CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
MESTRADO EM TECNOLOGIA

LEANDRO RAMOS DA SILVA

DESENVOLVIMENTO DE UM FRAMEWORK PARA O GERENCIAMENTO DA ORGANIZAÇÃO DE AGENTES EM SISTEMAS MULTI-AGENTES

SÃO PAULO
DEZEMBRO / 2008

LEANDRO RAMOS DA SILVA

DESENVOLVIMENTO DE UM FRAMEWORK PARA O GERENCIAMENTO DA ORGANIZAÇÃO DE AGENTES EM SISTEMAS MULTI-AGENTES

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 Desenvolvimento e Formação, sob orientação da Profª Drª Márcia Ito.

SÃO PAULO
DEZEMBRO / 2008

Dedicatória

Esse trabalho é dedicado à comunidade
acadêmico-científica brasileira.
Tão carente de recursos e,
por vezes, de reconhecimento
da sociedade e governantes;
e à minha mãe por ter estado sempre ao meu lado.

Agradecimentos

Começo esta lista de agradecimentos ciente de que não terei posto aqui todos aqueles que direta ou indiretamente me auxiliaram tanto no trabalho, como para que eu estivesse aqui hoje com disposição e vontade para atingir meus objetivos.

Acima de tudo, agradeço à minha mãe, Elisabete Ramos, que sem dúvida alguma é a pessoa que mais me auxiliou no dia-a-dia para que concluísse este trabalho. Outros familiares ajudaram diretamente essa conclusão e cito em especial a Mariana Versolato nas revisões, tanto da dissertação como de artigos provenientes dela, e Vanessa Ramos pela ajuda no projeto gráfico.

Não há como falar dos familiares sem demonstrar o carinho por toda a família do meu pai, Sérgio da Silva, à família da minha irmã, os Pedra e à família do Maurício onde juntos passamos por divertidíssimos finais de semana de descontração juntos.

Amigos são sempre muito bem vindos e me lembro de todos. Mas não os cito por falta de espaço, agradecendo de uma forma geral àqueles que fiz na época em que trabalhei na IBM e aos novos da Prodesp. Desta última, merecem um agradecimento especial Marcos Ide e Cristiane Almeida por viabilizarem que tivesse condições de me dedicar a este projeto.

Outras condições foram dadas a mim por laços inigualáveis que fiz durante o curso no Centro Paula Souza. Deste, quero agradecer em especial à minha grande tutora, a Profª Drª Márcia Ito, com a qual venho trabalhando desde 2003.

Por fim, agradeço à Deus pela vida, pela saúde e principalmente por colocar em meu caminho pessoas tão especiais e importantes. E claro, ao dom equilíbrio que me foi concedido para viver a vida de uma forma tão intensa e tão carinhosa todos os dias.

"O que sabemos é uma gota,
O que ignoramos é um oceano"
Isaac Newton
(1643- 1727)

Resumo

SILVA, L. R. da. **Desenvolvimento de um Framework para o Gerenciamento da Organização de Agentes em Sistemas Multi-agentes**. 2008. 96 f. Dissertação (Mestrado) – Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2008.

A Orientação a Agentes vem cada vez mais sendo difundida em pesquisas científicas e em resolução de problemas em sistemas aplicados. Assim, a aplicação de Sistemas Multi-agentes se torna uma vertente muito importante como objeto de pesquisa, assim como métodos, técnicas e ferramentas que facilitem lidar com a complexidade intrínseca nesta tecnologia. O presente trabalho apresenta um estudo acerca de conceitos e ferramentas de Orientação a Agentes com o objetivo de criar um *framework* onde seja facilitada a criação de Sistemas Multi-agentes. O objeto do estudo é o que diz respeito à criação de organizações de agentes onde ainda não se sabe de antemão quantos e quais são os agentes necessários para resolução de um determinado problema de forma distribuída. A abordagem utilizada se refere principalmente ao aspecto conversacional de agentes como forma de representação de seu conhecimento e definição de ações, que por sua vez estão baseadas em racionalização BDI (*belief, desires, intentions*) na forma de papéis. Desta forma o estudo apresenta organizações dinâmicas e autônomas de agentes baseadas em negociação Contract-Net através de uma comunicação baseada na padronização de FIPA (*Foundation for Intelligent Physical Agents).*

Palavras-chave: Sistemas Multi-agentes, SMA, Arquitetura de Software, Racionalização, Organização.

Abstract

SILVA, L. R. da. **Development of a Framework for Managing the Agents Organization in Multi-Agents Systems**. 2008. 96 f. Dissertation (Master's Degree) - Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2008.

The Agent Orientation paradigm has been published in scientific researches and used to solve problems in applied systems. Thus, the application of Multi-Agents Systems becomes a very important branch as an object of study, the same way as methods, techniques and tools are to facilitate how to deal with the inherent complexity involved in this technology. This project presents a study on the concepts and tools of Agent Orientation with the intention to create a framework in which it is easier to create Multi-Agents Systems. The object of study is creating organizations of agents when it is not known, beforehand, how many and which are the necessary agents for the resolution of a specific problem in a distributed way. The approach that has been used refers mainly to the conversational aspect of agents as a form of representing their knowledge and defining their actions, which are based in BDI rationalizing (*belief, desires, intentions*) as roles. Hence, the study presents dynamic and autonomous organizations of agents, based in Contract-Net negotiating through communication based in FIPA's patterns (*Foundation for Intelligent Physical Agents).*

***Keywords*:** Multi-agent Systems, MAS, Software Architecture, Rationalizing, Organization.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS

AI – *Artificial Intelligence* (Inteligência Artificial)
AP – *Application Plataform* (plataforma de aplicação)
AUML – *Agent UML* (Linguagem de Modelagem Unificada para Agentes)
IAD – Inteligência Artificial Distribuída
BDI – *Belief, desire, intention*
CMD – Central de Monitoração de Diabéticos
CNET – Contrac-Net
CRC – Central de Relacionamento com Pacientes Crônicos
FIPA – *Foundation for Intelligent Physical Agents*
GRPC – Gestão do Relacionamento com o Paciente Crônico
MAS - *Multi-agent Systems*
PLN – Processamento de Linguagem Natural
RDP = Resolução Distribuída de Problemas
SMA = Sistemas Multi-agentes
TIC = Tecnologia da Informação e Comunicação

SUMÁRIO

1. Introdução

Desenvolver *software* de alta qualidade para aplicações complexas não é fácil, o que implica na necessidade de modelos e técnicas que facilitam lidar com esta complexidade (JENNINGS, 1999).

Por sua vez a tecnologia de agentes, que segundo Janca (1995) é fruto da evolução da IAD (Inteligência Artificial Distribuída), possui características poderosas para o projeto de softwares complexos. Um agente é entendido como um software capaz de resolver um problema em particular de forma autônoma e ainda é capaz de interagir com um ambiente para possibilitar a resolução de problemas que individualmente não conseguiria (WOOLDRIDGE, 2002). Tal atividade é realizada através da intercomunicação que possibilita se organizarem e reorganizarem em grupos interativos, tanto entre si como para com um ambiente em que estejam inseridos.

Porém, de acordo com Hait (2000) sistemas que se utilizam do conceito de Orientação a Agente possuem uma alta complexidade agregada e seria de grande interesse reduzir a dificuldade de implementação de sistemas que utilizem esse paradigma através de uma arquitetura que facilite sua implementação e manutenção.

A arquitetura é a estrutura ou estruturas do sistema que abrangem os componentes de *software*, as propriedades externamente visíveis desses componentes e a relação entre eles, mas muitos sistemas têm sido construídos baseados exclusivamente em requisitos técnicos, deixando a arquitetura implícita no resultado final (BASS; CLEMENTS; KAZMAN, 2003; PRESSMAN, 2002).

Desse modo, a arquitetura torna-se um ponto essencial em um sistema de agentes, sendo ainda mais relevante quando tem-se em vista implementar diversos agentes cujas aplicações ainda não conhecemos em sua totalidade.

