representam a vontade do agente sobre a informação contida na mensagem. Uma mensagem FIPA-ACL pode conter um ou mais parâmetros. O único parâmetro que deve estar sempre presente é a performativa. Ao todo a FIPA-ACL possui um conjunto de vinte e duas performativas, são elas:

1. **Accept Proposal:** permite que um agente declare que aceita uma proposta feita por outro agente.
2. **Agree:** permite que um agente declare que concorda com o pedido feito por outro agente.

⁴ O IEEE é uma associação técnica sem fins lucrativos com mais de 365.000 sócios.

⁵ A teoria dos Atos da Fala iniciou-se com o trabalho do filósofo alemão John Austin e foi posteriormente estendida por John Searly. Essa teoria pode ser utilizada como uma forma de organizar a conversação tratando a comunicação como uma ação. Dessa maneira a teoria assume que os atos da fala são realizados pelos agentes da mesma forma que outras ações, auxiliando a realização de suas ações.

3. **Cancel:** utilizada por um agente para indicar que não deseja mais que uma ação solicitada seja realizada.
4. **Call for Proposal** ou **CFP:** permite a um agente solicitar um processo de negociação.
5. **Confirm:** permite que o agente que enviou uma mensagem confirme a veracidade do conteúdo para o receptor.
6. **Disconfirm:** similar ao **confirm**, indica ao receptor que o agente que enviou a mensagem não tem certeza do seu conteúdo.
7. **Failure:** informa que o agente pretendia executar uma ação mas, que por alguma razão isso não foi possível.
8. **Inform:** é o mecanismo básico para a comunicação de informações. O agente emissor deseja que o receptor acredite em seu conteúdo.
9. **Inform If:** o agente emissor solicita que o receptor que informe se uma proposição é verdadeira ou não.
10. **Inform Ref:** o agente emissor informa ao receptor algum objeto que corresponda a uma descrição, por exemplo, um nome.
11. **Not Understood:** permite que o agente informe que não entendeu o porquê de executar determinada tarefa.
12. **Propagate:** o agente que emite a mensagem pede que o receptor reenvie o conteúdo da mensagem a outros agentes.
13. **Propose:** permite que um agente faça uma proposta para outro agente, por exemplo, em resposta a uma mensagem CFP anteriormente enviada.
14. **Proxy:** o emissor pede ao receptor que selecione determinados agentes a fim de enviar a eles uma mensagem do emissor.
15. **Query If:** o emissor questiona ao receptor se uma determinada proposição é verdadeira.
16. **Query Ref:** o emissor solicita a outro agente uma nova alusão a um objeto referenciado pelo receptor.
17. **Refuse:** utilizada por um agente para informar a outro que não executará determinada ação.
18. **Reject Proposal:** utilizada por um agente para indicar a outro que não aceita uma proposta feita durante uma negociação.
19. **Request:** permite que um agente peça a outro que execute uma ação.

20. **Request When:** permite que um agente peça a outro que execute uma ação quando uma determinada proposição for verdadeira.
21. **Request Whenever:** permite que um agente peça a outro que execute uma ação quando uma determinada proposição se tornar verdadeira novamente.
22. **Subscribe:** permite que um agente solicite a outro que o informe o valor de uma determinada referência e notifique novamente caso esse valor se altere.

Participantes da comunicação

Existem ao todo três parâmetros nessa categoria, o *sender*, o *receiver* e o *reply to*.

- O *sender* indica quem está enviando a mensagem.
- O *receiver* indica para quem é a mensagem.
- O *reply to* indica que o nome indicado nesse campo também está recebendo a mensagem enviada pelo *sender*.

Conteúdo da mensagem

Essa categoria possui apenas um parâmetro, o conteúdo.

- O conteúdo de qualquer mensagem ACL pode ser interpretado pelo recebedor da mensagem.

Descrição do conteúdo

Nessa categoria estão inclusos os parâmetros: linguagem, codificação e ontologia.

- A linguagem indica o idioma no qual o parâmetro é expresso.
- A codificação mostra a codificação específica do conteúdo do idioma.
- A ontologia é o vocabulário de palavras e seus respectivos significados.

Controle da conversa

Nessa categoria estão os parâmetros: protocolo, *conversation-id*, *reply-with*, *in-reply-to* e *reply-by*.

- O protocolo é definido pela FIPA para que um agente requisite uma tarefa a outro agente, tais como o tipo e a seqüência das mensagens trocadas. Em cada protocolo de interação são definidos os papéis que podem ser desempenhados pelos agentes, os tipos de mensagens que podem ser enviadas e recebidas por cada papel, além da seqüência com a qual essas mensagens são trocadas.
- Um *conversation-id* introduz um identificador que é utilizado para a identificação da sucessão de mensagens trocadas pelos agentes.
- Um *reply-if* introduz uma expressão que deve ser utilizada ao se responder uma mensagem.
- Um *in-reply-to* denota uma expressão que referencia uma expressão tratada anteriormente.
- Um *reply-by* indica um tempo ou uma data na qual se gostaria de receber uma resposta.

2.6.2 PLATAFORMAS DE DESENVOLVIMENTO

Existem alguns ambientes de desenvolvimento os quais seguem as especificações da FIPA, dentre os quais podemos citar: JACK, FIPA-OS e JADE.

JACK [JACK05]

Jack *Intelligent Agents* é um ambiente de desenvolvimento totalmente integrado com Java. Inclui todos os componentes do ambiente de desenvolvimento Java, assim como oferece extensões específicas de implementação e comportamento dos agentes. Os agentes utilizados em JACK têm seus comportamentos modelados de acordo com o modelo BDI.

FIPA-OS [FIPA-OS05]

FIPA-OS é totalmente implementado em Java. O OS significa *Open Source*. Essa foi a primeira plataforma de desenvolvimento multi-agente com código aberto construída.

JADE [JADE04]

Java Agent Development Framework (JADE) é um ambiente para desenvolvimento de aplicações baseadas em agentes totalmente implementado em Java. JADE foi desenvolvido e continua sendo atualizado pela Universidade de Parma na Itália. JADE não apenas facilita o desenvolvimento como também é utilizado para o gerenciamento de agentes. Nele está incluso dois produtos um compilador (FIPA - *Compliant Agent Platform*) e um pacote de desenvolvimento.

No capítulo 3 esse ambiente será revisto com maior teor de detalhes.

3 DESENVOLVENDO SMA COM TROPOS E JADE

Este capítulo aborda o desenvolvimento de Sistemas multi-agentes. Inicialmente daremos uma descrição de como funciona a metodologia Tropos abordando suas fases de desenvolvimento. Adiante iremos expor as características da plataforma de desenvolvimento JADE.

3.1 INTRODUÇÃO

A engenharia de software orientada a agentes nos apresenta um novo nível de abstração para construir sistemas de software. Essa nova abstração permite que o engenheiro de software projete um sistema em termos de agentes que interagem entre si. No entanto, hoje em dia os projetistas de software ainda não podem explorar todos os benefícios oferecidos pelo paradigma de agentes devido à ausência de notações diagramáticas, metodologias e ferramentas de projeto reconhecidas para o desenvolvimento orientado a agentes [Bergenti00].

Para gerenciar com sucesso a complexidade associada ao desenvolvimento, manutenção e distribuição de sistemas multi-agentes, um conjunto de técnicas e ferramentas de engenharia de software é solicitado ao longo do ciclo de vida do software. Nesse capítulo iremos mostrar a metodologia Tropos e a plataforma de desenvolvimento JADE.

3.2 A METODOLOGIA TROPOS

A metodologia Tropos é baseada nos conceitos usados para modelar requisitos iniciais e complementa propostas para plataformas de programação orientada a objetos. Essa metodologia conta com um conjunto de ferramentas e técnicas que possibilitam a construção de modelos baseados nos conceitos oferecidos pelo i^* ⁶ [Yu95].

As fases cobertas por esta metodologia são as seguintes [Castro02]:

- A fase de Requisitos Iniciais está preocupada com o entendimento de um problema estudando uma configuração organizacional existente. Durante esta fase, os

⁶ i^* simboliza intencionalmente distribuídos e é um *framework* de modelagem que inclui os conceitos de ator e suas interdependências.

engenheiros de requisitos modelam os *stakeholders*⁷ como atores e suas intenções como metas. Cada meta é analisada do ponto de vista de seu ator resultando em um conjunto de dependências entre pares de atores. Como resultado desta fase dois modelos são construídos:

- O modelo de dependência estratégica (SD) que captura os atores relevantes, suas metas respectivas e suas interdependências; e
 - O modelo de razão estratégica (SR) que determina através de uma análise meios-fim como as metas podem ser cumpridas através das contribuições de outros atores.
- A fase de Requisitos Finais inclui o sistema de software que é introduzido como um outro ator no modelo de dependência estratégica [Castro01]. O ator que representa o sistema é relacionado aos atores sociais em termos de dependências. Este ator é detalhado através de uma análise meios-fim para produzir um novo modelo de razão estratégica. Suas metas são analisadas e revisadas.
- A fase de Projeto Arquitetural define a arquitetura global do sistema em termos de subsistemas, interconectados através de dado e fluxos de controle [Kolp01]. A arquitetura de um sistema constitui um modelo da estrutura relativamente pequeno e intelectualmente gerenciável, que descreve como os componentes do sistema trabalham juntos. Subsistemas são representados como atores e interconexões de dado/controle são representados como dependências de ator. Esta fase consiste de dois passos: 1. Selecionar o estilo arquitetural e; 2. Refinar os modelos de razão e dependência estratégica.
- Passo 1. Escolher o estilo arquitetural usando como critério as qualidades desejadas que foram identificadas na fase anterior e o *framework* NFR [Chung00] para conduzir esta análise de qualidade.

⁷ *Stakeholders* são pessoas ou organizações que serão afetadas pelo sistema e tem influência, direta ou indireta, sobre os requisitos do sistema – usuários finais, gerentes e outros envolvidos no processo organizacional influenciados pelo sistema, engenheiros responsáveis pelo desenvolvimento e manutenção do sistema, clientes da organização que usará o sistema para fornecer algum serviço, etc [Kotonya97].

Incluir novos atores, de acordo com a escolha de um estilo arquitetural específico de agente;

- Passo 2. Incluir novos atores e dependências, assim como decompor os atores e as dependências existentes em sub-atores e sub-dependências. Revisar os modelos de razão e dependência estratégica. As capacidades de ator são identificadas da análise das dependências indo e vindo do ator, bem como das metas e planos que o ator irá executar a fim de cumprir requisitos funcionais e não-funcionais.
- A fase Projeto Detalhado visa aplicar padrões sociais [Kolp01] ao projeto arquitetural do sistema multi-agente. Os padrões sociais são usados para solucionar uma meta específica que foi definida ao nível arquitetural através da identificação de estilos organizacionais e atributos de qualidade relevantes (*softgoals*). Uma análise detalhada de cada padrão social permite definir um conjunto de capacidades associadas com os agentes envolvidos no padrão. Uma capacidade estabelece que um ator é capaz de agir a fim de atingir uma meta determinada. Em particular, para cada capacidade o ator tem um conjunto de planos que podem aplicar em diferentes situações. Um plano descreve a seqüência de ações a executar e as condições sob o qual o plano é aplicável. Capacidades são coletadas num catálogo e associadas ao padrão. Isto permite definir os papéis e capacidade do ator que são adequados para um domínio particular. Esta fase também visa especificar minuciosamente o comportamento dos atores/agentes, em particular Tropos utilizando a AUML [Odell00].
- A fase de Implementação segue passo a passo, de forma natural, a especificação de projeto detalhado que é transformada em um esqueleto para a implementação. Isto é feito através de um mapeamento entre os modelos de projeto detalhado e os elementos de uma plataforma de implementação de agente tal como JADE [Bellifemine05] ou JADEX [Braubach04]; o código é adicionado ao esqueleto usando a linguagem de programação suportada pela plataforma de programação.

