CEETEPS – Programa de Pós-Graduação

Centro Estadual de Educação Tecnológica Paula Souza

Mestrado em Gestão e Desenvolvimento de Tecnologias da Informação Aplicadas

EDUARDO ENDO

ARQUITETURA DE AGENTES RACIONAIS UTILIZANDO A TECNOLOGIA ADAPTATIVA

SÃO PAULO

08-2010

EDUARDO ENDO

ARQUITETURA DE AGENTES RACIONAIS UTILIZANDO A TECNOLOGIA ADAPTATIVA

Dissertação apresentada como exigência parcial para obtenção do Título de Mestre em Tecnologia no Centro Estadual de Educação Tecnológica Paula Souza, no Programa de Mestrado em Tecnologia: Gestão e Desenvolvimento de Tecnologia Aplicada, sob orientação do Prof. Dr. Maurício Amaral de Almeida e co-orientação da Profa. Dra. Márcia Ito.

SÃO PAULO

08-2010

EDUARDO ENDO

ARQUITETURA DE AGENTES RACIONAIS UTILIZANDO A TECNOLOGIA ADAPTATIVA

PROF. DR. MAURÍCIO AMARAL DE ALMEIDA

PROFA. DRA. MARCIA ITO

PROF. DR. JOÃO JOSÉ NETO

São Paulo, __ de _____ de _____

FICHA ELABORADA PELA BIBLIOTECA NELSON ALVES VIANA
FATEC-SP / CEETEPS

Endo, Eduardo
E56a Arquitetura de agentes racionais utilizando a tecnologia adaptativa / Eduardo Endo. – São Paulo : CEETEPS, 2010.
146 f. : il.

Orientador: Prof. Dr. Maurício Amaral de Almeida.

Co-orientadora: Profª Dra. Márcia Ito.
Dissertação (Mestrado) – Centro Estadual de Educação Tecnológica Paula Souza, 2010.

1. Tecnologia adaptativa. 2. Agentes. 3. Sistemas multiagentes. 4. Racionalização. 5. Tomada de decisão. I. Almeida, Maurício Amaral de. II.Ito, Márcia. III. Centro Estadual de Educação Tecnológica Paula Souza. VI. Título.

Dedicatória

Dedico este trabalho à minha esposa que sempre me apoiou nesta jornada ficando ao meu lado nos momentos mais difíceis e compreendendo a importância deste trabalho para mim.

Aos meus pais que nunca mediram esforços para que eu pudesse ter tido uma boa educação, e assim, me tornar essa pessoa que sou hoje.

Agradecimento

Agradeço principalmente a meu orientador Professor Dr. Maurício Amaral de Almeida e a co-orientadora Professora Dra. Marcia Ito por me apoiarem durante a elaboração do trabalho, mostrando o caminho para que eu pudesse superar minhas fraquezas e abrindo as portas para que eu pudesse mostrar minhas qualidades.

Agradeço ao Professor Dr. João Jose Neto por me apoiar nos direcionamentos em relação a Tecnologia Adaptativa.

Agradeço à Professora Dra. Lucia Christina Iochida e à Professora Msc. Patrícia Albuquerque da UNIFESP pelo apoio no estudo de caso do trabalho.

Agradeço à Professora Msc. Cristiane Ikenaga pelo apoio fundamental na revisão deste trabalho.

Agradeço a minha esposa e a minha família pelo apoio durante esta jornada.

Enfim agradeço a todas as pessoas que direta ou indiretamente tornaram possível a elaboração deste trabalho.

"Só existem dois dias do ano em que nada pode ser feito, o dia de ontem e o dia de amanhã.

Portanto hoje é o dia certo. Sonhe, acredite e principalmente faça."

Dalai Lama

Resumo

ENDO, E. **Implementação da Racionalidade de Agentes utilizando a Tecnologia Adaptativa.** 2010. 146 f. Dissertação (Mestrado) – Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2010.

A utilização de agentes para a resolução de problemas em sistemas aplicados vem sendo difundida cada vez mais por meio de pesquisas científicas. Em uma dessas vertentes está a capacidade de racionalização de agentes que concentra um dos grandes desafios na implementação de um Sistema Multiagentes. A criação deste ambiente racional está relacionada diretamente com os aspectos de tomada de decisão, que tornou viável e pertinente a utilização da tecnologia adaptativa como base dos estudos deste trabalho. Esta viabilidade se tornou evidente pelo fato de que a utilização da tecnologia adaptativa na solução de problemas nas mais diversas áreas do conhecimento humano tem apresentado resultados que podem ser considerados satisfatórios. Desta forma, o objetivo deste estudo é propor a arquitetura de uma solução de racionalização de agentes por meio da Tecnologia Adaptativa. A Tecnologia Adaptativa tem um campo de estudo muito amplo. A abordagem utilizada para a implementação do estudo deste trabalho utilizou apenas dos aspectos de Tabelas de Decisão Adaptativas que teve o papel de servir como ferramenta de tomada de decisão e em relação aos aspectos da implementação de agentes foi utilizado o *framework Jade*. Os resultados obtidos no presente trabalho demonstram que a utilização da Tecnologia Adaptativa como ferramenta de racionalização de agentes é eficaz e viável, e sua utilização foi um diferencial nos resultados obtidos.

Palavras-chave: Agentes, Tecnologia Adaptativa, Sistemas Multiagentes, Racionalização, Tomada de Decisão.

Abstract

ENDO, E. **Implementation of Rational Agents using Adaptive Technology.** 2010. 146 f. Dissertation (Master´s Degree) – Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2010.

The use of agents to solve problems in applied systems is being increasingly disseminated through scientific research. In one of these aspects is the ability for create rational agents that is one of the major challenge in implementing a Multiagent System. Creating this rational environment is directly related to aspects of decision making, which became feasible and appropriate the use of adaptive technology as the basis of studies of this work. This viability became evident by the fact that the use of adaptive technology in solving problems in various areas of human knowledge has shown results that may be considered satisfactory. Thus, the purpose of this study is to propose the architecture of a solution to create rational agents through the Adaptive Technology. The Adaptive Technology has a very broad field of study. The approach used to study the implementation of this work used only those aspects of adaptive decision tables that had the role of serving as a tool for decision making and for aspects of the implementation of agents was used framework Jade. The results of this study show that the use of adaptive technology as a tool for rational agents is effective and feasible, their use make sense in the results obtained.

Keywords: Agents, Adaptive Technology, Multiagent System, Rational, Decision Making.

Lista de Quadros

Lista de Abreviaturas, Siglas e Símbolos

AMS - Agent Management System

AP – Agent Platform

API – Application Programming Interface

BRMS – Business Rules Management System

COOL - Cooperation Language

CORBA – Common Object Request Broker Architecture

DF – Directory Facilitator

DRL – Description Rules Language

EJB – Enterprise Java Beans

FIPA - Foundation for Intelligent Physical Agents

IA – Inteligência Artificial

IAD -Inteligência Artificial Distribuída

IEEE - Institute of Eletrical and Eletronics Engineers

JADE – Java Agent Development Framework

KQML - Knowledge Query Manipulation Language

KSE - Knowledge Sharing Effort

LGPL - Lesser General Public License

MTS – Message Transport Service

RDF – Resource Description Framework

SDP - Solução Distribuída de Problemas

SMA – Sistemas Multiagentes

SOA – Service Oriented Architecture

TDA – Tabela de Decisão Adaptativa

UNIFESP – Universidade Federal de São Paulo

XML - Extensible Markup Language

XSD - XML Schema Definition

Lista de Figuras

Lista de Gráficos

SUMÁRIO

1. INTRODUÇÃO

Os estudos relacionados às características do ser humano vêm desde a época renascentista, em que o homem era visto como o centro do universo. Apesar dessa época estar em um passado um tanto distante, a busca da criação de máquinas que pudessem criar cópias das características humanas é algo que nunca deixou de existir. Nos dias atuais vem crescendo uma vertente de estudo que deixa de lado as características físicas do seres humanos e converte seus esforços em aspectos relacionados à inteligência e à habilidade de adaptação, qualidades essas que fazem do ser humano, único perante todos os outros (NILSSON, 1998).

O desafio de entender a inteligência humana com o objetivo de mapeá-la e tornar possível sua reprodução é um assunto que não fica restrito apenas às áreas das ciências humanas, uma vez que diversos estudos das áreas de exatas têm como objetivo este mesmo contexto. Nas áreas de Engenharia e das Ciências da Computação, a linha de estudo responsável pelos assuntos relacionados à inteligência humana e organização do conhecimento é a Inteligência Artificial e todas as vertentes que a compõem (RUSSEL; NORVIG, 2003).

Em relação à definição do conceito de Inteligência Artificial, não existe um consenso dentre todos os estudiosos, pois suas definições estão estritamente relacionadas às suas respectivas áreas de conhecimento e o ponto de vista em que são objeto de estudo. Em uma definição genérica e simplificada, pode-se definir Inteligência Artificial como sendo uma área de pesquisa composta por conceitos multidisciplinares, responsáveis por investigar diferentes formas de habilitar o computador a realizar tarefas que, até o momento, o ser humano apresenta desempenho superior (MITCHELL, 1997; RUSSEL; NORVIG, 2003).

A contribuição de diversas áreas do conhecimento humano, tais como Biologia, Psicologia, Filosofia e a Lingüística, além obviamente das áreas de Engenharia, Ciências da Computação e Lógica Matemática, torna a natureza dos estudos relacionados à Inteligência Artificial multidisciplinar. Este aspecto é fundamental, visto que para a resolução de problemas deste tipo, somente a utilização das áreas de concentração dos estudos da área de exatas não tem alcance suficiente para pesquisar como funciona o conhecimento humano (RUSSEL;NORVIG, 2003).

Com o crescimento e amadurecimento das áreas de Engenharia e Ciências da Computação em conjunto com uma demanda crescente por soluções cada vez mais eficazes nos problemas relacionados à utilização de inteligência para sua resolução, tornou-se necessário um novo paradigma de computação, que deixou o processamento centralizado para um novo modelo de processamento distribuído. Essa mudança não se restringiu apenas aos aspectos clássicos da computação, mas também à área de Inteligência Artificial (IA), fomentando assim, o surgimento da Inteligência Artificial Distribuída (IAD) (COELHO; PAIVA, 1998) e, posteriormente, dos Sistemas Multiagentes (SMA).

Ao fazer uma comparação entre a Inteligência Artificial Clássica e a Inteligência Artificial Distribuída, pode-ser afirmar que as primeiras pesquisas sobre modelos de inteligência se baseiam no *comportamento individual humano,* que representa o conhecimento e métodos de inferência, enquanto que a segunda, baseia-se no *comportamento social* com enfoque nas ações e interações entre as entidades que compõem seu conjunto (SICHMAN; DEMAZEAU, 1992).

Proveniente dos estudos de IAD, um SMA é composto por entidades artificiais (agentes) que apresentam cada qual sua autonomia e capacidade de socialização, negociação, cooperação e competição entre si, refletindo como funciona a sociedade humana (WOOLDRIDGE, 2002). Pela maneira semelhante como funciona um SMA e uma sociedade

humana, é possível aplicar conceitos estudados nas mais diversas áreas, que não estão necessariamente relacionadas ao estudo de computação, tais como Psicologia e Ciências Sociais, vislumbrando a criação de um ambiente de interação entre os agentes, que os torna capazes de coordenar ações e negociações em caso de conflitos (WOOLDRIDGE, 2002).

Ao aplicar todos esses conceitos, um agente deve ter a capacidade de representar os outros agentes e a sua respectiva organização, além de explorar tais representações. Por explorar, entende-se utilizar tais representações em seus mecanismos de raciocínio e decisão como, por exemplo, decidindo quais objetivos devem alcançar, quais ações devem tomar, negando-se a realizar uma ação que não esteja habilitado na organização ou enviando um pedido de auxílio a um outro agente capaz de realizar a ação desejada (ANTOINE; BAUJARD; BOISSER *et al*., 1992).

A capacidade de racionalização dos agentes, principalmente nos aspectos de tomada de decisão, é um dos grandes desafios na implementação de um SMA, em que a responsabilidade de criar este ambiente racional fica a cargo de quem projeta a solução (FIPA, 2010). Nesta realidade, utilizar a tecnologia adaptativa para criar um ambiente racional em um SMA é pertinente.

A utilização da tecnologia adaptativa na solução de problemas nas mais diversas áreas do conhecimento humano tem apresentado resultados que podem ser considerados satisfatórios. Tais resultados são, em sua maioria, compatíveis com o resultado de técnicas clássicas de aprendizagem de máquina, apresentando, além disso, características desejáveis para o desenvolvimento da racionalização de agentes, visto sua relativa simplicidade na manipulação do dispositivo, seus conceitos maduros, e que algumas de suas vertentes já têm aplicações em um contexto real (PISTORI, 2003).

As pesquisas sobre Tecnologia Adaptativa têm seus estudos direcionados aos mais diversos campos do desenvolvimento científico e essa abrangência tem o objetivo de buscar

novos modelos de computação que possam atender os mais diferentes níveis de complexidade de problemas (PISTORI, 2003). Neste sentido, este trabalho faz uso de tabelas de decisões adaptativas, que é uma das linhas de pesquisa da Tecnologia Adaptativa para servir de apoio à criação do contexto racional.

Oferecer uma alternativa de implementação de todos esses aspectos de inteligência aos agentes é o problema central e o objeto de estudo nesta pesquisa. De uma maneira mais detalhada, pode-se afirmar ainda, que é encontrar alternativas viáveis e práticas para a questão de racionalização de agentes, ou seja, pretende-se obter um ambiente racional para as atividades de tomada de decisões dos agentes, cuja complexidade de programação dessas ações de racionalização seja simples e intuitiva, e que apresente comportamento similar aos clássicos, utilizados em inteligência de máquina.

1.1 OBJETIVO DO TRABALHO

O objetivo deste trabalho é propor uma arquitetura para solução de racionalização de agentes por meio da Tecnologia Adaptativa.

Neste trabalho, entende-se como racionalização, o ato de usar a razão para fazer uma dedução sobre uma determinada necessidade de tomada de decisão, na tentativa de estabelecer um paralelo com as faculdades que tem o homem em relação a este assunto.

1.2 MOTIVAÇÕES E CONTRIBUIÇÕES DO ESTUDO

Existem diversas pesquisas relacionadas à utilização da Tecnologia Adaptativa para a construção do conhecimento e os resultados obtidos nessas pesquisas são bastante animadores. Dentre elas, pode-se destacar sua aplicação nas áreas de processamento digital de imagens e interface homem-máquina através de reconhecimento de sinais e olhar (PISTORI, 2003). Neste mesmo sentido, a difusão de conhecimento relacionada aos sistemas multiagentes, principalmente em aplicações reais, já é uma realidade, portanto, utilizar a Tecnologia Adaptativa para inserção de uma racionalização em agentes pode ser considerada uma estratégia bem direcionada e serve de motivação para este trabalho.

Um dos fatores que influenciou a escolha pela Tabela de Decisão Adaptativa para este trabalho foi o fato de que a mesma utiliza como dispositivo subjacente uma Tabela de Decisão Convencional, que tem uma grande disponibilidade de recursos computacionais já estudados, com diversas implementações comerciais e acadêmicas já consolidadas, sendo que este aspecto será aproveitado, visto que o presente trabalho adiciona aspectos adaptativos a um dispositivo convencional já implementado. Além deste aspecto, outro que foi fundamental, é o aspecto intuitivo e de fácil manipulação, uma vez que o presente estudo pretende provar sua aplicabilidade em um cenário onde o conhecimento computacional não é fator fundamental.

Ainda neste cenário, visto que a área de racionalização possui um amplo campo de atuação, neste trabalho o foco foi dado principalmente na resolução de problemas de tomada de decisão.

1.3 MÉTODO E DESENVOLVIMENTO DO TRABALHO

A pesquisa se deu inicialmente pelo estudo dos conceitos de agentes e sistemas multiagentes com a finalidade de entender e organizar os processos de racionalização. Foram estudados os aspectos dos agentes relacionados à sua organização, arquitetura, representações e funcionamento, além dos aspectos relacionados à sua racionalização.

Após a conceituação de agentes, efetuou-se uma pesquisa sobre a Tecnologia Adaptativa, passando pelos conceitos que a compõe e suas diversas variações de implementações. Essa pesquisa teve como objetivo principal identificar quais as características da Tecnologia Adaptativa que melhor atendem as necessidades para a criação de um ambiente de racionalização.

Após os estudos anteriores, optou-se por pesquisar ferramentas, métodos, conceitos e técnicas relacionadas à tomada de decisão, uma vez que a implementação escolhida (Tabela de Decisão Adaptativa (TDA)) faz uso diretamente de conceitos fundamentais desse assunto.

Com o objetivo de consolidar os conceitos necessários para o desenvolvimento do trabalho, uma implementação em formato de estudo de caso foi desenvolvida.

Em relação ao estudo de caso, no qual os conceitos foram aplicados, o mesmo teve como objetivo, resolver um problema de priorização de atendimento a pacientes com problemas nefrológicos. Após as simulações terem sido realizadas, foram feitas análises dos resultados obtidos e considerações sobre a implementação criada, levando em consideração o modelo de criação de tabelas de decisão adaptativa junto à arquitetura de agentes.

1.4 ORGANIZAÇÃO DO TRABALHO

Além do presente capítulo que trata da introdução ao tema estudado, o trabalho está dividido em duas partes, sendo a primeira relacionada aos capítulos que abordam os principais conceitos e fundamentos do trabalho, e a segunda, relacionada aos capítulos que apresentam a proposta de implementação da solução de racionalização de agentes.

A primeira parte está composta por um capítulo que trata dos conceitos de agentes e sistemas multiagentes, um capitulo que trata da *Tecnologia Adaptativa* e, por último, um capítulo responsável por tratar da estratégia de racionalização de agentes com a *Tecnologia Adaptativa*.

A segunda parte do trabalho compreende um capítulo sobre a arquitetura do *framework* Jade, outro sobre a arquitetura do *framework* Drools e ainda a abordagem sobre a camada adaptativa para os propósitos deste trabalho. Compreende ainda o capítulo que apresenta o desenvolvimento da racionalização de agentes por meio da Tecnologia Adaptativa no contexto da área de Saúde.

Após estas duas principais partes, o trabalho apresenta as Conclusões bem como algumas sugestões para estudos futuros e as Referências de trabalhos e estudos sobre o tema pesquisado.

Por fim, o trabalho apresenta os Apêndices que contém informações adicionais sobre alguns conceitos discutidos no trabalho.

2 AGENTES E SISTEMAS MULTIAGENTES

Por séculos, a humanidade tem se preocupado no campo da ficção e muitas vezes no campo da engenharia em criar agentes artificiais que pudessem imitar o comportamento humano. Diversos dispositivos têm sido criados como frutos dessa necessidade: secretárias eletrônicas, bonecas que riem, instrumentos que tocam sozinhos. Durante quase todo o século XX, andróides, humanóides, robôs, *cyborgs* e computadores inteligentes foram personagens da ficção cientifica, vistos como criaturas que existiriam em um futuro não tão distante (BRADSHAW, 1996).

O personagem mais famoso da tela do cinema que representa bem este anseio do ser humano, o computador HAL 9000, do famoso filme de ficção científica de 1968, *'2001, uma odisséia no espaço'*, fazia a previsão de que um dia os computadores seriam inteligentes o suficiente a ponto de sentirem emoções e terem vontades próprias (CLARKE, 1972). Muitos anos antes, o matemático Alan Turing, precursor da Ciência da Computação, já propunha o Teste de Turing, em que as máquinas poderiam ser consideradas inteligentes se um ser humano não pudesse distinguir o comportamento de uma máquina do comportamento de outro ser humano (TURING, 1950).

O entusiasmo com a possibilidade de criar máquinas inteligentes fez pesquisadores do campo da Inteligência Artificial se reunirem na década de 1950 com o objetivo de determinar quais seriam as características necessárias para reproduzir a inteligência humana em computadores. Nesta época os computadores ainda estavam muito longe de oferecer o hardware necessário para implementar todas as idéias surgidas neste encontro que, anos mais tarde, surgiriam como técnicas de representação e manipulação do conhecimento (RUSSEL; NORVIG, 2003).

Os precursores iniciais da Inteligência Artificial não conseguiram alcançar todos os resultados esperados na pesquisa de máquinas inteligentes, que imitavam o comportamento humano, e apesar de todos os avanços na área realizados até hoje, o ano 2010 chegou sem que tivéssemos algo como o HAL 9000. E também, há algumas décadas, os esforços de pesquisa na área da Inteligência Artificial se voltaram para outros problemas mais específicos, em que os resultados tivessem mais aplicabilidade para a sociedade.

Novas propostas relacionadas à computação autônoma (sistemas que possuem a capacidade de se autogerenciar) começaram a surgir na década de 1990, quando uma das idéias relacionadas que passou a ser divulgada foi o termo *agente* como sendo uma entidade autônoma, tal qual um cérebro de um robô, capaz de processar e tomar decisões a partir de informações e agir em busca da solução de um objetivo (NEGROPONTE, 1995).

2.1 HISTÓRICO DE AGENTES

Até a década de 1980, os estudos relacionados à inteligência artificial trataram tradicionalmente de sistemas individuais, tentando imitar nesses sistemas as características específicas à inteligência humana, tais como a resolução de problemas e aprendizagem. Em decorrência desses estudos, ainda na década de 1980, mostrou-se necessário nos estudos básicos da inteligência, a inclusão de conceitos relacionados a aspectos da interação e distribuição. Surgiu assim, o conceito de que a inteligência é o resultado de inúmeros módulos interagindo entre si, cada um resolvendo tarefas primitivas específicas (MINSKY, 1985).

Esses estudos ganharam um lugar de destaque quando foram integrados com o modelo de objetos interativos, designados por atores, como anteriormente proposto por Hewitt (1973).

Surgiram, assim, as bases para a Inteligência Artificial Distribuída. Esses atores são caracterizados por um estado interno, operando em paralelo e interagindo com outros objetos por meio de mensagens. Este conceito de ator evoluiu para o que hoje é conhecido como agentes.

Do seu conceito inicial para os aspectos dos dias atuais, percebe-se que o conceito de agentes atualmente é utilizado nos mais diferentes contextos. O agente pode ser compreendido em uma perspectiva de um processo de software, executado concorrentemente, que encapsula algum estado e comunica com outros agentes através de mecanismos de troca de mensagens, como ocorre no âmbito da engenharia de software baseada em agentes, ou ainda, como um programa com alguma forma de controle persistente, como acontece em aplicações para a Internet. Esse tipo de caracterização corresponde ao que se designa por noção fraca de agente (WOOLDRIDGE; JENNINGS, 1995).

Em contrapartida, a noção forte de agente é característica de áreas como a Inteligência Artificial, na qual, além das características anteriores, conceitos como conhecimento, crenças, intenções, desejos, obrigações ou emoções, podem ser atribuídos aos agentes (WOOLDRIDGE, 2000; SCHWEITZER, 2003).

É possível perceber que o conceito de agentes é utilizado em diversas áreas com diferentes significados, fazendo assim, com que seja difícil definir a noção de agente de forma consensual (COELHO; PAIVA, 1998). Nesse sentido, o consenso que existe é que, apesar da noção da conceituação de agente fazer-se necessário, esta deve ser utilizada somente como ferramenta para classificar sistemas, de maneira a separar sistemas com características de agentes dos sistemas “tradicionais”, e não, como uma caracterização absoluta do que é agente e não agente (RUSSEL; NORVIG, 2003).

Ainda assim, é possível identificar algumas características gerais que delimitam o conceito de agentes, tais como os que são abordados neste estudo.

Pode-se definir como *Inteligência Artificial Distribuída* (IAD) o ramo da Inteligência Artificial que está relacionado com a solução de problemas por meio do cooperativismo em um determinado ambiente, fazendo o uso de agentes inteligentes. As utilizações destes mecanismos ocorrem nos mais diversos níveis (BITTENCOURT, 1998):

- A utilização de alguns conceitos e idéias da IAD no desenvolvimento de novas aplicações que utilizam metodologias tradicionais de desenvolvimento de software;
- Na melhoria das funcionalidades de sistemas existentes, através da inclusão destas aplicações em plataformas de IAD;
- No desenvolvimento de sistemas que incorporem as técnicas de IAD, da concepção até sua implementação.

A motivação da utilização dos conceitos da IAD baseia-se na idéia de que a flexibilidade, a agilidade, a inteligência e o desempenho de um sistema podem ser melhorados à medida que ela permite alcançar objetivos como (PARUNAK, 1994):

- Possibilidade da construção de sistemas descentralizados ao invés de centralizados;
- Obtenção de soluções que evoluem de forma autônoma (conhecimento resultado das interações entre agentes e/ou humanos) em vez de totalmente planejadas e estáticas;
- Possibilidade da execução de processos concorrentes em vez de seqüenciais.

Ao observar os problemas reais, pode-se perceber que muitos são natural e fisicamente distribuídos, por isso, a necessidade de soluções distribuídas e a exigência da capacidade de adaptação dos sistemas são alguns dos principais motivos que fazem com que o desenvolvimento de sistemas inteligentes distribuídos seja uma boa opção de escolha para a resolução de problemas reais (FERBER, 1999).

A IAD pode ser dividida em dois segmentos em relação aos seus conceitos: *Solução Distribuída de Problemas* (SDP) e *Sistemas Multiagentes* (SMA) (BITTENCOURT, 1998).

Pode-se definir uma SDP em relação a sua característica, tendo seu foco principal no problema. Como principais objetivos, pode-se citar a utilização da capacidade de processamento e a robustez oferecida pelas tecnologias de redes para atacar problemas de natureza distribuída ou muito complexa. Um exemplo clássico que soluções distribuídas de problemas resolvem é o problema do controle de tráfego aéreo (BITTENCOURT, 1999).

Os agentes envolvidos em SDP são programados para cooperar, dividir tarefas e comunicar-se de maneira confiável, entretanto, existe um grande desafio em estabelecer estas propriedades de maneira satisfatória. Desta maneira, fica a cargo dos SMA o estudo das pressuposições básicas sobre agentes que possam garantir todas as características necessárias em relação à cooperação em sociedade dos agentes, com o objetivo de atender as necessidades de uma SDP. De maneira direta, o foco das pesquisas são os agentes e suas interações (FRIGO; POZZEBON; BITTENCOURT, 2004).

Uma vez que os estudos relacionados à Inteligência Artificial são muito extensos, e se o foco deste trabalho estivesse direcionado apenas aos estudos da Inteligência Artificial Distribuída, ainda assim, o escopo seria muito amplo. A figura 1 demonstra como estão organizadas as áreas de atuação da inteligência artificial e fornece uma idéia da amplitude desses campos de atuação. Dessa forma, para o presente trabalho, os estudos estão restritos ao universo dos Sistemas Multiagentes e os aspectos que o compõem.

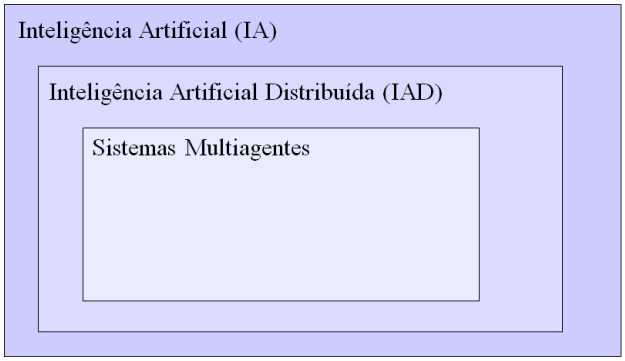

Figura 1- Organização das áreas de atuação da inteligência artificial.
Fonte: FRIGO; POZZEBON; BITTENCOURT (2004).