Para este trabalho, porém, levaram-se em consideração alguns aspectos da tecnologia de Orientação a Agentes e que são relevantes para a proposta. Em primeiro lugar, os agentes são componentes de softwares sociáveis; ou seja, eles são capazes de interagir com os demais agentes de forma a caracterizar uma "organização". Por outro lado, os agentes são capazes de racionalizar, pensar, acerca de algum assunto e produzir uma resposta direcionada por ações. E por último, os agentes estão inseridos em um ambiente que pode ser qualquer elemento externo ao agente capaz de interagir com ele.

Porém, a idéia de se abordar o problema com a orientação a agentes, por si só, não seria suficiente se os benefícios da orientação a agentes não fossem aparentes também para a utilização do *framework*, uma vez que muitos sistemas têm sido construídos deixando a arquitetura implícita no resultado final (BASS; CLEMENTS; KAZMAN, 2003; CLEMENTS; BACHMANN; BASS, 2003; PRESSMAN, 2002). Como o objetivo final desse projeto é facilitar a implementação de agentes, seria indesejável que a arquitetura do projeto não fosse fácil de manter.

Para o desenvolvimento do trabalho, dois pontos se mostraram essenciais: em primeiro lugar, a capacidade simular a capacidade de raciocínio que são alimentadas através de regras pré-definidas por seus administradores. Em segundo lugar, a distribuição dos agentes é levada amplamente em consideração, pois tem-se em vista o panorama de Sistemas Multi-agentes explanado em que a organização entre os agentes é um ponto crucial para o sucesso do modelo, afinal o conhecimento coletivo dos Agentes é que proporciona a maior parte dos resultados, e não os mesmos individualmente.

1.1. OBJETIVOS DO TRABALHO

O objetivo deste trabalho é propor uma arquitetura Multi-agentes capaz de instituir uma sociedade auto-organizável em número de agentes e construir um *framework* para implementação de sociedades de agentes dinâmicas composto por agentes racionais e com o ambiente representado por uma arquitetura orientada a objetos.

1.2. MOTIVAÇÕES E RESULTADOS ESPERADOS

Com a difusão dos conhecimentos a cerca de Orientação a Agentes e o aumento da demanda em aplicações reais (VENKATESAN, 2008) uma vertente vem sendo explorada: a aplicação de sistemas Multi-agentes. Venkatesan (2008) ainda explica que há um ganho na utilização de sistemas de IA de forma contextualizada na base de conhecimento do usuário: “como a especificação de objetivos a serem alcançados pela declaração do que querem em expressões da linguagem humana e a abstração de detalhes técnicos sobre o consumo de serviços e recursos” diz (VENKATESAN, 2008, p. 4). Com isso, se destacam estudos sobre técnicas e métodos que facilitem lidar com a complexidade intrínseca desta categoria de sistemas complexos. (MEA, 2001)

Atualmente já existem técnicas (BALDONI; BOELLA; GENOVESE; GRENNA; TORRE, 2008), métodos (HANNOUN; BOISSIER; SICHMAN, 2000; SCHWAMBACH, 2004) e inclusive ferramentas (BELLIFEMINE, 2003) para aplicação em Sistemas Multi-agentes. Nota-se até por esta diversidade que não há uma única teoria comumente aceita acerca de agentes. E por esta razão ao se deparar com uma implementação prática em Sistemas Multi-agentes se torna necessário adaptar algumas técnicas e até a criação de novos conceitos.

Motivando-se por essas questões, foi desenvolvido este trabalho que agrupa algumas técnicas, métodos e ferramentas para criação de um *framework* na expectativa de que possibilite utilizar o paradigma da Orientação a Agentes e Sistemas Multi-agentes instituindo agentes distribuídos em organizações gerenciadas, em que tais aplicações seja requerido o aspecto conversacional dos agentes e sua capacidade de racionalizar sobre *objetivos* através da seqüência linear ou não de *ações* mantendo sempre o estado do ambiente, sem a necessidade da utilização de uma linguagem de programação.

Duas soluções foram propostas: a idéia de *framework de racionalização*, e o conceito de *Broker* (para agências). A função dos agentes na sociedade pode mudar com o tempo, dependendo do papel que ele assume autonomamente. Para se organizarem, contam com a negociação entre eles e com um agente gerente denominado *broker*. E para agirem, os agentes possuem uma racionalização baseada em BDI e no aspecto Conversacional de Agentes.

1.3. MÉTODO E DESENVOLVIMENTO DO TRABALHO

A pesquisa se deu inicialmente pelo estudo da Orientação a Agentes e de Sistemas Multi-agentes e como criar organizações de agentes que possam crescer até um número indeterminado de agentes participantes e que aceitem novos tipos de agentes sem a necessidade de utilizar linguagens de programação de computadores.

Foram estudadas características funcionais e avaliações nos seguintes quesitos técnicos: facilidade de implementação de novos cenários, gerenciamento da sociedade e aderência aos padrões mais atuais de SMAs para interoperabilidade.

Após o estudo optou-se por pesquisar ferramentas, métodos e técnicas relacionadas à Orientação a Agentes de forma a permitir a sua utilização de uma maneira facilitada e gerenciada. Para viabilizar a implementação optou-se por utilizar uma arquitetura *OpenSource*

para a implementação do sistema *framework* proposto, a arquitetura JADE (BELLIFEMINE, 2003) que utiliza-se de padrões de comunicação amplamente divulgados e testados na implementação de agentes, onde foi possível estabelecer um plano de ação para a adaptação da arquitetura preexistente utilizando artefatos que já existem no Jade como o *Directory Facilitator* para registro de serviços, existência do conceito de comportamento e existência do conceito de ontologia.

Foram estudadas técnicas de gerenciamento da distribuição de agentes (BALDONI; BOELLA; GENOVESE; GRENNA; TORRE, 2008; HANNOUN; BOISSIER; SICHMAN, 2000) como base comparativa a essa nova proposta de *framework*, além de aspectos de racionalização de agentes (WOOLDRIDGE, 2000).

Por fim, métodos de representação do ambiente foram aplicados (ITO, 2006; SCHWAMBACH, 2004) ao *framework* que foi testado com base em simulações.

As simulações foram realizadas no domínio do acompanhamento de pacientes crônicos, o modelo GRPC (ITO, 2006), e foram realizadas análises dos resultados obtidos e considerações sobre o *framework* implementado, que levaram em consideração a disponibilidade das organizações e o gerenciamento da distribuição realizada sem intervenção humana.

2. Agentes

A Orientação a Agentes surgiu como uma linha de pesquisa em Inteligência Artificial, e três fortes correntes de pensamento guiaram os pesquisadores partindo do final da década de 1980, são elas (WOOLDRIDGE, 2002, p. 89):

- rejeição de representações simbólicas, e decisões tomadas a partir da manipulação de tais representações;
- como proposta, a comunidade de pesquisadores sugere que a inteligência dos agentes são fruto, na verdade, da interação que o agente mantém com o ambiente;
- a idéia de que o comportamento inteligente emerge da interação de vários comportamentos simples.

Para a arquitetura proposta neste trabalho utilizaremos a seguinte definição de Wooldridge (WOOLDRIDGE, 2002, p. 15, tradução nossa): “Um agente é um sistema computacional que está situado em algum *ambiente* e que é capaz de *ações autônomas* neste ambiente de modo a alcançar os seus objetivos”. E é justamente nas ações dos agentes e sua organização e no ambiente que este estudo envereda seus esforços.

2.1. AGENTES E AMBIENTE

Ambiente é tudo que possa interagir com um agente, e que não faça parte da sua estrutura interna (a estrutura interna do agente será explicada no capítulo seguinte “CLASSIFICAÇÃO DE AGENTES”).