3.3 A PLATAFORMA JADE

JADE é um *middleware*⁸ totalmente desenvolvido em Java, que implementa uma plataforma distribuída e um *framework* de desenvolvimento para sistemas multi-agentes. JADE é composto por um ambiente de execução, uma biblioteca de classes e um pacote de ferramentas de suporte.

Jade é um software aberto, distribuído sob licença *Lesser General Public License* (LGPL) em [JADE04].

3.3.1 AMBIENTE DE EXECUÇÃO

O ambiente de execução JADE é baseado no conceito de *containers*⁹, que são uma instância de JADE rodando em uma *Java Virtual Machine* (JVM). Uma plataforma é composta por um conjunto de *containers* ativos. Numa plataforma JADE há sempre um único *container* principal responsável por centralizar certos serviços da plataforma. Esse *container* abriga o *Agent Management System* (AMS) e o *Directory Facilitador* (DF), os demais *containers* se conectam ao *container* principal, formando assim um ambiente de execução completa para rodar sistemas multi-agentes.

3.3.2 BIBLIOTECA DE CLASSES

Existem vários pacotes Java para JADE. Os principais pacotes de JADE são:

- O pacote **jade.core**, que implementa o kernel do sistema. Ele inclui a classe de agente (*Agent Class*), uma classe de comportamento (*Behaviour Class*) que está contida no sub-pacote **jade.core.behaviours**.

⁸ *Middleware* é um neologismo criado para designar camadas de software que não constituem diretamente aplicações. O objetivo do *middleware* é facilitar a integração de sistemas legados ou desenvolvidos de maneira não integrada.

⁹ O nome *container* se deve ao fato de que cada instância em JADE abriga agentes

Behaviour não é propriamente comportamento e sim uma implementação de uma tarefa ou intenção do agente, os programadores definem os agentes e escrevem os *behaviours* esperados.

- O sub-pacote **jade.lang.acl** é provido para processar as ACLs de acordo com as especificações da FIPA.
- O pacote **jade.content** é utilizado para permitir a utilização das ontologias e conteúdos de linguagem (CL – *Content languages*) definidos pelo usuário.
- O pacote **jade.domain** contém todas as classes Java que definem os agentes de gerenciamento de entidades estabelecidos pela FIPA, em particular o AMS e DF, além de outros conceitos como mobilidade e *sniffers*.
- O pacote **jade.gui** contém um conjunto de classes úteis para criar as GUIs para exibir e editar *Agent-Identifiers*, Agentes de descrição (*Agent Descriptions*), *ACLMessages*, dentre outros.
- O pacote **jade.mtp** contém uma interface Java que todo protocolo de transporte de mensagens deve implementar para ser integrado a JADE.
- O pacote **jade.proto** contém classes para modelar protocolos de interação (*fipa-request*, *fipa-query*, *fipa-subscribe*, além de muitos outros definidos pela FIPA)
- O pacote FIPA contém o módulo IDL para mensagens baseadas no IIOP-transporte.
- O sub-pacote **jade.tools** contém algumas ferramentas que facilitam a administração da plataforma.

3.3.3 PACOTE DE FERRAMENTAS DE SUPORTE

JADE prove um conjunto de ferramentas gráficas de suporte. São elas *Remote Monitoring Agent (RMA)*, *Directory Facilitador (DF)*, *Dummy Agent*, *Sniffer Agent* e *Introspector Agent*.

Remote Monitoring Agent (RMA)

Console gráfico para administração e controle da plataforma. Múltiplas instâncias podem ser abertas (em diversos *hosts*), JADE garante a coerência entre elas. A partir dessa ferramenta é que são ativadas as outras. (ver figura 3.1)

Directory Facilitador (DF)

Prove o serviço de páginas amarelas em JADE. É automaticamente carregado quando o ambiente é iniciado. (ver figura 3.2)

Dummy Agent

Ferramenta que possibilita o monitoramento e *debugg*. Possibilita a composição de mensagens ACL e envio para outros agentes, para observar como eles reagem ou respondem à mensagem. (ver figura 3.3)

Sniffer Agent

Ferramenta capaz de interceptar mensagens ACL vindas de agentes de quaisquer plataformas. As mensagens são mostradas graficamente usando uma notação similar à dos diagramas de seqüência de UML. (ver figura 3.4)

Introspector Agent

Permite analisar o ciclo de vida de um agente, as mensagens trocadas e os *behaviours* em execução. (figura 3.5)

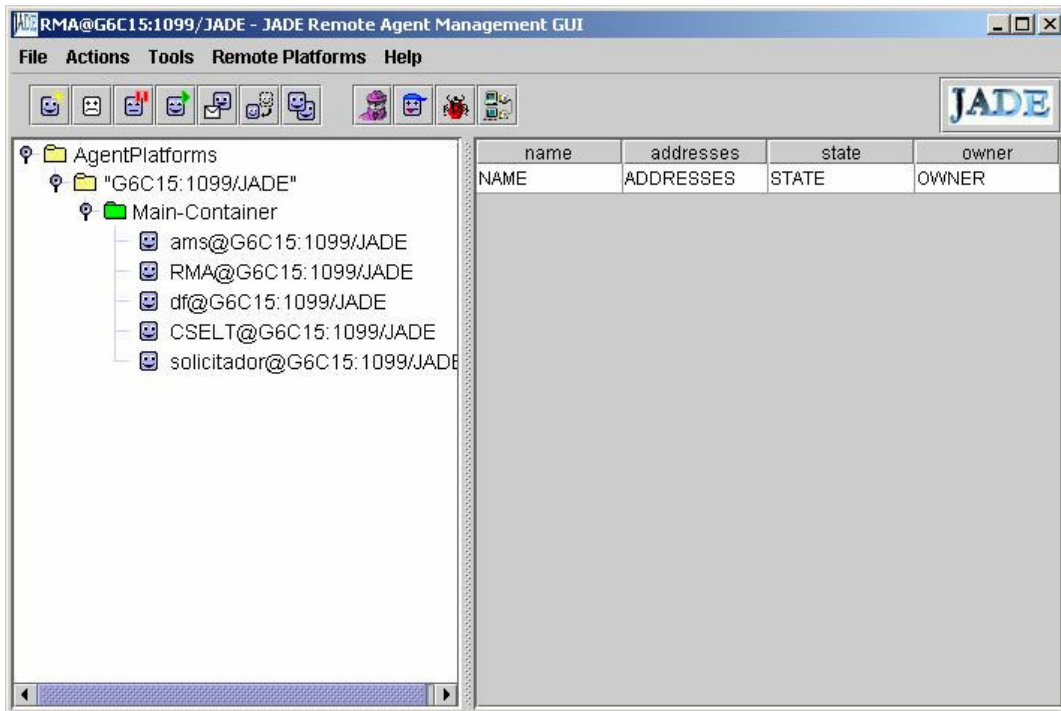


Figura 3.1: Remote Monitoring Agent (RMA)

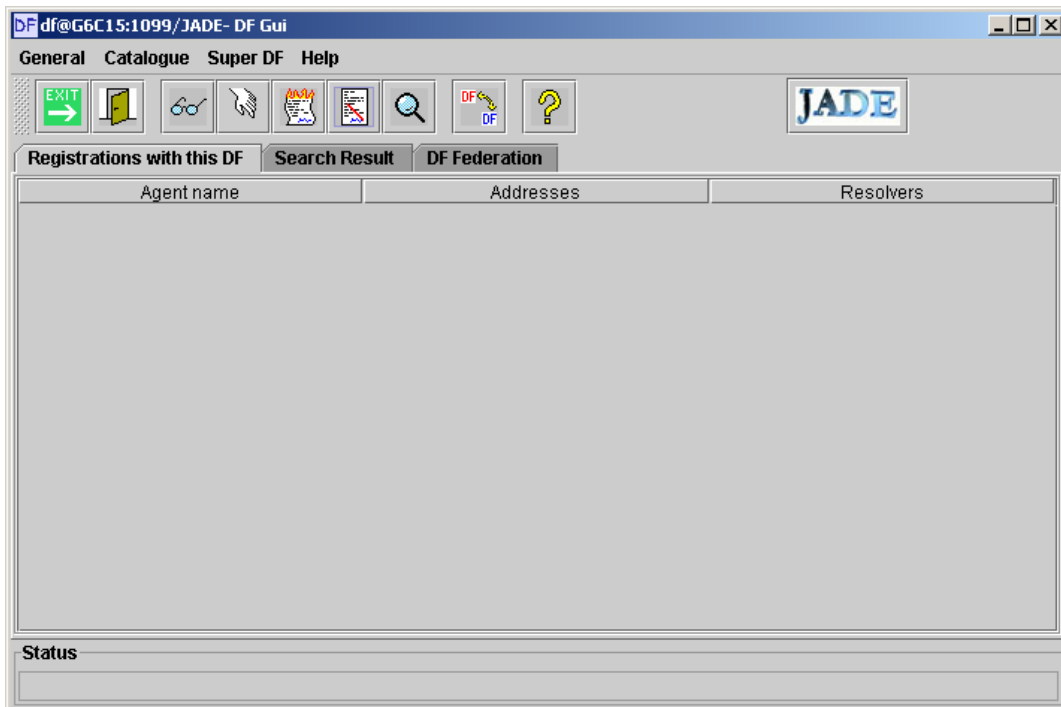


Figura 3.2: Directory Facilitador (DF)