2.2 CONCEITO DE AGENTES

Pode-se definir agentes como entidades autônomas e sociais utilizadas para desenvolver aplicações complexas. A idéia por trás do conceito de agentes está relacionada com o fato da cooperação e competição que envolve o ambiente dos agentes. Os agentes têm características tais como reativação, autocontrole, orientação a objetivo, execução contínua, aprendizado de comunicação, mobilidade, flexibilidade e habilidade social. Agentes são autônomos porque podem decidir para onde vão e o que podem fazer, além de controlarem seu ciclo de vida. Em relação ao aspecto de comunidade, podem receber mensagens de diversas entidades externas, tais como outros agentes, mas fica a cargo de cada agente decidir se atende ou não tal requisição (GAWALO; MESHRAM, 2009).

Embora não existia um alinhamento universal preciso sobre a definição do termo "agente", muitas das definições concordam em muitos pontos e discordam de poucos outros

(BONABEAU, 2001). Alguns pesquisadores consideram qualquer tipo de componente independente como um agente, sendo que o conceito de componente independente varia desde uma ação reativa baseada em regras de decisão a ações complexas baseadas em inteligência flexível; outros ainda acreditam que o comportamento do componente tem que ser adaptativo para que possa ser considerado um agente (MELLOULI; MINEAU *et al*., 2003).

De acordo com Casti (1997), agentes devem conter comportamento baseado em regras, assim como um conjunto de alto nível de "regras que modificam regras", nos quais as regras de base têm como responsabilidade a interação com o ambiente, enquanto que as "regras que modificam regras" têm como responsabilidade a flexibilidade.

Jennings (2000) fornece uma visão computacional que enfatiza as características essenciais de um comportamento autônomo, o qual está relacionado diretamente ao conceito de agentes, visto que a característica fundamental é a capacidade de tomar decisões independentes, que torna necessário que um agente seja ativo, ao contrário de ser simplesmente passivo.

De um ponto de vista prático, pode-se modelar os agentes como elementos com as seguintes características (JENNINGS, 2000):

- Um agente é identificável, um indivíduo discreto com um conjunto de características e regras que controlam seus comportamentos e capacidades de decisão. Agentes são autocontidos, o que os torna com fronteiras em que facilmente pode-se determinar o que é parte de um agente, o que não é parte e o que são características compartilhadas.
- Um agente vive em um ambiente no qual pode interagir com outros agentes. Agentes têm protocolos para interação com outros agentes, tais como protocolos de comunicação e a capacidade de responder para o ambiente. Agentes têm a habilidade de reconhecer e distinguir outros agentes.

- Um agente é direcionado a tarefas.
- Um agente é autônomo e autodirecionado, pode trabalhar independente ou em conjunto com outros agentes.
- Um agente é flexível e tem a habilidade de aprender e adaptar seus comportamentos através das interações, baseando-se em experiência. Para isso é necessário algum tipo de memória. Um agente pode ter algumas regras que modificam seu comportamento.

Para que o agente possua as características descritas é necessário que algumas funcionalidades façam parte da sua estrutura. A figura 2 representa uma proposta de uma estrutura de um agente e suas funcionalidades (JENNINGS, 2000).

Figura 2 – Estrutura de um agente.
Fonte: JENNINGS (2000).

2.2.1 REGRAS DE COMPORTAMENTO

Em geral, os agentes têm seus atributos e regras de comportamento implementados de diversas maneiras, com o objetivo de criar um ambiente com maior ou menor racionalidade. A seguir são apresentadas algumas estratégias de implementações de comportamento (JENNINGS, 2000):

- Criação de regras de comportamento dinâmicas, nas quais variam sua sofisticação, de acordo com a quantidade de informação que é considerada nos aspectos de decisão dos agentes (aspecto cognitivo).
- Implementação de um modelo de interação do agente com o ambiente e com outros agentes por meio de padrões predefinidos (socialização).
- A utilização de uma memória de extensão de eventos passados que armazena e utiliza as informações para decisões futuras (aprendizado).

Observando em mais detalhes as características apresentadas em relação às regras de comportamento de um agente, pode-se afirmar que a sua implementação está relacionada diretamente à estratégia de construção desses aspectos, portanto, não existe uma regra predefinida para esta construção, ficando a cargo de quem está construindo o agente decidir qual será a melhor solução para sua implementação. Neste trabalho, a estratégia de construção do agente está baseada nos conceitos da Tecnologia Adaptativa, mais especificamente a utilização de um dispositivo adaptativo dirigido por regras, pois este tem a característica de utilizar em sua operação qualquer dispositivo formal que possua um conjunto fixo e finito de regras. Estas regras, no entanto, podem variar de acordo com um outro conjunto de regras, denominadas ações adaptativas, que agindo sobre o conjunto de regras original (através de ações de consulta, inserção, remoção destas regras), permite que um dispositivo altere sua

estrutura interna durante seu funcionamento (PISTORI, 2003). Desta forma, a utilização de um dispositivo adaptativo dirigido por regras em conjunto aos conceitos de agentes consegue auxiliar a criação de regras de comportamento dinâmico (aspecto cognitivo), e ainda, por meio de algumas modificações, auxiliar na criação de uma memória de extensão de eventos passados para decisões futuras (aprendizado). Os conceitos que direcionam a solução da racionalização de agentes em relação à Tecnologia Adaptativa estão detalhados no capítulo 3 deste trabalho.

Visto que as características de um agente estão relacionadas diretamente com sua implementação, surgem inúmeras possibilidades de formas de configurações dos agentes. Apesar disso, pode-se perceber que existem características que fazem com que seja possível efetuar agrupamentos por tipos de agentes.

2.2.2 TIPOS DE AGENTES

Em relação aos tipos de agentes, pode-se classificá-los nas seguintes categorias: persistentes ou temporários, estacionários ou móveis, reativos ou cognitivos (FERBER, 1999).

Agentes persistentes são agentes de software que, quando relacionados a um dado ambiente computacional, não podem ser excluídos do sistema. Agentes temporários são agentes de software que têm uma vida finita, normalmente de duração igual ao tempo de uma dada tarefa, ou seja, tão logo finalizam sua missão, eles são excluídos do sistema, normalmente por eles próprios (ITO, 2009).

Agentes estacionários são agentes de software que, quando relacionados a um dado ambiente computacional, não têm a habilidade de se moverem pela rede para outros computadores. Por outro lado, agentes móveis são agentes de software que podem se mover para outros ambientes através da rede e, quando se movem, levam consigo seus estados internos, ou seja, sua representação mais a memória (RABELO, 2009).

Em relação ao seu modelo de organização e funcionamento os agentes podem ser classificados como reativos ou cognitivos. Os agentes reativos são baseados em modelos de organização biológica ou ecológica, como por exemplo, as sociedades de formigas ou cupins. Uma formiga sozinha não é considerada uma entidade inteligente, mas o formigueiro sim, pois apresenta comportamento inteligente relacionado à busca e armazenamento de alimentos, além da organização dos berçários. O modelo de funcionamento de um agente reativo é o de estímulo-resposta. Em geral, estes agentes não apresentam memória, não planejam suas ações futuras e não se comunicam com outros agentes, tomando conhecimento das ações dos outros agentes pelas mudanças ocorridas no ambiente. Normalmente existem em grande quantidade no sistema e possuem baixa complexidade (BITTENCOURT, 1998).

Os agentes cognitivos são baseados em organizações sociais humanas como grupos, hierarquias e mercados. Esses agentes possuem uma representação explícita do ambiente e dos outros agentes, dispõem de memória e por isto são capazes de planejar ações futuras. Agentes cognitivos podem comunicar-se entre si diretamente, isto é, seus sistemas de percepção e de comunicação são distintos, o que não acontece nos reativos. Normalmente estão em pequena quantidade no sistema e são de média ou alta complexidade (BITTENCOURT, 1998). Além disso, requerem sofisticados mecanismos de coordenação e protocolos de alto nível para suporte à interação (FERBER, 1999).

Com base nas definições apresentadas, é possível afirmar que os agentes cognitivos aumentam a qualidade do sistema sob o ponto de vista do conhecimento porque permitem

gerar um sistema mais perceptivo com autonomia, flexibilidade e colaboração, o que torna essa categoria como a ideal para a utilização neste trabalho.

O quadro 1 mostra de uma maneira resumida os tipos de agentes e suas características.

Quadro 1 – Tipos de Agentes.

Tipo	**Descrição**
Estacionários	Não têm a capacidade de se moverem pela rede.
Móveis	Têm a capacidade de se moverem para outros computadores ou ambientes pela rede.
Persistentes	Não podem ser excluídos do sistema.
Temporários	Têm vida finita. Normalmente após concluir sua tarefa são excluídos do sistema.
Reativos	Baseados em estímulo-resposta; não trabalham em organizações.
Cognitivos	Baseados em organizações.

2.3 SISTEMA MULTIAGENTES

Um Sistema Multiagentes (SMA) pode ser definido como um grupo de entidades autônomas interativas compartilhando um ambiente comum (WEISS, 1999). Aplicações que utilizam sistemas multiagentes podem ser encontradas em uma vasta variedade de finalidades incluindo robótica, controles distribuídos, gerenciamento de recursos, sistemas de suporte a decisão colaborativo, *data mining,* dentre outros. (PARUNAK; WEISS, 1999; STONE; VELOSO, 2000).

As características de um Sistema Multiagente tornam-o mais próximo dos aspectos humanos em relação aos outros sistemas, ou ainda pode prover uma perspectiva alternativa

em sistemas que originalmente são centralizados, por exemplo, em times de robôs, em que o controle de autoridade é naturalmente distribuído entre cada robô, ou ainda em gerenciamento de recursos, no qual a utilização de cada recurso em conjunto com um agente pode prover uma ajuda na criação de uma perspectiva de sistema distribuído (STONE; VELOSO, 2000; CRITES; BARTO, 1998).

Os agentes em um sistema multiagentes podem ser programados com comportamentos complexos e freqüentemente faz-se necessário que eles aprendam também novos comportamentos *online*, de tal maneira que o agente ou todo o sistema multiagentes gradualmente se aperfeiçoe. Essa necessidade se torna corriqueira, uma vez que a complexidade do ambiente pode tornar a criação do comportamento de um agente um processo extremamente difícil, ou até mesmo impossível (SEN; WEISS, 1999).

2.3.1 ESTRATÉGIAS DE RESOLUÇÃO DE PROBLEMAS EM SISTEMAS MULTIAGENTES

Em relação à abordagem de resolução de problemas, um SMA tem uma característica diferenciada que é a filosofia de resolução distribuída de problemas, que consiste em adotar uma estratégia de dividir para conquistar. Esta estratégia consiste em dividir o problema em subproblemas de tal maneira que cada um pode ser direcionado para um agente específico, cada um destes comunicando ou cooperando entre si quando necessário, utilizando assim, a idéia básica de que a soma de resultados de cada um dos agentes corresponderá à solução do problema geral. Este resultado é obtido por meio da camada de cooperação, que é uma

camada encontrada apenas em sistemas baseados em agentes, nos quais a cooperação reflete uma estratégia de ação decidida pelo agente, permitindo uma negociação (JENNINGS, 1994).

Em relação aos tipos de cooperação, existem dois tipos: partilha de informação ou partilha de tarefas. A partilha de informação se dá quando um dado agente dispõe ou produz informações parciais que julga serem úteis a partilhar e as envia para os outros agentes. A partilha de tarefa é efetuada quando um dado agente, ao decompor uma dada tarefa, detecta subtarefas que não pode ou não quer realizar, sendo necessário procurar outros agentes que possam auxiliá-lo (DURFEE, 1987).

Em um projeto que utiliza conceitos de um SMA devem ser considerados vários aspectos, como por exemplo, tipo ou classe do problema a ser tratado, níveis de autonomia dos agentes, representação do conhecimento, protocolos de comunicação, definição da organização do sistema, modelagem de dados, modelos de coordenação, cooperação e resolução de conflitos (RABELO, 2009).

2.3.2 SOCIEDADE DE SISTEMAS MULTIAGENTES

As sociedades de sistemas multiagentes se diferenciam quanto à possibilidade de comunicação e a complexidade do agente. Para a realização de determinada tarefa, é necessário que seja adotada uma política de cooperação, de modo que seja possível ao agente demonstrar as suas necessidades aos demais. Este compartilhamento pode ocorrer de duas maneiras: por meio do compartilhamento de tarefas, quando o agente tem uma tarefa a cumprir e necessita da ajuda de outros agentes da sociedade ou por meio de compartilhamento

de resultados, em que os agentes disponibilizam informações para a sociedade, de tal maneira que outros agentes podem fazer uso de tal informação (FARACO, 1998).

A seguir são apresentadas algumas arquiteturas de SMA, desde o precursor de todas elas, o Sistema Quadro-Negro, até os atuais Sistemas Multiagentes Abertos.

De acordo com Ferber (1999), em relação à classificação de sua arquitetura, um SMA pode ser dos seguintes tipos: democrática, federada ou aberta. Aquém dessa classificação existe a arquitetura do Sistema Quadro-Negro (*BlackBoard System*), precursora dos sistemas multiagentes, que se caracteriza pelo fato de não existir comunicação direta entre os agentes. O "Quadro-Negro" é uma estrutura de dados persistente, dividida em regiões e níveis, visando servir de meio para busca de informações. Para fazer suas interações, os agentes fazem uso desta estrutura, na qual podem ler ou escrever informações que são compartilhadas para todo o grupo, de acordo com a necessidade da sociedade, de maneira que esta estrutura se torna uma memória de compartilhamento (FARACO, 1998).

Nesta organização, quando um agente necessita realizar uma tarefa mas não possui as habilidades necessárias, coloca uma "requisição" no "quadro-negro", no qual outro agente que possui as habilidades necessárias pode retirar o pedido, processar e devolver o resultado de volta para o "quadro-negro", de maneira que o agente requisitante possa resgatar o resultado (COSTA, 2001).

O fato dos agentes não necessitarem se conhecer mutuamente é a principal vantagem desta arquitetura, porém, como desvantagem decorre o fato dos agentes não possuírem capacidade de comunicação entre si, ficando dependendo do "quadro-negro" para assegurar a percepção e a comunicação entre eles.

Na organização democrática, os agentes possuem o mesmo nível hierárquico. As ações coletivas, como a comunicação, são realizadas de forma assíncrona. Para que a comunicação seja feita de maneira eficiente deve-se obedecer algumas regras de comunicação de agentes,

tais como a KQML (*Knowledge Query Manipulation Language*), a CooL (*Cooperation Language*) ou a Parla. Esta organização possui como vantagem a flexibilidade e a modularidade, e como desvantagem, a necessidade de que o agente precisa conhecer a identidade e as habilidades de outros agentes da sociedade (COSTA, 1997).

Na organização federada, os agentes da sociedade são divididos em grupos ou federações, de acordo com critérios predefinidos. Em cada grupo existe a presença de agentes facilitadores e agentes realizadores. Os agentes facilitadores possuem o conhecimento das habilidades dos agentes realizadores. Seu papel consiste em organizar o trabalho entre eles, por meio do recebimento das mensagens e pelo encaminhamento ao respectivo agente destinatário, o qual possui as habilidades necessárias para realizar a tarefa (COSTA, 2001).

A vantagem deste tipo de abordagem está no reduzido número de mensagens desnecessárias, resultado da comunicação entre os agentes, uma vez que os facilitadores têm a habilidade de identificar quem deve receber a mensagem, tornando assim, a comunicação mais eficiente (MARTIAL, 1992).

Como desvantagem, o fato de toda comunicação passar pelo agente facilitador torna o sistema sensível, uma vez que, caso ocorra um mau funcionamento do agente facilitador, a organização toda é penalizada.

Na organização federada existe algum tipo de hierarquia. Há a presença de agentes facilitadores, agentes intermediários entre o cliente e o supervisor. Grupos de agentes da federação têm graus similares de independência e autonomia, e normalmente são bastante heterogêneos. Os procedimentos de coordenação são de média complexidade, dada à própria existência do facilitador. Entretanto, as arquiteturas hierárquicas são as mais utilizadas em aplicações industriais, em que o sistema multiagentes está estruturado em níveis de hierarquia (COSTA, 2001).

Na organização aberta, os agentes que compõem a sociedade não são fixos, pois é permitido que os agentes possam se subscrever e desligar-se da comunidade de maneira dinâmica. Deste modo, os agentes podem se adaptar, escolhendo diferentes metas, planos de execução ou padrões de cooperação. A vantagem desta abordagem é a robustez e a sua desvantagem é a excessiva troca de mensagens realizada por um complexo protocolo de comunicação (COSTA, 2001).

2.3.3 COORDENAÇÃO ENTRE AGENTES EM UM SISTEMA MULTIAGENTES

De acordo com Weiss (1999), a coordenação é uma característica fundamental para um sistema de agentes que executa alguma atividade em um ambiente compartilhado.

A coordenação está diretamente relacionada com o compartilhamento de conhecimento entre os agentes, e o seu principal objetivo é coordenar as ações individuais de cada agente para atingir o objetivo final de um SMA. A coordenação entre agentes se torna ainda mais importante pelo fato de que um só agente dentro de um SMA não tem informação ou capacidade suficiente para resolver muitos dos problemas, visto que muitos desses objetivos não podem ser atingidos por agentes agindo isoladamente (WEISS, 1999).

Desta maneira, a coordenação é a capacidade dos agentes de trabalharem em conjunto e combinar seus objetivos de forma a concluírem o objetivo final do problema. Uma das formas, para uma cooperação ser bem sucedida, cada agente deve manter um "modelo" do outro agente e também desenvolver um modelo de interações futuras ou possíveis, nas quais as futuras podem ser divididas em cooperação e negociação (WEISS, 1999).

De acordo com Weiss (1999), a negociação é a coordenação entre agentes antagônicos ou simplesmente egoístas (*self-interested*), ou seja, a negociação está relacionada à coordenação de agentes competitivos. Geralmente são usados protocolos de negociação para determinar as regras de negociação e são definidos conjuntos de atributos sobre os quais se pretende chegar a um acordo.

E a cooperação é a coordenação entre agentes não antagônicos (WEISS, 1999), ou seja, uma coordenação com agentes que não possuem objetivos conflitantes geralmente é chamada de cooperação. Neste caso, agentes cooperativos auxiliam-se mutuamente para um objetivo comum. Fica assim, a cargo da coordenação, controlar o conjunto de agentes de tal maneira que esses realizem suas tarefas como um "trabalho de equipe", visando sempre realizar o objetivo.

2.3.4 COMUNICAÇÃO DE SISTEMAS MULTIAGENTES

A maneira como ocorre a interação entre os agentes é um fator determinante na sua integração. A capacidade de comunicação de cada agente está estritamente relacionada à sua linguagem de comunicação e sua expressividade. Estes conceitos devem ser universais e partilhados por todos os agentes, ser concisos e ter um número limitado de padrões de comunicação.

Como modelo de comunicação de agentes são utilizadas a comunicação humana e a teoria dos atos comunicativos (*Speech ActTheory*) (MENESES, 2001). Esta teoria usa o conceito de performativas para a condução das ações. Gudwin (2003) define certas características desta teoria:

- É derivada da análise lingüística da comunicação humana;

- Como em uma linguagem, o comunicador de uma língua não somente efetua uma declaração, mas realiza uma ação;
- As mensagens são consideradas ações ou atos comunicativos.

De acordo com Gudwin (2003), os atos comunicativos são interpretados a partir da mensagem de contexto e nem sempre essa interpretação é obvia. Neste sentido, significa que atos comunicativos são sujeitos a interpretações dúbias que podem ter significados diferentes de acordo com o ponto de vista.

As seguintes frases ilustram alguns exemplos de tipos de ações para o mesmo termo "Sair da frente":

- "Saia da minha frente !" (Comando)
- "Por favor, saia da minha frente." (Pedido)
- "Você poderia sair da minha frente ?" (Pergunta)
- "Eu gostaria que você saísse da minha frente." (Informação)

Desta maneira, percebe-se que é necessário sempre deixar bem claro o ato comunicativo relacionado à mensagem na comunicação entre agentes.

Uma das propriedades essenciais de um agente é a sua capacidade de comunicar-se com outros agentes, usuários e sistemas visando atingir um de seus objetivos, a interação. Durante o planejamento da interação, deve-se lembrar que a informação pode ser incompleta, imprecisa e/ou prevista, e sua qualidade pode variar de acordo com o tipo de agente. A interação pode ser dividida em quatro camadas de complexidade: comunicação, coordenação, cooperação e colaboração (WORTMANN; SZIRBIK, 2001).

A camada comunicação é básica para qualquer software que precisa interagir. A camada coordenação define as regras de interação, considerando as agendas dos agentes, de forma a

se evitar comportamentos indesejados. A camada cooperação é uma camada encontrada apenas em sistemas em que a cooperação reflete uma estratégia de ação decidida pelo agente, permitindo a negociação. A camada mais refinada é a camada colaboração, na qual um agente tem capacidade de detectar possíveis objetivos comum, e de planejar sua agenda com os outros, de forma a atingir o objetivo da melhor forma possível, aproveitando ao máximo a partilha de informações (WORTMANN; SZIRBIK, 2001).

2.4 A LINGUAGEM PADRÃO FIPA-ACL

Diante da grande variedade de pesquisas e usos da tecnologia de agentes, tornou-se importante o estabelecimento de padrões para orientar os modelos de desenvolvimento de agentes. Com o intuito de possibilitar a interação entre agentes de diferentes origens, foi criada a FIPA (*Foundation for Intelligent Physical Agents*), que é uma fundação internacional sem fins lucrativos que tem como objetivo exclusivo criar padrões concretos de comunicação para tornar possível a implementação de agentes abertos e interoperáveis. De acordo com esta instituição (FIPA, 2010), sua missão oficial é: "A promoção de tecnologia e especificações de interoperabilidade que facilitem a comunicação entre sistemas de agentes inteligentes no contexto comercial e industrial moderno.". Sua criação se deu em 1996, através da união de esforços de diversas empresas, universidades e centros de pesquisa. Seus padrões são, atualmente, os mais utilizados na pesquisa de sistemas multiagentes (GLUZ; VICCARI, 2003). A FIPA é uma organização membro do IEEE (*Institute of Eletrical and Eletronics Engineers*), a principal autoridade em áreas da engenharia como Computação, Biomedicina, Aeronáutica, Telecomunicações, Elétrica, entre outras.

Segundo dados do site da FIPA (FIPA, 2010), antes do surgimento dessa instituição existiam cerca de 60 sistemas proprietários de agentes cada um com características próprias, sendo que a maioria destes eram sistemas “fechados” e incompatíveis. Este fato fez com que houvesse um atraso no desenvolvimento da tecnologia de agentes, ficando claro que a necessidade da criação de um padrão era indispensável.

As primeiras especificações da FIPA, de 1997 (padrão FIPA-97), definiam pouco mais que a linguagem de comunicação entre agentes (*Agent Communication Language*, ACL) FIPA-ACL e algumas aplicações. O padrão atual FIPA-2000, está bem evoluído em suas especificações de padrões de agentes se comparado com o passado, pois durante esses anos recebeu diversas contribuições da comunidade que permitiu criar um conjunto de especificações extenso e maduro que atende as mais diversas necessidades.

A FIPA-ACL é uma linguagem utilizada no padrão FIPA de comunicação entre agentes, baseada em ações de fala. Sua especificação consiste de um conjunto de tipos de mensagens e descrições dos efeitos das mensagens sobre os agentes que as enviam e sobre os que as recebem. Possui uma semântica definida precisamente como uma linguagem de descrição semântica.

As mensagens FIPA-ACL são compostas por um identificador do tipo de ato comunicativo, seguido de um conjunto de *slots* com parâmetros. O ato comunicativo da mensagem representa a vontade dos agentes sobre determinada informação, carregada pela mensagem. Alguns exemplos de atos comunicativos são: *Propose*, *Accept-Proposal*, *Request*, *Inform*, *Cancel*, *Confirm*, *Agree*. Já os parâmetros das mensagens trazem informações relativas à conversa realizada, como remetente, destinatário, o conteúdo da mensagem, a linguagem em que o conteúdo está expresso, e ainda outros parâmetros relativos ao controle da conversação (GLUZ; VICCARI, 2003).

Na figura 3 apresenta-se uma estrutura de uma mensagem em FIPA-ACL, com algumas das partes que compõem uma mensagem básica.

Figura 3 - Formato de uma mensagem FIPA-ACL
Fonte: FIPA (2010)

Além do padrão de mensagens, a FIPA também especifica um conjunto de padrões de protocolos de interação que podem ser usados como base para a estruturação das conversas entre agentes[1]. Estes protocolos são padrões definidos para trocas de mensagens FIPA-ACL, compondo conversações específicas entre os agentes. Os protocolos são importantes para garantir que os agentes tenham conhecimento do efeito do envio de sua mensagem.

As mensagens FIPA-ACL são compreendidas como representantes de um ato comunicativo, uma dentre as ações que um agente pode executar. O padrão da FIPA especifica, para cada ato comunicativo, as Precondições de Factibilidade (as condições que devem ser verdadeiras para que o agente possa executar a ação, ou seja, enviar a mensagem) e o Efeito Racional (*Rational Effect*), que é o efeito esperado da ação, ou seja, a razão para seu envio. O padrão ainda preconiza que, após ter executado o ato comunicativo (envio da mensagem), o remetente não pode tomar como certo que seu efeito racional agora vale, pois, dada a autonomia dos agentes, o destinatário pode ter simplesmente decidido descartar a

[1] Disponível em: http://www.fipa.org/repository/ips.html

mensagem recebida e não realizar seu efeito (BERNARDES, 2004). Por esta razão, os protocolos de interação são importantes. Em vez de enviar uma mensagem simples, os agentes que desejarem saber o efeito de suas ações devem iniciar um protocolo, já que eles permitem ao primeiro remetente verificar se o efeito racional foi cumprido.

A linguagem FIPA-ACL pode ser considerada uma evolução e expansão do KQML (FININ et al., 1994), uma linguagem mais antiga que teve seu desenvolvimento na iniciativa KSE (*Knowledge Sharing Effort* – Esforço para o Compartilhamento de Conhecimento), patrocinada pela agência de pesquisas norte-americana DARPA, no início da década de 1990.

Gudwin (2003) afirma que a FIPA-ACL deve vir a substituir o KQML, pois resolve a maioria dos problemas criticados por diferentes autores na concepção do KQML. Durante as atividades de pesquisa para o presente trabalho, foi possível perceber que existe uma forte tendência na comunidade em utilizar a FIPA-ACL para o desenvolvimento de projetos de agentes. Por este motivo, a linguagem utilizada para a comunicação de agentes neste trabalho foi a linguagem no padrão FIPA-ACL.

2.5 O MODELO DE REFÊRENCIA PARA GERENCIAMENTO DE AGENTES DA FIPA

Na especificação *FIPA Agent Management*[2] é apresentado o Modelo de Referência para o Gerenciamento de Agentes, que estabelece o modelo lógico para a criação, registro, localização, comunicação, migração e exclusão de agentes. No Modelo de Referência FIPA, os agentes devem, antes de trocar mensagens entre si, registrar-se numa plataforma, que é o

[2] Disponível em: http://www.fipa.org/specs/fipa00023/

componente do sistema multiagentes que fornece suporte às linguagens, protocolos e serviços FIPA para o sistema.