Russel e Norvig (RUSSEL, 2003) propuseram a seguinte classificação para ambientes de agentes:

- acessível e inacessível: um ambiente acessível é aquele em que o agente obtém informação sobre o estado do ambiente de forma atual, precisa e completa (que não é o caso da maioria dos ambientes reais, como a Internet);
- determinístico e não-determinístico: determinístico é um ambiente onde uma ação tem somente um resultado possível. Não existe incerteza sobre o resultado proveniente das ações do agente;
- estático e dinâmico: um ambiente estático é aquele que só altera seu estado mediante ações de agentes, e nunca por fatores externos;
- discreto e contínuo: um ambiente discreto é aquele em que existe um número finito de ações e percepções sobre ele.

Wooldridge (WOOLDRIDGE, 2002) fez algumas considerações sobre estas definições. Em primeiro lugar, um ambiente determinístico não tem relevância em um ambiente complexo. Em segundo lugar, o não-determinismo está intimamente associado ao aspecto dinâmico dos agentes. E a partir dos estudos contidos em (Allen, 1990) é possível concluir que o estado do ambiente passa a interferir na racionalização do agente e, por conseguinte um agente passa a ser obrigado a coletar informações do ambiente antes de selecionar uma ação a executar (MOORE, 1990) já que não pode pressupor que o ambiente em um instante t1 é o mesmo ainda em um instante t0, e não tem como ter certeza de que o estado permanece o mesmo enquanto está executando suas ações, o que implica em um ambiente muito mais difícil de controlar e implementar no caso de ambientes dinâmicos [WOO02, p. 19].

Models for Actions é tida como a abordagem clássica para lidar com ações, tendo uma ação definida como uma transição de estado, ou seja, como um operador que executa e produz um novo estado. O resultado de um agente – de sua ação – é diretamente modelado a partir da modificação das variáveis de estado do ambiente. Apesar de essa abordagem ser suficiente na

IA clássica onde somente um agente atua, ela falha em SMAs onde muitos agentes atuam concorrentemente em um ambiente compartilhado (WEYNS, 2005).

Uma versão deste conceito é o modelo de ação de J. Ferber e J.P. Muller (1996) (***synchronous Model for Action***) baseado em três princípios básicos. Primeiro, distingue-se entre influencias e reações: influencias vêm de dentro dos agentes e são tentativas de modificar o curso de eventos no mundo; reações, que resultam em mudanças de estados, são produzidas pelo ambiente combinando influências de todos os agentes, dado o estado local do ambiente e as regras do mundo. **Essa distinção clara entre produto do comportamento do agente e reação ao ambiente permite a manipulação de ações simultâneas.** Os outros dois princípios são complementares e terminam por dar solidez ao modelo. O segundo princípio define que o modelo decompõe a dinâmica do sistema em duas partes, a dinâmica do ambiente e a dinâmica dos agentes situados no ambiente. E para tornar o modelo funcional o terceiro princípio descreve as diferentes dinâmicas dos SMAs traduzindo como um abstrato estado de máquinas. Contrariamente às teorias clássicas que só usam o estado do mundo para descrever a evolução dos SMAs, o modelo de evolução de Felber e Muller é descrito como sendo a transformação do que eles chamam estado, ou ambiente, dinâmico.

2.2. CLASSIFICAÇÃO DE AGENTES

Algumas definições delineiam um histórico acerca da teoria de agentes e facilitam o entendimento da evolução do conceito. Tais definições são centradas basicamente na estrutura interna de um único agente sem ainda se preocupar com sua organização em uma sociedade Multi-agentes. Essa diferenciação é importante para o trabalho, pois a partir dela é contornada a estrutura interna do agente e separada do modelo organizacional de sociedade Multi-agentes na qual estão inseridos. E tendo definido o termo “agente” passamos a denominar a estrutura interna que viabiliza a construção de um único agente de “agente” e de “organização” a existência de diversos agentes propondo-se a realizar um objetivo e comunicando-se entre si. (NWANA, 1996).

A definição de agente de Russel (2003) **AIMA** (*Artificial Intelligence: a Modern Approach*) é que um agente é qualquer coisa que pode perceber seu ambiente através de sensores e atuar neste ambiente através de atuadores. Já Brustolini (1991; FRANKLIN, 1995, p. 265) complementa que tais agentes são autônomos e capazes de ações intencionais e autônomas no mundo real. Outros autores ainda como Maes (1995) e Hayes-Roth (1995) compartilham dessa definição:

> Agentes autônomos são sistemas computacionais que habitam um ambiente complexo e dinâmico, percebem e atuam autonomamente neste ambiente, e fazendo isso realizam determinados objetivos ou tarefas para os quais foram projetados. (MAES, 1995, p. 108, tradução nossa).

> Agentes inteligentes realizam continuamente três funções: percepção de condições dinâmicas do ambiente; atuam de forma a afetar condições no ambiente; e racionalizam para interpretar percepções, resolver problemas, inferir e determinar ações. (HAYES-ROTH, 1995, p. 3, tradução nossa).

Outras propostas ainda acrescentam critérios à definição de agente como a habilidade de execução autônoma ou a habilidade de agir orientado por domínio:

> Vamos definir um agente como uma entidade de software persistente dedicada a um propósito específico. ‘Persistente’ distingue agentes de sub-rotinas; agentes têm suas próprias idéias sobre como cumprir tarefas, suas próprias agendas. ‘Propósito especial’ os distingue de todas as aplicações multifunções: agentes são tipicamente muito menores. (SMITH; CYPHER; SPOHRER, 1994, p. 7, tradução nossa).

Outras propostas se concentram ainda na interação dos agentes com usuários, como um formato de instruir os agentes de forma que estes possam auxiliar nas atividades dos usuários: “aplicar regras como um método de o usuário instruir um agente facilita o trabalho do usuário em modificar suas instruções e entender o que um agente faz”. (GROSOF, 1997, p. 4). Ou ainda no modo de comunicação entre os agentes e os usuários baseando-se na definição de Coen (1994) ao afirmar que agentes de software são programas que se envolvem em diálogos, negociam e coordenam a transferência de informações.

Wooldridge e Jennings (1995, p. 2) sintetizam essas definições: “um hardware ou (mais usualmente) um sistema de computador baseado em software que goza das seguintes propriedades: autonomia [...], habilidade social [...], reatividade [...] e pró-atividade [...].” e ainda destacam os detalhes destas características onde **autonomia** diz como agentes operam sem a direta intervenção de humanos ou outros, e tem algum tipo de controle sobre suas ações e estados internos; A **habilidade social:** como agentes interagem com outros agentes (e possivelmente humanos) via algum tipo de linguagem de comunicação entre agentes (ACL – *Agent Communication Language*); r**eatividade:** percepção do ambiente por partes dos agentes (que podem ser um mundo físico, um usuário via interface gráfica, uma coleção de outros agentes, a Internet, ou talvez todos esses combinados), e responde em tempo hábil para mudar o que ocorre nele; e por fim a **pró-atividade**, uma vez que agentes não agem simplesmente em resposta ao ambiente, mas também são capazes de demonstrar comportamento orientado a objetivos tomando a iniciativa. (WOOLDRIDGE; JENNINGS, 1995).