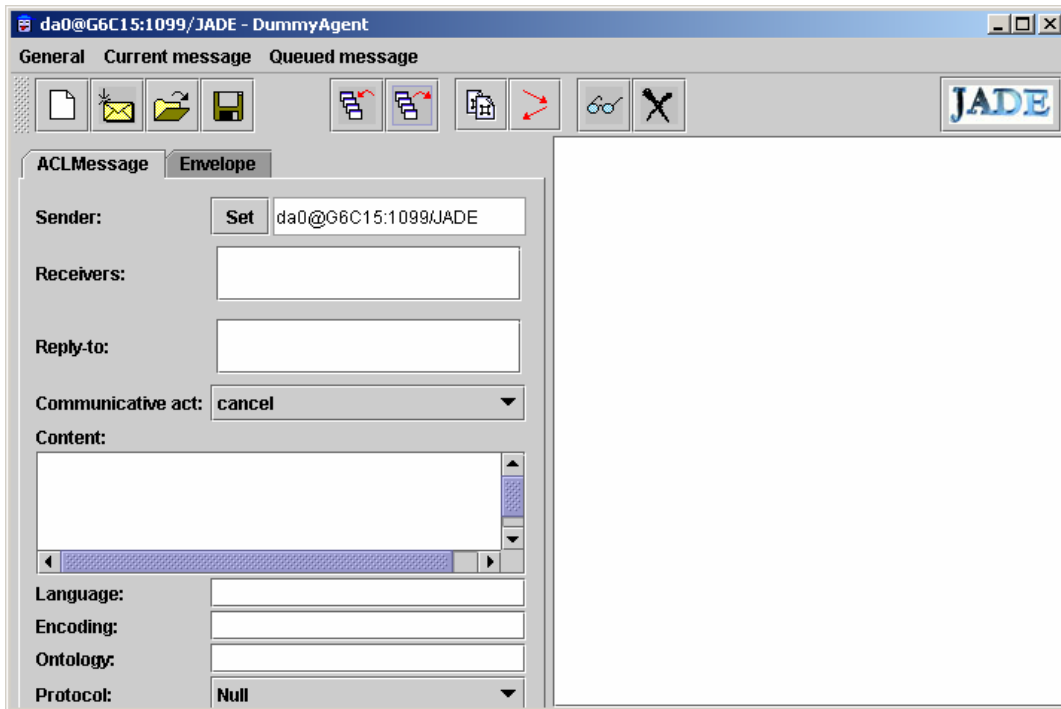


Figura 3.3: Dummy Agent

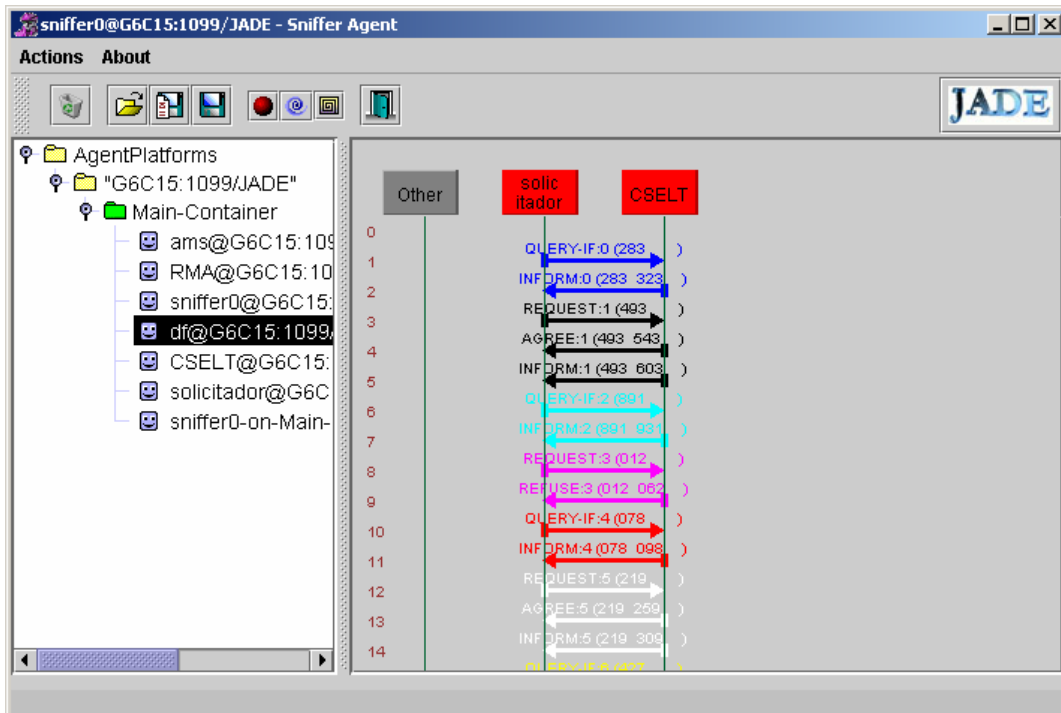


Figura 3.4: Sniffer Agent

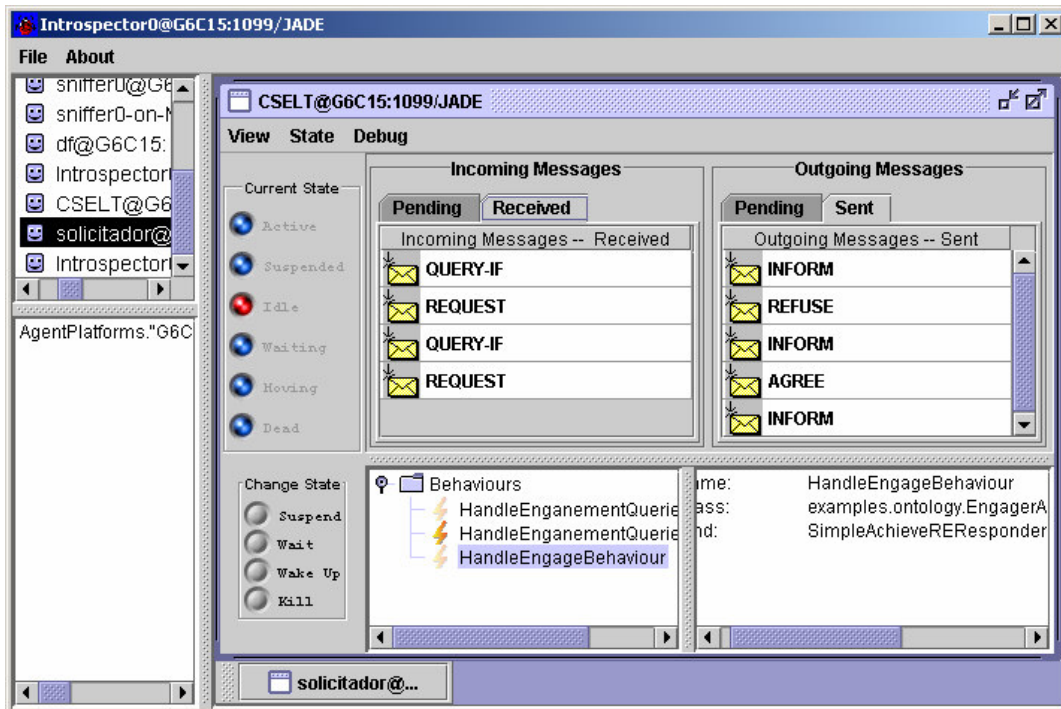


Figura 3.5: Introspector Agent

4 ESTUDO DE CASO

Nesse capítulo apresentaremos um estudo de caso, com o objetivo de apresentar nossa proposta de complementação da fase de projeto detalhado da metodologia Tropos. O estudo de caso escolhido baseia-se no sistema de controle de tráfego aéreo (SCTA) brasileiro. Inicialmente faremos uma descrição de como funciona SCTA. Em seguida, modelaremos o sistema utilizando a metodologia Tropos. Depois, mostraremos a implementação de um dos padrões propostos na plataforma JADE. Por fim, apresentaremos algumas considerações a respeito do nosso estudo de caso.

4.1 INTRODUÇÃO

Na maior parte do mundo, os padrões de aviação civil utilizados são regulamentados pelo ICAO (*International Civil Aviation Organization*) [ICAO05]. Essa organização, idealizada em 1944 e instituída em outubro de 1947 é sediada na cidade de Montreal no Canadá. Ao todo são 18 anexos com normas e métodos recomendados a serem seguidos pelos países membros da ICAO. O Brasil é membro da ICAO desde a sua fundação.

Nesse trabalho vamos nos basear no sistema de controle de tráfego aéreo brasileiro, o qual segue as normas da ICAO, para estabelecer a arquitetura do sistema.

Para este trabalho propusemos a utilização de dois padrões de projeto com mediadores orientados a agentes em um sistema de controle de tráfego aéreo, o padrão monitor e o padrão mediador.

4.2 SISTEMA DE CONTROLE DE TRÁFEGO AÉREO

Nesse capítulo descreveremos os componentes do sistema de controle de tráfego aéreo bem como seu funcionamento.

4.2.1 COMPONENTES DO SISTEMA

O sistema de controle de tráfego aéreo é composto por:

AIS - Sala de Informações Aeronáuticas de Aeródromo.

É nessa sala para onde o piloto se dirige antes do voo para obter informações a respeito das condições meteorológicas atuais e do funcionamento dos aeroportos (se as aeronaves estão podendo pousar/decolar e em quais condições isso está acontecendo). Nessa sala, o piloto também preenche o plano de voo.

TWR - Torre de Controle

É responsável pelo controle da aeronave junto ao piloto durante o tempo no qual ela se encontra no chão até que a mesma levante vôo ou se estiver pousando até que termine o pouso. Se a aeronave estiver decolando, o controle externo da aeronave passa a ser feito pela APP.

APP - Centro de Controle de Aproximação

É responsável pelo controle da aeronave junto ao piloto durante a aproximação e saída na área chamada de zona de aproximação ou “tambor” (área crítica na qual a aeronave levanta vôo até se estabilizar ou no momento do pouso até colocar os pneus na pista). Se a aeronave estiver decolando, quando sair do “tambor” o controle externo da aeronave passará a ser do ACC. Se a aeronave estiver pousando, no momento em que colocar os pneus na pista seu controle externo passará à TWR.

ACC - Centro de Controle de Área

Faz parte do CINDACTA (Centro Integrado de Defesa Aérea) e existem três centros desse tipo no Brasil: Brasília, Recife e Curitiba. Cada um desses centros é responsável por atuar em uma área, juntos eles cobrem todo o território nacional. É responsável pelo controle da aeronave na estrada aérea até o momento em que a aeronave entra na área crítica novamente (“tambor”) quando volta a ser controlada pela APP.

4.2.2 FUNCIONAMENTO DO SISTEMA

O Controle de Tráfego Aéreo se inicia pelo menos 45 minutos antes da aeronave levantar vôo. O piloto responsável pela aeronave deve ir até a sala AIS e receber informações de segurança, como informações meteorológicas e listagem de todos os aeroportos. Para a AIS conseguir a listagem, os funcionários da AIS se comunicam via rádio com os funcionários dos aeroportos. Nessa listagem o piloto vai poder ver como os aeroportos estão operando. Existem três possibilidades de estado para os aeroportos:

- Aberto: o aeroporto está funcionando normalmente possibilitando pousos e decolagens. Toda a aparelhagem do aeroporto também está funcionando.

- Operando manualmente: o aeroporto está possibilitando pousos e decolagens. Porém a aparelhagem do aeroporto não está funcionando ou está funcionando parcialmente e por isso o aeroporto opera manualmente.
- Fechado: o aeroporto não permite pousos nem decolagens.

De posse dessas informações, o piloto preenche o plano de vôo, dando todas as informações a respeito do vôo, como horários de saída e chegada, locais de chegada (um aeroporto de destino principal e mais dois alternativos) e quantidade de combustível presente na aeronave.

Na sala AIS as informações do plano de vôo são passadas ao ACC, o qual autoriza ou fornece alguma alternativa ao plano de vôo proposto pelo piloto. Caso o plano não seja autorizado o piloto pode modificar o plano de vôo, seguindo as sugestões da ACC e novamente entregam-lo na sala AIS até que o mesmo seja aprovado pelo ACC.