A plataforma de agentes (AP – *Agent Platform*), segundo a especificação FIPA *Agent Management*, fornece a infraestrutura física em que os agentes operam, consistindo da(s) máquina(s), do sistema operacional, do software de suporte aos agentes, dos componentes de gerenciamento de agentes FIPA (AMS, DF e MTS) e dos próprios agentes:

O AMS (*Agent Management System* - Sistema de Gerenciamento de Agentes) é a entidade responsável pela supervisão do acesso à plataforma de agentes. Os agentes, quando são criados, devem se registrar com o AMS, para obter uma identificação válida dentro da plataforma. O AMS mantém uma lista com estas identificações de agentes e seus respectivos endereços, de modo que possa oferecer um serviço de localização de agentes.

O DF (*Directory Facilitator* - Facilitador de Diretório) é o responsável por oferecer um serviço de "páginas amarelas" à comunidade de agentes. Os agentes podem registrar no DF os serviços que oferecem, de modo que outros agentes possam consultá-lo em busca de um serviço específico que precisem, e obtenham uma identificação do agente que está oferecendo este serviço.

O MTS (*Message Transport Service* - Serviço de Transporte de Mensagens) é um serviço fornecido pela plataforma à qual o agente está ligado. O MTS tem como função suportar o transporte de mensagens FIPA-ACL entre agentes de uma mesma plataforma e também entre agentes de plataformas diferentes.

A figura 4 ilustra o relacionamento entre os diversos elementos que compõem o modelo de referência de gerenciamento de agentes, formando desta maneira, o padrão estabelecido pela FIPA para a plataforma de agentes.

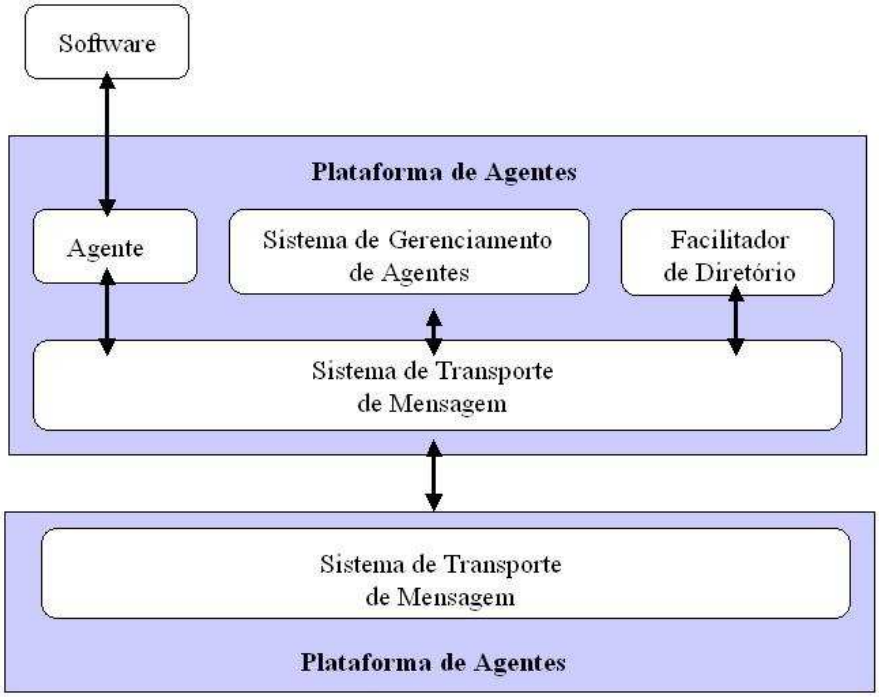

Figura 4 - Modelo de Referência de Gerenciamento de Agentes
Fonte: FIPA [a] (2010)

Existem diversos *frameworks* tecnológicos que implementam o padrão FIPA. Foram realizados estudos com o objetivo de escolher o que melhor atendia as necessidades do presente trabalho e o escolhido foi o *framework JADE* (*Java Agent Development Framework*), o qual será detalhado no capítulo 4. A escolha do *framework JADE* foi fundamentada no fato de que sua documentação é abrangente e detalhada facilitando o aprendizado e a construção de projetos, outro fator importante está relacionada a sua construção que utiliza a tecnologia *Java* que é de conhecimento do autor.

3 TECNOLOGIA ADAPTATIVA

Com o crescente desenvolvimento de novos e diferentes sistemas computacionais, foi natural o surgimento também de ambientes cada vez mais heterogêneos, gerando assim, a necessidade de integração entre eles. Acompanhando este crescimento, a qualidade das interfaces homem-máquina também aumentou, com linguagens cada vez mais próximas da linguagem natural, ampliando assim, a necessidade de dispositivos que atendessem tal premissa.

Com o objetivo de endereçar tais necessidades, os trabalhos e pesquisas sobre dispositivos formais foram intensificadas, especificamente para o tratamento de linguagens de programação, desenvolvimento de compiladores e modelos de computação, formando assim, uma nova classe de dispositivos (NETO; MAGALHÃES, 1981).

Um autômato de pilha convencional é um autômato finito que pode fazer uso de uma memória auxiliar em forma de pilha e pode fazer uso da informação que está no topo dessa pilha para decidir qual transição deve ser efetuada, além de manipular a pilha ao efetuar uma transição (SIPSER, 2007).

O autômato de pilha estruturado foi o primeiro dispositivo formal desenvolvido, com algumas características que atendem essas novas necessidades, apresentando uma opção mais didática e de manipulação mais intuitiva em comparação ao autômato de pilha convencional (NETO; MAGALHÃES, 1981).

Por sua vez, pode-se definir um autômato de pilha estruturado como uma coleção de submáquinas que operam como autômatos finitos mutuamente recursivos. A chamada de uma submáquina por outras se dá sempre que for executada uma transição de chamada de submáquina. Essa chamada implica em empilhar uma indicação do estado para onde o retorno deve ser feito ao final da operação da submáquina (NETO, 1987). Ao término do

processamento desta cadeia, e sendo o estado corrente um estado final, a análise da cadeia continua, então, a partir do estado armazenado no topo da pilha, o qual é desempilhado. Este procedimento é chamado de transição de retorno de submáquina.

No autômato de pilha estruturado, a pilha é utilizada somente para o controle do aninhamento sintático, que é a característica que diferencia uma construção livre de contexto de uma construção regular. Desta forma, a cada nova instância de aninhamento, uma chamada à submáquina será efetuada.

Pode-se definir linguagem livre de contexto como uma linguagem gerada por alguma gramática livre de contexto com a característica de ser aceita por um autômato de pilha (NETO; MAGALHÃES, 1981).

Autômatos de pilha estruturados permitem o reconhecimento de linguagens livre de contexto. Para o reconhecimento de linguagens dependentes de contexto pode-se utilizar uma extensão deste modelo os autômatos adaptativos (NETO; MAGALHÃES, 1981).

Segundo Neto (2001), pode-se definir o dispositivo adaptativo como um formalismo que tem a capacidade de alterar dinamicamente sua topologia e seu comportamento, de forma autônoma. Essa capacidade do dispositivo adaptativo de se automodificar ocorre em situações que exijam mudanças nas reações em resposta aos estímulos de entrada. É possível afirmar, então, que o dispositivo deve se adaptar, reagindo adequadamente às mudanças impostas, de maneira a atender todas as situações.

Como o dispositivo adaptativo tem a característica de, a cada nova transição, alterar sua própria estrutura, ou seja, adicionar ou remover estados e transições por meio de chamadas às funções adaptativas, faz-se necessário a criação de uma simbologia para a representação formal dessas ações. Portanto, pode-se definir como ações adaptativas elementares as ações de pesquisa, de inclusão e a de exclusão, além de uma estrutura de suporte, com declaração de variáveis, geradores e parâmetros, dentre outro recursos (NETO, 1994).

O conceito de adaptatividade de dispositivos teve sua origem no modelo formal de um autômato adaptativo, que é uma extensão do autômato de pilha estruturado, que altera seu comportamento em resposta às entradas recebidas através da execução de regras adaptativas predefinidas. Desta forma, surgiam os conceitos que estariam compondo as características do que seria chamada de Tecnologia Adaptativa (NETO, 2007).

Como a característica principal de um sistema adaptativo está relacionada à possibilidade de se automodificar, reprogramando automaticamente sua própria estrutura e comportamento, baseando-se no conhecimento adquirido durante sua operação, faz-se necessário a inclusão de recursos de inteligência para que todo o mecanismo de aprendizado funcione de maneira coerente (TCHEMRA, 2007).

Segundo Tchemra (2007), a aplicabilidade e a versatilidade da Tecnologia Adaptativa levaram a sua utilização nos mais diversos projetos, cuja relação custo-benefício se mostrou bastante interessante se comparada às técnicas mais usuais. Podem-se destacar os seguintes formalismos adaptativos: Gramáticas Adaptativas, Redes de Markov Adaptativas, *Statecharts* Adaptativos, Árvores de Decisão Adaptativas e os Dispositivos guiados por Regras Adaptativos (ou apenas Dispositivos Adaptativos), Tabelas de Decisão Adaptativas, entre outros. Alguns formalismos adaptativos já foram definidos, dentre eles o formalismo relacionado a Tabelas de Decisão Adaptativas, no sítio do Laboratório de Linguagens e Técnicas Adaptativas pode-se encontrar mais detalhes e referências (LTA, 2010).

Como o direcionamento deste trabalho está para o formalismo adaptativo conhecido como Tabelas de Decisão Adaptativas, seus conceitos serão mais detalhados no decorrer do trabalho.

3.1 DISPOSITIVO ADAPTATIVO

A definição de dispositivo adaptativo, também denominado dispositivo guiado por regras adaptativo, caracteriza-se por um sistema cuja operação é definida por um conjunto finito de regras que se automodifica dinamicamente, dando nova forma ao dispositivo (NETO, 2001).

Este dispositivo é formado por duas camadas, sendo que a primeira compõe o núcleo do sistema, composto por um dispositivo convencional não-adaptativo e por uma camada externa a este núcleo que acrescenta um mecanismo adaptativo que atua no núcleo, modificando sua estrutura e comportamento, atuando no seu conjunto de regras.

Portanto, o dispositivo adaptativo é formado por um núcleo, chamado dispositivo subjacente, coberto por uma camada adaptativa que contém as ações adaptativas que operam sobre o núcleo quando necessário, com o objetivo de criar as características de modificação autônoma, conforme ilustrado na figura 5.

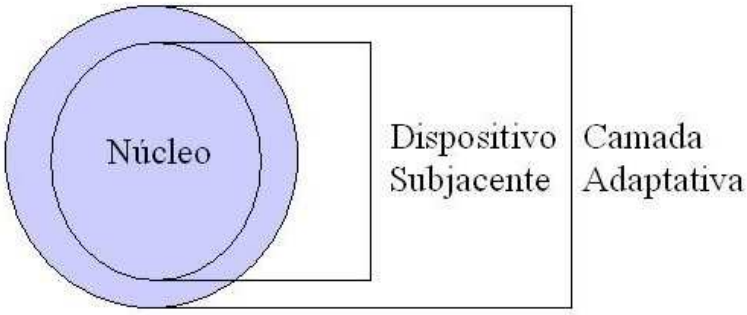

Figura 5 - Estrutura Geral de um Dispositivo Adaptativo
Fonte: TCHEMRA (2007)

O grande diferencial deste tipo de abordagem é a facilidade de uso, uma vez que existe a possibilidade de se empregar formalismos conhecidos e tradicionais, tais como tabelas de

decisão, principalmente dispositivos não-adaptativos subjacentes e acrescentar as características de um dispositivo adaptativo por meio da inclusão de uma camada adaptativa.

Pode-se consultar em Neto (2001) um detalhamento profundo do formalismo e definições de um dispositivo adaptativo conduzido por regras.

3.2 PROCESSO DE TOMADA DE DECISÃO

O processo de tomada de decisão é algo que faz parte do cotidiano do ser humano desde os primórdios de sua existência, e essa situação perdura até hoje. Tomar decisão compõe as mais diversas atividades diárias de qualquer pessoa, mesmo que muitas vezes sejam feitas utilizando o bom senso.

Embora algumas situações que exigem um processo de tomada de decisão sejam resolvidas de maneira simples e objetiva, com a evolução da humanidade, situações mais complexas começaram surgir, trazendo consigo novos cenários de tomada de decisão, nos quais, para sua resolução, é necessário um estudo mais profundo e uma melhor organização do processo. Este tipo de processo mais complexo pode ser encontrado nas mais diversas situações, como por exemplo, na gestão de negócios, no planejamento estratégico das empresas, na busca de melhores investimentos financeiros, na análise de custos e lucros, no controle do tráfego aéreo, nos diagnósticos médicos, dentre outros.

Os estudos na área de tomada de decisão definem decisão como um processo que começa na identificação das melhores alternativas e termina na escolha das alternativas possíveis que mais atendem os objetivos daquela tomada de decisão (CERTO, 2000; HARRIS, 2009).

Segundo Yates (2003), existem duas teorias de como os, seres humanos, toma decisão. O modelo racional de tomada de decisão e o modelo de tomada de decisão naturalista.

O modelo *Racional de Tomada de Decisão* analisa as opções disponíveis e calcula os níveis de riscos, permitindo assim, escolher a opção ou as opções que possuem um grau de confiança satisfatório para atender os objetivos propostos. A busca da opção com este grau de confiabilidade consiste dos seguintes passos:

1- Escolha dos critérios de confiabilidade que serão avaliados nas opções disponíveis.
2- Definição de critérios de escolha dos melhores resultados por meio de pontuação dos critérios de confiabilidade.
3- Análise das opções disponíveis com o objetivo de verificar o grau de confiabilidade de cada uma delas.
4- Determinar dentre as opções disponíveis os resultados escolhidos.

Este modelo tem sido utilizado nos mais diferentes campos econômicos, políticos/governamentais e até mesmo em processos de planejamento militar (BROWN, 2005; KLEIN, 1997).

O modelo de *Decisão Naturalista* analisa as experiências das pessoas na tomada de decisão em uma situação proposta abruptamente, normalmente relacionada à pressão de tempo ao tomador de decisão, fazendo com que o processo de tomada de decisão seja baseado na melhor opção, dada sua experiência em alguma outra situação parecida (ZSAMBOCK, 1997).

A diferença entre as duas abordagens está em como elas direcionam o processo de tomada de decisão. O modelo *Racional de Tomada de Decisão* direciona o processo de estratégia de como os indivíduos deliberam sobre suas escolhas, enquanto que o modelo de

Decisão Naturalista baseia-se em tomar decisões utilizando percepções das experiências em situações passadas que sejam semelhantes às propostas.

O presente estudo tem como base a abordagem do modelo *Racional de Tomada de Decisão* para a racionalização dos agentes. Apesar da abordagem do trabalho ser o modelo racional a utilização do aspecto adaptativo permite criar um ambiente de decisão naturalista, visto que gera um mecanismo de aprendizado dentro da proposta de solução. Este modelo racional de aprendizado tem relação direta com a utilização da Tecnologia Adaptativa neste trabalho, pois a aprendizagem proporcionada pela adaptatividade permite tomar decisões com base na experiência anterior, desta forma melhorando a assertividade do processo.

O modelo *Racional de Tomada de Decisão* é composto de cinco etapas e está ilustrado na figura 6 (CERTO, 2000). A seguir é apresentado o detalhamento das etapas:

1 – Identificar o problema – Identificar a existência de um problema e os objetivos da decisão.

2 – Criar alternativas – Listar as possíveis alternativas para a solução do problema.

3 – Selecionar a melhor alternativa – De posse das alternativas, selecionar qual produz os melhores resultados em relação aos objetivos da decisão.

4 – Implementar a decisão – Colocar a alternativa selecionada em ação.

5 – Analisar *feedback* – Capturar o *feedback* após a execução da ação, com o objetivo de avaliar se a alternativa implementada está solucionando o problema identificado.

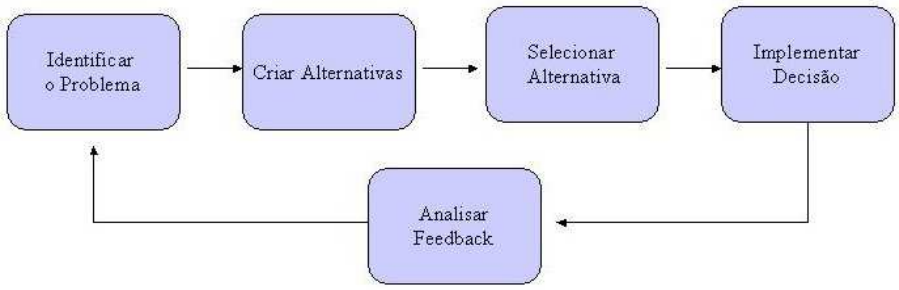

Figura 6 - Fluxo do processo do modelo racional de tomada de decisão.
Fonte: CERTO (2000).

Apesar do modelo da figura 6 resolver grande parte dos problemas de tomada de decisão, é importante salientar que este somente é válido quando possui um critério de escolha prefixado em relação ao qual a avaliação é realizada. Caso esta premissa não seja verdadeira, o modelo não funciona.

O modelo *Racional de Tomada de Decisão* é baseado em três premissas básicas. Primeiro, que os tomadores de decisão são seres econômicos que sempre objetivam a maximização da satisfação e/ou retorno. Segundo, é assumido que dentro da situação proposta, todas as alternativas e suas possíveis ações são conhecidas. E por último, que os tomadores de decisão consigam identificar claramente as prioridades em relação à utilidade de cada alternativa. Caso todas as premissas sejam atendidas, os tomadores de decisão provavelmente irão produzir o melhor resultado possível dentro do cenário apresentado (CERTO, 2000; SHIMIZU, 2001).

3.2.1 TIPOS E CONDIÇÕES PARA TOMADA DE DECISÃO

De acordo com Simon (1970) e Chiavenato (1997), o processo de decisão pode ser classificado em dois tipos que não são mutuamente exclusivos, mas representam dois pontos extremos em relação a programação das decisões:

Decisões programadas: São as decisões caracterizadas pela rotina e repetitividade. São adotadas mediante uma regra, com dados evidentes, condições estáticas, certeza, previsibilidade. Acontecem com certa freqüência nas organizações. Exemplos: fazer pedido de estoque sempre que o nível cair para 100 unidades; liquidar mercadorias de lojas do vestuário próximo à troca de estação.

Decisões não programadas: São as decisões caracterizadas pela não-estruturação, dados inadequados, únicos ou imprevisíveis. Estes tipos de decisões estão ligados às variáveis dinâmicas, tornando-se difícil de gerenciar. Seu intuito é a resolução de problemas incomuns, marcados pela inovação e incerteza.

Visto que, na maioria dos cenários de decisão no mundo real, é praticamente impossível prever exatamente quais as conseqüências da implementação de uma alternativa escolhida, foi proposta uma classificação das condições de previsibilidade das conseqüências das alternativas da decisão (CERTO, 2000; PIDD, 2001).

As condições para a decisão ser tomada são as seguintes (GOMES; GOMES; ALMEIDA, 2006):

Condição de certeza: Ocorre quando a decisão é feita com pleno conhecimento de todos os estados da natureza. Existe a certeza do que irá ocorrer durante o período em que a decisão é tomada. Nesta condição, tudo o que os decisores têm a fazer é listar os resultados das alternativas, e então, escolher a que melhor atende. Como exemplo desta condição, pode-se citar como resultado de uma alternativa o financiamento com juros preestabelecidos, uma vez que, desta maneira, o resultado se torna previsível, visto que as taxas já são conhecidas antes mesmo da alternativa ser escolhida.

Condições de risco: Ocorre quando são conhecidas as probabilidades associadas a cada um dos estados de natureza. O número total de estados da natureza é conhecido. Ao contrário da condição anterior, que dispunha de 100% de certeza no resultado final, aqui, essa certeza

varia de 0 a 100%. Como exemplo, pode-se supor a situação em que um dono de uma empresa de entregas tem como objetivo aumentar sua capacidade de atuação, e assim, compre mais três caminhões. Ele pode acreditar que é alta a probabilidade que o volume de entregas aumente, porém, não existem garantias que isso ocorra.

Condições de incerteza ou em condições de ignorância: Ocorre quando não se obteve o total estado da natureza, ou mesmo quando a parcela dos estados conhecidos da natureza possui dados obtidos com probabilidade incerta, ou a probabilidade associada aos eventos é desconhecida, muitas vezes por não existir nenhum histórico sobre o qual a decisão possa ser baseada.

Em situações de falta de conhecimento em relação ao que pode ocorrer, os decisores podem assumir que a decisão correta está relacionada a uma questão de sorte. Felizmente, poucas decisões do mundo real fazem parte desta categoria, uma vez que, pelo fato do resultado de uma decisão ser totalmente inesperado, é inviável controlar essa situação (CERTO, 2000). Pode-se citar como exemplo de tomada de decisão em uma situação de incerteza completa, fazer um jogo da mega sena, visto que não se pode saber qual é o resultado que vai ocorrer.

Em uma situação real, o grau de risco associado à decisão tomada está diretamente relacionado à qualidade da informação disponível para a tomada de decisão, sendo assim, quanto menor a qualidade da informação, mais perto da incerteza completa será a tomada de decisão, e maior o risco de tal alternativa ser escolhida (CERTO, 2000).

3.2.2 CRITÉRIOS PARA TOMADA DE DECISÃO

No processo de tomada de decisão é importante determinar quais são os critérios que devem ser atendidos para uma solução aceitável do problema. A solução escolhida é resultado da avaliação de duas ou mais alternativas possíveis que atendam às condições e objetivos do problema, dados esses que formam o critério.

Normalmente, processos decisórios que envolvem múltiplos critérios podem apresentar maior quantidade de alternativas, elevando assim, o nível de complexidade na escolha da solução (CERTO, 2000).

Problemas de decisão com apenas um critério são problemas clássicos de otimização, cuja função objetiva é o critério e as limitações são os requisitos das alternativas. Existem diferentes técnicas de otimização para resolver tais problemas, dentre elas, a análise de custo e benefício, programação linear, entre outras, todas com o objetivo de achar o valor ótimo. Por outro lado, para problemas de tomada de decisão multicritérios, existem métodos particulares que se caracterizam por utilizar critérios quantitativos e qualitativos, permitindo a análise e a confiabilidade da decisão por meio de julgamentos do decisor (FÜLÖP,2005).

Os métodos multicritérios são classificados através das características inerentes que os compõem. Por exemplo, alguns têm o objetivo de identificar a melhor alternativa, outros classificam ou categorizam as alternativas, ou apresentam uma lista de alternativas possíveis para análise subseqüente, ou para filtrar aquelas que são aceitáveis e inaceitáveis dentro de um padrão pré-estabelecido. Dentre os métodos multicritérios formais pode-se citar: AHP (*Analytic Hierarchy Process*), PROMETHE (*Preference Ranking Method for Enrichment Evaluation*), ELECTRE (*Elimination and Choice Translating Reality*), MACBETH

(*Measuring Attractiveness by a Categorical Based Evaluation Technique*), TOPSIS (*Technique for Order Preference by Similarity to Ideal Solution*), dentre outros (TCHEMRA, 2007).

Os problemas de tomada de decisão multicritérios com m critérios e n alternativas podem ser representados pelos conjuntos de critérios C= {C1,C2,C3,..., C_i, ..., C_m} e de alternativas A = {A1,A2,A3,..., A_i, ..., A_n}, formando, dessa forma, uma matriz de decisão conforme a figura 7.

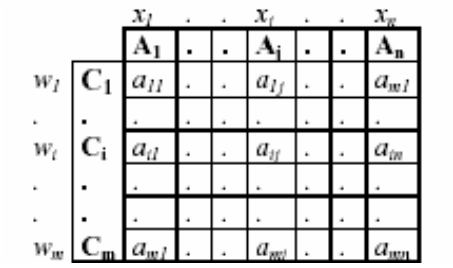

Figura 7 - Matriz de decisão.
Fonte: FÜLÖP (2005)

Apesar de os elementos já servirem de base para uma tomada de decisão, em um método multicritério é necessário atribuir pesos w_i (ou força) a cada um dos critérios C_i da matriz, sendo assim, uma ferramenta para o decisor torná-la mais aderente à realidade do problema. A maneira como são utilizados esses pesos varia de um método para outro e um exemplo dessa utilização é a lógica *fuzzy*. Em relação às alternativas A_i, é feita uma pontuação a_{ij} para descrever o desempenho da alternativa em relação ao critério C_j. Com base nesses pesos e pontuações, cada alternativa recebe um valor que servirá de avaliação para a tomada de decisão (FIGUEIRA; GRECO; EHRGOTT, 2004).

De uma maneira geral, a decisão por uma determinada alternativa está relacionada com o maior valor atribuído, gerando assim, uma alternativa de solução ou uma alternativa de execução de ação (KEENEY; RAIFFA, 1976).

Existem cenários em que a escolha de somente uma alternativa não serve como solução. Para esses casos, existe outro método multicritério que se baseia nos métodos de sobreclassificação (*Outranking*), no qual, por meio dos valores w_i e a_{ij}, as alternativas A_i e A_j são comparadas com o objetivo de determinar a preferência entre elas, ou ainda, determinar o nível de concordância ou discordância entre uma em relação à outra, levando-se em conta os critérios do problema. Diferentemente dos métodos em que uma e somente uma alternativa é viável, este método determina um subconjunto de alternativas viáveis para um determinado cenário, ou ainda, possíveis ações a serem tomadas (FIGUEIRA; GRECO; EHRGOTT, 2004; TCHEMRA, 2007).

3.3 TABELAS DE DECISÃO ADAPTATIVAS

Conforme mostrado na Seção anterior, existem diversas técnicas para resolução de problemas de tomada de decisão.

A implementação destes conceitos conduz para a criação de dispositivos padrões, tais como as tabelas de decisões, de tal forma que o presente trabalho faz uso para a implementação do objetivo proposto. Para isso será apresentado o conceito do dispositivo, sua

base de funcionamento, sua estrutura, e um exemplo, com o intuito de mostrar seu funcionamento.

3.3.1 TABELAS DE DECISÃO

De acordo com Vanthienen (2008), pode-se definir tabela de decisão como uma tabela que representa um conjunto completo de expressões condicionais mutuamente exclusivas em uma área predefinida. Sua utilização está relacionada ao seu alto poder de visualização e à facilidade na prevenção de erros, sendo que seu conceito de modularidade facilita a organização de idéias e seu desempenho, uma vez que suporta a implementação de diversos algoritmos.

As tabelas de decisão são instrumentos que têm sua aplicação não somente restrita às áreas de tecnologia, tais como Sistemas de Informação e Engenharia de Software, sendo utilizada também em outras áreas do conhecimento humano, tais como Economia, Medicina, Administração e Finanças, ou seja, em aplicações onde é necessária a classificação do conhecimento.

Pode-se definir como sendo uma tabela de decisão típica um dispositivo composto por um conjunto de regras, no qual cada regra é composta por duas partes, sendo a primeira formada pelas condições a serem testadas e a segunda, composta pelas ações a serem executadas, e ainda, esta segunda parte é executada se e somente se todas as condições forem satisfeitas. Em um detalhe maior em relação às regras, pode-se observar que a parte das

condições pode conter três valores distintos: Verdadeiro (V), Falso (F) ou Nulo (-). No processo de execução de teste das condições da regra, caso esta tenha o seu valor preenchido com o valor Nulo, a mesma se torna irrelevante em sua avaliação. (MONTALBANO, 1974).