Essa última definição (WOOLDRIDGE; JENNINGS, 1995) se apresentou como a abordagem mais aderente ao que se deseja neste trabalho, pois sugere uma classificação de

agentes heterogêneos. Franklin (1996) partindo de diversas definições chegou a alguns parâmetros através dos quais também conseguia classificar os agentes segundo critérios comuns. Na tabela 2-1 estão descritos os critérios, que são ilustrados na Figura 2-1 (NWANA, 1996):

Propriedade	Outros nomes	Significado
Reativo	Percepção e atuação	Responde em um espaço de tempo capaz de realizar mudanças no ambiente
Autônomo		Demonstra controle sobre suas próprias ações
Orientado a objetivo	Pró-ativo intencional	Não age simplesmente em resposta ao ambiente
Temporalmente contínuo		É um processo que roda continuamente
Comunicativo	Socialmente capaz	Comunica-se com outros agentes, possivelmente com pessoas
Aprendizado	Adaptativo	Muda seu comportamento baseado em experiências prévias
Móveis		Capaz de se transportar de uma máquina para outra
Flexível		Ações não são pré-descritas
Caráter		"Personalidade" confiável e estado emocional

Tabela 2-1 - Critérios para classificação de agentes (FRANKLIN, 1996)

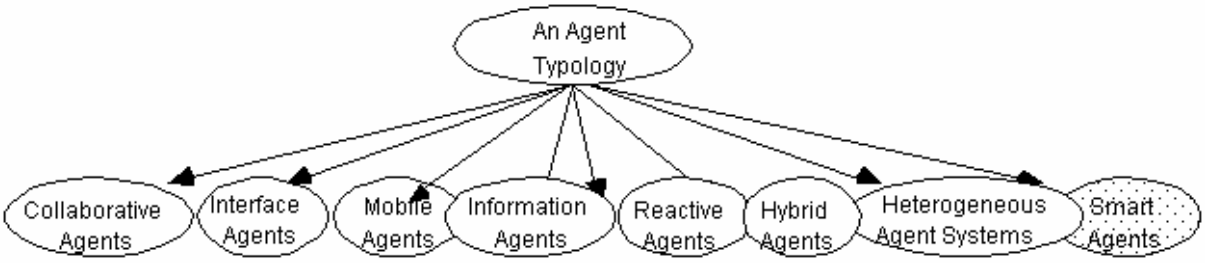


Figura 2-1 – Uma classificação de agentes de software (NWANA, 1996)

"*Smart agents* está mais relacionado à *aspiração* dos pesquisadores de agentes ao invés de uma *realidade*" (NWANA, 1996, cap. 5), porém conforme as pesquisas em torno da Orientação a Agentes evoluíram em prol dessa aspiração, duas principais correntes se destacaram: a de agentes reativos representada principalmente pela proposta de Brooks

(*Subsumption Architecture*) e a de agentes pró-ativos, com um exemplo em destaque de Vere e Bickmore (WOOLDRIDGE, 2002, p. 97).

No entanto, não se tratavam de teorias excludentes, e sim complementares. Ou seja, a partir das duas correntes foi sendo contornado um conceito de que os agentes podem ser, ao mesmo tempo, ativos e reativos.

Esta idéia deu a origem ao termo cunhado por (WOOLDRIDGE, 2002, p. 97) de "Agentes híbridos" que se refere à idéia de existir uma aplicação de muitas camadas, com cada uma das camadas responsável por um aspecto da aplicação. Este conceito é utilizado em muitos sistemas atuais (BELLIFEMINE, 2003) (BALDONI; BOELLA; GENOVESE; GRENNA; TORRE, 2008) e é um tema que se refere diretamente a "organização de agentes", pois a partir de (SILVA; CASTRO, 2002) é possível concluir que a arquitetura de um sistema de agentes influencia a organização dos mesmos. Porém, antes de continuar é necessário estudar o comportamento dos agentes individualmente para que faça sentido discutir suas interações.

2.3. O AGENTE COMO SOFTWARE "PENSANTE": BDI – BELIEFS, DESIRES E INTENTIONS

Sistemas Multi-agentes são compostos por agentes inteligentes que racionalizam a respeito do ambiente em que executam e também sobre suas intenções (ou vontade de agir).

Segundo Wooldridge (2000, p. 21) a racionalização dos agentes tem sua origem na filosofia, analogamente à racionalização humana. Ele distingue o processo de racionalização em dois: prática (*pratical reasoning agents)* e teórica (*theorical reasoning agents*). Segundo ele a diferença essencial é que a racionalização prática sempre é direcionada por ações, ao contrário da teórica que altera a visão de mundo somente para o agente que racionaliza.

Como o que nos importa para este trabalho é a deliberação prática, ou seja, aquela que pode implicar em alguma ação por parte dos agentes, vamos nos ater em discuti-la apenas tendo em vista que a racionalização teórica não seria útil em termos de modificação do ambiente (WOOLDRIDGE, 2000).

Observando o processo de racionalização prática existem ao menos duas atividades distintas: a primeira envolve decidir *quais* estados de satisfação pretendemos atingir, e o segundo envolve decidir *como* queremos atingi-los. Esse processo de decidir *quais* estados de satisfação atingir é denominado *deliberação.* Já a escolha de *como* atingir os estados de ação nos leva até um *plano*.

Muita atenção tem sido dada ao "problema de planejamento" em IA, particularmente ao problema de automatizar a racionalização orientada a ações de soluções em IA e principalmente no que diz respeito ao objetivo de selecionar ações particulares em um universo de ações possíveis, um *goal*, ou um conjunto de *goals*. Os trabalhos nessas áreas resultaram em várias técnicas úteis de representação e racionalização a respeito de ações e *goals* como (FIKES; NILSSON, 1971; SACERDOTI, 1977; GEORGE; LANSKY, 1987; GEORGE, 1987b) apud (WOOLDRIDGE, 2000).

Uma forma reduzida de representar a racionalização é adaptada de Wooldridge (2000, o. 31) e é apresentada no quadro 2-1:

```
1. Percepcao perception;
2. Belief[] beliefs = recuperarCrencasIniciais();
3. Intention[] intentions = recuperarIntencoesIniciais();
4. Plan plan;
5. While(true)
6. {
7.    perception = recuperarPercepcaoAtual();
8.    beliefs = avaliarCrencasAtuais(beliefs,perception);
9.    intentions = deliberar(beliefs, intentions);
10.   plan = planejar(beliefs,intentions);
11.   executar(plan);
12. }
```

Quadro 2-1 - Processo de racionalização em Agentes (WOOLDRIDGE, 2000)

O quadro 1 é totalmente baseado no *Pratical Reasoning Agents* (WOOLDRIDGE, 2002), a versão de BDI do Wooldridge. De acordo com ele, nas linhas 8 e 9 o agente *delibera* sobre os seus *beliefs* - crenças (*quais* condições satisfariam o agente) e decide o que vai fazer, após decidir monta seu plano na linha 10 (*como* essas condições satisfazem o agente).

Para completar o sistema BDI se introduz o conceito de "*desires*". Segundo Wooldridge, toda vez que um agente decide executar uma ação ele possui um universo de *desires* (*goals*) distintos que poderia executar e que o satisfazem no seu término. No entanto, deve-se avaliar desse universo de *desires* distintos qual, ou quais, deles é útil dadas as condições atuais do ambiente. Todo esse mecanismo faz parte da *deliberação*, e está representado no quadro 2-2:

```
1.  Intention[] deliberar(Belief[] beliefs, Intention[] intentions)
2.  {
3.       Desire[] desires;
4.       desires = avaliarPossibilidades(beliefs, intentions);
5.       return filtrar(beliefs, desires, intentions);
6.   }
```

Quadro 2-2 – Deliberação: a definição dos goals (WOOLDRIDGE, 2000)

O agente realiza um filtro levando em consideração todas os *beliefs* – crenças que são válidas neste momento. Tudo o que o agente acredita e considera sobre si mesmo, de outros agentes e do ambiente em que está executando. Dadas todas essas crenças, ele consegue selecionar objetivos para interação neste ambiente dentre um universo que possui alguns *desires*. Esses objetivos são recuperados na linha 4. Por fim, o agente realiza um filtro sobre todos esses objetivos que, nesse momento, são alternativas de execução e escolhe dentro desse universo quais são suas *intentions* – planos de execução, ou seja, o que ele pretende realizar de fato.

Essa técnica se chama BDI (*belief-desire-intention*) e outros autores ainda sustentam a mesma idéia de deliberação prática sobre *belief, desire e intention*: Segundo Thomason (THOMASON, 2000) a deliberação prática de planejar está centrada justamente na existência

de crenças - *beliefs* (de “idéias” pré-existentes) e desejos - *desires* (de como gostaria que o ambiente ficasse após a execução). O conceito permite a um agente pensar tendo apenas parte do conhecimento ao invés de conhecer todos os outros agentes e ambiente (WOOLDRIDGE, 2000).