Tendo sido provado o plano de vôo, a aeronave pode iniciar seu processo de partida. Em todos os momentos o piloto irá ser assistido por um controle externo à aeronave, o qual irá auxiliar o vôo passando ao piloto todas as informações necessárias.

Inicialmente, quando o avião ainda está na pista de pouso e decolagem, a torre de controle (TWR) é quem fica responsável por auxiliar o piloto no controle da aeronave. A TWR é responsável pela organização das filas de partida e chegada e pelo controle da aeronave junto ao piloto até que a aeronave decole. A TWR passa ao piloto o tipo de saída que o piloto deve utilizar e o código identificador do vôo.

Logo após a decolagem, o Centro de Controle de Aproximação (APP) passa a guiar a aeronave junto ao piloto. Passada a área da zona de aproximação, o controle passa do APP para o Centro de Controle de Área (ACC). O ACC controla as estradas aéreas, são ao todo três áreas cobertas em todo o território nacional (Recife, Brasília e Curitiba). Caso a aeronave esteja em uma dessas áreas e passe para outra, o controle externo passa automaticamente para a ACC responsável pela área na qual a aeronave se encontra. Quando a aeronave for pousar e entrar na zona de aproximação (“tambor”) o controle

passa novamente ao APP e quando terminar a zona de aproximação, a TWR novamente passará a auxiliar o controle da aeronave.

Durante o voo, o controle externo da aeronave (a TWR, o APP ou o ACC) tem acesso ao plano de voo da aeronave. Assim, dentre outras informações o controle externo da aeronave está ciente do destino da aeronave, por isso em intervalos regulares os funcionários do controle externo se comunicam via rádio com os funcionários dos aeroportos a fim de tomar conhecimento de alguma mudança de estado nos mesmos. Dessa maneira, quando em algum momento do voo, o aeroporto de destino da aeronave muda de estado, o controle externo atual da aeronave (a TWR, o APP ou o ACC) deve informar ao piloto, também se comunicando via rádio, sobre a mudança de estado do aeroporto.

Ao saber que o aeroporto de destino não está aberto, cabe ao piloto solicitar ao controle externo o estado dos aeroportos alternativos. Em caso de haver necessidade de uma solicitação de pouso de emergência, o piloto deverá se comunicar via rádio com o controle externo solicitando pouso. O controle externo então se dirige ao aeroporto (novamente via rádio) a fim de negociar um local para o pouso da aeronave.

4.3 MODELAGEM COM TROPOS

O sistema foi modelado utilizando a metodologia Tropos. A modelagem inclui requisitos iniciais, requisitos finais, projeto arquitetural e projeto detalhado.

4.3.1 REQUISITOS INICIAIS

Durante a fase de requisitos iniciais procuramos modelar a estrutura organizacional do sistema de controle de tráfego aéreo.

Os atores encontrados para a aplicação são aeronave, AIS, ACC, TWR, APP e Aeroporto, nomeados respectivamente de *aeronave*, *AIS*, *ACC*, *TWR*, *APP* e *Aeroporto*.

A aeronave é o elemento central dessa organização, só não apresentando dependência com o Aeroporto. (ver figura 4.1)

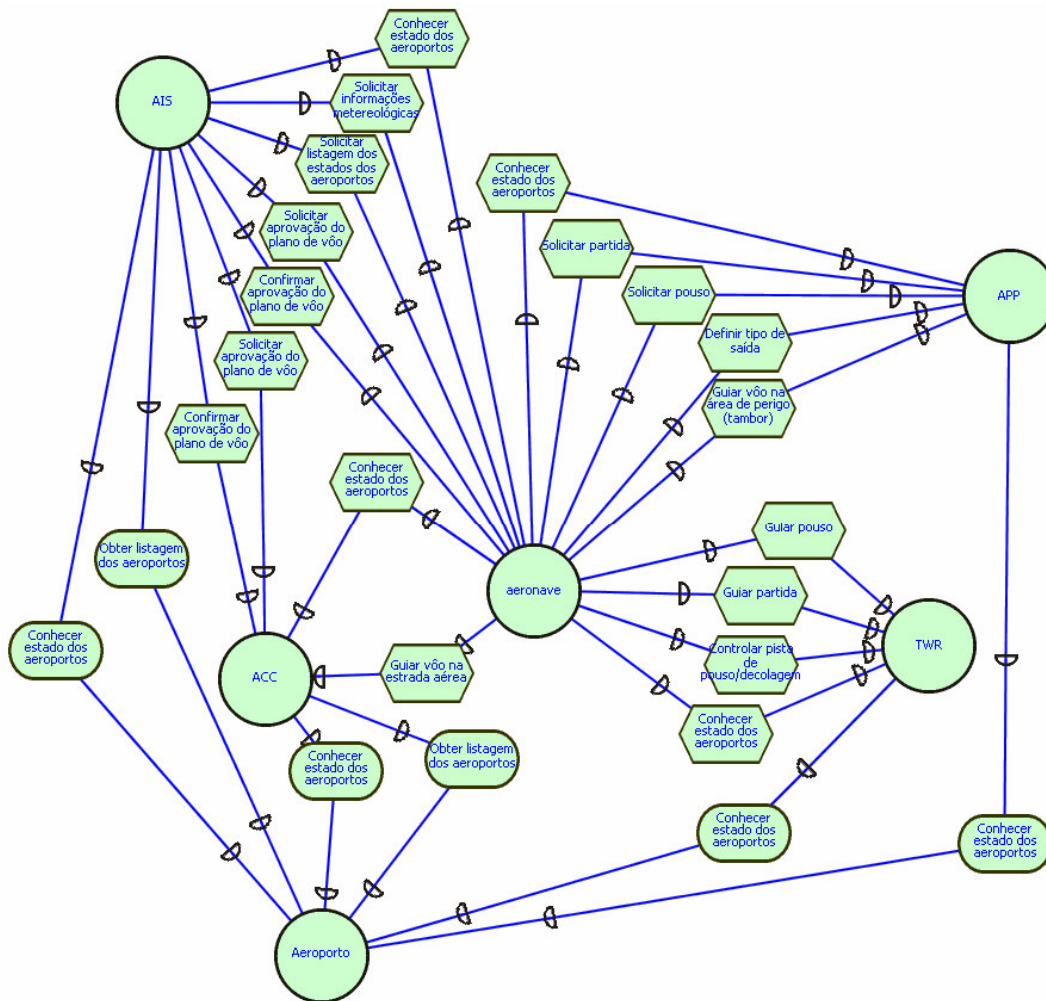


Figura 4.1: Modelagem Organizacional do Sistema de Controle de Tráfego Aéreo

A aeronave depende da AIS para resolver as tarefas de *Conhecer estados dos aeroportos*, *Solicitar informações meteorológicas*, *Solicitar listagem dos aeroportos*, *Solicitar aprovação do plano de vôo* e *Confirmar aprovação do plano de vôo*. A AIS, por sua vez, depende da ACC para executar as tarefas de *Solicitar aprovação do plano de vôo* e *Confirmar a aprovação do plano de vôo*.

A aeronave também possui dependências de tarefas com o ACC a fim de *Conhecer estados dos aeroportos* e *Guiar vôo na estrada aérea*.

As dependências existentes entre a aeronave e a TWR são as tarefas de *Conhecer estados dos aeroportos, Controlar pista de pouso/decolagem, Guiar partida e Guiar pouso.*

Entre a aeronave e o APP existem as dependências das tarefas de *Conhecer estados dos aeroportos, Solicitar partida, solicitar pouso, Definir tipo de saída e Guiar vôo na área de perigo (tambor).*

A AIS e o ACC possuem as dependências com o Aeroporto para atingir as metas *Conhecer estado dos aeroportos e Obter listagem dos aeroportos.*

A TWR e o APP também possuem dependências com o Aeroporto para obter a meta de *Conhecer estado dos aeroportos.*

Nessa etapa de requisitos iniciais, foi possível identificar algumas falhas de comunicação nessa organização:

- Para ter conhecimento dos estados dos aeroportos, os agentes AIS, ACC, APP e TWR comunicam-se via rádio com os aeroportos. Em locais onde existem mudanças bruscas de clima com uma curta distância geográfica isso têm sido relatado como um problema bastante persistente.
- A comunicação entre os aeroportos também acontece via rádio.
- Ao tomar conhecimento de aeroportos fechados os agentes AIS, ACC, APP e TWR comunicam-se via rádio com outros aeroportos e com a aeronave a fim de negociar o pouso.

São relatadas falhas nessa transmissão de informações, tanto devido ao próprio sistema de rádio o qual não é totalmente robusto, quanto por falhas humanas.

4.3.2 REQUISITOS FINAIS

Devido aos problemas encontrados durante a modelagem organizacional, nessa etapa nós propomos um sistema que substitui a comunicação via rádio, automatizando a organização para resolver as seguintes necessidades: (ver figura 4.2)

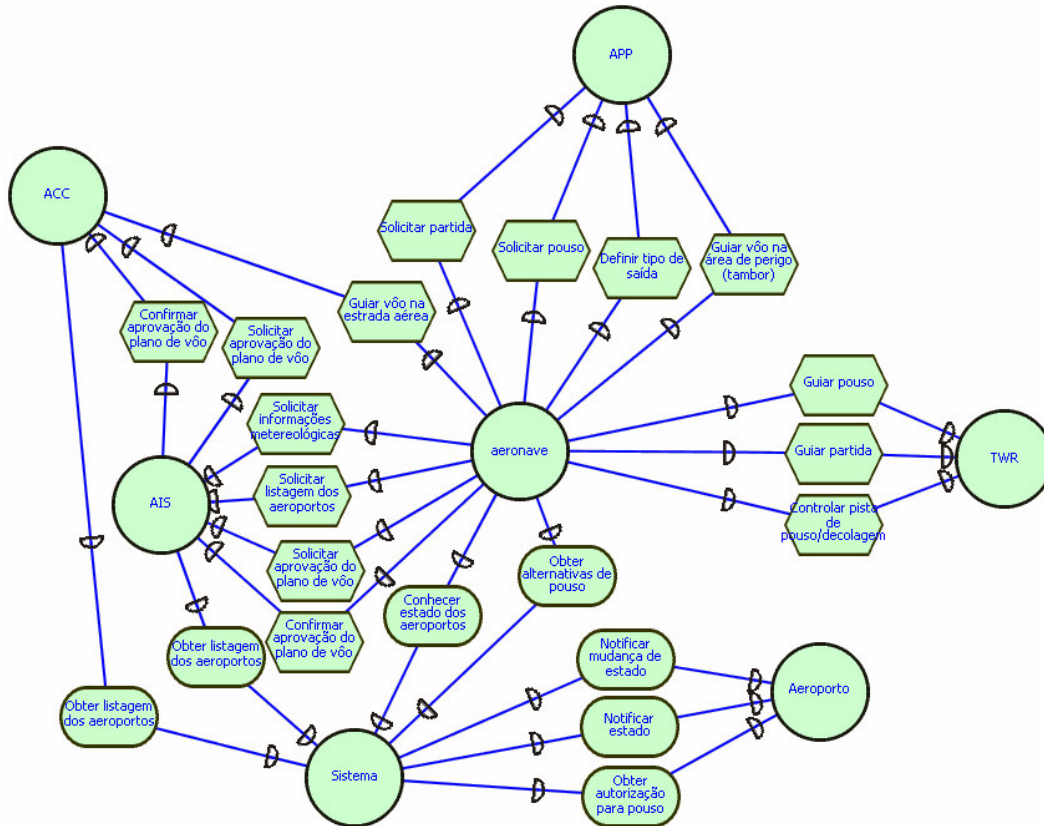


Figura 4.2: Modelagem dos Requisitos Finais do Sistema

Antes de levantar vôo o piloto solicita a AIS a listagem dos aeroportos e posteriormente solicita através da AIS a aprovação do plano de vôo pelo ACC

Originalmente vimos que a AIS assim como o ACC se comunicavam com o aeroporto para obter a listagem dos estados dos aeroportos. A AIS solicitava a listagem para passá-la ao piloto da aeronave e a ACC utilizava a listagem para a aprovação do plano de vôo.