O formato tradicional de representação de uma tabela de decisão é composto por colunas e linhas, no qual as colunas representam as regras, sendo cada coluna uma regra independente, e as linhas representam as condições de testes e as ações a serem executadas por cada regra (GILDERSLEEVE, 1970). A figura 8 representa uma tabela de decisão onde as linhas representam as condições e ações a serem tomadas caso a regra seja executada e cada coluna representa uma regra que é formada pela combinação das diversas condições e ações.

		Regras									
		0	1	2	3	4	5	6	7	8	9
Linha de Condições	c1										
	c2										
	...										
	cn										
Linha de Ações	a1										
	a2										
	...										
	am										

Figura 8 - Tabela de decisão convencional.

A manipulação de uma Tabela de Decisão pode se apresentar da seguinte maneira:

- Uma vez definida a situação atual do ambiente, é feita uma busca nas linhas que representam as condições, de tal maneira que seja encontrada uma ou mais

condições que sejam compatíveis com a situação apresentada. Existe ainda a possibilidade de que nenhuma condição seja encontrada.

- Caso não seja encontrada alguma regra compatível, nenhuma ação é executada.
- Caso exista apenas uma regra compatível, a ação correspondente é selecionada.
- Caso existam mais de uma regra compatível, pode-se obter uma situação conhecida como configuração não-determinística, em que são selecionadas todas as ações correspondentes ou pode ocorrer a escolha de qual ação será selecionada de acordo com um critério preestabelecido.
- Uma vez selecionadas as ações, as mesmas são executadas; havendo mais de uma, as ações podem ser executadas em paralelo ou seguindo uma ordem predeterminada por meio de um critério, ou ainda, caso suas execuções sejam conflitantes, são escolhidas quais ações serão executadas e quais não.

Dadas as diversas condições de utilização de tabelas de decisão e suas propriedades, torna-se possível sua classificação em categorias, como pode ser observado em Nagayama (1990). Um exemplo destas classificações é a existência ou não de uma regra do tipo ‘else’, que pode ser executada caso nenhuma das outras regras seja compatível, ou ainda, a possibilidade do uso de variáveis ou conjuntos valorados nos campos de condições das regras (NAGAYAMA, 1990).

A Seção a seguir apresenta o detalhamento de tabelas de decisão adaptativas, que é foco da abordagem do presente estudo.

3.3.2 CONCEITUAÇÃO EM TABELAS DE DECISÃO ADAPTATIVAS

De acordo com Neto (2001), pode-se definir Tabela de Decisão Adaptativa como sendo uma aplicação dos conceitos de dispositivos adaptativos que utiliza como dispositivo subjacente uma tabela de decisão convencional. Por suas características, uma Tabela de Decisão Adaptativa tem sua estrutura muito similar à estrutura de uma tabela de decisão convencional, uma vez que, de uma maneira sintetizada, é composta pela associação de funções adaptativas às regras da tabela de decisão convencional, possibilitando assim, mudanças no conjunto de regras da tabela de forma dinâmica.

A utilização de um dispositivo dirigido por regras adaptativas pode dar maior flexibilidade às tabelas de decisão convencionais, permitindo a inclusão e a exclusão de regras durante sua execução, tornando-a uma ferramenta ainda mais poderosa (NETO, 2001). Para fins de comprovação, pode-se encontrar o trabalho desenvolvido por Neto (2001) para simular um autômato adaptativo que reconhece sentenças de linguagens dependentes de contexto.

Um estudo detalhado da estrutura geral e do formalismo da Tabela de Decisão Adaptativa (TDA) também pode ser consultado em Neto (2001).

A figura 9 apresenta a composição de uma tabela de decisão adaptativa, com suas condições, ações e regras, e a camada adaptativa, com as funções e ações adaptativas.

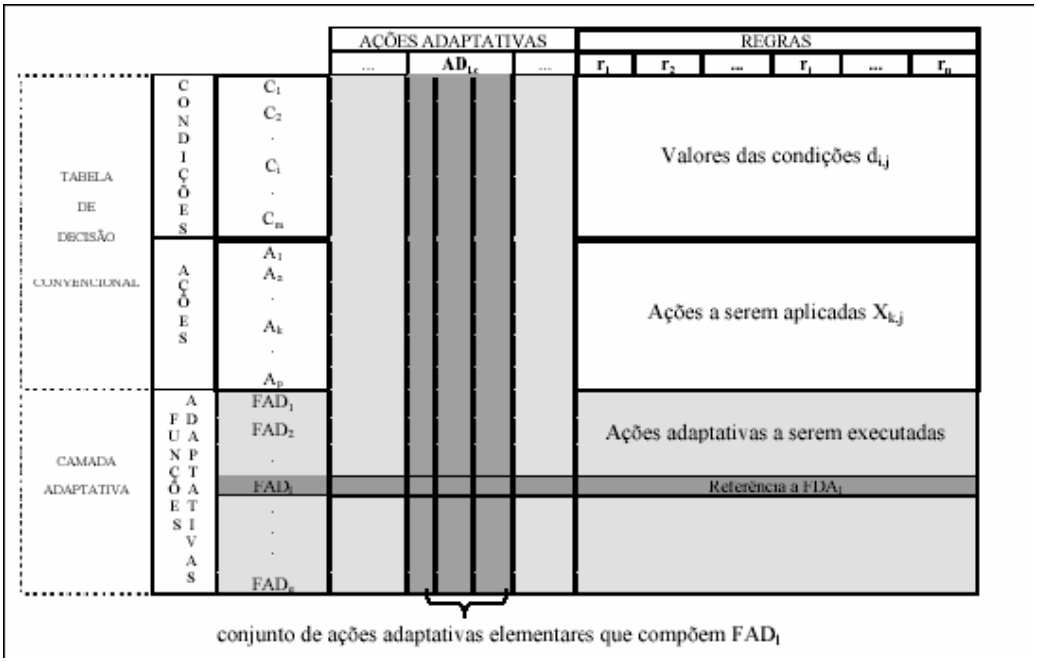

Figura 9 - Estrutura geral de uma tabela de decisão adaptativa.
Fonte: TCHEMRA (2007).

Em relação à operação de uma Tabela de Decisão Adaptativa, é necessário primeiramente identificar e estruturar as condições e ações de um problema na tabela de decisão, juntamente com suas funções e ações adaptativas (TCHEMRA, 2007). Para explicar o funcionamento de qualquer dispositivo dirigido por regras, Neto (2001) considera um algoritmo. A seguir, é apresentada uma versão adaptada deste algoritmo especificamente para as Tabelas de Decisão Adaptativas (PEDRAZZI, 2007).

- Primeiramente, executa-se uma busca nas condições de todas as regras da tabela de decisão, verificando se há alguma regra compatível com a situação do ambiente.
- Caso não exista alguma regra compatível, nenhuma ação é executada.
- Caso exista apenas uma regra compatível, esta regra é selecionada para a aplicação.

- Caso existam mais de uma regra compatível, obtém-se uma configuração não-determinística. Neste caso, selecionam-se todas as regras compatíveis para a aplicação e é determinada qual será a estratégia para a execução das regras.
- Em relação à aplicação da regra selecionada, essa é feita em três partes:

    - O dispositivo verifica se existe uma função adaptativa anterior. Sendo essa chamada diferente de vazia, o conjunto de regras do dispositivo sofrerá alteração, e assim, pode existir a possibilidade de novas regras serem incluídas à tabela, ou ainda, a exclusão de regras já existentes. Neste caso, existe a possibilidade da execução da ação adaptativa excluir a própria regra chamadora. Desta forma, a execução do processamento desta regra é interrompida após a eliminação da regra, assim a Tabela de Decisão Adaptativa procura outra regra para execução.

    - Executam-se todas as ações cujo campo da regra fora solicitado, da mesma maneira que as ações são executadas em uma tabela de decisão convencional.

    - O dispositivo verifica se existe uma função adaptativa posterior. Sendo essa chamada diferente de vazia, o conjunto de regras do dispositivo sofrerá alteração, e assim, pode existir a possibilidade de novas regras serem incluídas à tabela, ou ainda, a exclusão de regras já existentes; Neste caso, existe a possibilidade da execução da ação adaptativa excluir a própria regra chamadora. Desta forma, a execução do processamento desta regra é interrompida após a eliminação da regra. Assim, a Tabela de Decisão Adaptativa procura outra regra para execução.

- Havendo mais de uma regra selecionada para aplicação, todas as regras são executadas ao mesmo tempo.
- Uma vez executadas todas as chamadas às ações adaptativas e todas as ações solicitadas correspondentes à regra selecionada, o processo de operação da tabela de decisão é reiniciado.

Esta Seção encerra a primeira parte do presente trabalho, com a apresentação dos principais conceitos e fundamentos que são utilizados como base para o presente estudo.

4 RACIONALIZAÇÃO DE AGENTES COM USO DA TECNOLOGIA ADAPTATIVA

Este capítulo é responsável pela apresentação dos elementos que compõem a proposta do projeto deste estudo. São detalhados o plano de implementação da proposta, a descrição das características e funcionalidades do *framework* JADE (*Java Agent Development Framework),* a descrição das características e funcionalidades do *framework Drools* que, neste trabalho, são responsáveis pela implementação de tabelas de decisão. Também são detalhadas as modificações feitas no *framework Drools* para a inclusão das características da Tecnologia Adaptativa e, por fim, a estratégia de racionalização de agentes é abordada em detalhes.

4.1 SOLUÇÃO PROPOSTA

Conforme apresentado no primeiro capítulo, o objetivo do presente trabalho é apresentar a implementação de uma solução de racionalização de agentes por meio da Tecnologia Adaptativa, mais especificamente, a inclusão de um núcleo adaptativo dentro de um agente, com o objetivo de criar um ambiente racional.

Para atingir tal objetivo, foi utilizado a estratégia de escolher um *framework* que implementasse uma solução de tabela de decisão tradicional para que este faça o papel de dispositivo subjacente. Para criar o contexto adaptativo, a arquitetura do *framework* foi

estudada, com o objetivo de localizar uma maneira de adaptá-la para criar a camada adaptativa. Com a junção dessas duas camadas foi criado o dispositivo adaptativo que foi integrado a um *framework* de Sistemas Multiagentes com a proposta de produzir um ambiente racional.

Para o estudo de agentes e sistema multiagentes foram pesquisadas algumas implementações de sistemas multiagentes os quais, em uma seleção preliminar, foram considerados os aspectos de conformidade com os padrões da FIPA. Nesta primeira lista foram identificadas as seguintes implementações: *JADE* (JADE, 2010), FIPA-OS (FIPA-OS, 2010) e Zeus (ZEUS, 2010). Para a escolha da implementação a ser utilizada no presente trabalho foram considerados aspectos relacionados à aderência a padrões de mercados consolidados, quantidade e qualidade de documentação disponível e o atendimento às necessidades propostas para a implementação do projeto deste estudo, sendo o *framework JADE* o escolhido dentre as opções apresentadas.

4.2 ARQUITETURA DO FRAMEWORK JADE

O conteúdo desta Seção está baseado principalmente no sítio do *framework* Jade (JADE, 2010), que contém informações relacionadas à arquitetura do *framework* a ser apresentado.

A implementação do *framework* tem como objetivo fazer o papel de *middleware* para o desenvolvimento e a execução de aplicações baseadas no paradigma de agentes, que podem, de maneira transparente, trabalhar em conjunto em ambientes com ou sem rede.

O termo *middleware* está relacionado a bibliotecas de alto nível que tornam fácil e efetivo o desenvolvimento de aplicações, por meio da disponibilização de serviços genéricos que podem ser utilizados, por uma variedade de aplicações. Os serviços de acesso a dados, comunicação e controle de recursos são alguns exemplos de serviços. A proposta dos serviços de *middleware* é disponibilizar serviços de mais alto nível, servindo de camada entre o sistema operacional e as aplicações, principalmente com características de reutilização. Conforme se pode observar na figura 10 o *middleware* fica na camada intermediaria entre as aplicações e o sistema operacional, para que desta forma possa fornecer serviços padrões para as aplicações de maneira transparente.

Em um desenvolvimento tradicional de aplicações (sem o uso de *middlewares*), a criação destes serviços normalmente leva um tempo considerável, de tal maneira que sua utilização reduz consideravelmente o tempo de desenvolvimento dessas aplicações.

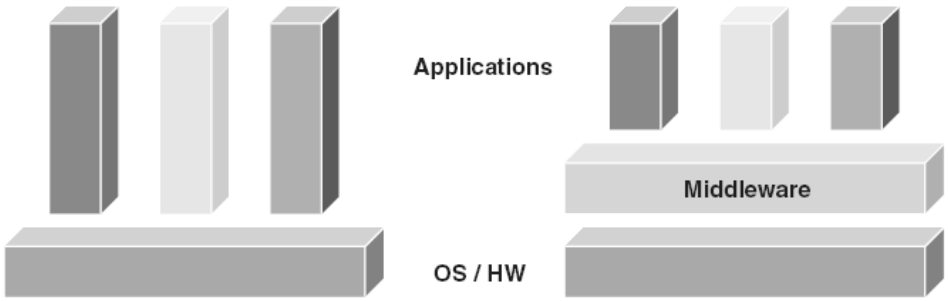

Figura 10 - Arquitetura de um *Middleware*.
Fonte: JADE (2010).

Desta maneira, pode-se definir o Jade como um *middleware* que permite um rápido desenvolvimento de aplicações distribuídas de sistemas multiagentes e que tem as seguintes características: baseia-se na arquitetura de comunicação *peer-to-peer*; tem sua arquitetura de

acordo com as especificações da FIPA; e tem sua distribuição seguindo a licença *LGPL* (*Lesser General Public License*) pois segue os padrões de uma aplicação *Open Source*.

Pode-se definir *peer-to-peer* com a relação de comunicação de nós em uma rede, onde a comunicação pode ser implementada por meio de diversos modelos, os quais não influenciam diretamente na arquitetura do Jade.

Em relação à sua característica de mobilidade, o Jade permite que a inteligência, as informações, os recursos e o controle possam vir a serem distribuídos entre os nós, de tal maneira a criar um ambiente no qual cada nó, possa interagir de maneira transparente por toda a rede. Esta mobilidade é possível devido à padronização na implementação do ambiente dos agentes, que ocorre por meio de órgãos que gerenciam esses padrões, tornando possível a transparência na comunicação entre os agentes.

A implementação do Jade foi desenvolvida completamente em Java e segue os seguintes princípios:

- Interoperabilidade – A implementação do Jade está de acordo com as especificações da FIPA, tendo como característica a capacidade de que seus agentes podem se comunicar com outros agentes em outras implementações, desde que ambos sigam os mesmos padrões.
- Uniformidade e portabilidade – O Jade disponibiliza um conjunto de interfaces de programação de aplicativos (*APIs – Application Programming Interface*) que são independentes de rede e de versão do Java; o ambiente de execução do Jade utiliza as mesmas *APIs* para os ambientes *JEE (Java Enterprise Edition)*, *JSE (Java Standard Edition)* e *JME (Java Móbile Edition)*, aplicações desenvolvidas que podem decidir em qual ambiente serão executadas.

- Facilidade de uso – A complexidade da utilização do *middleware* é escondida pela utilização de *APIs* simples e intuitivas.

O *framework* Jade é composto pelas bibliotecas *APIs* Java necessárias para o desenvolvimento de aplicações de agentes e pelo ambiente de execução que fornece os serviços básicos para a execução de agentes, o qual, por esta característica, deve ser executado antes da execução de agentes.

Cada um dos ambientes de execução do Jade é definido como *container* e o conjunto de todos os *containers* é chamado de plataforma, que fornece uma camada homogênea que esconde dos agentes a complexidade e a diversidade das camadas (*hardware,* sistema operacional, redes e JVM – *Java Virtual Machine*).

Conforme pode ser visualizado na figura 11, o Jade é compatível com as diversas plataformas Java, graças ao seu ambiente de execução que necessita de poucos recursos de *hardware*, tornando-o versátil o suficiente para a sua execução em dispositivos móveis que podem se comunicar com dispositivos tradicionais. Em relação aos aspectos de comunicação, uma vez que sua criação segue os padrões da FIPA, é possível que exista a comunicação entre *containers* escritos em outra linguagem (exemplo: .NET), desde que estes também sigam os padrões FIPA.

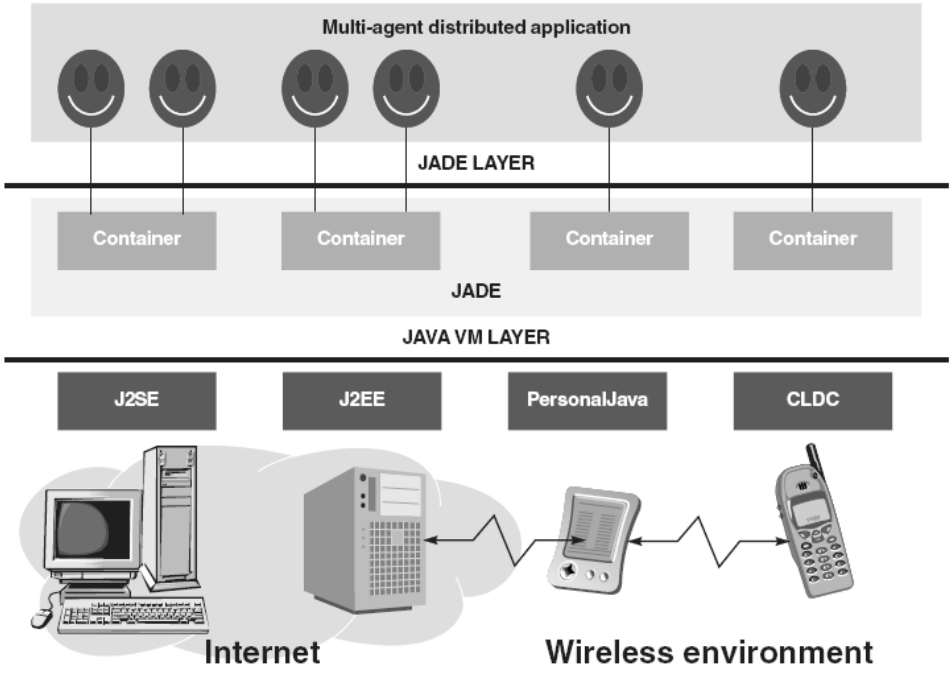

Figura 11 – Arquitetura do Jade
Fonte: JADE (2010).

Em uma perspectiva funcional, o Jade fornece os serviços básicos necessários para distribuir aplicações *peer-to-peer* em ambientes fixos e móveis. O Jade permite que cada agente possa descobrir dinamicamente outros agentes e comunicarem entre si.

De uma perspectiva de aplicação, cada agente é identificado com um único nome e fornece um conjunto de serviços. Cada agente pode registrar e modificar seus serviços e/ou procurar por agentes que têm os serviços necessários para as suas necessidades. Além disso, cada agente tem o controle de seu ciclo de vida.

Em relação à sua comunicação, os agentes trocam mensagens assíncronas, sendo que este é o modelo que tem os benefícios de distribuição e baixo acoplamento de comunicação. Por exemplo, em um ambiente no qual um agente que não conhece nada sobre os outros agentes, quando esse agente deseja se comunicar, apenas envia uma mensagem para um

destino determinado que sirva de meio transmissor para os mais diversos agentes. Como os agentes são identificados por nomes, não é necessário que exista uma dependência entre o agente chamador e o receptor, uma vez que o chamador e o receptor podem não estar disponíveis no mesmo tempo, ou ainda, o receptor nem ao menos existir, portanto, neste modelo, quando um agente processar a requisição, ele devolve o resultado em formato de mensagem para o agente chamador. Todas as mensagens trocadas pelos agentes são enviadas em um envelope, que é composto de informações da camada de transporte e do conteúdo propriamente dito. Este modelo permite que, caso seja necessário, o conteúdo possa vir a ser criptografado para aspectos de segurança.

A estrutura das mensagens trocadas segue os padrões ACL (*Agent Communication Language*) que são definidos pela FIPA e inclui campos, variáveis que indicam o contexto das mensagens, *timeout* de recebimento e resposta de mensagens, suporte a interações complexas, comunicações múltiplas paralelas. Com o objetivo de suportar comunicações múltiplas paralelas, o Jade fornece padrões de execução de tarefas específicas, tais como negociação entre agentes e delegação de tarefas.

O Jade fornece suporte para conversão automática do conteúdo das mensagens, incluindo os formatos *XML* (*eXtensible Markup Language*) e *RDF* (*Resource Description Framework*), dentre outros. Este suporte está relacionado diretamente à possibilidade da utilização de ontologias no conteúdo das mensagens.

Com o objetivo de melhorar a escalabilidade ou atender as necessidades em ambientes com recursos limitados, o Jade fornece a possibilidade de executar múltiplas tarefas em paralelo em uma mesma linha de execução (*Thread Java*).

A plataforma ainda inclui um serviço de localização (garantindo que cada agente tenha um nome único) e um serviço de *"páginas amarelas"*, que pode ser distribuído em vários nós da rede e serve de ponto estratégico para que os agentes possam se localizar entre si.

Como forma de facilitar o controle, pode-se recorrer ao *debugging* e ao gerenciamento/monitoramento de aplicações. O Jade também disponibiliza uma ferramenta gráfica onde os agentes podem vir a ser controlados remotamente, com o objetivo de emular conversações, troca de mensagens, monitorar tarefas e controlar seu ciclo de vida.

4.2.1 A PLATAFORMA DE AGENTES NO JADE

Conforme apresentado na Seção 2.5, uma plataforma de agentes segundo a especificação FIPA contém os seguintes elementos: o AMS (*Agent Management System)*, o DF (*Directory Facilitator*) e o MTS (*Message Transport Service)*.

Para a implementação do trabalho foi escolhida a versão 4.0.1 do Jade pela sua maturidade e por conter todas a funcionalidades necessárias para a implementação da proposta.

O Jade implementa completamente essa arquitetura de referência e quando a plataforma é executada, o *AMS* e o *DF* são imediatamente criados e o *MTS* é configurado para permitir as comunicações de mensagens. A plataforma de agentes pode ser executada em diversos locais, porém, somente uma aplicação Java e uma *Java Virtual Machine* são executadas em cada local.

Cada *JVM* é um *container* de agentes que fornece um ambiente de execução de agentes e permite que os agentes possam executar concorrentemente no mesmo local. Apesar de ser permitida a execução de diversos *containers,* sempre um é escolhido para ser o principal, onde são executados o *AMS* e o *DF*. Os outros *containers* se conectam ao principal e fornecem informações para que assim, todos os agentes possam se comunicar.

A figura 12 representa um exemplo de como é a execução de diversos *containers* ao mesmo tempo, com a possibilidade de comunicação entre si.

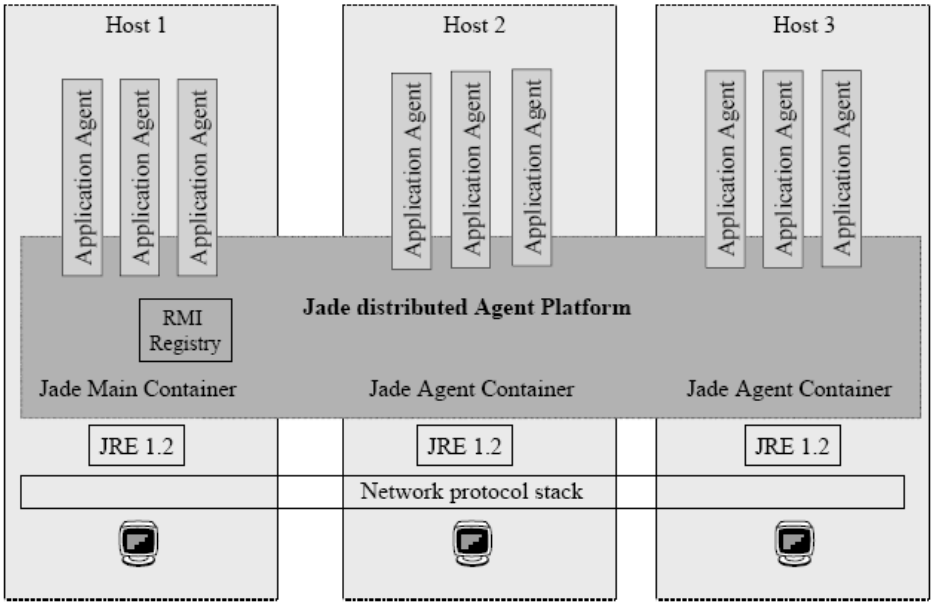

Figura 12 - Arquitetura de uma plataforma de agentes distribuídas em vários *containers*.
Fonte: JADE (2010).

A classe Java "agente" (*Agent*) representa uma classe base para que os usuários possam definir seus agentes. Do ponto de vista do programador, um agente Jade é simplesmente uma instância de uma classe Java que estende (*extends)* a classe base *Agent*. Isto implica que

funcionalidades básicas de interação (registro, configuração, gerenciamento remoto etc.) e de comportamento (enviar/receber mensagens, utilização de protocolos de interação etc) são herdadas automaticamente da classe “pai”.

Um agente no Jade pode ter alguns estados, seguindo os padrões da especificação FIPA para o ciclo de vida. Estes estados podem ser visualizados na figura 13 e seus detalhamentos, a seguir:

- Inicializado (*Initiated*): O agente está criado, mas ainda não está registrado no *AMS*, portanto, ainda não tem um nome ou um endereço e não pode se comunicar com outros agentes.
- Ativo (*Active*): O agente está registrado no *AMS*, tem um nome e um endereço e tem acesso a todas as funcionalidades do ambiente.
- Suspenso (*Suspended*): O agente está parado. Sua *thread* interna está suspensa e nenhum comportamento pode ser executado.
- Esperando (*Waiting*): O agente está bloqueado, esperando por algo. Sua *thread* interna está dormindo e um monitor Java o acorda quando alguma condição ocorre (tipicamente quando uma mensagem chega).
- Apagado (*Destroyed)*: O agente está morto. A *thread* interna terminou a execução e o agente não está mais registrado no *AMS*.
- Em trânsito (*Transit*): Quando um agente está migrando de um local para o outro.

É importante observar que um agente tem permissão de executar algum comportamento somente no estado Ativo.

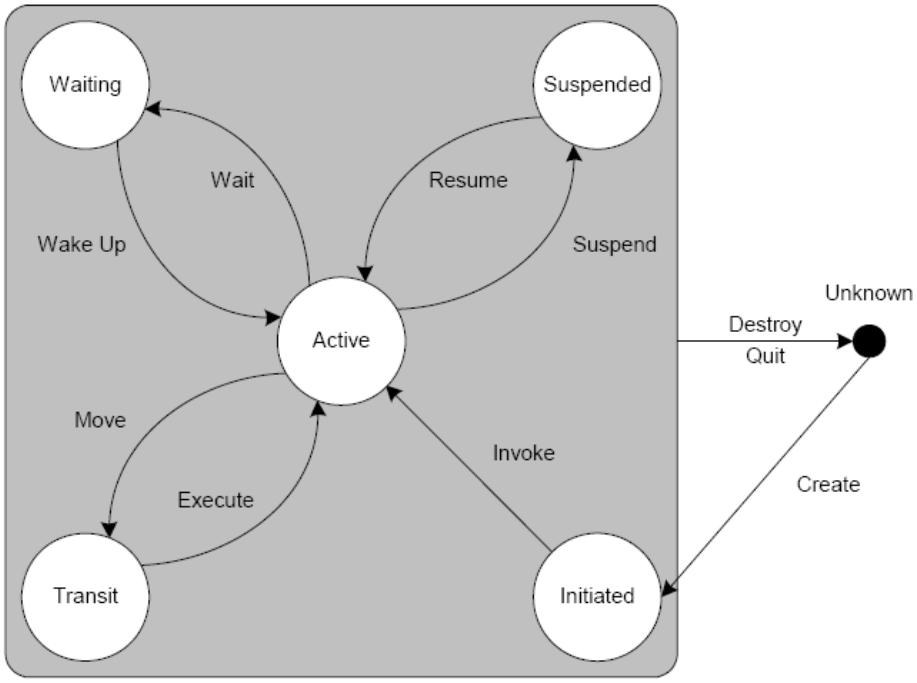

Figura 13 - Ciclo de vida de um agente definido pela FIPA.
Fonte: JADE (2010).