Enfim, o mecanismo de BDI se apresenta como uma solução que, segundo Wooldridge (2002), explica a necessidade dos pesquisadores em agentes de simular comportamentos humanos para torná-los práticos, concluindo ainda que “a evolução da ciência tem sido marcada por uma mudança gradual de explicações teológicas ou figurativas para matemáticas” (WOOLDRIDGE, 2002, p. 29, tradução nossa).

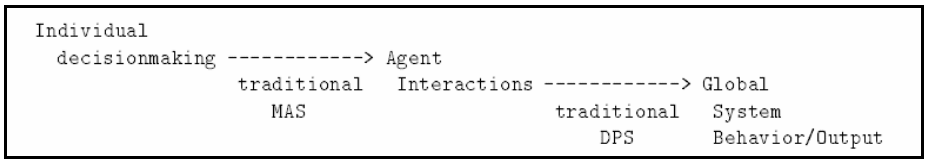
3. Organização de Agentes

O desenvolvedor de agentes implementa um único agente e o faz de forma que possa interagir com os demais, implicando que as organizações de agentes sejam compreendidas como um conglomerado de agentes autônomos que podem interagir para atingir um objetivo particular, e possivelmente comum.

3.1. ORGANIZAÇÕES SMA E RDP

Segundo Durfee (DURFEE, 1994) duas vertentes de Sistemas de Agentes são vislumbradas: SMA, ou Sistemas Multi-agentes e RDP, ou Resolução Distribuída de Problemas. Estudos em SMAs estão centrados em como agentes automotivados e tomadores de decisão individuais podem descobrir formas de interagir entre si de forma estável e até prevista. Já pesquisas em RDP se interessam mais em considerar como essas interações podem ser iniciadas, controladas e exploradas de forma a permitir que o sistema como um todo atinja um objetivo necessário externamente.

Para começar, é difícil se não impossível caracterizar essa pesquisa exclusivamente no campo de SMAs ou RDPs de acordo com a definição de Durfee (DURFEE, 1994) que categoriza os sistemas de acordo com as características do ambiente e do agente, quadro 3-1 e tabela 3-1. Segundo Durfee (1994) propriedades de agentes estão relacionadas à sua individualidade, racionalização, compartilhamento do conhecimento e capacidades. Por outro lado, quando nos referimos às características do ambiente estamos se pensa se um ambiente é estático, fechado e na capacidade de previsão das alterações.



Quadro 3-1 – SMA e RDP (DURFEE, 1994)

	Agent Properties	Environ Properties	System Properties
MAS	variable	fixed	fixed (internal)
DPS	fixed	variable	fixed (external)
?	fixed	fixed	variable

Table 1: Matrix of Research Agendas and Properties to Vary.

Tabela 3-1 - Proposta de classificação para de sistemas em SMA ou RDP (DURFEE, 1994)

A dicotomia acerca de SMA e RDP sob a ótica dos agentes é discutida por Rosenschein e Genesereth (ROSENSCHEIN; GENESERETH, 1985) ao relatar que RDP é uma categoria especial de SMAs que são benevolentes (sem interesses próprios), e portanto agentes em sistemas RDP poderiam ser gerenciados pelo ambiente de execução. Complementa ainda Durfee (1994, p. 4) que *benevolência* significa que "um agente *deseja* ajudar-se mutuamente sempre que possível".

Em termos de ambiente e organização, Jaime em (HUBNER; SICHMAN, 2003) discute a diferença entre SMA e RDP definindo quatro tipos de SMAs baseados na sua estrutura organizacional.

- Organizações implicitamente definidas
    - AR – não existe motivação para se organizar;
    - AC – organizam de acordo com suas próprias motivações, cada agente pode ter uma representação da sociedade diferente das demais.
- Organizações instituídas
    - OR – obriga os agentes a adotarem determinada organização, sem que os mesmos possam decidir sobre isso;

- OC – possui uma organização instituída, mas os agentes podem escolher mudá-la ou não.

Sob o ponto de vista organizacional organizações do tipo OR não deveriam ser considerados SMAs e sim RDP (HUBNER; SICHMAN, 2003) deixando assim implícita uma distinção entre SMA e RDP sob o ponto de vista organizacional de agentes.

No entanto, analisando mais características de SMAs e RDPs, Sichman em (ÁLVARES, 1997) cita a obra (DURFEE, 1994) e reforça as afirmações de Rosenschein e Genesereth (ROSENSCHEIN; GENESERETH, 1985) explicando que às vezes é difícil dizer se um sistema é construído sobre RDP ou SMA, sendo mais viável discutir à luz das motivações científicas e técnicas dos projetistas do sistema.

Esse formato de interação entre os agentes é condicionado por diversos fatores, tais como a interação com o ambiente e também a forma como eles têm suas responsabilidades divididas dentro da sociedade o que, por conseguinte influencia a forma como são construídos.

Assim, entende-se que em um mesmo sistema pode haver tanto características SMAs ou RDPs dependendo da abordagem utilizada.

3.2. ARQUITETURA DE SISTEMAS DE AGENTES

Quando se trata de organização de agentes, estuda-se o formato como esses agentes se comportam para atingirem objetivos. Por outro lado ao tratar da arquitetura, na verdade, estuda-se como o sistema é construído e como são esquematizadas suas estruturas lógicas que possibilitam que ele funcione.

Independentemente da abordagem de organização, objeto de estudo acerca de sistemas de agentes, sejam elas Sistemas Multi-agentes (SMAs) ou Resolução Distribuída de

Problemas (RDP), decisões arquiteturais sobre como organizar esses agentes acabam partindo de experiências próprias, e não de uma organização particular, o que pode se tornar crítico se estivermos tratando de diversos tipos de agentes distintos (SILVA; CASTRO, 2002).

FIPA, por exemplo, é um padrão de arquitetura de sistemas de agentes e inclui um padrão de comunicação entre agentes, indicado por (WOOLDRIDGE, 2002). O foco principal que define o modelo de agentes por eles pregado é a *comunicação entre agentes*, pois segundo FIPA (2008a, p. 30, tradução nossa) os atos, ações, dos agentes podem ser representados a partir da troca de mensagens entre eles: "agentes comunicam-se através da troca de mensagens que representam ações, e que são codificadas em uma linguagem de comunicação de agentes". Essa representação é apresentada na figura 3-1, no "*Message Transport System*". Enquanto o "*Agent Management System*" se comporta como um mecanismo interno da Plataforma de Agentes (AP) (que implementa FIPA) e se preocupa em registrar todos os agentes que existem na plataforma para garantir o gerenciamento de nomes. Por fim, o "Directoty Facilitator" (*yellow pages* ou páginas amarelas) disponibiliza recursos para que os agentes publiquem os serviços para os quais estão disponíveis (FIPA, 2008b, p.5, tradução nossa):

> O DF [*directory facilitator*] provê serviços de páginas amarelas a outros agentes. Os agentes podem registrar seus serviços junto ao DF ou perguntar ao DF para encontrar por serviços que são oferecidos por outros agentes. Múltiplos DFs podem existir em uma AP [*agent plataform*, plataforma de agentes].

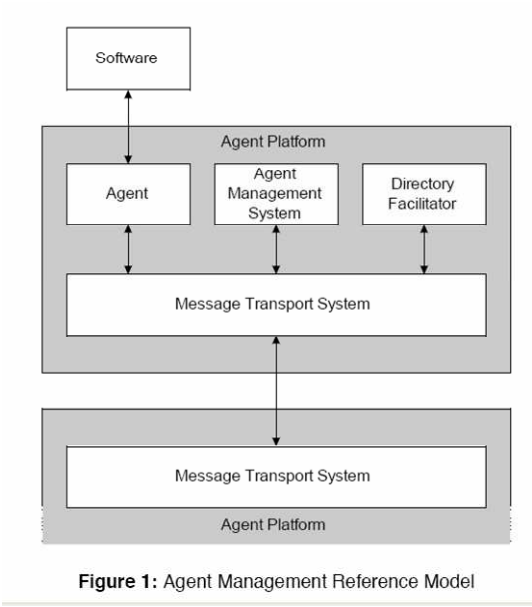


Figura 3-1 - Agent management reference model (FIPA, 2008a)

No entanto, a abordagem centrada na comunicação dos agentes da FIPA é uma abordagem dentro de um universo de abordagens de arquitetura em Sistemas de Agentes que é detalhado a seguir.