Com o sistema proposto, a comunicação feita pela AIS e pelo ACC para obter a listagem dos Aeroportos também foi substituída pelo Sistema. Assim, surge a meta *Obter listagem dos aeroportos* entre a AIS e o Sistema e entre a ACC e o Sistema.

Durante o vôo o piloto da aeronave precisa ser avisado de mudanças de estado do aeroporto de destino

Originalmente vimos que essa tarefa era realizada pelo controle externo da aeronave. Dependendo do local onde a aeronave se encontra a TWR, o ACC ou o APP faria esse papel, sempre se comunicando tanto com o piloto da aeronave quanto com os aeroportos via rádio.

Com o sistema proposto, agora a aeronave comunica-se diretamente com o sistema o qual passa as informações de mudança de estado dos aeroportos. Para isso, o sistema comunica-se também com os aeroportos a fim de obter qualquer mudança de estado que haja. Assim, as tarefas *Conhecer estado dos aeroportos* existente entre a aeronave e a AIS, o ACC, a TWR e o APP passam a não ser mais necessárias, existindo agora a meta *Conhecer estado dos aeroportos* entre a aeronave e o Sistema.

Durante o vôo surge a necessidade de um pouso de emergência

Quando havia alguma necessidade de aterrissagem de emergência o piloto da aeronave se comunicava com os controladores externos para que esses negociassem com os aeroportos uma pista de aterrissagem disponível para a aeronave.

Com o sistema proposto quando for necessário um pouso emergencial a comunicação aparece a aeronave e o Sistema. O próprio sistema negociará com os aeroportos uma pista de pouso disponível. Assim, a aeronave depende do Sistema para atingir a meta *Obter Alternativas de pouso*.

Com esse sistema, esperamos remover o problema de comunicação via rádio. Evitando as falhas de comunicação abordadas anteriormente.

4.3.3 PROJETO ARQUITETURAL

Para o sistema proposto utilizamos dois padrões de projeto, o monitor e o mediador. Assim, foram criados os agentes Monitor e Mediador. (ver figura 4.3)

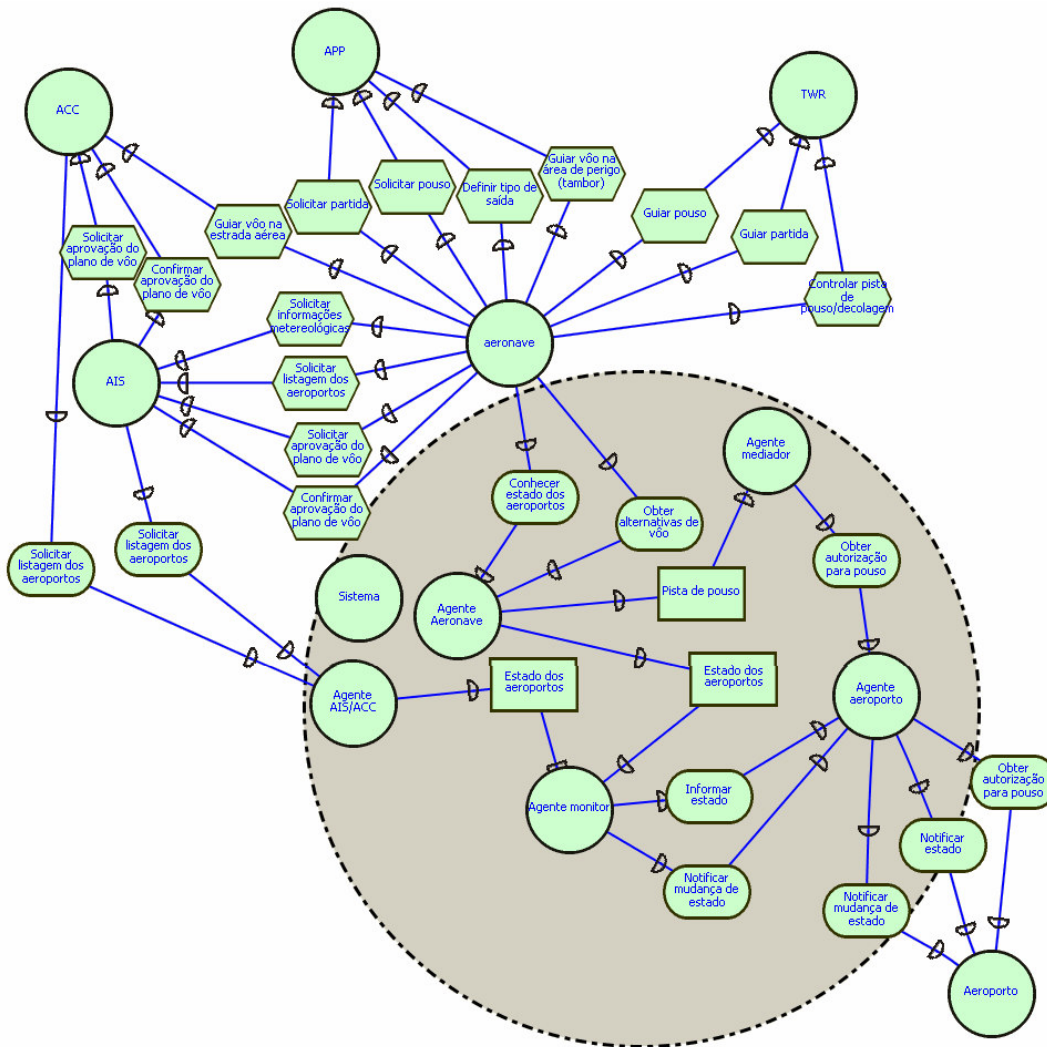


Figura 4.3: Modelagem do Projeto Arquitetural do Sistema

Agente Monitor

O agente Monitor foi proposto para resolver dois dos problemas citada na etapa de requisitos finais:

- Antes de levantar vôo o piloto solicita a AIS a listagem dos aeroportos e posteriormente solicita através da AIS a aprovação do plano de vôo pelo ACC e

- Durante o voo o piloto da aeronave precisa ser avisado de mudanças de estado do aeroporto de destino.

Nessa etapa, foram criados além do agente Monitor, os agentes aeronave, AIS/ACC, e agente Aeroporto. Inicialmente o agente Aeroporto se registra no sistema, depois o agente Monitor solicita ao sistema a listagem de aeroportos existentes. De posse dessa informação o agente Monitor faz uma solicitação aos agentes Aeroportos sobre seus estados e os armazena. Sempre que uma aeronave necessita de informações sobre o Aeroporto, solicita ao agente Monitor que monitore o aeroporto de destino. Assim, o agente aeronave se registra no agente Monitor para receber notificações de mudanças de estado do aeroporto de destino. O agente Monitor então passa a solicitar ao agente aeronave seu estado atual regularmente. Assim, a listagem dos aeroportos está sempre sendo atualizada. Quando a AIS ou a ACC necessitam de uma listagem dos aeroportos solicitam ao agente AIS/ACC e este entra em contato com o agente Monitor, o qual se encarrega de obter a listagem de todos os estados dos aeroportos.

Agente Mediador

O agente Mediador foi proposto para resolver o terceiro problema citado durante a etapa de requisitos finais:

- Durante o voo surge a necessidade de um pouso de emergência.

Nessa etapa a aeronave, sempre que necessite de um pouso emergencial, se comunica com o agente Mediador através do agente aeronave. O papel do agente Mediador é conseguir negociar com os aeroportos um local mais adequado para o pouso de maneira rápida e informar ao agente aeronave o resultado da negociação.

4.3.4 PROJETO DETALHADO

Nessa fase nós aplicamos o padrão social monitor ao projeto arquitetural do sistema. Apesar de a metodologia Tropos propor padrões sociais, ela não adota um modelo que descreva esses padrões. Nesse trabalho utilizaremos diagramas UML estendidos para representar a colaboração e a estrutura dos agentes.

4.3.4.1 MODELO DO PADRÃO MONITOR

Abaixo, a tabela 4.1 mostra um modelo mais detalhado do padrão monitor utilizado nesse trabalho.

Elemento do Modelo	Descrição
<i>Nome</i>	Monitor
<i>Objetivo</i>	Possibilitar ao agente (subscriber) monitorar outro agente (fonte) sem que para isso tenha que interagir diretamente com o mesmo.
<i>Aplicação</i>	Quando um agente (subscriber) necessita ser avisado a respeito da modificação do estado de algum assunto de seu interesse, mas não deseja interagir diretamente com o agente que sofre a modificação de estado.
<i>Exemplo</i>	Um agente (subscriber) necessita de ser notificado de alguma modificação do estado de um outro agente (fonte). O agente monitor age como um agente intermediário o qual envia mensagens ao agente fonte e recebe dele a informação do seu estado. Quando há alguma modificação do estado do agente fonte, o agente monitor envia ao subscriber a notificação do novo estado do agente fonte.
<i>Participantes</i>	O subscriber que requer a notificação da mudança de estado de outro agente (fonte). O monitor que prover o serviço de monitorar o agente fonte e envia ao subscriber notificação de qualquer modificação de estado. O fonte o qual passa a responder pedidos de notificação de estado ao agente monitor.