A figura 14 apresenta um exemplo de uma implementação de um agente no *framework* Jade.

```
package examples.hello;

import jade.core.Agent;

public class HelloWorldAgent extends Agent {

  protected void setup() {
        System.out.println("Hello World! My name is "+getLocalName());

        // Make this agent terminate
        doDelete();
  }
}
```

Figura 14 - Código fonte da uma implementação de um agente.

Em relação à comunicação, o *framework* Jade utiliza o padrão ACL (*Agent Communication Language*) para a criação de suas mensagens. A classe Java *ACLMessage* das *APIs* do *framework* Jade representa as mensagens ACL que podem ser trocadas entre os agentes.

Para um agente criar uma mensagem, deve-se criar um novo objeto do tipo *ACLMessage*, colocar atributos com valores apropriados e, finalmente, chamar o método *Agente.send()*. Por outro lado, para um agente receber uma mensagem, deve-se chamar o método *receive()* ou *blockingReceive(),* onde ambos são implementados na classe *Agent*. Além disso, esta classe define um conjunto de constantes que devem ser utilizadas como referências para aos tipos da FIPA (exemplo: *REQUEST*, *INFORM* etc).

Quando um novo objeto do tipo *ACLMessage* é criado, uma dessas constantes deve ser passada, com o objetivo de que o receptor saiba qual o tipo da mensagem recebida. Para responder uma mensagem recebida, o programador pode utilizar o método *createReply()* da classe *ACLMessage*, pois esta cria um objeto de resposta válido para o agente que o enviou. A figura 15 mostra um trecho de código de programa em que uma mensagem é construída.

```
protected void beginConversation() {
   // start a rumour
   ACLMessage rumour = new ACLMessage( ACLMessage.INFORM );
   rumour.setContent( RUMOUR );
   rumour.addReceiver( randomGuest( null ) );
   send( rumour );

   // introduce two agents to each other
   doIntroduction( randomGuest( null ) );
   setPartyState( "Swinging" );
}
```

Figura 15 - Trecho de código da criação de uma mensagem.

Esta Seção apresentou as funcionalidades básicas do *framework* Jade. Existem diversas outras que o compõe e que podem ser estudadas em mais detalhes no sítio do Jade (2010). Para o presente trabalho, estas funcionalidades básicas são suficientes para o entendimento da proposta.

4.3 ARQUITETURA DO FRAMEWORK DROOLS

O conteúdo desta Seção está baseado principalmente no sítio do *framework Drools* (DROOLS, 2010), que contém informações relacionadas à arquitetura e ao funcionamento deste *framework*. O projeto do *framework Drools* pertence à divisão *Jboss* e tinha o antigo nome de *Jboss Rules,* porém após a compra da empresa *Jboss* (que ocorreu no decorrer deste trabalho) pela empresa norte americana *Red Hat*, alguns *frameworks* da divisão *Jboss* mudaram de nome, pois agora existe a versão licenciada e a versão gratuita. O *framework Drools* utilizado neste trabalho é gratuito. A versão 5.0 do *framework Drools* foi escolhida

para ser utilizada no trabalho por ser uma implementação madura e com as funcionalidades necessárias para a construção da proposta.

O *Drools* é um *framework* que compõe um sistema de gerenciamento de regras (BRMS – *Business Rules Management System*) e um mecanismo de regras em código aberto baseado em padrões, que desta forma, pretende simplificar o gerenciamento e a modificação das regras.

De acordo com as informações do sítio do *framework Drools*, a utilização de um sistema de gerenciamento de regras ainda é uma incógnita, pois o discernimento de como melhor utilizá-la está relacionada ao conhecimento profundo de suas principais características, que por enquanto, não são de conhecimento coletivo, uma vez que estes conceitos são novos. De maneira geral, sua utilização deve ser feita em situações em que as regras de funcionamento devem ser exteriorizadas ao sistema, de tal forma que o usuário tenha facilidade de manipulá-las a qualquer momento, principalmente para resolver problemas complicados nos quais um conjunto de regras e fatos interagem entre si na busca de uma solução.

Um BRMS é um "motor" de execução de regras de negócio. Regras de negócio são estruturas que codificam decisões do negócio de alguma maneira, normalmente de uma maneira simples, tal qual uma condicional se/então .

Como um exemplo, segue uma regra hipotética relacionada a um seguro de carro:

- *Se o carro é azul,*
- *Se o carro é um carro esportivo,*
- *Se o motorista é homem,*
- *Se o motorista tem idade entre 18 e 25 anos,*
- *Se a cidade do motorista é São Paulo,*
- **Então, aumentar 25% do valor do seguro**.

A utilização desse tipo de regra não é novidade e normalmente este tipo de inteligência pode ser encontrado nos programas de sistemas tradicionais. A mudança de paradigma na utilização de um BRMS não está no significado das regras, mas em como essas são expressas no sistema. Neste novo paradigma de tecnologias de agentes, as regras são expressas por meio de padrões se/então/senão, além de estarem localizadas em uma entidade fora do sistema.

O principal elemento que compõe um BRMS é o algoritmo interno que direciona sua execução. Alguns BRMS mais simples utilizam em seu funcionamento uma cadeia procedural lógica, em conjunto com a ordem em que as regras são especificadas. Outros, mais sofisticados, utilizam algoritmos como *Rete*, *Treat* e *Leaps* para conectar os fatos às regras, determinando qual regra deve ser executada e em que ordem. Pode-se definir os fatos como os valores que são submetidos às regras. Conforme o exemplo anterior, seria algo como:

- *Carro (Azul, Esportivo)*
- *Motorista (Homem, 20 anos)*

O *Drools*, assim como outros BRMS, utiliza o algoritmo *Rete*, um algoritmo desenvolvido por Charles Forgy em 1974. Embora um detalhamento do algoritmo *Rete* esteja fora do escopo deste trabalho, de uma maneira simplificada, em relação ao seu funcionamento, o algoritmo *Rete* cria uma árvore a partir das regras, como uma máquina de estados. Os fatos entram no topo desta árvore como parâmetros das regras, e vão descendo até a base, conforme encontram as condições em que são verdadeiras.

A estratégia de utilizar um algoritmo específico para direcionar a execução de regras torna o BRMS mais complexo, porém, além de torná-lo mais eficiente, resolvem o problema de regras repetidas, que são tratadas de maneira transparente para que cada regra seja executada somente uma vez.

A escolha de onde utilizar um BRMS deve ser feita de maneira cautelosa, uma vez que utilizá-lo para resolver qualquer solução pode tornar os sistemas mais lentos em diversos aspectos. Um exemplo de construção em que a utilização de um BRMS não deve trazer grandes vantagens é a criação de um calendário de professores e disciplinas, pois as regras para resolução do problema não são repetitivas e quando alteradas não geram mudanças na execução. Em contrapartida, em construções tais como a cotação de um seguro, em que as regras são repetitivas e funcionam por meio de combinações entre si, o cenário é mais favorável à utilização de tal solução.

4.3.1 FUNCIONAMENTO DO FRAMEWORK DROOLS

O *Drools* utiliza uma notação própria para a criação de suas regras, apesar de sua execução estar relacionada diretamente com a linguagem Java. Essa linguagem própria de descrição das regras é denominada DRL (*Description Rules Language*) e controla a edição de todas as regras que podem ser executadas no *framework*.

A figura 16 apresenta, no formato DRL, a regra descrita anteriormente.

```
package com.insurance.auto
import com.insurance.auto.Driver
import com.insurance.auto.Car
import com.insurance.auto.Policy
import com.insurance.auto.Style
import com.insurance.auto.Color
import com.insurance.auto.Percentage;
  rule "High Risk"
        when
                Car ( style == Style.SPORT, color==Color.BLUE )
                Driver( male=true, age >= 18, age <= 25, city = 'Sao Paulo' )
                $p: Policy ()
        then
                $p.increasePremium( new Percentage( 25 ) );
```

Figura 16 - Regra escrita em formato DRL.

Para a edição das regras, o *Drools* possui um *plug-in* que funciona com o *Eclipse*, um ambiente de desenvolvimento integrado (IDE – *Integrated Development Environment*) Java, que tem o objetivo de facilitar essa tarefa.

Para a execução das regras não é necessário um servidor. O BRMS funciona na forma de um conjunto de *APIs* que pode ser empacotado e executado junto ao sistema, porém, em alguns casos, necessita-se de uma estratégia para que as regras estejam em um lugar centralizado. Para isso, pode-se utilizar tecnologias tais como CORBA (*Common Object Request Broker Architecture*), EJB (*Enterprise JavaBeans*) ou SOA (*Service-Oriented Architecture*), com o objetivo de centralizar essas informações. Essa característica, entretanto, não faz parte do escopo do BRMS.

Um dos desafios na utilização de um BRMS é o gerenciamento das regras, tornando aconselhável a utilização de um repositório centralizado. A construção deste repositório pode utilizar diversas estratégias, que vão desde a criação de arquivos textos até repositórios em banco de dados. Em qualquer uma dessas estratégias existem pontos fortes e fracos, uma vez que alguns ganham na simplicidade, mas perdem no controle, e outros ganham no controle, mas perdem na simplicidade. O *Drools* aceita as mais diversas abordagens, porém, o mais comum em sua utilização é a estratégia de arquivo texto.

Para a proposta de solução do presente trabalho será adotada a estratégia de criação de regras em arquivo texto. Esta foi escolhida pela sua característica de simplicidade pois neste momento não é necessário um alto nível de controle, um vez que as regras em um primeiro momento não serão alteradas com freqüência, porém, é aconselhável que dependendo das características de um trabalho, as regras devam ficar em repositórios centralizados para facilitar o gerenciamento e o controle.

Ainda, para atender a proposta de projeto deste trabalho, foram pesquisadas diversas implementações de tabelas de decisão que mais se adequavam às necessidades do presente estudo. Desta maneira, optou-se por utilizar o *framework Drools*, que tem como uma das suas principais características utilizar uma arquitetura *OpenSource*. Esta característica foi item fundamental na decisão, uma vez que existia uma premissa de que o *framework* pudesse ser modificado para a criação dos aspectos relacionados à *Tecnologia Adaptativa* com o objetivo de criar um ambiente adaptativo.

4.3.2 TABELAS DE DECISÃO

As regras de negócio deveriam ser editadas por pessoas que conhecem o negócio, porém, pelo fato da linguagem de descrição de regras não ser intuitiva, os usuários, que muitas vezes não têm conhecimento de programação, acabam delegando esta responsabilidade de editar as regras para os programadores. Esta situação cria um alto risco de falhas, uma vez que podem ter regras que não condizem com a realidade que o solicitante pediu. Por outro lado, quem conhece o negócio muitas vezes não é motivado a escrever as regras, uma vez que a utilização de uma notação se/então/senão nem sempre é clara para uma pessoa que não

conhece um pouco de lógica *booleana*, criando assim, um ambiente de desconfortável e sem um responsável.

Com o objetivo de tornar esse ambiente menos desconfortável e tornar mais próxima as duas perspectivas (usuários e programadores), o *Drools* permite a utilização de tabelas de decisões que, conforme abordado anteriormente, tem uma notação mais fácil e intuitiva, podendo ser compreendida facilmente.

Uma boa estratégia para a criação de uma tabela de decisão é que ela seja criada em conjunto com os desenvolvedores, de maneira a agregar e organizar as informações que serão as entradas de dados e as regras de negócio, para que as aplicações atendam suas necessidades. Em seguida, é necessário preocupar-se com o gerenciamento dessas regras.

O *Drools* permite que seja utilizada como ferramenta de edição o *Microsoft Excel®*, com o objetivo de fazer o gerenciamento das regras, como pode-se ver no exemplo da figura 17.

		Regras		
Condições	Idade maior igual	18	26	45
	Idade menor igual	25	40	60
	Sexo do condutor	Masculino	Feminino	Masculino
	Cor do carro	Azul	-	-
	Tipo do carro	Esporte	Esporte	Tradicional
	Cidade	São Paulo	São Paulo	São Paulo
Ação		Aumenta 25% o prêmio	Aumenta 15% o prêmio	Aumenta 5% o prêmio

Figura 17 - Tabela de decisão convencional.

Esta estratégia garante uma maior facilidade entre os responsáveis pela criação das regras, porém, ela somente não atende completamente o processo. É necessário que quem edita as regras tenha um ambiente de simulação, uma vez que ao utilizar essa abordagem, as regras possam vir a ser testadas e desta forma melhorar a assertividade do problema.

A vantagem da utilização de planilhas eletrônicas como o *Microsoft Excel®* para a criação de tabelas de decisão está no fato de facilitar a inclusão das regras por parte do usuário, porém, o *Drools* aplica um processo de transformação nessas tabelas e as transforma em regras no formato padrão de execução DRL, uma vez que, em sua essência, ambas são equivalentes.

Uma vez que o *Drools* tem a capacidade de gerar regras a partir de uma tabela de decisão, observou-se que seria possível a criação de uma estratégia para a construção de um dispositivo adaptativo a partir dele, fazendo com que desta maneira, o *Drools* tenha o papel de dispositivo subjacente, criando uma camada adicional ao *framework,* de tal forma que este faça o papel de camada adaptativa. Esta estratégia se torna interessante, uma vez que são utilizadas todas as funcionalidades da implementação do *Drools* em conjunto com as contribuições que a Tecnologia Adaptiva tem a oferecer.

A Seção a seguir detalha o desenvolvimento da estratégia de criação da camada adaptativa e sua inserção no *framework Drools* para a criação do dispositivo adaptativo.

4.4 CAMADA ADAPTATIVA

Conforme apresentado no Capítulo 3, um dispositivo adaptativo é composto por uma camada subjacente e uma camada adaptativa. No presente trabalho, a camada subjacente é implementada pelo *framework Drools* em sua implementação padrão. E para a criação da camada adaptativa, será feita uma implementação adicional que será agregada ao *framework Jboss Rules*.

Em relação aos elementos que compõem um DRL, o *Drools* tem os seguintes elementos:

- *package*– pacote que organiza o conjunto de regras.
- *rule* – nome da regra.
- *activation*-group – o grupo ao qual a regra pertence.
- *When* – condição em que a regra será executada.
- *Then* – ação que irá ocorrer como resultado da execução da regra.

A figura 18 apresenta um trecho de código de programa com os elementos do DRL destacados.

```
package org.unifesp.sisnefro.rules

import org.unifesp.sisnefro.common.Paciente;

/*
Até 40 anos 0
41-49   1
50-59   2
60-69   3
Maior que 70    4
*/
rule "idade1"
activation-group "idade"
    when
        p : Paciente(idade > 40,idade<50)
    then
        p.somaPontos(1);
end
rule "idade2"
activation-group "idade"
    when
        p : Paciente(idade >= 50,idade<60)
    then
        p.somaPontos(2);
end
rule "idade3"
activation-group "idade"
    when
        p : Paciente(idade >= 60,idade<70)
    then
        p.somaPontos(3);
end
rule "idade4"
activation-group "idade"
    when
        p : Paciente(idade >= 70)
    then
        p.somaPontos(4);
end
```

Figura 18 - Elementos de um arquivo DRL.

Neste contexto, para a execução de um conjunto de regras, um arquivo com o formato apresentado é gerado e sua execução ocorre de maneira linear, ou seja, uma regra de cada vez.

Para a execução de um determinado conjunto de regras, o *Drools* funciona por meio da união de um arquivo DRL, os fatos que deverão ser submetidos às regras e o motor de execução (*APIs* do *Drools*), tal como pode ser observado na figura 19, que exemplifica este esquema.

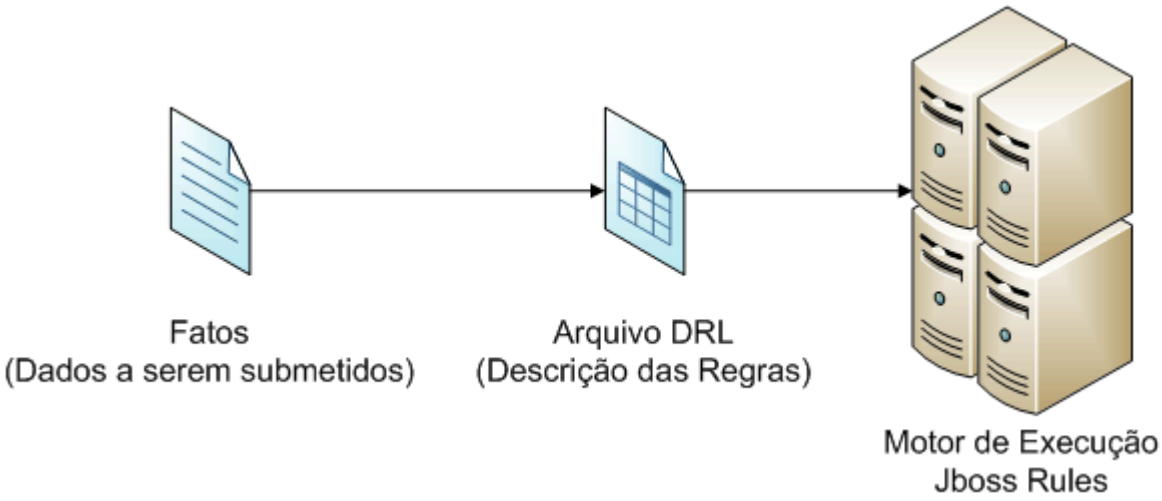

Figura 19 - Ambiente de execução de regras.

Com o objetivo de incluir novas funcionalidades ao motor de execução do *Drools*, seus criadores desenvolveram o conceito de *Listeners*, que são elementos que podem ser incluídos dinamicamente a uma determinada execução com a finalidade de agregar funcionalidades que a configuração padrão não atende.

Neste trabalho, pretende-se utilizar esta estratégia para a criação da camada adaptativa, conforme detalhamento a seguir:

1 – Criação de um arquivo para especificação de regras adaptativas.

Será criado um arquivo do tipo *XSD (XML Schema Definition)*, que será utilizado como esquema de validação para um padrão de criação de regras adaptativas.

2 – Criação de um arquivo com as regras adaptativas.

Será criado um arquivo com as regras adaptativas que devem ser incluídas na execução de um conjunto de regras tradicionais. Essas regras adaptativas devem seguir o padrão criado no arquivo *XSD* de especificação de regras.

3 – Criação de um *Listener* para execução de regras adaptativas.

Será criado um *listener* que intercepte as execuções das regras, de tal forma, que ao executar cada regra, seja verificado se existem ações adaptativas associadas e, caso existam, elas sejam executadas.

4.5 IMPLEMENTAÇÃO DO DISPOSITIVO ADAPTATIVO

A construção do dispositivo adaptativo utilizou a estratégia de manter a camada subjacente como sendo o *framework Drools* em sua construção padrão, uma vez que este tem as vantagens de utilizar padrões abertos e estabilizados. Desta forma, no aspecto da construção do dispositivo adaptativo, o presente trabalho direcionou seus esforços para a criação de uma camada adicional ao *framework Drools* que fez o papel de camada adaptativa.

Em relação à criação da camada adaptativa, adotou-se a utilização de uma Tabela de Decisão Adaptativa, dentre as diversas implementações relacionadas à Tecnologia Adaptativa. Esta escolha ocorreu devido à facilidade de implementação das características da Tabela de Decisão dentro do *framework Drools*, porém, essa facilidade se restringe apenas à criação da Tabelas de Decisão e fez-se necessário a criação de uma funcionalidade adicional para que o *framework* pudesse contemplar características de adaptabilidade.

Para a criação de um ambiente adaptativo a partir de uma tabela de decisão, primeiramente foi necessário procurar uma tecnologia que garantisse a integridade dessas

normativas, e em seguida, definir uma linguagem para descrição das normativas de regras adaptativas. Em relação à tecnologia para descrever as regras, foi utilizada a tecnologia *XSD (XML Schema Definition)* que tem a característica de definir esquemas em arquivos *XML* estabelecendo padrões a serem seguidos. Este mecanismo foi criado com o objetivo de garantir que a criação dos arquivos de descrição de regras adaptativas seguisse um padrão preestabelecido.

Portanto, após a definição da tecnologia, foi criado um arquivo em formato *XML* com os esquemas que modelam as regras adaptativas a partir de regras tradicionais e relacionando as regras tradicionais a possíveis ações adaptativas. Este arquivo recebeu o nome de *adaptive.xsd* (figura 20) e tem papel fundamental no direcionamento da construção da Tabela de Decisão Adaptativa da proposta.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://endo.com/xml/ns/adaptive"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified" >

    <xsd:element name="adaptive-rules">
       <xsd:complexType>
          <xsd:sequence>
             <xsd:element name="adaptive-rule"  minOccurs="1" maxOccurs="unbounded" >
               <xsd:complexType>
                  <xsd:sequence>
                  <xsd:choice>
                     <xsd:element name="target-rule"   type="xsd:string" />
                     <xsd:element name="target-group-rule" type="xsd:string"/>
                  </xsd:choice>
                     <xsd:element name="before" type="xsd:string"/>
                     <xsd:element name="after" type="xsd:string"/>
                  </xsd:sequence>
               </xsd:complexType>
             </xsd:element>
          </xsd:sequence>
       </xsd:complexType>
    </xsd:element>