A arquitetura flat (figura 3-2), por exemplo, é um exemplo onde nenhum controle de um ator sobre o outro é assumido. A principal vantagem desta arquitetura é que ela suporta autonomia, distribuição e evolução contínua da arquitetura de um ator, porém a principal desvantagem é que ela requer uma elevada racionalização e comunicação por parte de cada ator participante para se identificarem e organizarem.

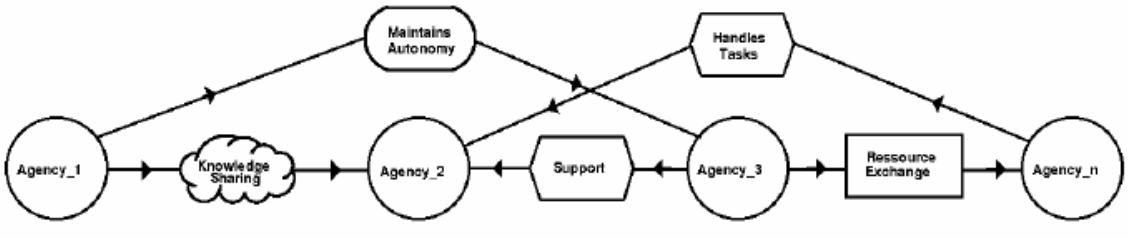


Figura 3-2 - Estrutura flat para agentes (SILVA; CASTRO, 2002)

Para alguns autores (NWANA, 1996) quando falamos em termos de uma arquitetura de agentes completa pensamos em agentes que tem exacerbadas essas suas características, e

segundo (SILVA; CASTRO, 2002) podem possuir e para possibilitar estruturas de organização como FIPA, Contract-Net ou Flat, estaremos tratando com alguns agentes com características particulares. Vamos citar cinco tipos de agentes (monitor, matchmaker, wrapper, mediator e broker) e comentar quais são as arquiteturas que segundo Silva e Castro (2002) estes são mais recorrentes.

Um **monitor** (SILVA; CASTRO, 2002) (figura 3-3) aceita inscrições de diversos candidatos e ao menos um assunto de interesse, recebe solicitações para envio de notificações como eventos e alertas de subscritos para eventos relevantes. Um subscrito registra-se para notificação de mudanças de estado para assuntos distribuídos, recebe notificações com a informação do estado corrente, e atualiza o local de informação. Este funcionamento e uma possível arquitetura está representado na figura 3-3:

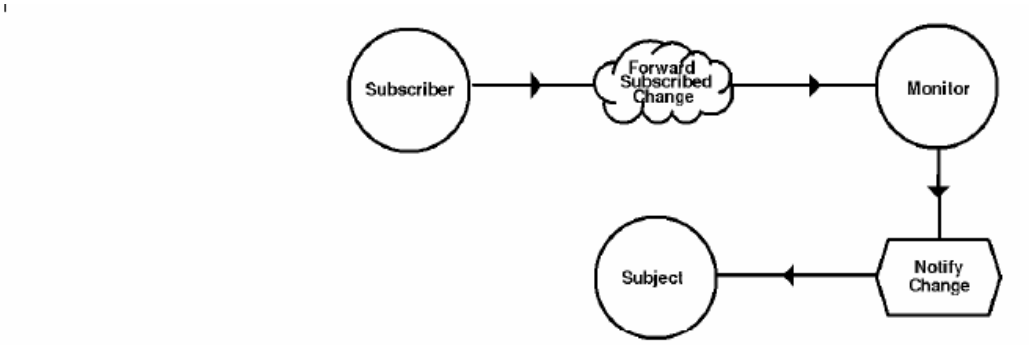


Figura 3-3 - Agente monitor (SILVA; CASTRO, 2002)

Um **broker** (SILVA; CASTRO, 2002) (figura 3-4) é um árbitro e intermedeia acessos a serviços de um ator para satisfazer as requisições de outros consumidores que por sua vez podem também assumir o papel de provedor de serviços, e inversamente provedores podem ser também consumidores. Papéis de cada ator são estabelecidos no contexto de um diálogo particular.

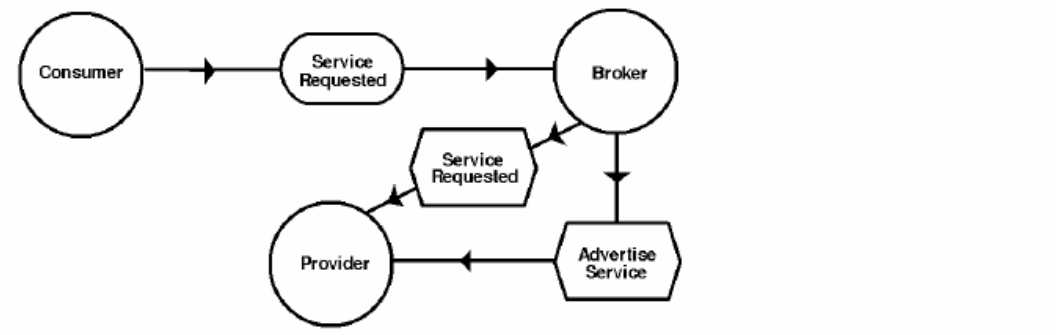


Figura 3-4 - Agente broker (SILVA; CASTRO, 2002)

Um **mediator** (SILVA; CASTRO, 2002) (figura 3-5) e um número de colegas representam tanto o papel de iniciador como de trabalhador, onde o iniciador endereça o mediador na posição de questionar diretamente outro colega (trabalhador). Ele tem modelos de amizade de colegas e coordena a cooperação entre eles sendo que cada colega possui o modelo de amizade do mediador. Comparativamente um *broker* vincula provedores e consumidores, enquanto um mediador encapsula interações e mantém modelos de iniciadores e comportamentos de todos trabalhadores ao mesmo tempo para reforçar mecanismos de autoridade.

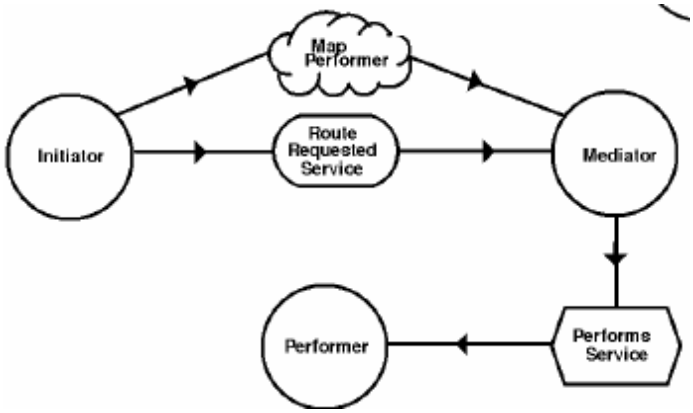


Figura 3-5 - Agente mediator (SILVA; CASTRO, 2002)

Matchmaker (SILVA; CASTRO, 2002) (figura 3-6) busca um provedor que corresponda à requisição de serviço de um consumidor, e então encaminha o consumidor para o provedor escolhido diretamente. Um *broker* pode ser utilizado em conjunto para manipular diretamente todas as interações ente o consumidor e o provedor. Aqui a negociação por serviços e a provisão de serviço atual são separadas em duas fases distintas.