Tabela 4.1: Modelo de descrição do agente monitor

4.3.4.2 COLABORAÇÃO

A fim de melhor visualizar a colaboração entre os agentes, utilizamos um diagrama de seqüência que descreve o padrão Monitor através do agente Monitor, agente aeronave e agente Aeroporto. (ver figura 4.4)

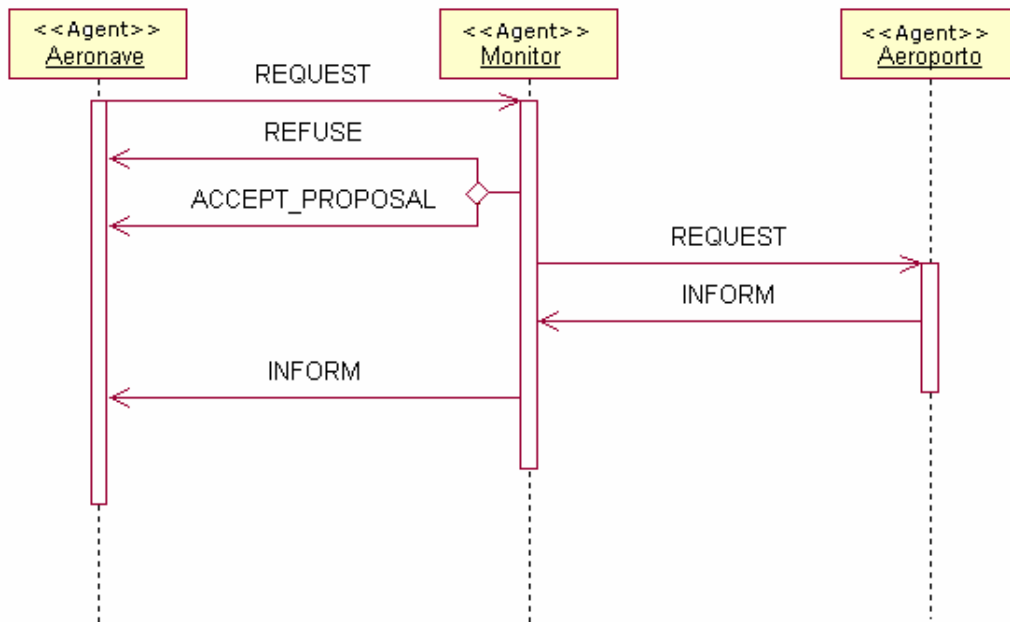


Figura 4.4: Diagrama de Seqüência

Os agentes trocam mensagens caracterizadas por performativas JADE. Essas mensagens definem o protocolo do padrão monitor. Um protocolo descreve um conjunto de possíveis mensagens trocadas entre agentes.

O agente aeronave pode enviar uma mensagem de REQUEST ao agente Monitor solicitando a inscrição em seu serviço de supervisão de aeroportos. O agente Monitor pode responder a mensagem com uma mensagem caracterizada por:

- i. Uma mensagem do tipo ACCEPT_PROPOSAL, indicando que o agente Monitor aceitou o pedido de inscrição do agente aeronave.
- ii. Uma mensagem do tipo REFUSE, indicando que o agente Monitor recusou o pedido de inscrição do agente aeronave.

Tendo aceito o pedido do agente aeronave, o agente Monitor envia uma mensagem do tipo INFORM ao agente aeronave informando o estado do aeroporto de destino da aeronave, e passa a enviar mensagens de REQUEST solicitando ao agente Aeroporto seu estado atual. O agente Aeroporto responde à requisição do agente Monitor enviando-lhe uma mensagem do tipo INFORM contendo seu estado atual. Se o estado do agente Aeroporto mudar o agente Monitor envia novamente uma mensagem de INFORM ao agente aeronave. Quando o agente aeronave recebe um INFORM com a notícia de que seu aeroporto de destino está fechado, reinicia a troca de mensagens com o agente Monitor enviando agora o aeroporto de destino alternativo.

4.3.4.3 ESTRUTURA

O diagrama mostrado na figura 4.5 descreve a estrutura do padrão monitor. Esse diagrama é uma extensão do diagrama de classes UML e procura fornecer uma visão dos agentes envolvidos do padrão utilizado na implementação em JADE.

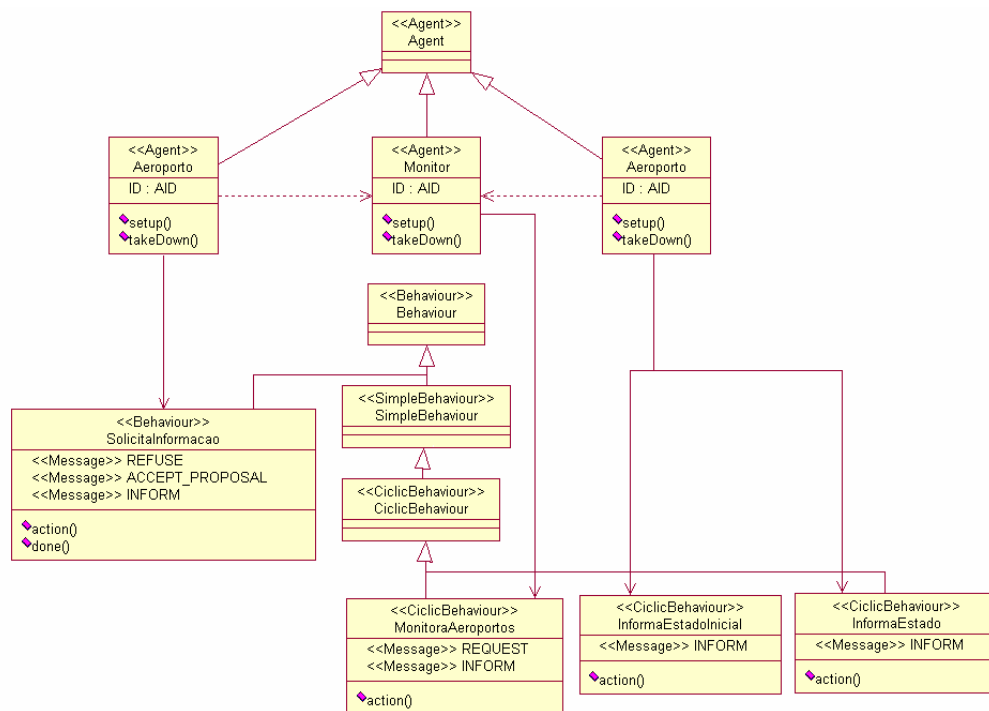


Figura 4.5: Estrutura do padrão monitor

O agente é caracterizado por uma classe com o estereótipo <<Agent>>. Todo agente executa um ou mais comportamentos. Os comportamentos são caracterizados pelos estereótipos <<Behavior>>, <<SimpleBehaviour>> ou <<CyclicBehaviour>>, dependendo do tipo de comportamento que apresentem. Para representar que um agente possui um comportamento, inserimos uma seta entre o agente e o comportamento do agente.

A habilidade de um agente de estabelecer comunicação com outros agentes é representada por uma dependência entre os agentes. Observe que <<Agent>> e <<Behaviour>> são classes que estendem respectivamente a classe Agent do JADE e a classe Behaviour.

Os estereótipos <<Message>> indicam as performativas JADE com as quais o agente que executa aquele comportamento pode lidar.

Na seção seguinte nós mostraremos dois exemplos da implementação do padrão monitor feitos utilizando a plataforma JADE. Dessa maneira, será mais fácil tanto o entendimento da colaboração entre os agentes quanto o comportamento de cada um deles.

4.3.5 IMPLEMENTAÇÃO EM JADE

Os agentes foram implementados utilizando a plataforma JADE. Adiante damos uma descrição dos comportamentos de cada agente.

Agente aeronave

Comportamento: SolicitaInformacao

Através desse comportamento o agente aeronave envia aos agentes monitores disponíveis uma mensagem do tipo REQUEST solicitando ao agente Monitor a supervisão do aeroporto de destino da aeronave. Caso a resposta do agente Monitor seja um REFUSE, o agente reenvia o pedido de REQUEST até que o mesmo seja atendido. Caso a resposta seja um ACCEPT_PROPOSAL, o agente aeronave se prepara para receber mensagens de INFORM com notícias a respeito do agente Aeroporto. Se na

mensagem INFORM vier a notícia de que o agente Aeroporto encontra-se fechado, o agente aeronave reenvia o pedido de REQUEST, desta vez solicitando a supervisão do primeiro aeroporto alternativo. Se o aeroporto monitorado for o primeiro aeroporto alternativo e ele estiver fechado, o agente aeronave envia outro pedido de REQUEST solicitando a supervisão do segundo aeroporto alternativo.

Agente Monitor

Comportamento: MonitoraAeroportos

Através desse comportamento, o agente Monitor envia uma mensagem de REQUEST a todos os agentes Aeroportos solicitando seus estados. Os agentes Aeroportos respondem ao pedido com uma mensagem de INFORM contendo o estado do aeroporto. Quando um agente aeronave envia ao agente Monitor uma mensagem de REQUEST solicitando o serviço de supervisão dos aeroportos ao agente Monitor, o agente monitor responde a mensagem com uma mensagem do tipo ACCEPT_PROPOSAL, caso o aeroporto de destino da aeronave esteja entre os aeroportos supervisionados ou, caso contrário, com uma mensagem de REFUSE. Se o agente Monitor houver enviado uma mensagem de ACCEPT_PROPOSAL, em seguida ele enviará uma mensagem do tipo INFORM com o estado do aeroporto de destino da aeronave. A partir desse momento, o agente Monitor passa a enviar periodicamente ao agente Aeroporto mensagens de REQUEST solicitando o estado do aeroporto de destino da aeronave. Caso haja alguma modificação no estado do aeroporto, o agente Monitor envia ao agente aeronave o novo estado do aeroporto de destino da aeronave.

Agente Aeroporto

Comportamento 1: InformaEstadoInicial

Esse comportamento possibilita ao agente aeronave responder ao agente Monitor quando inicialmente ele solicita através de uma mensagem do tipo REQUEST o estado do aeroporto. O agente Aeroporto responde a mensagem com uma mensagem do tipo INFORM com seu estado atual.

Comportamento 2: InformaEstado

Esse comportamento é utilizado quando o agente Monitor aceita o pedido do agente aeronave e passa a solicitar ao agente Aeroporto seu estado atual. O agente Aeroporto

recebe do agente Monitor um pedido de REQUEST e responde com uma mensagem do tipo INFORM contendo seu estado atual.

Para um melhor entendimento dos comportamentos entre os agentes mostraremos dois exemplos da implementação do padrão monitor na plataforma JADE.

Exemplo 1:

Nesse primeiro exemplo temos apenas um agente aeronave, um agente Monitor e um agente Aeroporto. O agente Aeroporto possui estado “Aberto”. (ver figura 4.6)



Figura 4.6: Agente Aeroporto em estado Aberto

O agente aeronave tem como destino principal o aeroporto “Aeroporto” e como destinos alternativos principal e secundário os aeroportos “Aeroporto2” e “Aeroporto3” respectivamente. (ver figura 4.7)



Figura 4.7: Aeroportos de destino do agente aeronave

O agente monitor solicita inicialmente ao *Directory Facilitador* (DF) a listagem dos agentes Aeroportos inscritos através de uma mensagem do tipo REQUEST. O DF age como um agente *matchmaker* fornecendo um serviço do tipo páginas amarelas. Assim

que o agente Monitor recebe a listagem dos aeroportos com o Aeroporto registrado, envia ao mesmo uma mensagem do tipo REQUEST solicitando seu estado. O agente Aeroporto responde à mensagem com uma mensagem do tipo INFORM contendo o estado do seu aeroporto. Ao solicitar ao agente Monitor seu serviço, o agente aeronave envia um pedido de REQUEST ao agente Monitor com nome do seu aeroporto de destino. Como o agente Monitor já tem na sua listagem de aeroportos o aeroporto de destino da aeronave, envia em resposta uma mensagem de ACCEPT_PROPOSAL. Em seguida, o agente Monitor envia ao agente aeronave uma mensagem do tipo INFORM com o estado do aeroporto de destino da aeronave (“Aberto”). O agente Monitor passa então a enviar periodicamente mensagens do tipo INFORM ao agente Aeroporto com a finalidade de observar mudanças de estado no aeroporto de destino da aeronave.