</xsd:schema>
```

Figura 20 - Arquivo de definição de esquema de regras adaptativas (*adaptive.xsd*).

O quadro 2 descreve em mais detalhes os elementos que compõem o esquema de definição do arquivo de regras adaptativas.

Quadro 2 – Elementos do arquivo de definição de regras adaptativas.

Nome	**Descrição**
adaptative-rules	Permite a criação de um conjunto de regras adaptativas
adaptativa-rule	Representa uma regra adaptativa que irá ser executada a partir de uma ou mais regras tradicionais.
target-rule	Nome da regra tradicional que irá iniciar a execução da regra adaptativa descrita.
target-group-rule	Nome do grupo de regras tradicionais que irá iniciar a execução da regra adaptativa descrita (opcional).
Before	Nome do arquivo de regra que deve ser executado antes da execução da regra tradicional.
After	Nome do arquivo de regra que deve ser executado após a execução da regra tradicional.

O *framework Drools* tem a linguagem Java como base de implementação e é formado por uma arquitetura de características de padrões abertos e conceitos de Orientação a Objetos, portanto, torna-se possível a extensão de suas funcionalidades por meio do uso de conceitos de Herança e, principalmente, de Polimorfismo.

Em relação a sua arquitetura, o *framework Drools* tem a característica de possuir um ambiente de execução que pode ser facilmente modificado. Dentre os diversos elementos que permitem criar este ambiente flexível destaca-se a possibilidade da inclusão dos chamados

listeners que têm a responsabilidade de interceptar as regras em tempo de execução com o objetivo de modificar o seu fluxo ou seu comportamento (figura 21).

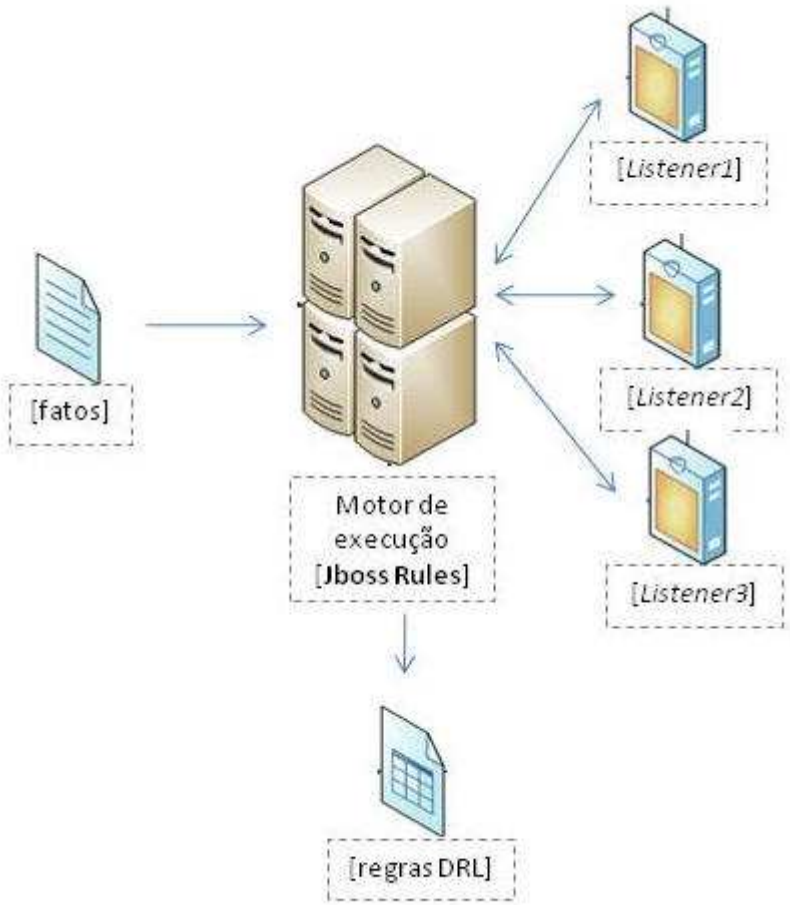

Figura 21 - – Relacionamento entre o motor de regras e os *Listeners*.

Existem diversos *listeners* previamente construídos no *framework Drools* , relacionados principalmente a criação de *logs* e monitoramente de desempenho. A arquitetura de utilização dos *listeners* permite que seus componentes apesar de estarem disponíveis possam ou não ser utilizados na execução de um programa, criando assim um ambiente de execução flexível para atender as mais diversas necessidades.

A estrutura de um *listener* é basicamente composta por métodos que são executados antes da execução de uma regra e métodos que são executados após a execução de uma regra.

Um fato importante nesta estrutura é que os métodos têm acesso a todos os dados da execução (dados dos fatos, regras, etc.).

A estratégia utilizada no presente trabalho foi a criação de um *listener* com o objetivo de ser o intermediador entre o *framework Drools* e as regras adaptativas. Essa estratégia foi executada por meio da construção do *listener AdaptativeListener*, que na arquitetura do *framework Jboss Rules*, é uma classe Java, filha da classe *AgendaEventListener*.

O *AdaptativeListener* foi construído com a finalidade de fazer a comunicação entre o *framework Drools* e implementações externas, mais especificamente com a linguagem de regras adaptativas criadas por meio das definições feitas pelo arquivo *XSD*. A característica de ser um *listener* permite que a execução de um programa possa ou não ter um ambiente adaptativo facilitando assim os testes e permitindo uma maior flexibilidade.

Como a definição do padrão para a criação das regras adaptativas, foi estabelecido e o intermediador entre as regras adaptativas e o motor de execução de regras foi construído (*listener AdaptativeListener)*, o próximo passo foi estabelecer um processo para a criação de tabelas de decisão adaptativa, conforme segue abaixo.

1- Criar o conjunto de regras tradicionais que atendam as necessidades do problema a ser resolvido. O *framework Drools* permite a criação de tabelas de decisão em seu modelo tradicional com o objetivo de fazer o papel de regras de decisão, porém, no motor de execução do *framework*, as tabelas são convertidas para regras em modelos tradicionais (*if then else*).

2- Criar um arquivo padrão *XML* seguindo as definições descritas no arquivo *adaptative.xsd* (Conforme exemplo da figura 20) com as informações relacionadas às regras adaptativas e suas relações/ações com as regras tradicionais descritas anteriormente.

3- Criar um ou mais arquivos com as regras adaptativas (em formato tradicional do *framework Drools - if then else*) que serão executadas de acordo com os direcionamentos feitos no arquivo de definição (passo anterior).

4- Adicionar no programa de execução das regras o *listener AdaptativeListener* para que assim seja criado o contexto adaptativo dentro do *framework Drools*.

A figura 22 demonstra os elementos que compõem o processo de execução do *Adaptative Listener*.

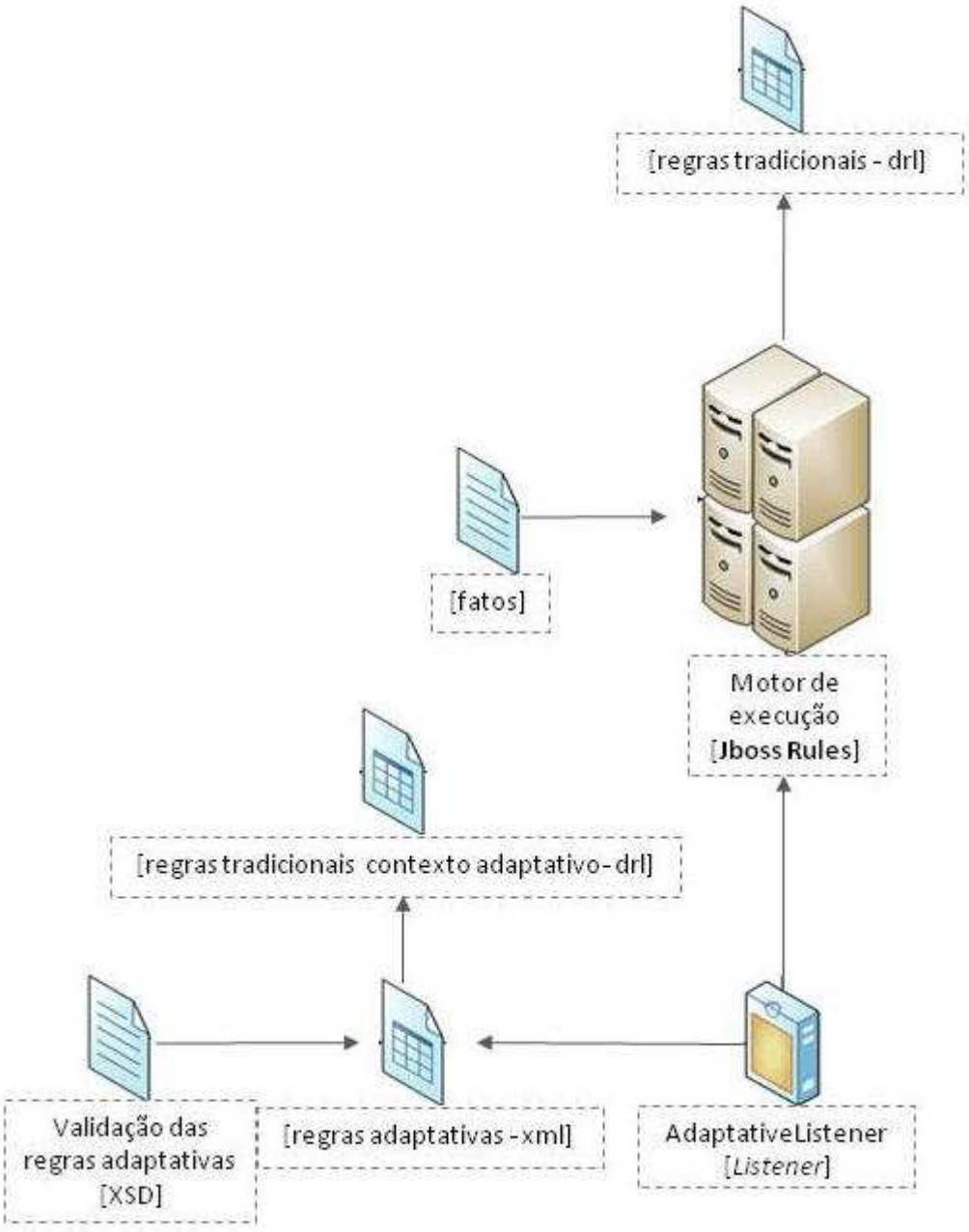

Figura 22 - Elementos que compõem a execução do *AdaptiveListener*.

Ao final de todas as ações descritas neste item, foi finalizada a descrição de como se dá a criação de uma Tabela de Decisão Adaptativa com a responsabilidade de ser o motor de execução das regras.

4.6 RACIONALIZAÇÃO DE AGENTES

Conforme abordado na primeira parte deste trabalho, uma das características que compõem um agente é seu aspecto cognitivo. Nos modelos cognitivos, normalmente os agentes possuem um conhecimento e funcionam racionalmente, isto é, raciocinam para construir um plano de ações para resolver um determinado problema, sendo essas características que os diferem de um programa convencional e dos agentes reativos.

Em relação às características de um agente cognitivo, pode-se citar: sua capacidade de alterar seu funcionamento a fim de adaptar-se melhor ao seu ambiente; estão continuamente em funcionamento; são sociáveis (possuem a capacidade de comunicação entre si); possuem representação de conhecimento explícita no código (conhecimento introspectivo); o mecanismo de controle é deliberativo (o agente raciocina sobre que ações realizar); tem memória e, portanto, pode lembrar de ações realizadas no passado; e normalmente, as sociedade criam um meio de interação entre os agentes (RAO; GEORGEFF, 1995).

De acordo com o modelo proposto por Demazeau e Muller (1990), o conhecimento que o agente possui é formado a partir da sua percepção do ambiente e da comunicação com outros agentes. Dado este conhecimento e uma meta, o agente gera um conjunto de possíveis planos que atingem esta meta. Dadas estas possibilidades, o agente delibera sobre o melhor plano a ser executado. Aqui se pode dizer que “o agente fez x porque tem por objetivo y e x faz parte de um plano que leva à satisfação de y”. Este processo pode ser visualizado na figura 24.

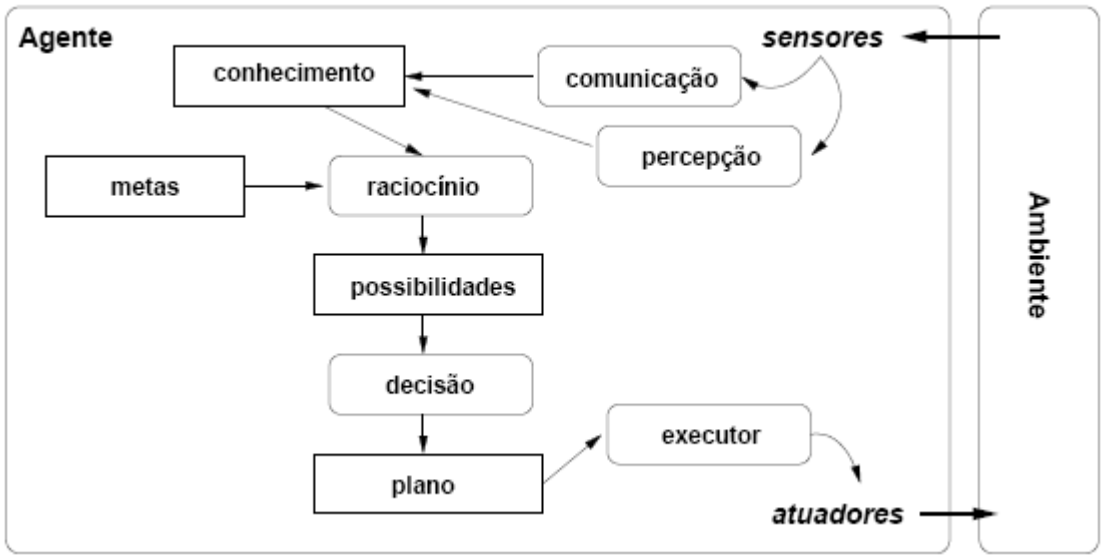

Figura 23 - Arquitetura de um agente cognitivo.
Fonte: DEMAZEAU;MULLER (1990).

Na proposta deste trabalho, pretende-se criar o dispositivo adaptativo tornando-o possível de ser inserido dentro de um agente em um sistema multiagentes.

Desta forma, ao inserir o dispositivo adaptativo dentro de um agente, pretende-se que este possa ser localizado e executado seguindo os padrões dos sistemas multiagentes, de tal maneira que estarão sendo agregadas as características da Tecnologia Adaptativa e um SMA com o objetivo de criar um agente com contexto racional (figura 24).

A estratégia proposta pretende criar um agente racional que pode ter seu comportamento manipulado externamente, isto é, uma vez que suas regras de racionalização estarão fora do contexto do agente, essas podem ser manipuladas pelo usuário com a finalidade de efetuar ajustes.

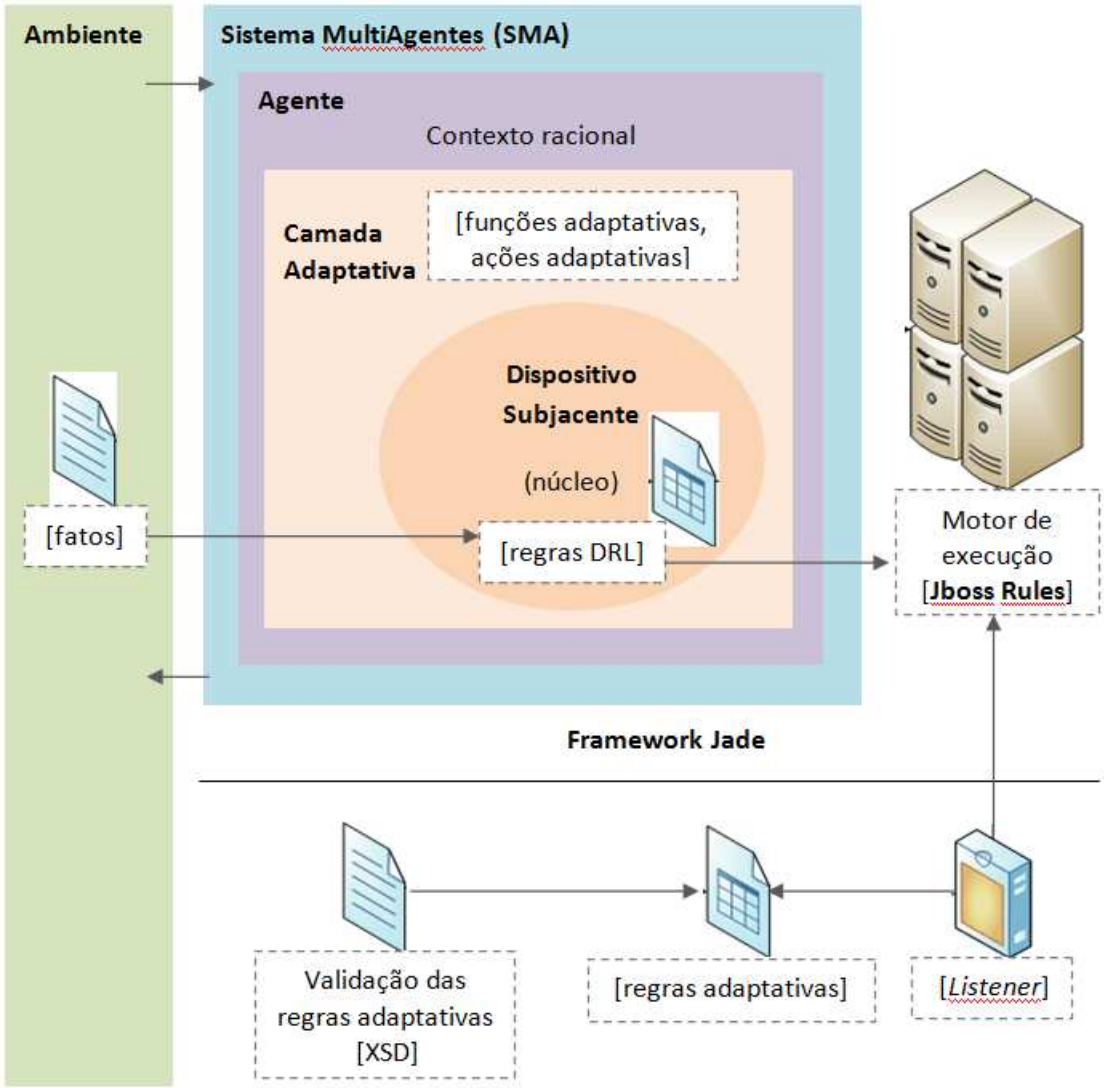

Figura 24 - Modelo de implementação do trabalho.

4.7 IMPLEMENTAÇÃO DO AGENTE RACIONAL

A construção do agente racional teve como base a utilização do *framework Jade*, uma vez que, desta forma, grande parte das implicações relacionada à criação de um SMA e dos aspectos dos agentes são resolvidos pelo *framework*, tornando assim, viável que o trabalho seja direcionado para a criação de comportamentos racionais.

A execução de um agente no *framework Jade* está relacionada ao que é denominado internamente de comportamento (*Behavior*), que tem a responsabilidade de controlar as rotinas de conversação entre os mais diversos agentes. Além das rotinas de conversa entre os agentes, é necessária a criação de rotinas que tornam o este fluxo de conversação dinâmico, isto é, dependendo da mensagem de entrada, uma mensagem de resposta é enviada. Portanto, a racionalização de um agente no *framework* Jade está relacionada diretamente à criação de comportamentos, de tal forma que esses possam se interagir e realizar suas tarefas criando assim um ambiente racional.

Com o objetivo de criar um agente racional, foi criado o *RacionalAgent* que tem como responsabilidade ficar disponível para troca de mensagens com outros agentes que necessitam de uma interação para comportamentos racionais predeterminados (figura 25). Para que esses comportamentos predeterminados possam ser inseridos dinamicamente sem a necessidade de alterações no agente racional, foi utilizada uma estratégia de carregamento dinâmico de classes do *Java* através de arquivos de propriedades, tornando assim, possível a inclusão e a retirada de comportamentos a qualquer momento do ciclo de vida do agente.

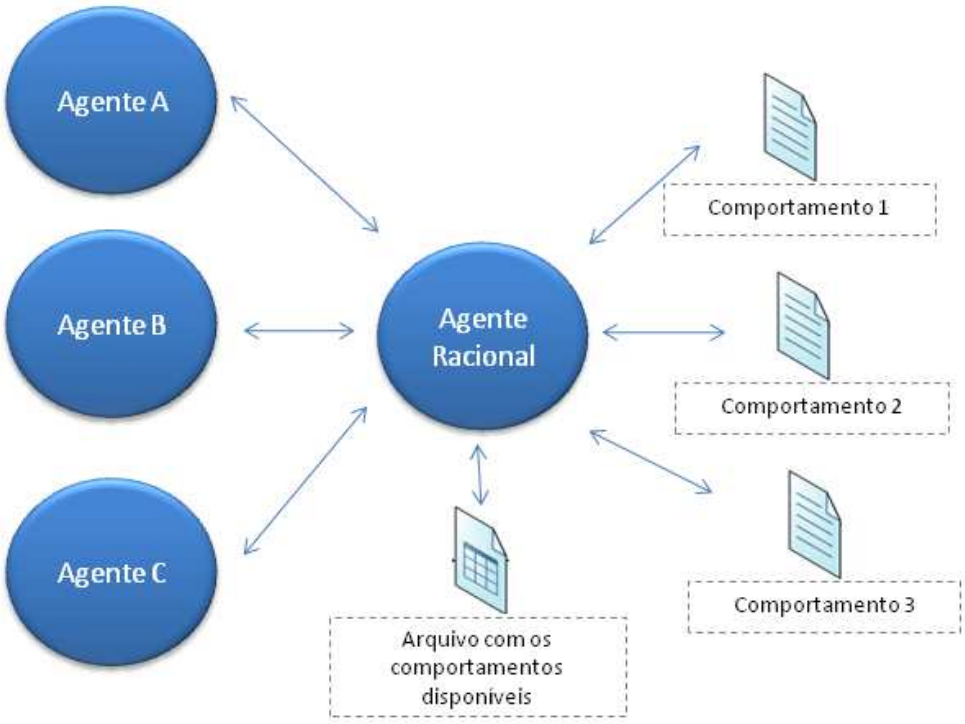

Figura 25 – Modelo do Agente Racional.

Os agentes têm sua racionalidade relacionada aos comportamentos que o compõem e a troca de mensagens entre si. Neste trabalho, foi direcionada a racionalidade relacionada aos comportamentos e não a troca de mensagens entre si, sendo que este assunto pode vir a fazer parte de um trabalho futuro.

5 SIMULAÇÕES DE CASOS ESPECÍFICOS

De acordo com a OMS, que é a Organização Mundial da Saúde (2006), o colapso nos sistemas de saúde deve ocorrer por volta de 2015. Apesar dos esforços de médicos e pesquisadores na tentativa de melhorar os sistemas de saúde tornando-os cada vez mais eficientes, os gastos mundiais em saúde aumentam de maneira desenfreada. No Brasil, os investimentos feitos pelo governo em relação à saúde são precários, a quantidade de profissionais não atende as necessidades de todos os pacientes e a qualidade do atendimento não é adequada. Diante deste cenário, uma parcela da população que pode, utiliza-se de empresas privadas.

A utilização da tecnologia da informação para influenciar os mais diversos segmentos da sociedade tem um papel fundamental nos mais diversos segmentos da sociedade e, como não poderia ser diferente, sua utilização nas atividades relacionadas à Saúde tem tido um papel de destaque (CASTELLS, 1999).

Entidades como a Organização Mundial da Saúde afirmam que a tecnologia da informação tem papel-chave no aumento da qualidade do sistema de saúde (OMS, 2006), por meio do direcionamento de esforços para utilizar essas tecnologias no apoio, expansão e aumento da qualidade dos cuidados e serviços aos pacientes (CHAU; HU, 2004).

Segundo Porter e Teisberg (2007), a percepção de qualidade por parte dos pacientes está cada vez pior. Neste sentido, a inexistência de um processo na criação de uma fila de atendimento de pacientes pode estar relacionada diretamente com a percepção de falta de qualidade na prestação de serviços médicos.

Considerando o problema da fila de atendimento, tal como mencionado, estudou-se a necessidade do Departamento de Nefrologia da UNIFESP em criar um sistema de agendamento de consulta de pacientes com problemas nefrológicos. Este sistema de agendamento se difere dos sistemas de agendamento tradicionais, pois deve ter a capacidade de ordenar os pacientes de acordo com critérios de prioridade preestabelecidos. Esses critérios utilizam diversas informações dos pacientes, além de aspectos clínicos laboratoriais.

A criação desse sistema visa sanar a dificuldade em identificar, dentre os pacientes, aqueles que devem ser atendidos com prioridade, daqueles que podem ter um atendimento não prioritário. Além disso, o sistema tem como objetivo melhorar a assertividade do grau de conhecimento do médico que irá atender o paciente, uma vez que, além de existir um número limitado de profissionais, esses têm um grau de conhecimento diferente entre si, e poucos são os que possuem um alto nível de experiência, uma vez que dentre os profissionais, existem diversos residentes de medicina.

Esta simulação de caso específico contempla a criação de um conjunto de agentes com inteligência para categorizar os pacientes por meio de alguns critérios preestabelecidos.

Esses critérios, bem como os dados de pacientes e as regras para este trabalho, foram informados por uma médica da UNIFESP que atualmente está desenvolvendo sua tese de doutorado, e que tem como um dos objetivos estudar uma maneira de categorizar os pacientes com problemas nefrológicos em relação à gravidade de seu estado de saúde.

Em relação à implementação dos critérios preestabelecidos no sistema, estes foram inseridos em uma Tabela de Decisão Adaptativa, com o objetivo de representar a parte racional da execução do agente que pontua os pacientes.

O detalhamento das informações dos pacientes e do cálculo de pontuação pode ser encontrado no Apêndice B.

5.1 DADOS DE ENTRADA

Com o objetivo de validar a efetividade do cálculo de pontuação de pacientes com problemas nefrológicos (Apêndice B), foi feita uma coleta de dados de pacientes com todas as informações necessárias (Apêndice A) para pontuá-lo. Este estudo foi realizado no segundo semestre de 2010.

Com os dados dos pacientes já tabulados, estes foram submetidos a três especialistas da área de nefrologia com objetivo de se categorizar cada um dos pacientes em três tipos, em relação à gravidade do problema renal:

1 – Paciente com leve risco renal.

2 – Paciente com moderado risco renal.

3 – Paciente com grave risco renal.

Desta forma, foram obtidos os dados necessários para a comparação de resultado entre a categorização feita pelos médicos e a categorização feita pelo Agente Racional, com o objetivo de verificar a efetividade das regras preestabelecidas e do funcionamento da Tabela de Decisão Adaptativa.

5.2 IMPLEMENTAÇÃO DO AGENTE RACIONAL DE PONTUAÇÃO

A estratégia de implementação do agente racional de pontuação contempla a criação de um comportamento racional dentro do agente *RacionalAgent*. Para isso, fez-se necessário a criação da classe *NefroBehavior*, que tem como responsabilidade receber as mensagens oriundas de outros agentes, verificar se essas mensagens contêm dados válidos de pacientes, e por fim, utilizar o dispositivo de regras para gerar a pontuação do paciente.

A comunicação entre os agentes ocorre por meio de troca de mensagens, porém, o *framework Jade* categoriza essas mensagens em relação ao conteúdo em dois tipos:

- Mensagem do Tipo Caractere - O conteúdo da mensagem tem o formato texto.
- Mensagem do Tipo Objeto – O conteúdo da mensagem é um objeto Java sem restrição de tipo.

Uma vez que as mensagens do tipo objeto não restringem o tipo de objeto que está sendo trafegado, o *framework* Jade permite a criação de ontologias. Desta forma, é possível criar um vocabulário e semânticas próprias com o objetivo de que o conteúdo das mensagens trafegadas entre os agentes tenha, obrigatoriamente, um padrão predefinido.

Como o comportamento *NefroBehavior* está relacionado diretamente com a manipulação dos dados de pacientes com características nefrológicas e, pensando que em um trabalho futuro, as funcionalidades de comunicação entre os agentes podem ser estendidas, percebeu-se que seria necessário a criação de uma ontologia para modelar este paciente. Esta implementação foi feita por meio da classe *PacienteOntology* que descreve o vocabulário e as semânticas relacionadas aos pacientes relacionados ao presente trabalho. O conteúdo desta classe pode ser vista em detalhes no Apêndice C.

O processo de funcionamento do comportamento *NefroBehavior* pode ser descrito da seguinte maneira:

a) Por meio de um agente requisitante o Agente Racional recebe uma solicitação de cálculo de pontuação contendo supostamente os dados de um paciente nefrológico.
b) Os dados enviados são verificados se seguem o vocabulário predefinido por meio da ontologia *PacienteOntology*.
c) O *NefroBehavior* envia os dados recebidos para o dispositivo adaptativo que contém as regras de pontuação.
d) O Agente Racional recebe a pontuação do paciente em questão.

e) O Agente Racional retorna o paciente já pontuado para o agente requisitante.

Com o objetivo de executar a simulação do caso, foi criado um agente requisitante chamado *MedicoAgent*, que tem como responsabilidade carregar os dados de todos os pacientes e passar um a um para o *AgenteRacional*, que por sua vez, executa o comportamento *NefroBehavior*, que pontua o paciente. Uma vez que todos os pacientes estejam pontuados, o agente *MedicoAgent* classifica os pacientes em uma categoria (paciente com leve risco renal, paciente com moderado risco renal, paciente com grave risco renal), de acordo com uma faixa preestabelecida. A faixa de categorização foi proposta pela professora MSc. Patrícia Albuquerque, conforme apresentado no quadro 3.

Quadro 3 – Faixa de categorização dos pacientes em relação aos pontos alcançados.

Faixa	**Descrição**
De 0% a 25% dos pontos totais possíveis	Paciente com leve risco renal.
Maior que 25% até 50% dos pontos totais possíveis.	Paciente com moderado risco renal.
Maior que 50% até 100% dos pontos totais possíveis.	Paciente com grave risco renal.

Para as regras preestabelecidas no trabalho, o máximo de pontos possíveis (100% dos pontos) são 44 pontos, porém, a estratégia de porcentagem foi estabelecida devido à possibilidade da criação de novas regras de pontuação, gerando assim, uma mudança na pontuação máxima possível.

Esta categorização dos pacientes tem o objetivo de fazer um estudo comparativo entre os resultados apresentados pelo sistema de agentes e a categorização feita pelos médicos.

Após o cruzamento dos resultados da categorização de pacientes pelo sistema e pelos médicos, é possível validar a efetividade da tabela de pontos e do sistema de pontuação.

5.2.1 IMPLEMENTAÇÃO DA TABELA DE DECISÃO ADAPTATIVA NO AGENTE RACIONAL DE PONTUAÇÃO

A estratégia de implementação da tabela de decisão adaptativa contempla primeiramente a criação de uma tabela de decisão tradicional utilizando os critérios discutidos na Seção 5.1. Para a inclusão dessas informações foi criado o arquivo *pontos.drl* (formato de regras padrão do *framework Drools)*, que descreve todas as regras de pontuação dos pacientes. O conteúdo detalhado deste arquivo pode ser consultado no Apêndice B.

Em execuções do processo de pontuação em fases preliminares utilizando somente a tabela de decisão tradicional como contexto racional, observou-se que o resultado da categorização dos pacientes dada pelo Agente Racional apontava categorias maiores ou menores em comparação às classificações realizadas pelos especialistas, com a porcentagem de acertos muito abaixo do esperado. Para que os resultados obtidos pelo Agente Racional fossem mais próximos dos resultados feitos pelos especialistas, foi necessário efetuar ajustes nas regras de pontuação, porém, o autor não possui conhecimento na área para executar tais ajustes. Desta forma, a estratégia adotada foi a criação de um conjunto de regras adaptativas para ajustar a categorização feita pelo modelo tradicional, com objetivo de que em um trabalho futuro, esta estratégia possa servir de ferramenta para o ajuste de regras, ou ainda, para a criação de um contexto de inteligência em que os ajustes são feitos de forma automatizada, por meio de aprendizado.

A estratégia de implementação do contexto adaptativo foi a criação de regras adaptativas para os pacientes que possuíam categorias abaixo ou acima dos categorizados pelos especialistas. Esta construção foi feita por meio da criação dos arquivos *adaptive1.xml* e

adaptive2.xml, que tem em seu conteúdo, todas as regras adaptativas utilizadas na proposta. Além dos arquivos de descrição de regras adaptativas, foi criado um arquivo de regras tradicionais para cada regra adaptativa. Os arquivos *adaptive1.xml* e *adaptive2.xml* podem ser consultados no Apêndice D.

A figura 26 ilustra o esquema de pontuação do paciente mostrando todos os seus elementos e suas interações.

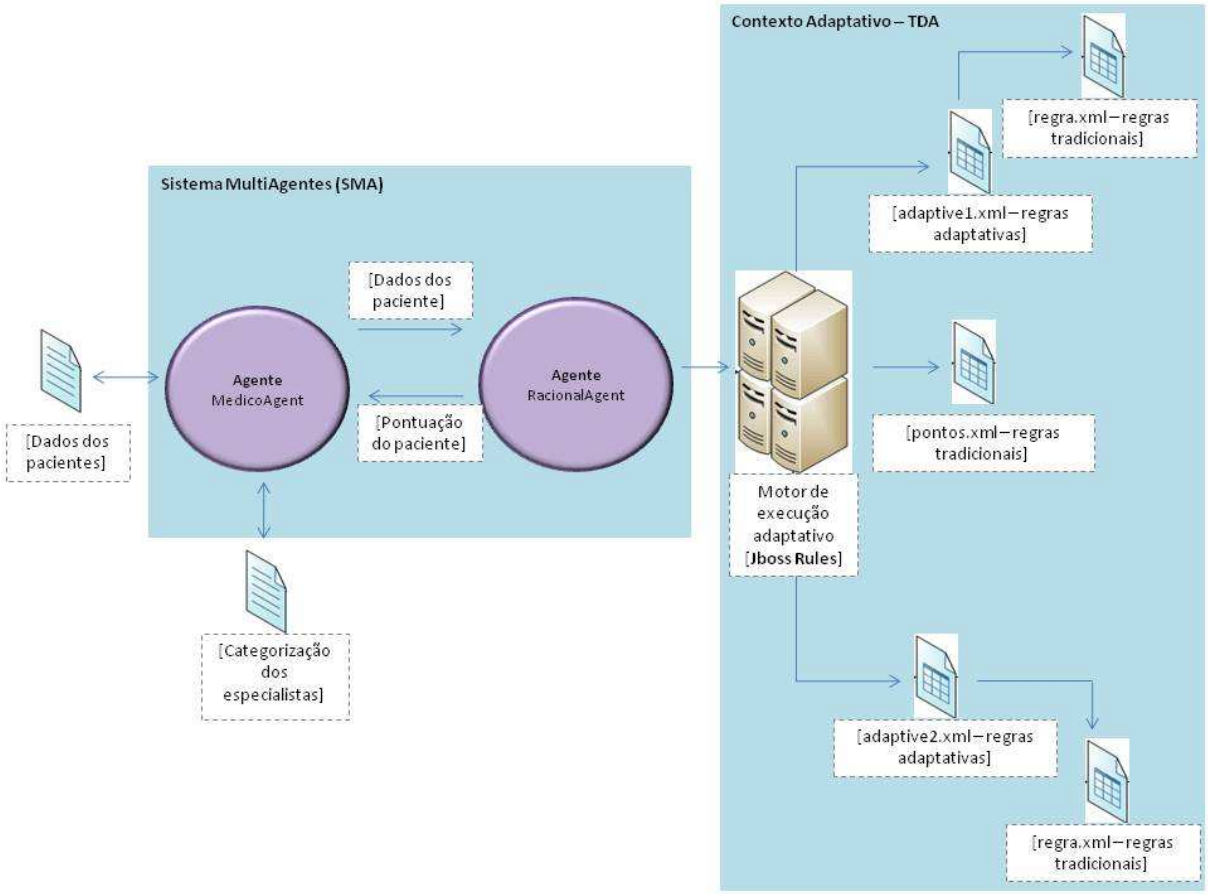

Figura 26 - Esquema do processo de pontuação do paciente.

5.3 RESULTADOS DA PONTUAÇAO DOS PACIENTES

Foram utilizados dados de 147 pacientes, contemplando as mais diversas características, com objetivo de que diversos cenários pudessem ser atingidos.

Após submeter todos os pacientes ao processo de pontuação utilizando somente o dispositivo de regras tradicional, isto é, sem a aplicação das regras adaptativas, obteve-se a pontuação de cada um dos pacientes e suas categorias. Com estes dados, foi feito um confronto entre as categorias designadas pelo agente e a categorização feita por cada um dos três médicos especialistas. O gráfico 1 mostra o resultado obtido deste confronto, no qual cada uma das colunas representa a porcentagem de acerto das categorizações.

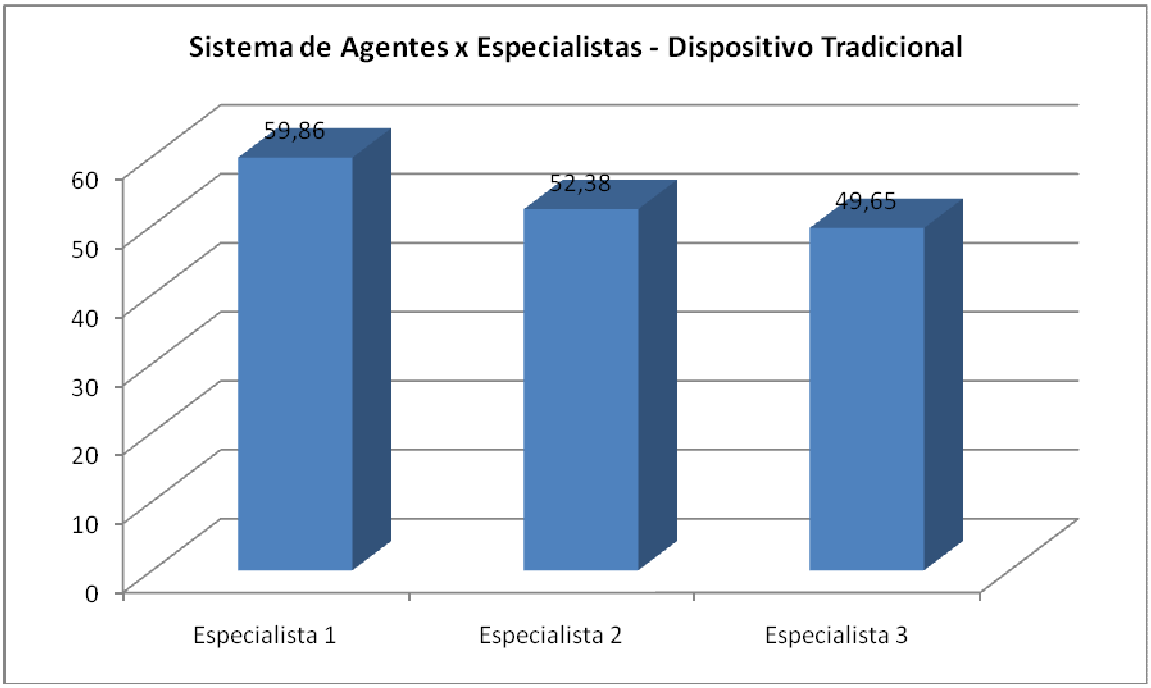

Gráfico 1 - Porcentagem de acerto da categorização de pacientes.

Os resultados obtidos demonstram que a porcentagem de acerto do agente sem a utilização de um dispositivo adaptativo fica abaixo de um nível aceitável, provando que é necessário que fossem feitos ajustes nas regras e/ou na quantidade de informações recebidas.

Em uma análise detalhada dos resultados do experimento, observou-se que as categorizações obtidas pelo agente ficavam às vezes abaixo e às vezes acima da categorização feita pelos especialistas, portanto sem seguir um padrão de comportamento.

Para isso, foram criadas regras adaptativas que foram aplicadas tanto aos pacientes categorizados abaixo quanto aos categorizados acima da classificação indicada pelos especialistas.

A mesma lista de pacientes utilizada pelos especialistas foi submetida ao Agente Racional para pontuação e categorização, porém, quando um paciente recebia uma categoria diferente da dada pelos especialistas, era submetido novamente, utilizando o dispositivo com as regras adaptativas. O resultado desta segunda execução foi então comparado com a categorização feita pelos especialistas.

O gráfico 2 mostra o resultado obtido deste confronto, no qual cada uma das colunas representa a porcentagem de acerto das categorizações após a execução das regras adaptativas.

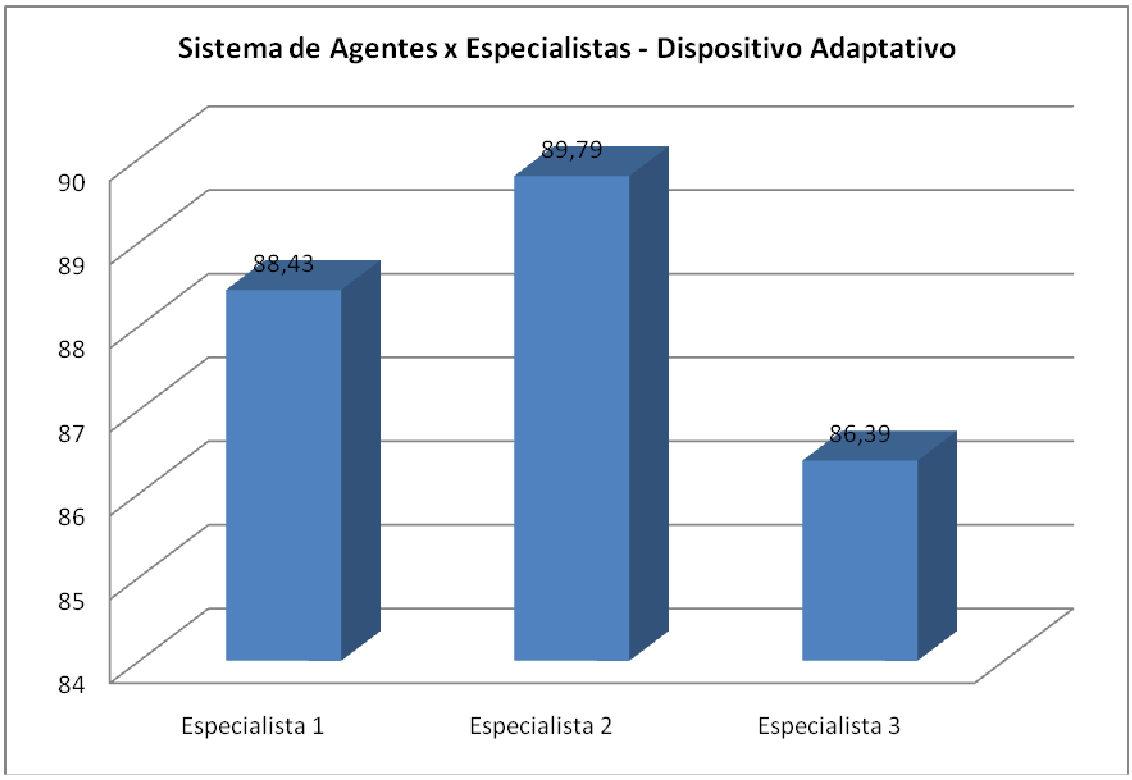

Gráfico 2 - Porcentagem de acerto da categorização de pacientes após a utilização do dispositivo adaptativo.

O gráfico 3 mostra uma comparação entre as porcentagens de acerto entre as duas estratégias, utilizando o dispositivo tradicional e utilizando o dispositivo adaptativo.

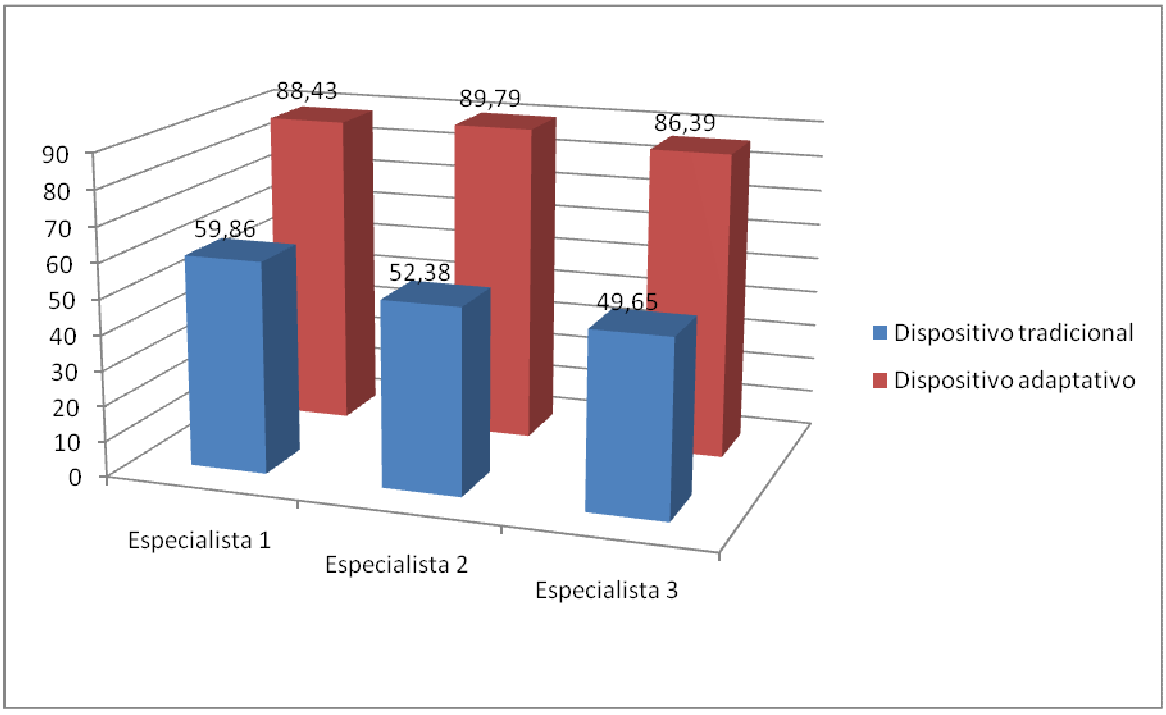

Gráfico 3 - Gráfico comparativo entre as porcentagens de acerto das estratégias.

Observando os resultados, pode-se chegar à conclusão de que, com a utilização da tabela de decisão adaptativa, ocorreu um crescimento médio de 34,24% na porcentagem de acertos.

Este crescimento ocorreu pela inclusão de regras adaptativas criadas pelo autor, porém, acredita-se que caso as regras adaptativas fossem estudas e criadas por um médico especialista, os resultados obtidos poderiam ser ainda mais animadores. Em outro momento, a utilização das regras adaptativas poderia servir de base para estudos na melhoria das regras de decisão tradicionais, uma vez que as regras adaptativas alteram o comportamento do dispositivo sem alterar o funcionamento das regras ditas tradicionais, servindo de base de experimento.

6 CONCLUSÕES E DISCUSSÕES