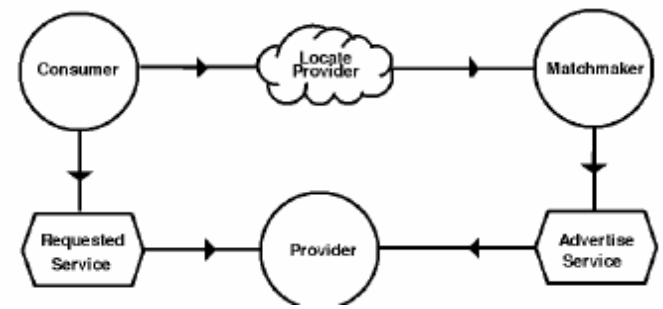


Figura 3-6 - Agente matchmaker (SILVA; CASTRO, 2002)

Por fim, um **wrapper** (SILVA; CASTRO, 2002) (figura 3-7) possui a característica marcante de ser "domínio-específico" e interagir somente com um sistema legado permitindo que uma aplicação legada seja acoplada a um SMA e realiza a interface tanto com o cliente como com o legado, como um tradutor. Assim, garante que os protocolos de comunicação sejam respeitados e o sistema legado permanece separado dos clientes.

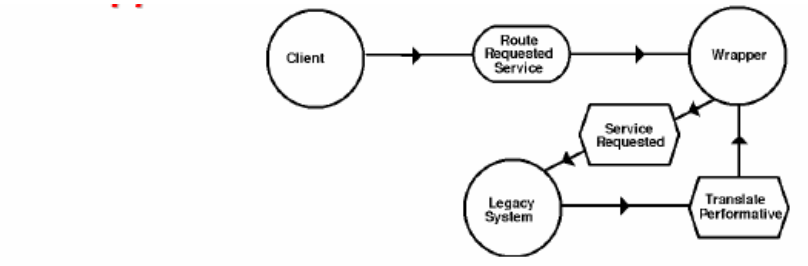


Figura 3-7 - Agente wrapper (SILVA; CASTRO, 2002)

4. Framework para o gerenciamento da organização de agentes

Para essa implementação deve-se observar que muitos sistemas têm sido construídos baseados exclusivamente em requisitos técnicos, deixando a arquitetura implícita no resultado final (BASS; CLEMENTS; KAZMAN, 2003; JANCA, 1995).

Assim, a idéia de se abordar o problema com a orientação a agentes, por si só, não seria suficiente se os benefícios da orientação a agentes não fossem aparentes também para a aplicação de Sistemas Multi-agentes e sendo assim, algumas decisões cruciais em torno da teoria de agentes foram tomadas.

Para desenvolvimento do trabalho, dois pontos se mostraram essenciais para facilitar a aplicação em Sistemas Multi-agentes: em primeiro lugar, a possibilidade de proporcionar a simulação de capacidade de raciocínio que seriam alimentadas através de regras pré-definidas. Em segundo lugar, a distribuição dinâmica dos agentes incluindo a criação de novos agentes com novos papéis, pois considera-se tendo em vista o panorama de Sistemas Multi-agentes já explanado que a organização entre os agentes é um ponto crucial para o sucesso do modelo, já que o conhecimento coletivo dos Agentes que proporciona a correta interação com os usuários, e não os mesmos individualmente.

Assim, em Durfee (DURFEE, 1994, p. 4) diz que *benevolência* significa que “um agente *deseja* ajudar-se mutuamente sempre que possível” já que Sistemas Multi-agentes são compostos por agentes inteligentes que racionalizam a respeito do ambiente em que executam e também sobre suas intenções (ou vontade de agir).

Os agentes propostos não são capazes de racionalizar sobre suas intenções - retomando a afirmação de Wooldridge (2000, p. 45) ao tratar intenções como “estados da mente” (ou *beliefs*) - sendo assim de certa forma benevolentes. Então na solução proposta os agentes

ficam em uma posição de atingir a objetivos específicos sem saber o quanto determinado objetivo vai ou não agradá-lo.

E em termos de ambiente, ou sociedade dependendo do autor, uma vez que tratamos de agentes benevolentes RDP (ROSENSCHEIN; GENESERETH, 1985) disponibilizamos a correspondência de um conjunto de planos para um objetivo e por papel que podem integrar o ambiente a qualquer momento dependendo da necessidade.

A categorização de sistemas de agentes em SMAs ou de RDPs é difícil sem o auxilio dos projetistas e desenvolvedores do programa, e tanto um como outro conceito possuem aplicações e recursos interessantes ao desenvolvimento de software de agentes.

Relembrando a classificação de sistemas de agentes em SMA ou RDP de Durfee (1994) onde de acordo com as características dos agentes (se são fixas ou variáveis) e do ambiente (se são fixas ou variáveis), propõe-se uma nova categoria com propriedades de agentes *variáveis* e propriedades de ambiente *variáveis* classificando assim o *framework* em uma categoria que não é nem exclusivamente SMA ou exclusivamente RDP.

Retomando ainda Rosenschein e Genesereth (ROSENSCHEIN; GENESERETH, 1985) que definem RDP como uma categoria especial de SMAs (e não uma abordagem distinta) que são benevolentes (sem interesses próprios), se pode concluir que os agentes estão dispostos, conforme a necessidade, a assumir diversos papéis pré-definidos para atuar na sociedade. A seguir é apresentada em detalhes a solução proposta no trabalho.

4.1. ORGANIZAÇÃO BASEADA EM PAPÉIS

Algumas abordagens em Sistemas Multi-agentes tratam de organizações ou arquiteturas, como CNET, FIPA e Flat. Além disso, foram citados tipos de agentes para compor qualquer arquitetura e organização independente da sua abordagem e foram eles: mediator, matchmaker, monitor, broker e wrapper.

Dentre as abordagens citadas, a arquitetura JADE (BELLIFEMINE, 2003) foi utilizada para implementação do *framework*, e esta utiliza comunicação FIPA *Compilant language* cuja a proposta se baseia no formato de padronização FIPA, tendo em vista que necessitamos dos mecanismos de *directory facilitator* para possibilitar a implementação de papéis, pois é ali que os agentes registram os serviços que sabem oferecer toda vez que se registram em um papel, ou caso estejam “livres”. Mas tendo em vista os diversos critérios de classificação de agentes e de organização de agentes, classificar a proposta somente sob a ótica FIPA é o mesmo que deixar de lado diversas características importantes do *framework* proposto, que são discutidas a seguir.

Os tipos de agentes anteriormente relatados *matchmaker* e *mediator* foram descartados, pois deseja-se que a proposta tenha um gerente da organização que não fique aguardando todas as requisições dos demais agentes. Já os agentes ativos da organização, realizam também o papel de *monitor*.

E por fim, ambos negociam a partir do *contract-net.* O CNET (figura 4-1) envolve um gerente que cria uma proposta de requisição a um serviço particular para todos os participantes. Este gerente recebe “propostas” para atingir a requisição de serviço a um “custo” particular. O gerente então seleciona uma destas propostas e indica a aceitação para um único participante que realiza o serviço contratado que posteriormente informa o gerente quando o serviço estiver finalizado.

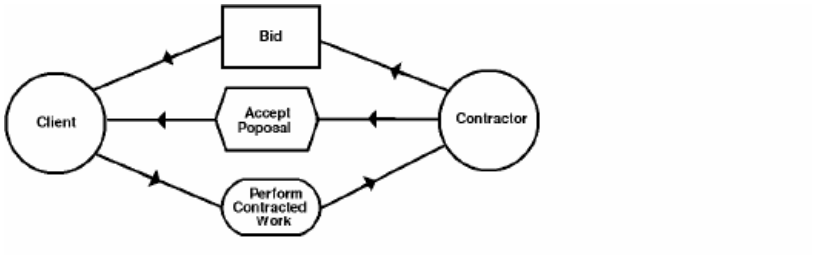


Figura 4-1 - Arquitetura Contrac-Net (SILVA; CASTRO, 2002)

Logo, a partir de todas as abordagens estudadas, o *broker* se destaca como uma possibilidade de criar uma sociedade que tenha seu crescimento em termos de **participantes ilimitada**, sendo os agentes do mesmo tipo ou não, sem influenciar na construção interna dos agentes atuando como gerente..