Na figura 4.8 é possível observar a troca de mensagens ocorrida entre os agentes do exemplo 1.

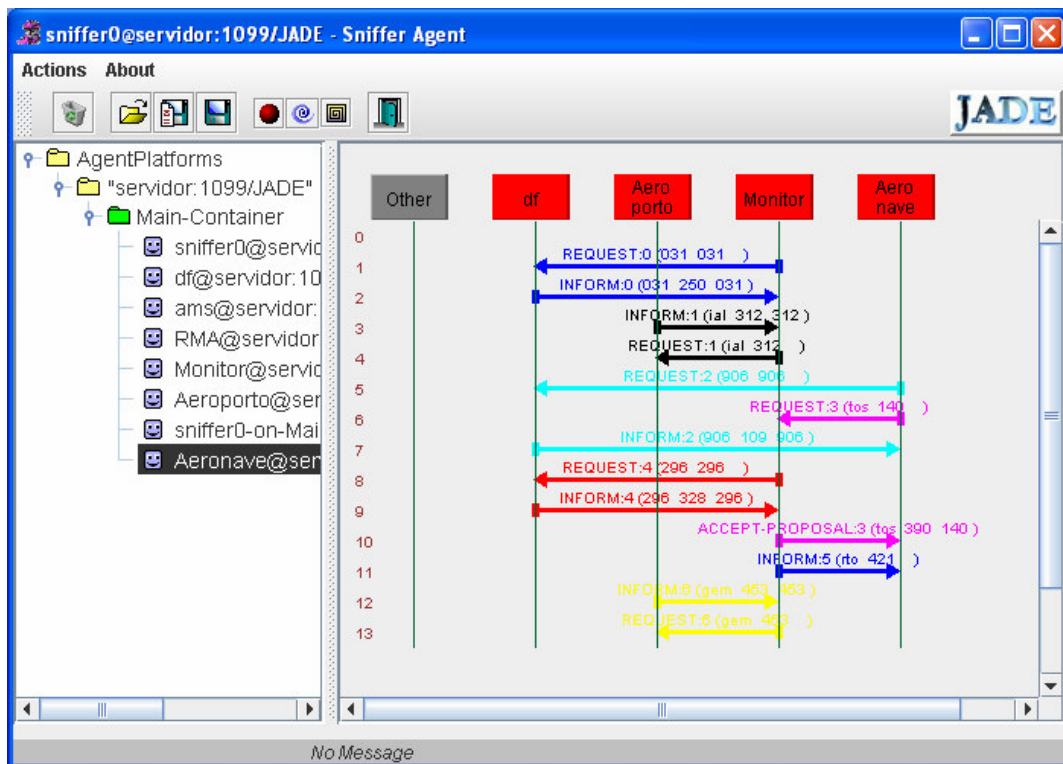
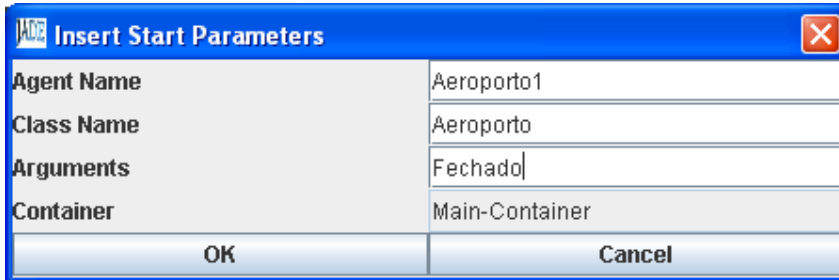


Figura 4.8: Troca de mensagens entre os atores

Exemplo 2:

No segundo exemplo iremos mostrar o comportamento do sistema quando temos um agente aeronave, um agente Monitor e dois agentes Aeroportos. O primeiro agente Aeroporto representa o aeroporto “Aeroporto1” e está no estado “Fechado”. (ver figura 4.9)



The image shows a dialog box titled "Insert Start Parameters" with a blue header and a close button in the top right corner. The dialog contains four input fields: "Agent Name" with the value "Aeroporto1", "Class Name" with the value "Aeroporto", "Arguments" with the value "Fechado", and "Container" with the value "Main-Container". At the bottom of the dialog, there are two buttons: "OK" on the left and "Cancel" on the right.

Agent Name	Aeroporto1
Class Name	Aeroporto
Arguments	Fechado
Container	Main-Container
OK Cancel	

Figura 4.9: Agente Aeroporto1 em estado Fechado

O segundo agente Aeroporto representa o aeroporto “Aeroporto2” e está no estado “Aberto”. (ver figura 4.10)



The image shows a dialog box titled "Insert Start Parameters" with a blue header and a close button in the top right corner. The dialog contains four input fields: "Agent Name" with the value "Aeroporto2", "Class Name" with the value "Aeroporto", "Arguments" with the value "Aberto", and "Container" with the value "Main-Container". At the bottom of the dialog, there are two buttons: "OK" on the left and "Cancel" on the right.

Agent Name	Aeroporto2
Class Name	Aeroporto
Arguments	Aberto
Container	Main-Container
OK Cancel	

Figura 4.10: Agente Aeroporto2 em estado Aberto

O agente aeronave tem como destino principal o aeroporto “Aeroporto1” e como destinos alternativos principal e secundário os aeroportos “Aeroporto2” e “Aeroporto3” respectivamente. (ver figura 4.11)

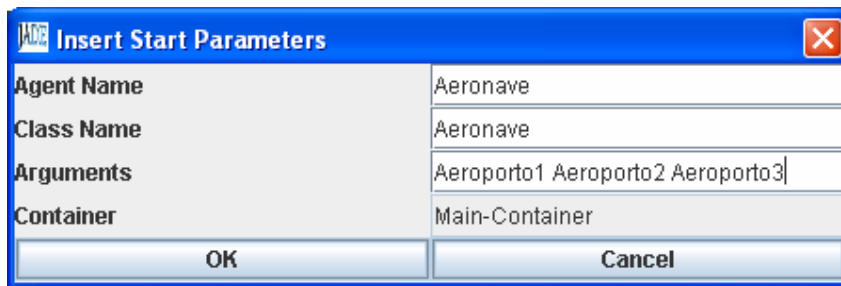


Figura 4.11: Aeroportos de destino do agente aeronave

O agente monitor solicita inicialmente ao *Directory Facilitador* (DF) a listagem dos agentes Aeroportos inscritos através de uma mensagem do tipo REQUEST. Assim que o agente Monitor recebe a listagem dos aeroportos com os Aeroportos registrados, envia aos mesmos uma mensagem do tipo REQUEST solicitando seus estados. Os agentes Aeroportos “Aeroporto1” e “Aeroporto2” respondem à mensagem com uma mensagem do tipo INFORM contendo o estado dos seus respectivos aeroportos. Ao solicitar ao agente Monitor seu serviço, o agente aeronave envia um pedido de REQUEST ao agente Monitor com nome do seu aeroporto de destino, o “Aeroporto1”. Como o agente Monitor já tem na sua listagem de aeroportos o aeroporto de destino da aeronave, envia em resposta uma mensagem de ACCEPT_PROPOSAL. Em seguida, o agente Monitor envia ao agente aeronave uma mensagem do tipo INFORM com o estado do aeroporto de destino da aeronave.

Como o aeroporto de destino está fechado, o agente aeronave reenvia a solicitação de pedido de serviço do agente Monitor através de uma mensagem do tipo REQUEST. Desta vez a mensagem contém o novo aeroporto de destino da aeronave, o “Aeroporto2”. O agente Monitor responde ao pedido do agente aeronave com um ACCEPT_PROPOSAL e em seguida envia ao agente aeronave uma mensagem do tipo INFORM com o estado do “Aeroporto2”.

O agente Monitor passa então a enviar periodicamente mensagens do tipo INFORM ao agente Aeroporto2 com a finalidade de observar mudanças de estado no aeroporto de destino da aeronave.

Na figura 4.12 é possível observar a troca de mensagens ocorrida entre os agentes do exemplo 2.

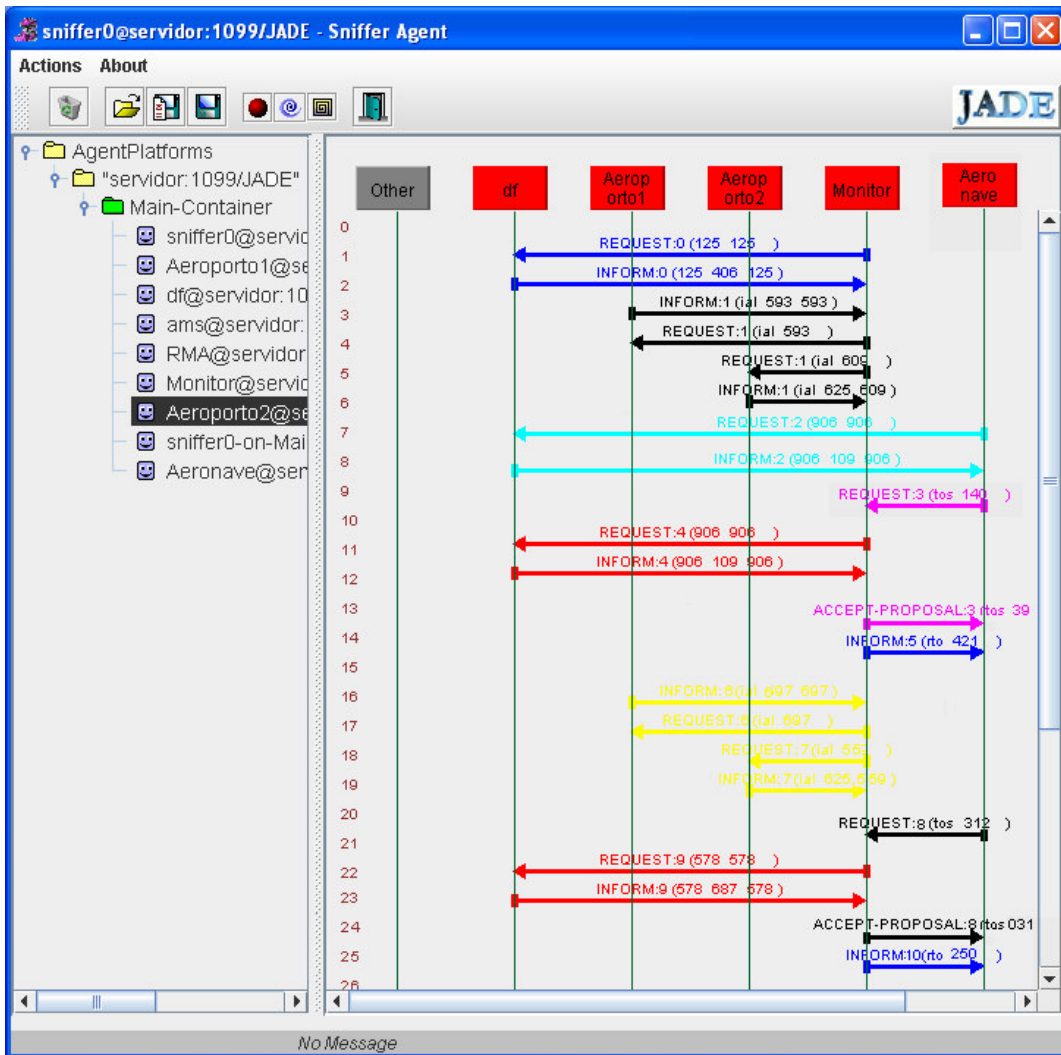


Figura 4.12: Troca de mensagens entre os atores

4.4 CONSIDERAÇÕES FINAIS

Nesse capítulo apresentamos a aplicação prática da nossa proposta de detalhamento da arquitetura organizacional através de sua representação em UML. Exemplificamos nesse capítulo o desenvolvimento de um sistema multi-agente desde a fase dos requisitos iniciais até sua implementação através de um estudo de caso referente ao sistema de controle de tráfego aéreo. Através dos exemplos mostrados aqui foi possível observar o também o funcionamento dos padrões sociais de maneira mais clara. Como

resultado, conseguimos descrever o padrão monitor com maior precisão e riqueza de detalhes.