O principal objetivo deste trabalho foi apresentar uma das possíveis aplicações da Tecnologia Adaptativa em conjunto com o conceito de agentes para a criação de uma arquitetura de agentes racionais. Para comprovar tal proposição utilizou-se a questão do apoio à tomada de decisão em um ambiente de priorização de atendimento a pacientes com problemas nefrológicos.

A viabilidade da proposta foi comprovada por meio do estudo de caso, a partir do qual, foi possível comprovar que a utilização de agentes racionais para o cálculo de pontuação para o problema de classificação de prioridades em filas de atendimento de pacientes facilitou o processo e permite possíveis extensões. Verificou-se, porém, que seria necessário o ajuste das regras para alcançar o objetivo de comparação entre a categorização feita pelo agente e pelos médicos especialistas. Pelo fato de que a estratégia tradicional dificultaria a resolução de tal problema foi utilizada a Tecnologia Adaptativa, que se mostrou uma ferramenta eficaz para criar este ambiente flexível e racional. Em uma abordagem tradicional seria necessário ajustar a pontuação das regras alterando o contexto inicialmente proposto, apesar de ser uma abordagem simples seria necessário que fossem feitas modificações até que o modelo matemático mais próximo da realidade pudesse ser encontrado, mesmo assim, podendo não acontecer. Na abordagem apresentada à utilização da Tecnologia Adaptativa criou um ambiente flexível por meio de sua característica de criar um cenário diferente após a execução de um cenário inicial sem a necessidade de que este seja modificado permanentemente, isto permitiu que novas regras de ajustes fossem incluídas ao conjunto de regras propostas, com o objetivo de aproximar a pontuação gerada pelo sistema de agentes da proposta pelos especialistas da área da saúde. No ajuste feito pelo autor o resultado foi satisfatório, porém

acredita-se que este mecanismo deve servir de ferramenta para os especialistas criarem novos cenários com o objetivo de ajustar o modelo de pontuação.

A utilização da Tecnologia Adaptativa como ferramenta de racionalização de agentes mostrou-se eficaz e viável, visto que sua utilização tornou-se um diferencial nos resultados obtidos, comprovando assim, as propostas feitas durante o trabalho. Este diferencial foi percebido pela facilidade de incluir e alterar regras dentro do contexto de execução dos agentes, o qual permitiu fazer simulações e ajustes para alcançar um resultado satisfatório.

Em relação à utilização das Tabelas de Decisão Adaptativas, pode-se perceber por meio do referencial teórico apresentado, que há diversos trabalhos comprovando sua eficácia, até mesmo em comparação às tabelas de decisão convencionais.

A implementação utilizando a estratégia de uso de tecnologia adaptativa neste trabalho foi comprovada ao final deste estudo de categorização (priorização) de pacientes com problemas nefrológicos.

Dentre os objetivos traçados, a realização do presente trabalho seguiu por meio de três grandes áreas de estudo: a racionalização de agentes, a Tecnologia Adaptativa e os conceitos de tomada de decisão. O estudo foi finalizado apresentando um resultado interessante, demonstrando a viabilidade da utilização da Tecnologia Adaptativa como fator de melhoria na eficácia da solução de problemas de tomada de decisão. Além disso, a utilização desses conceitos em conjunto com os sistemas multiagentes torna a solução viável e robusta.

Apesar do estudo de caso em questão ilustrar apenas os aspectos relacionados ao estudo da organização do atendimento de pacientes com problemas nefrológicos, é possível que a idéia da solução apresentada neste trabalho possa ser utilizada também, com as respectivas adaptações necessárias, em outros campos de atuação, principalmente em problemas relacionados à tomada de decisão.

7 TRABALHOS FUTUROS

No decorrer do trabalho e durante o processo de qualificação, foram vislumbrados alguns temas que poderiam servir de trabalhos futuros.

A Tabela de Decisão Adaptativa construída no trabalho adquire conhecimento adaptativo predeterminado antes da execução de suas regras. Uma proposta de trabalho futuro é a implementação de uma base de conhecimento dinâmica para armazenamento e a utilização do conhecimento adquirido durante a execução dos processos. Desta forma, a Tabela de Decisão Adaptativa pode adquirir conhecimento com o tempo e não se restringir apenas a um conhecimento predeterminado e utilizar este conhecimento adquirido para criar ou excluir regras adaptativas sem a necessidade de intervenção humana.

O processo de comunicação entre os agentes, apesar de padronizado, tem a troca de mensagens feita por meio de programação, sendo desta forma estática. Outra proposta para um trabalho futuro refere-se à criação de um *framework* utilizando a Tecnologia Adaptativa para tornar essa comunicação entre os agentes mais racional e dinâmica. Ainda nos aspectos dos agentes, outra proposta é a criação de um agente adaptativo que consegue absorver conhecimento das mais diferentes áreas, de maneira que outros agentes possam lhe passar informações e receber "conselhos", que este agente seja o núcleo de conhecimento do processo e possa ser utilizado nas mais diversas situações.

Em relação ao estudo de caso, a continuação do trabalho com os médicos especialistas se mostra interessante, com o objetivo de criar regras de pontuação mais assertivas, fazendo uso do mecanismo de regras adaptativas como base de experimento para o aprimoramento das regras tradicionais. Uma vez que a utilização deste processo chegasse ao seu limite, seria

interessante a implementação de um mecanismo de aprendizado (redes neurais, por exemplo) para que as regras adaptativas fossem criadas e excluídas de forma automática e racional.

Em relação à proposta do trabalho, uma sugestão de trabalho futuro é expandir os estudos para a criação de um ambiente de diagnóstico auxiliado por computador, de tal forma que possa ser configurável a cada particular necessidade. Isto pode ser feito por meio da utilização do Agente Racional e da criação do contexto de memória no dispositivo adaptativo.

8 REFERÊNCIAS

ANTOINE, J. Y.; BAUJARD, O.; BOISSER, O.; CAILLOT, B.; CHAILLOT, M.; DEMAZEAU, Y.; PESTY, S.; SICHMAN, J. S.; STEFANINI, M. H.; ZIEBELIN, D**..** Vers une taxinomie du vocabulaire pour les systemes multi-agents. In: JOURNÉE SYSTÈMES MULTI-AGENTS. Nancy, 1992. **PRC GDR 92**, Paris: PRC GDR, 1992.

BERNARDES, V. B**. Implementação de um Sistema Baseado em Agentes para o Suporte à Análise de Participações em Ambientes de EaD**. Relatório Final de Iniciação Científica. (FAPESP),XII Congresso Interno de Iniciação Cientifica - UNICAMP, Campinas, julho de 2004.

BITTENCOURT, G. Inteligência Artificial Distribuída. In: Workshop de Computação do ITA, São José dos Campos, 1998. **Título do documento (anais, proceedings etc.)**. São José dos Campos, janeiro de 1998.

BITTENCOURT, G. **Inteligência Artificial: ferramentas e teorias.** Florianópolis: Editora da UFSC, 1999. 371 p.

BONABEAU, E. Agent-based modeling: methods and techniques for simulating human systems**.** In: **Proc. National Academy of Sciences**, vol. 99, nº 3, pp. 7280-7287, 2001.

BRADSHAW, J. M. KAoS**:** An Open Agent Architecture Supporting Reuse, Interoperability, and Extensibility. In: Knowledge Acquisition Workshop (Know' 96), Banff, Alberta, Canada, 1996, **Proc. of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop**, Banff, Alberta, Canada, 1996.

BROWN, R. **Rational Choice and Judgment – Decision Analysis for the Decider.** Hoboken, NJ: John Wiley& Sons, 2005.

CASTELLS, M. **A sociedade em rede.** 1ª Edição. São Paulo: Editora Paz e Terra. 1999. v. 1.

CASTI, J. **Would-be worlds: how simulation is changing the world of science**, New York: Wiley, 1997.

CERTO, S. C., **Modern management: diversity, quality, ethics and the global environment.** New Jersey: Prentice Hall, 2000.

CHAU, P. Y. K.; HU, P. J. H. Technology implementation for telemedicine programs**.** **Communication of the ACM**, New York, nº 2, February 2004.

CHIAVENATO, I. **Introdução à Teoria da Administração.** 5 ed. São Paulo: Makron Books, 1997.

CLARKE, A. C. **Report on planet three and other speculations**. New York, NY (USA): Harper and Row, 1972, 250 p.

COSTA, A. L**. Expert-Coop: Um ambiente para desenvolvimento de sistema multiagentes cognitivos**. 1997. 117 f. Dissertação (Mestrado) – Curso de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, 1997.

COSTA, A. L. **Conhecimento social dinâmico: Uma estratégia de cooperação para sistemas multiagentes cognitivos.** 2001. 121 f. Tese (Doutorado) – Curso de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catariana, Florianópolis, 2001.

COELHO, H; PAIVA, A. **A mente e o mundo lá fora: uma introdução a inteligência artificial distribuída.** Faculdade de Ciências da Universidade de Lisboa / Instituto Superior Técnico, 1998.

CRITES, H.; BARTO, A. G. Elevator group control using multiple Reinforcement learning agents. **Mach. Learn**., vol. 33, no. 2–3, pp. 235–262, 1998.

DEMAZEAU, Y.; MULLER, J. P. **Decentralized Artificial Intelligence.** Amsterdam: Elsevier, 1990.

DROOLS. **The Business Logic Integration Platform**. JBoss Community. Disponível em: http://jboss.org/drools/. Acesso em: 12/07/2010.

DURFEE, E. H**.** A. **Unified Approach to Dynamic Coordination: Planning Actions and Interactions in a Distributed Problem Solving Network**, COINS Technical Report 87-84, University of Massachusetts, September 1987.

FARACO, R. A**. Uma arquitetura de agentes para negociação dentro do domínio do comércio eletrônico.** 1998. 120 f. Dissertação (Mestrado) – Curso de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 1998.

FERBER, J. **Multi-Agent System. An Introduction to Distributed Artificial Intelligence**. Addison-Wesley Publishers, 1999.

FIGUEIRA, J.; GRECO, S.; EHRGOTT, M. **Multiple Criteria Decision Analysis: State of the Art Surveys**, Springer, New York, 2004.

FININ, T.; FRITZON, R.; MCKAY, D.; MCENTIRE, R. **KQML - A Language and Protocol for Knowledge and Information Exchange.** Computer Science Department. Universit of Maryland Baltimore, 1994.

FIPA. **Foundation for Intelligent Physical Agents**. Disponível em: http://www.fipa.org/ . Acesso em: 12/07/2010.

FIPA [a]. **FIPA Agent Management Specification**, Disponível em: http://www.fipa.org/specs/fipa00023/. Acesso em: 12/07/2010.

FIPA-OS. **Framework de desenvolvimento de sistemas multiagentes- FIPA OS**. Disponível em: http://fipa-os.sourceforge.net/. Acesso em: 12/07/2010.

FRIGO, L. B.; POZZEBON, E.; BITTENCOURT, G**.** O Papel do Agentes Inteligente em Sistemas Tutores Inteligentes. In: World Congress on Engineering and Technology

Education, Guarujá, 2004, **Anais do Engineering Education in the Changing Society**, Guarujá Council of Researches in Education and Sciences, Guarujá, 2004.

FÜLÖP, J. **Introduction to Decision Making Methods**. Working Paper of the Laboratory of Operations Research and Decision Systems, Computer and Automation Institute, Hungarian Academy of Sciences, 2005.

GAWALO, R. D.; MESHRAM B. B. **Agent-Based Autonomous Examination Systems.** JAMA. 2009.

GILDERSLEEVE, T. R. **Decision Tables and Their Practical Application in Data Processing.** Englewood Cliffs, N. J., Prentice Hall, 1970.

GLUZ, J. C.; VICCARI, R. M. Linguagens de Comunicação entre Agentes: Fundamentos Padrões e Perspectivas. In Vieira, **Proceedings of III Jornada de Mini-Cursos de Inteligência Artificial,** vol. 8. Campinas: SBC. 53-102, July, 2003.

GOMES, L. F.; GOMES, C. F. S.; ALMEIDA, A. T.. **Tomada de Decisão Gerencial: um enfoque multicritério.** 2 ed. São Paulo: Atlas, 2006.

GUDWIN, R. R. **Introdução à teoria de agentes**. DCA-FEEC-UNICAMP, 2003. Disponível em http://www.dca.fee.unicamp.br/~gudwin/courses/IA009/. Acesso em 12/07/2010.

HARRIS, R. **Introduction to Decision Making**. VirtualSalt, 2009. Disponível em: http://www.virtualsalt.com/crebook5.htm. Acesso em 17/07/2010.

HEWITT, C. **A universal modular ACTOR formalism for AI.** Proceedings of the 3rd International Joint Conference on Artificial Intelligence, 1973. P. 235-245.

IBM ; Executive Summary. Healthcare 2015**: Win-win or Lose-lose ?,** 2006. Disponível em: http://www.ibm.com/healthcare/hc2015. Acesso em 12/07/2010.

ITO, M. ; SILVA, L. R. ; MARTINI, J. S. C.; IOCHIDA, L. C. **Um Sistema Multi-agente para a Monitoração de Pacientes Diabéticos com Base no Modelo GRPC.** In: IX Workshop de Informática Médica, 2009, Bento Gonçalves. Anais do XXIX Congresso da Sociedade Brasileira de Computação, 2009. p. 1935-1944.

JADE. **Framework de desenvolvimento de sistemas multiagentes - JADE,** 2010, Disponível em http://jade.tilab.com/. Acesso em 12/07/2010.

JENNINGS, N. R. **"Cooperation in Industrial Multi-agent Systems",** World Scientific, 1994.

JENNINGS, N. R. **On agent-based software engineering, Artificial Intelligence,** 2000.

KEENEY, R.L.; RAIFFA, H. **Decisions with Multiple Objectives: Performances and Value Trade-Offs**, Wiley, New York, 1976.

KLEIN, G. **An Overview of Naturalistic Decision Making Applications, Naturalistic Decision Making.** Gary Klein & Caroline Zsambockeds. Mahway, NJ: Lawrence Erlbaum Associates, 1997.

LTA. **Laboratório de Linguagens e Tecnicas Adaptativas**. Disponível em: http://www.pcs.usp.br/~lta/ . Acesso em: 12/07/2010.

MARTIAL, F. V. **Coordinating Plans of Autonomous Agents.** Berlin, Spring-Verlag Berlin Heidelberg, 1992. (Lecture Notes in Artificial Intelligence – Subseries of Lecture Notes in Computer Science)

MELLOULI, S.; MINEAU, G. ; et al. Laying the foundations
for an agent modeling methodology for fault tolerant multi-agent systems, In **Fourth International Workshop Engineering Societies in the Agents World**, Imperial College London, UK, 2003.

MENESES, E. X. 2001. **Jornada de atualização em inteligência artificial – integração de agentes de informação,** 2001. Disponível em http://www.ime.usp.br/~eudenia/jaia/. Acesso em 12/07/2010.

MINSKY, M. **The Society of Mind.** New York: Simon & Schuster, 1985.

MITCHELL, T. M. **Machine Learning.** New York: McGraw-Hill, 1997.

MONTALBANO, M. **Decision Tables.** Chicago: Science Research Associates, 1974.

NAGAYAMA, S. **Tabelas de decisão e implementação do gerador i-m-e.** 1990. 237p. Dissertação (Mestrado) – Instituto de Matemática e Estatística, Universidade de São Paulo. São Paulo, 1991.

NEGROPONTE, N. **Being Digital**, New York: Vintage Books, 1995. 255p.

NETO, J. J.; MAGALHÃES, M. E. Reconhecedores sintáticos uma alternativa didática para uso em cursos de engenharia. In: **Anais da XIV Congresso Nacional de Informática - SUCESU.** São Paulo, SP: [s.n.], 1981. p. 171-181.

NETO, J. J. **Introdução à Compilação.** Rio de Janeiro: LTC, 1987. pp. 222. ISBN 85-216-0483-1

NETO, J. J. **Adaptive automata for context-dependent languages.** ACM SIGPLAN Notices, v. 29, n. 9, p. 115-124, Sep. 1994.

NETO, J. J., **Adaptive Rule-Driven Devices – General Formulation and Case Study.** Lecture Notes in Computer Science. Watson, B.W. and Wood, D. (Eds.): Implementation and Application of Automata 6thInternationalConference, CIAA 2001, Vol. 2494, Pretoria, South Africa, July 23-25, Springer-Verlag, 2001, pp. 234-250.

NETO, J. J. **Adaptatividade e Tecnologia Adaptativa - Tutorial baseado no WTA 2007**. Revista IEEE América Latina. Vol. 5, Num. 7, ISSN: 1548-0992, Novembro 2007. (p. 495)

NILSSON, N. **Artificial Intelligence: A New Synthesis.** San Mateo: Morgan Kaufmann Publishers, 1998.

PARUNAK, H. V. D. **"Applications of Distributed Artificial Intelligence in Industry" Foundations of Distributed Artificial Intelligence.**Wiley Inter-Science, cap 4, 1994.

PARUNAK, H. V. D.; WEISS, G. **"Industrial and practical applications of DAI,"** in Multi- Agent Systems: A Modern Approach to Distributed Artificial Intelligence**,** Ed. Cambridge, MA: MIT Press, 1999, ch. 9, pp. 377–412.

PEDRAZZI, T. C. **Um Ambiente de Desenvolvimento Baseado em Tabelas Adaptativas.** 2007. 120 p. Dissertação (Mestrado) - Universidade de São Paulo, São Paulo, Brasil, 2007.

PIDD, M. **Modelagem empresarial:** ferramentas para tomada de decisão. Imprenta Porto Alegre: Bookman, 2001.

PISTORI, H. **Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações.** 2003. 180 p. Tese (Doutorado) - Universidade de São Paulo, São Paulo, Brasil, 2003.

PORTER, M. E.; TEISBERG, E. O. **Repensando a saúde: estratégias para melhorar a qualidade e reduzir os custos.** Traducao de Cristina Bazan. Porto Alegre, 2007.

RABELO, R. J. **"Sistemas Multi-agentes"**. 2009. Disponível em: http://www.das.ufsc.br/~rabelo. Acesso em: 12/07/2010.

RAO, A. S.; GEORGEFF, M. P. BDI agents: from theory to practice. In: **INTERNATIONAL CONFERENCE ON MULTIAGENT SYSTEMS (ICMAS'95)**, San Francisco, USA. Proceedings. [S.1.]: AAAI Pess, 1995. P. 312-319, 1995.

RUSSEL, S. J.; NORVIG, P. **Artificial intelligence**: a modern approach. Upper Saddle River: Prentice Hall, 2003.

SCHWEITZER, F. **Brownian Agents and Active Particles: Collective Dynamics in the Natural and Social Sciences**. Springer, 2003.

SEN, S. ; WEISS, G. **"Learning in multiagent systems,"** in Multiagent
Systems: A Modern Approach to Distributed Artificial Intelligence, Ed. Cambridge, MA: MIT Press, 1999.

SHIMIZU, T. **Decisão nas organizações: introdução aos problemas de decisão encontrados nas organizações e nos sistemas de apoio à decisão.** São Paulo: Atlas, 2001.

SICHMAN, J. S.; DEMAZEAU, Y. **When can knowledge-based systems be called agents ?.** 9th Brazilian Symposium on Artificial Intelligence. Rio de Janeiro, 1992. Proceedings. Rio de Janeiro, SBIA, 1992.

SIMON, H. A. **Comportamento Administrativo.** 2ª ed. Rio de Janeiro: FGV, 1970.

SIPSER, Michael. **Introdução à teoria da computação.** São Paulo: Thomson Learning, 2007.

STONE, P. ; VELOSO, M. **"Multiagent systems: A survey from the machine learning perspective,"**Auton. Robots, vol. 8, no. 3, 2000.

TCHEMRA, A. H. **Aplicação da Tecnologia Adaptativa em Sistemas de Tomada de Decisão. WTA 2007 – III.3 .** IEE América Latina, v. 5, n. 7, 2007.

TURING, A. **The imitation game, Computing Machinery and Intelligence Mind**, Vol. 59, No. 236, 1950.

VANTHIENEN, J. **Decision Tables and Business Rules**, Katholieke Universite it Leuven, Belgium, 2008.

WEISS, G. **Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.** Cambridge, MA: MIT Press, 1999.

WOOLDRIDGE, M., JENNINGS, N. **Intelligent Agents: theory and practice**. Knowledge Engineering Review. vol. 10., no. 2., pp. 115-152. 1995. Disponível em: http://citeseer.ist.psu.edu/97055.html . Acesso em: 12/07/2010.

WOOLDRIDGE, M. **Reasoning about rational agents.** EUA: MIT Press, 2000.

WOOLDRIDGE, M. **An Introduction to MultiAgent Systems.** Chischester, England: John Wiley& Sons, 2002. Paperback. ISBN 047149691X.

WORTMANN, H. ; SZIRBIK, N. **"ICT Issues among Collaborative Enterprises: from Rigid to Adaptive Agent-Based Technologies"** International Journal of Production Planning and Control, Taylor & Francis Publishers, 2001.

YATES, F. J. Decision Management - **How to Assure Better Decisions in Your Company.** San Francisco CA: John Wiley& Sons, 2003.

ZEUS. **Framework de desenvolvimento de sistemas multiagentes - ZEUS,** 2010. Disponível em http://sourceforge.net/projects/zeusagent/. Acesso em 12/07/2010.

ZSAMBOCK, C. **Naturalistic Decision Making Where Are We Now?, Naturalistic Decision Making.** Gary Klein & Caroline Zsambockeds. Mahway, NJ: Lawrence ErlbaumAssociates, 1997.

Apêndice A – Informações dos Pacientes

As informações coletadas dos pacientes com problemas nefrológicos para obtenção dos critérios de categorização de pacientes são:

Nome - Nome do paciente.

Idade – Idade do paciente.

Peso – Peso do paciente.

Altura – Altura do paciente.

Sexo – Sexo do paciente.

IMC – * Este dado não será solicitado ao paciente; o sistema irá calcular por meio da fórmula $PESO/ALT(m)^2$

Encaminhamento – Este dado informa qual o tipo de enfermidade que serviu de base para o encaminhamento do paciente ao tratamento. Os tipos listados são: nefrite, hipertensão arterial, diabetes, cálculos renais de repetição, infecção do trato urinário e outros.

HAS – Se o paciente possui Hipertensão Arterial Sistêmica. E qual a gravidade da doença: leve, moderado e grave.

DM - Se o paciente possui Diabetes Mellitus (Sim/Não). E qual a gravidade sendo: leve (tratamento apenas com dieta), moderada, grave (tratamento com insulina).

Ultra-som renal - Se o paciente fez o exame e se o resultado mostrou problemas ou não.

Creatinina Sérica – Índice encontrado no resultado do exame feito pelo paciente. Este dado é obrigatório caso os dados HAS, DM, Cálculo ou ITU sejam informados como positivos.

Urina 1 – Se o paciente fez o exame e se o resultado do exame feito pelo paciente teve alterações.

Clearence (ml/min/1,73m²) - Para o cálculo do *clearence* será utilizada a seguinte fórmula:

Ccr (mL/min) = (140 - idade (anos) X peso (kg)/ (Crs (mg/dL) x 72) para homens; Para mulheres, basta multiplicar o resultado por 0,85.

Apêndice B – Cálculo de Pontuação

A estratégia para a criação de uma ordem de atendimento dos pacientes teve como base uma tabela de decisão, de tal forma que, por meio das informações coletadas dos pacientes sejam contabilizados pontos. De posse da pontuação, os pacientes serão categorizados de acordo com uma faixa preestabelecida com o objetivo de criar uma ordem de atendimento onde os pacientes categorizados como mais críticos possam ter prioridade de atendimento.

As regras de pontuação de pacientes foram criadas por uma professora médica da UNIFESP e fazer parte de seu estudo da tese de doutorado. Essas regras são as seguintes:

Em relação à idade:

Quadro 4 – Faixa etária para cálculo de pontuação.

Critério	Pontos
Até 40 anos	0
41-49	1
50-59	2
60-69	3
Maior que 70	4

Em relação ao IMC:

Quadro 5 - Faixa de IMC para cálculo de pontuação.

Critério	Pontos
Abaixo de 18,5	1
entre 18,5 e 25	0
entre 25 e 30	2
acima de 30	3

Em relação ao encaminhamento:

Quadro 6– Motivo do encaminhamento para tratamento.

Critério	Pontos
Nefrite	4
Hipertensão arterial	5
Diabetes	6
Cálculo renais de repetição	1
Infecção do trato urinário	2
Outros	3

Em relação ao sexo:

Quadro 7 – Sexo do paciente para cálculo de pontuação.

Critério	Pontos
Masculino	1
Feminino	0

Em relação à hipertensão arterial:

Quadro 8 – Nível de alteração de hipertensão arterial para cálculo de pontuação.

Critério	Pontos
Sem alteração	0
Leve	2
Moderada	4
Grave	6

Em relação à diabetes:

Quadro 9 – Nível de diabetes para cálculo de pontuação.

Critério	Pontos
Não apresenta diabetes	0
Leve (tratamento com dieta)	2
Moderada	4
Grave (tratamento com insulina)	6

Em relação ao exame de ultra-som renal:

Quadro 10 – Resultado do exame de ultra-som renal para cálculo de pontuação.

Critério	Pontos
Não fez	0
Resultado sem alteração	2
Resultado com alteração	4

Em relação ao exame de urina:

Quadro 11 – Resultado do exame de urina para cálculo de pontuação.

Critério	Pontos
Não Fez	0
Resultado sem alteração	2
Resultado com alteração	4
Proteinúria > 0,05	6

Em relação à creatinina:

Quadro 12– Faixa de resultados de creatinina para cálculo de pontuação.

Critério	Pontos
0,5 – 1,0	0
1,1 – 1,5	1
1,6 – 2,2	2
2,3 – 3,0	3
> 3,0	4

Em relação ao *clearance*:

Quadro 13 – Faixa de resultados de *clearence* para cálculo de pontuação.

Critério	Pontos
> 90	0
60-89	1
30-59	2
15-29	3
< 15	4

A implementação das regras relacionadas à pontuação de pacientes com problemas nefrológicos foi feita respeitando os padrões criados pelo *framework Drools* e refletem exatamente a proposta feita pela professora da UNIFESP para a pontuação de pacientes.

A figura 27 mostra o conteúdo do arquivo *pontos.drl* com todas as regras.