5 CONCLUSÃO

Nesse trabalho nós mostramos a importância do paradigma de orientação a agentes além de vislumbrarmos a importância dos sistemas multi-agentes para a resolução de sistemas complexos de software.

Através de um exemplo prático nós mostramos uma aplicação desses sistemas utilizando a metodologia Tropos e a plataforma de desenvolvimento JADE. A metodologia utilizada mostrou-se bastante eficaz, por focar no ambiente organizacional do problema, diminuindo assim o distanciamento existente entre o sistema e seu ambiente.

A agregação dos conceitos resultantes da análise realizada com a metodologia Tropos e a plataforma de desenvolvimento JADE, mostrou-se bastante eficiente. Entretanto, o desempenho e execução do sistema gerado na plataforma JADE ainda deixa muitas expectativas não alcançadas.

Para trabalhos futuros, vemos a completude do sistema de controle de tráfego aéreo mostrado aqui e uma maior especulação no sistema da própria aeronave, a qual possibilita a aplicação eficiente de um sistema multi-agente.

REFERÊNCIAS BIBLIOGRÁFICAS

- [Alamillo03] Alamillo, T., Alsinet, T., Béjar, R., Ansótegui, C., Fernández, C., Manyà, F.: **Automated monitoring of medical protocols: a secure and distributed architecture**. Eds.: A. Moreno, C. Garbay. (2003) (in press)
- [Aldea01] Aldea, A., López, B. Moreno, A., Riaño, D., Valls, A. **A Multi-Agent System for Organ Transplant Co-ordination**. In Artificial Intelligence in Medicine. Eds: S. Quaglini P. Barahona, S. Andreassen. Lecture Notes in Computer Science 2101, Springer Verlag, 413-416. (2001)
- [Agentlink03] The Agentcities-Agentlink: **Agents for Commercial Applications**. Agent Technology Conference. Barcelona, Spain. (2003)
- [Baujard98] Baujard, O., Baujard, V., Aurel, S., Boyer, C., Appel, R. D.: **MARVIN, a multi-agent softbot to retrieve multilingual medical information on the Web**. Medical Informatics 23, 187-191, Taylor & Francis. London. (1998)
- [Bellifemine05] Bellifemine, F., Caire, G., Trucco, T., Rimassa, G.: **Jade Programmer's Guide – JADE 3.3**.
<http://sharon.cselt.it/projects/jade/>, último acesso em março de 2005.
- [Bergenti00] Bergenti, F., Poggi, A.: **Exploiting UML in the Design of Multi-Agent Systems**. In Engineering Societies in the Agent Word, First Internacional Workshop – ESAW2000 (160-113), Berlin, Germany, August (2000)

- [Braubach04] Braubach, L., Pokahr, A. and Lamersdorf, W.: Jadex: A Short Overview, In: Main Conference Net.ObjectDays 2004, AgentExpo. (2004)
- [Caire01] Caire, G. , Leal, F. Chainho, P., Evans, R., Jorge, F. G., Juan Pavon, G., Kearney, P., Stark, J., Massonet, P.: **Agent Oriented Analysis using MESSAGE/UML**. In Proceedings of the 2nd International Workshop on Agent-Oriented Software Engineering (AOSE 2001), Montreal. (2001)
- [Castro01] J. Castro, M. Kolp and J. Mylopoulos. **A requirements-driven development methodology**. In Proc. of the 13th Int. Conf. on Advanced Information Systems Engineering, CAiSE'01, pages 108–123, Interlaken, Switzerland. (2001)
- [Castro02] Castro, J., Kolp, M., Mylopoulos, J.: **Towards Requirements-Driven Information Systems Engineering: The Tropos Project**. In Information Systems Journal, Vol. 27:365-389, Elsevier, Amsterdam, The Netherlands. (2002)
- [Decker98] Decker, K., Li, J.: **Coordinated hospital patient scheduling**. Proceedings of the 3rd International Conference on Multi-Agent Systems, ICMAS-98. Paris, France. (1998)
- [DeLoach01] DeLoach, S.: **Analysis and Design using MaSE and agentTool**. Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference – MAICS. (2001)
- [FIPA-OS05] FIPA-OS: **A component-based toolkit enabling rapid development of FIPA compliant agents**. Disponível em: <http://fipa-os.sourceforge.net/>, 2005.

- [Godo03] Godo, L., Puyol-Gruart, J., Sabater, J., Torra, V.: **A multi-agent system approach for monitoring the prescription of restricted use antibiotics**. Eds: A. Moreno, C. Garbay. (2003) (in press).
- [ICAO05] **ICAO – International Civil Aviation Organization**. Disponível em: <http://www.icao.int/> (2005)
- [JACK05] **JACK Intelligent Agents**. Disponível em: <http://www.agent-software.com/> (2005)
- [JADE04] **JADE (Java Agent Development Framework)**, Disponível em: <http://jade.cselt.it/> (2004)
- [Jennings96a] Jennings, N. R., Faratin P., Johnson M., O'brien P., Wiegand M. E. : **Using Intelligent Agents to Manage Business Process**. In: Proceedings of the International Conference on Practical Applications of Ontelligent Agens and Multiagent Technology Anais. P. 345-360, London, UK. (1996)
- [Jennings96b] Jennings, N. R.: **Foundation of Distributed Artificial Intelligence**. Coordination Techniques for DAI. In O'HARE, Greg; Jennings, N. (Eds.). [S. I.]: John Wiley and Sons, cap. 6. (1996)
- [Jennings00] Jennings, N. R.: **On Agent-Based Software Engineering. Artificial Intelligence**. 117 (2) 277-296. (2000)
- [Jennings03a] Jennings, N. R.: **Applying Agent Technology**. Proceedings of the 5th European Agent Systems Spring School. EASSS 2003. Barcelona, Spain. February (2003)
- [Jennings03b] Jennings, N. R., Bussmann, S.: Agent-based control systems. IEEE Control Systems Magazine (2003)(to appear)

- [Kolp01] Kolp, M., Castro, J., Mylopoulos, J.: **A social organization perspective on software architectures**. In Proc. of the 1st Int. Workshop From Software Requirements to Architectures, STRAW'01, pages 5–12, Toronto, Canada. (2001)
- [Kostkova02] Kostkova, P., Mani-Saada, J., Weinberg, J.: **Agent-based up-to-date data management in the National Electronic Library for Communicable Disease**. At the 15th Conference on Artificial Intelligence, ECAI., 59-63. France. (2002)
- [Kotonya97] Kotonya, G., Sommerville, I.: **Requirements engineering – Processes and Techniques**. Chichester, John Willy & Sons, (1997)
- [Moreno2002] Moreno, A., Isern, D.: **Accessing distributed health-care service through smart agents**. Proceedings of the 4th IEEE International Workshop on Enterprise Networking and Computing in the Health Care Industry (HealthCom 2002), 34-41. Nancy, France. (2002)
- [Nealon2002] Nealon, J., Moreno, A. **The Application of agent technology to health care**. Proceedings of the Workshop “AgentCities: research in large scale open agent environments, in the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002) 169-173. Medical Applications of Multi-Agent Systems. Bolonga, Italy. (2002)
- [Odell99] Odell, J.: Objects and Agents: How do they differ?, working paper v2.2, September (1999)

- [Odell00] Odell, J., Parunak, H. V. D., Bauer, B.: **Extending UML for Agents**. Proc. of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence, Gerd Wagner, Yves Lesperance, and Eric Yu eds., Austin, TX, pp. 3-17accepted paper, AOIS Workshop at AAAI 2000. (2000)
- [Odell01] Odell, J., Prunak, H. V. D., Bauer, B.: **Representing Agent Interaction Protocols in UML**. Agent-Oriented Software Engineering, Paolo Ciancarini and Michael Woodridge eds. (121-140), Springer-Verlag, Berlin (Held at the 22nd Internacional Conference on Software Engineering (ISCE)). (2001)
- [O'hare96] O'hare, G., Jennings, N. R.: **Foundations of Distributed Artificial Intelligence**. [S. I.]: John Wiley and Sons (1996)
- [Pressman95] Pressman, R.: **Engenharia de Software**. São Paulo: Makron Books. (1995)
- [Silva03] Silva, C. T. L. L.: **Detalhando o Projeto Arquitetural no Desenvolvimento de Software Orientado a Agentes: O Caso Tropos**. Dissertação de Mestrado. Brasil. (2003)
- [Vázquez03] Vázquez-Salceda, J., Padget, J. A., Cortés, U., López-Navidad, A., Caballero, F.: **Formalizing an electronic institution for the distribution of human tissues**. Eds: A. Moreno, C. Garbay. (2003)
- [Weiss99] Weiss, G.: **Multiagent systems: A modern approach to Distributed Artificial Intelligence**. M.I.T. Press. (1999)

- [Wooldridge00] Wooldridge, M., Jennings, N. R., Kinny, D.: **The Gaia Methodology for Agent-Oriented Analysis and Design**, Journal of Autonomous Agents and Multi-Agent Systems 3 (3): 285-312. (2000)
- [Wooldridge02] Wooldridge, M. **An introduction to MultiAgent Systems**. Wiley Ed. (2002)
- [WQoS03] Projeto WQoS, Núcleo de JADE (Java Agent Development Framework). Disponível em <http://qos.tecnolink.com.br> (2003)
- [Yu01] Yu, E.: **Agent Orientation as a Modelling Paradigm**, Wirtschaftsinformatik 43 (2): 123-132, (2001)
- [Zambonelli00] Zambonelli, F., Jennings, N. R., Wooldridge, M.: **Organizational Abstractions for the Analysis and Design of Multi-Agent Systems**. In Proceedings of The 1st International Workshop on Agent-Oriented Software Engineering, Anais, p. 127-141. Limerick, Irland (2000)