```
package org.ceeteps.rules

import org.ceeteps.common.Paciente;
import org.ceeteps.common.Doenca;
import org.ceeteps.common.Exame;

global java.lang.String  arquivoAdaptativo;

/*
Até 40 anos 0
41-49 1
50-59 2
60-69 3
Maior que 70      4
*/
rule "idade1"
activation-group "idade"
      when
            $p: Paciente(idade > 40,idade<50)
      then
            $p.somaPontos(1);
end
rule "idade2"
activation-group "idade"
      when
            $p: Paciente(idade >= 50,idade<60)
      then
            $p.somaPontos(2);
end
rule "idade3"
activation-group "idade"
      when
            $p: Paciente(idade >= 60,idade<70)
      then
            $p.somaPontos(3);
end
rule "idade4"
activation-group "idade"
      when
            $p: Paciente(idade >= 70)
      then
            $p.somaPontos(4);
end


/*
Abaixo de 18,5    1
```

```
entre 18,5 e 25   0
entre 25 e 30     2
acima de 30       3
*/
rule "imc1"
activation-group "imc"
      when
            $p: Paciente(imc > 0 && imc < (18.5))
      then
            $p.somaPontos(1);
end
rule "imc2"
activation-group "imc"
      when
            $p: Paciente(imc >= (18.5) && imc < 25)
      then
            $p.somaPontos(0);
end
rule "imc3"
activation-group "imc"
      when
            $p: Paciente(imc >= 25 && < 30)
      then
            $p.somaPontos(2);
end
rule "imc4"
activation-group "imc"
      when
            $p: Paciente(imc >= 30)
      then
            $p.somaPontos(3);
end



/*
Masculino   1
Feminino    0
*/
rule "sexo1"
activation-group "sexo"
      when
            $p: Paciente(sexo == "M")
      then
            $p.somaPontos(1);
end
rule "sexo2"
activation-group "sexo"
      when
            $p: Paciente(sexo == "F")
      then
            $p.somaPontos(0);
end


/*
Encaminhamento
      1- nefrite
      2- HAS de difícil controle c/ ou s/ cr alterada
      3- DM de dífícil controle c/ ou s/ cr alterada
      4- cálculos renais de repetição
```

```
      5- ITU de repetição
      6- Outros
*/
rule "encaminhamento1"
activation-group "encaminhamento"
      when
            $p: Paciente( doenca.encaminhamento  == 1)

      then
            $p.somaPontos(4);
end
rule "encaminhamento2"
activation-group "encaminhamento"
      when
            $p: Paciente( doenca.encaminhamento  == 2)

      then
            $p.somaPontos(5);
end
rule "encaminhamento3"
activation-group "encaminhamento"
      when
            $p: Paciente( doenca.encaminhamento  == 3)

      then
            $p.somaPontos(6);
end
rule "encaminhamento4"
activation-group "encaminhamento"
      when
            $p: Paciente( doenca.encaminhamento  == 4)

      then
            $p.somaPontos(1);
end
rule "encaminhamento5"
activation-group "encaminhamento"
      when
            $p: Paciente( doenca.encaminhamento  == 5)

      then
            $p.somaPontos(2);
end
rule "encaminhamento6"
activation-group "encaminhamento"
      when
            $p: Paciente( doenca.encaminhamento  == 6)

      then
            $p.somaPontos(3);
end

/*
HAS
      0 - NÃO
      1- LEVE  1 A 2 DROGAS
      2- MODERADA > 2 E <= 3DROGAS
      3- GRAVE + 4 DROGAS

*/
rule "has1"
```

```
activation-group "has"
	when
		$p: Paciente( doenca.has == 1)
	then
		$p.somaPontos(2);
end
rule "has2"
activation-group "has"
	when
		$p: Paciente( doenca.has == 2)
	then
		$p.somaPontos(4);
end
rule "has3"
activation-group "has"
	when
		$p: Paciente( doenca.has == 3)
	then
		$p.somaPontos(6);
end

/*
DM
	0 - Não
	1- Leve dieta
	2- moderada hipoglicemiante
	3- grave insulinoterapia
*/
rule "dm1"
activation-group "dm"
	when
		$p: Paciente( doenca.dm == 1)
	then
		$p.somaPontos(2);
end
rule "dm2"
activation-group "dm"
	when
		$p: Paciente( doenca.dm == 2)
	then
		$p.somaPontos(4);
end
rule "dm3"
activation-group "dm"
	when
		$p: Paciente( doenca.dm == 3)
	then
		$p.somaPontos(6);
end


/*
Renal:
	0 - não fez
	1- sim normal
	2- sim alterado
*/
rule "renal1"
activation-group "renal"
	when
		$p: Paciente( exame.renal == 1)
```

```
	then
		$p.somaPontos(2);
end
rule "renal2"
activation-group "renal"
	when
		$p: Paciente( exame.renal == 2)
	then
		$p.somaPontos(4);
end


/*
Urina:
	0 - não fez
	1- sim normal
	2- sim alterado
	3- proteinúria > 0,05
*/
rule "urina1"
activation-group "urina"
	when
		$p: Paciente( exame.urina == 1)
	then
		$p.somaPontos(2);
end
rule "urina2"
activation-group "urina"
	when
		$p: Paciente( exame.urina == 2)
	then
		$p.somaPontos(4);
end
rule "urina3"
activation-group "urina"
	when
		$p: Paciente( exame.urina == 3)
	then
		$p.somaPontos(6);
end


/*
Creatinina:
0,5 – 1,0   0
1,1 – 1,5   1
1,6 – 2,2   2
2,3 – 3,0   3
> 3,0 4
*/
rule "creatinina1"
activation-group "creatinina"
	when
		$p: Paciente( exame.creatinina  >=(0.5) && exame.creatinina
<=1)
	then
		$p.somaPontos(0);
end
rule "creatinina2"
activation-group "creatinina"
	when
```

```
            $p: Paciente( exame.creatinina >=(1.1) && exame.creatinina
<=(1.5))
      then
            $p.somaPontos(1);
end
rule "creatinina3"
activation-group "creatinina"
      when
            $p: Paciente( exame.creatinina >=(1.6) && exame.creatinina
<=(2.2))
      then
            $p.somaPontos(2);
end
rule "creatinina4"
activation-group "creatinina"
      when
            $p: Paciente( exame.creatinina >=(2.3) && exame.creatinina<=3)
      then
            $p.somaPontos(3);
end
rule "creatinina5"
activation-group "creatinina"
      when
            $p: Paciente( exame.creatinina >3)
      then
            $p.somaPontos(4);
end



/*
Clearance:
> 90  0
60-89       1
30-59 2
15-29 3
< 15  4
*/

rule "clearance1"
activation-group "clearance"
      when
            $p: Paciente( exame.clearance >= 90)
      then
            $p.somaPontos(0);
end
rule "clearance2"
activation-group "clearance"
      when
            $p: Paciente( exame.clearance >= 60 && exame.clearance < 90)

      then
            $p.somaPontos(1);
end
rule "clearance3"
activation-group "clearance"
      when
            $p: Paciente( exame.clearance >= 30 && exame.clearance < 60)

      then
            $p.somaPontos(2);
```

```
end
rule "clearance4"
activation-group "clearance"
      when
            $p: Paciente( exame.clearance >= 15 && exame.clearance < 30)

      then
            $p.somaPontos(3);
end
rule "clearance5"
activation-group "clearance"
      when
            $p: Paciente( exame.clearance < 15)
      then
            $p.somaPontos(4);
end
```

Figura 27 – Arquivo *pontos.drl*.

Apêndice C – A classe *PacienteOntology*

A classe *PacienteOntology* representa a ontologia criada para modelar os pacientes com doenças nefrológicas tratadas no trabalho.

A figura 28 demonstra a classe Java que implementa a ontologia do paciente.

```
package org.ceeteps.agent;

import jade.content.onto.*;
import jade.content.schema.*;

import java.util.*;

import org.ceeteps.common.Doenca;
import org.ceeteps.common.Exame;
import org.ceeteps.common.Paciente;

import examples.ontology.employment.Engage;

/**
 * Ontologia para a classe de paciente
 * @author Eduardo Endo
 *
 */
public class PacienteOntology extends Ontology {

  /**
    Uma constante simbolica com o nome da ontologia.
   */
  public static final String NAME = "paciente-ontology";

  /**
   * Dados do paciente
   */
  public static final String PACIENTE = "PACIENTE";
  public static final String PACIENTE_NOME = "nome";
  public static final String PACIENTE_IDADE = "idade";
  public static final String PACIENTE_PESO = "peso";
  public static final String PACIENTE_IMC = "imc";
  public static final String PACIENTE_LOCALENCAMINHAMENTO =
"localencaminhamento";
  public static final String PACIENTE_SEXO = "sexo";
  public static final String PACIENTE_DOENCA = "doenca";
  public static final String PACIENTE_EXAME = "exame";
  public static final String PACIENTE_PONTOS = "pontos";
  public static final String PACIENTE_CATEGORIA = "categoria";

  /**
   * Dados da doenca
   */
  public static final String DOENCA = "DOENCA";
```

```
  public static final String DOENCA_ENCAMINHAMENTO = "encaminhamento";
  public static final String DOENCA_HAS = "has";
  public static final String DOENCA_DM = "dm";

  /**
   * Dados do exame
   */
  public static final String EXAME = "EXAME";
  public static final String EXAME_CREATININA = "creatinina";
  public static final String EXAME_RENAL = "renal";
  public static final String EXAME_URINA = "urina";
  public static final String EXAME_CLEARANCE = "clearance";

  public static final String PACIENTECONCEPT = "pacienteconcept";
  public static final String PACIENTECONCEPT_PACIENTES = "pacientes";

  private static Ontology theInstance = new PacienteOntology();

   public static Ontology getInstance() {
            return theInstance;
   }

  /**
   * Constructor
   */
  private PacienteOntology() {
    //__CLDC_UNSUPPORTED__BEGIN
      super(NAME, BasicOntology.getInstance());


    try {
            add(new ConceptSchema(PACIENTE), Paciente.class);

            ConceptSchema cs = (ConceptSchema)getSchema(PACIENTE);

            cs.add(PACIENTE_NOME,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
            cs.add(PACIENTE_IDADE,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));
            cs.add(PACIENTE_PESO,
(PrimitiveSchema)getSchema(BasicOntology.FLOAT));
            cs.add(PACIENTE_IMC,
(PrimitiveSchema)getSchema(BasicOntology.FLOAT));
            cs.add(PACIENTE_LOCALENCAMINHAMENTO,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));
            cs.add(PACIENTE_SEXO,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
            cs.add(PACIENTE_PONTOS,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));
            cs.add(PACIENTE_CATEGORIA,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));

            add(new ConceptSchema(DOENCA), Doenca.class);
            ConceptSchema csa = (ConceptSchema)getSchema(DOENCA);
            csa.add(DOENCA_ENCAMINHAMENTO,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));
            csa.add(DOENCA_HAS,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));
            csa.add(DOENCA_DM,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));
```

```
            add(new ConceptSchema(EXAME), Exame.class);
            csa = (ConceptSchema)getSchema(EXAME);
            csa.add(EXAME_CREATININA,
(PrimitiveSchema)getSchema(BasicOntology.FLOAT));
            csa.add(EXAME_CLEARANCE,
(PrimitiveSchema)getSchema(BasicOntology.FLOAT));
            csa.add(EXAME_RENAL,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));
            csa.add(EXAME_URINA,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));

      cs.add(PACIENTE_DOENCA, (ConceptSchema)getSchema(DOENCA),
ObjectSchema.OPTIONAL);
      cs.add(PACIENTE_EXAME, (ConceptSchema)getSchema(EXAME),
ObjectSchema.OPTIONAL);

      add(new AgentActionSchema(PACIENTECONCEPT), PacienteConcept.class);
            AgentActionSchema as =
(AgentActionSchema)getSchema(PACIENTECONCEPT);

      as.add(PACIENTECONCEPT_PACIENTES,(ConceptSchema)getSchema(PACIENTE),1
,ObjectSchema.UNLIMITED);

    }
    catch(OntologyException oe) {
      oe.printStackTrace();
    }
  }
}
```

Figura 28 – Classe *PacienteOntology*.

Apêndice D – Arquivos de Regras Adaptativas

Para a implementação do contexto adaptativo do estudo de caso foram criados os arquivos *adaptive1.xml* e *adaptive2.xml* que representam as regras adaptativas. Os quadros 14 e 15 mostram o conteúdo destes arquivos.

No quadro 14 pode-se observar que existem duas regras adaptativas:

- A regra que será executada após a execução do grupo de regras tradicionais "encaminhamento" e como conseqüência será executado o arquivo de regras *encaminhamento.drl*.

- A regra que será executada após a execução do grupo de regras tradicionais "clearance" e como conseqüência será executado o arquivo de regras *clearance.drl*.

```
<?xml version="1.0" encoding="UTF-8"?>

<p:adaptive-rules xmlns:p="http://endo.com/xml/ns/adaptive"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://endo.com/xml/ns/adaptive adaptive.xsd ">
  <p:adaptive-rule>
      <p:target-group-rule>encaminhamento</p:target-group-rule>
    <p:before></p:before>
    <p:after>encaminhamento.drl</p:after>
  </p:adaptive-rule>

  <p:adaptive-rule>
      <p:target-group-rule>clearance</p:target-group-rule>
    <p:before></p:before>
    <p:after>clearance.drl</p:after>
  </p:adaptive-rule>

</p:adaptive-rules>
```

Quadro 14– Arquivo *adaptive1.xml*

No quadro 15 pode-se observar que existem três regras adaptativas:

- A regra que será executada após a execução do grupo de regras tradicionais “urina” e como conseqüência será executado o arquivo de regras *urina.drl*.

- A regra que será executada após a execução do grupo de regras tradicionais “idade” e como conseqüência será executado o arquivo de regras *idade.drl*.

- A regra que será executada após a execução do grupo de regras tradicionais “imc” e como conseqüência será executado o arquivo de regras *imc.drl*.

```
<?xml version="1.0" encoding="UTF-8"?>

<p:adaptive-rules xmlns:p="http://endo.com/xml/ns/adaptive"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://endo.com/xml/ns/adaptive adaptive.xsd ">
  <p:adaptive-rule>
      <p:target-group-rule>urina</p:target-group-rule>
    <p:before></p:before>
    <p:after>urina.drl</p:after>
  </p:adaptive-rule>

    <p:adaptive-rule>
      <p:target-group-rule>idade</p:target-group-rule>
    <p:before></p:before>
    <p:after>idade.drl</p:after>
  </p:adaptive-rule>

    <p:adaptive-rule>
      <p:target-group-rule>imc</p:target-group-rule>
    <p:before></p:before>
    <p:after>imc.drl</p:after>
  </p:adaptive-rule>
</p:adaptive-rules>
```

Quadro 15– Arquivo *adaptive2.xml*