A série NorMAS "*normative multiagent systems*" (BALDONI; BOELLA; GENOVESE; GRENNA; TORRE, 2008; BOELLA; TORRE, 2004a; BOELLA; TORRE, 2004b; BOELLA; TORRES, 2005) propõe e implementa uma arquitetura baseada em *papéis* e *broker*, na qual é possível separar a criação dos agentes do seu mecanismo de racionalização, fazendo assim com que seja criados tantos novos agentes quantos necessários baseados em regras de raciocínios que já existem, que é a idéia principal do *framework* de como suportar o gerenciamento de uma organização onde não se sabe todos o agentes participantes, garantindo flexibilidade ao mesmo tempo que permite a expansão da aplicação.

Os papéis no NorMAS são difusos em toda a arquitetura, e em alguns momentos os conceitos de organização, e estrutura de papel de cada agente se misturam:

> Classes que extendem a classe `Role` precisam ser *inner classes* de uma classe que estenda a classe `Organization`. Assim o `Role` [papel] pode acessar os estados privados da organização e de outros papéis. Uma vez que papéis são implementados como *inner classes* [de organização], uma instância de papel precisa estar na mesma instância da plataforma a qual pertença à organização. Além do mais, o papel do agente pode ser visto como um objeto d ponto de vista da organização e dos papéis que tenham referência a ele, e mandem mensagens a ele. (BOELLA;TORRES, 2005, tradução nossa).

Este trabalho não pretende ir por esse caminho, pois o modelo NorMAS possui implicâncias indesejáveis para a proposta do *framework*, como a possibilidade de comportamentos dos agentes manipularem diretamente características da organização social e de demais agentes. Este ponto é crucial no tocante da diferença entre as duas arquiteturas. De acordo com (FRANKLIN, 1996, p. 10, tradução nossa):

> Alguns agentes com a arquitetura fragmentada não são sistemas Multi-agentes. Muller, Pischel e Thiel (1995) classificam tais arquiteturas em verticalmente e horizontalmente fragmentada. Em sistemas com arquitetura horizontalmente fragmentada cada camada tem acesso a perceber e agir, realizando uma decomposição em sub-agentes. Em sistemas com a arquitetura verticalmente fragmentada, somente a camada mais baixa percebe, e somente a camada mais alta age, fazendo assim uma decomposição do conceito de Multi-agentes

No caso deste trabalho, os papéis e *players* não são distribuídos. Eles personificam o mesmo agente, assim a organização entre os agentes não interfere em sua estrutura interna, incorrendo-se o risco de descaracterizar Sistemas Multi-agentes.

Outra abordagem também relacionada ao *framework* proposto exemplifica um mecanismo de racionalização relacionado a papéis (URUGUAY; HIRATA, 2006) e nesse caso os *goals* são mapeados para as ações, ao invés das *intentions* que é o caso do estudo aqui apresentado. Porém Uruguay e Hirata não deixam explícita a relação entre *intentions* e ações, mas para cada *goal* algumas ações são tomadas, como o caso do *goal* decolar (*take off*) - que implica na ação atualizar movimento (*update movement*), figura 4-2, deixando claro que o conceito de ações a serem realizadas para cada *goal* existe.

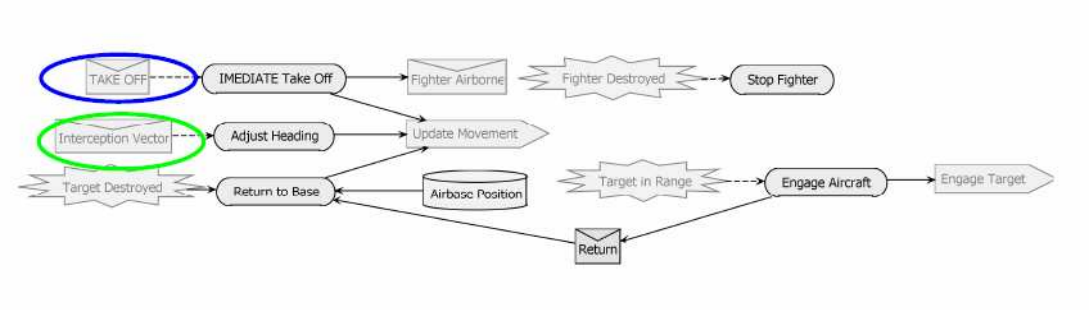


Figura 4-2 - Ações realizadas a partir de goals (URUGUAY; HIRATA, 2006)

O Moise (HANNOUN; BOISSIER; SICHMAN, 2000) é o ponto de partida para a construção desse mecanismo BDI baseado em papéis (URUGUAY; HIRATA, 2006). Nele pode-se entender melhor o porquê de determinadas decisões. Essa categoria de agentes demonstra que as missões dos agentes podem possuir dependências “a existência de um link de autoridade ente [...] dois papéis permitirá a ele enviar esta requisição com uma maior força, comandando sua delegação” (HANNOUN; BOISSIER; SICHMAN, 2000, p. 6, tradução nossa). Apesar de à primeira vista parecer uma invasão à racionalização do agente, a delegação através de *links* no Moise é totalmente justificada pelo fato de ser uma proposta centrada na organização no qual o modelo de organização é imposto ao agente (HANNOUN; BOISSIER; SICHMAN; SAYETTAT, 2000, p. 1).

Apesar da grande semelhança entre Moise e o proposto neste estudo no que se refere aos *goals* e às ações (também agrupadas para cada papel), no *framework* proposto não existe a possibilidade de um agente acionar parte de seu plano presente em outro agente, ficando a cargo de cada agente racionalizar sobre seus relacionamentos, já que trata-se de organizações *agent-centered* onde “os agentes computam, eles próprios, suas relações” (HANNOUN; BOISSIER; SICHMAN; SAYETTAT, 2000, p. 1, tradução nossa) já que são eles os responsáveis por negociarem os papéis a serem assumidos. E assim, sempre que os agentes precisem se relacionar ocorre uma negociação entre eles baseada em CNET (*Contract-Net*) de forma que um agente não interfira na estrutura de um outro agente.

4.2. DISTRIBUIÇÃO DOS PAPÉIS NA SOCIEDADE

O mecanismo central de organização das sociedades de agentes instituídas nesta arquitetura é a possibilidade dos agentes se candidatarem e se descandidatarem a papéis, gerenciados pelo *broker* que é responsável por criar novos agentes “livres” (sem papéis).

Esta proposta não possui representação formal da sociedade deixando com que os agentes se agrupem em um SMA de acordo com as necessidades, ao estilo das organizações implicitamente definidas do tipo AC definidas por Jaime em (HUBNER; SICHMAN, 2003): os agentes se organizam de acordo com suas próprias motivações e cada agente pode ter uma representação da sociedade diferente das demais.

Assim sendo, toda a representação formal na proposta está nos agentes, e nunca a respeito da sociedade. Apesar disso, na sua estrutura básica também não existe uma classe de agente concreto, devido a existência do *broker* e *papéis*.

Os agentes do *framework* proposto são *benevolentes* (como explicado na introdução do capítulo 4) e assumem novos papéis caso estejam disponíveis, e realizam isso a através de um diálogo baseado em CNET (**Contract-net**).

Os agentes se organizam através da emissão de uma proposta para algum agente livre atender a determinado serviço e conseqüentemente assumirem papéis. Esse comportamento é a parte de racionalização que é configurável e cadastrável por papéis e é representado na figura 4-3. A ressalva com relação ao CNET, porém, é que no *framework* proposto os Agentes não consideram diferentes pesos para aceitarem papéis justamente por serem *benevolentes*.

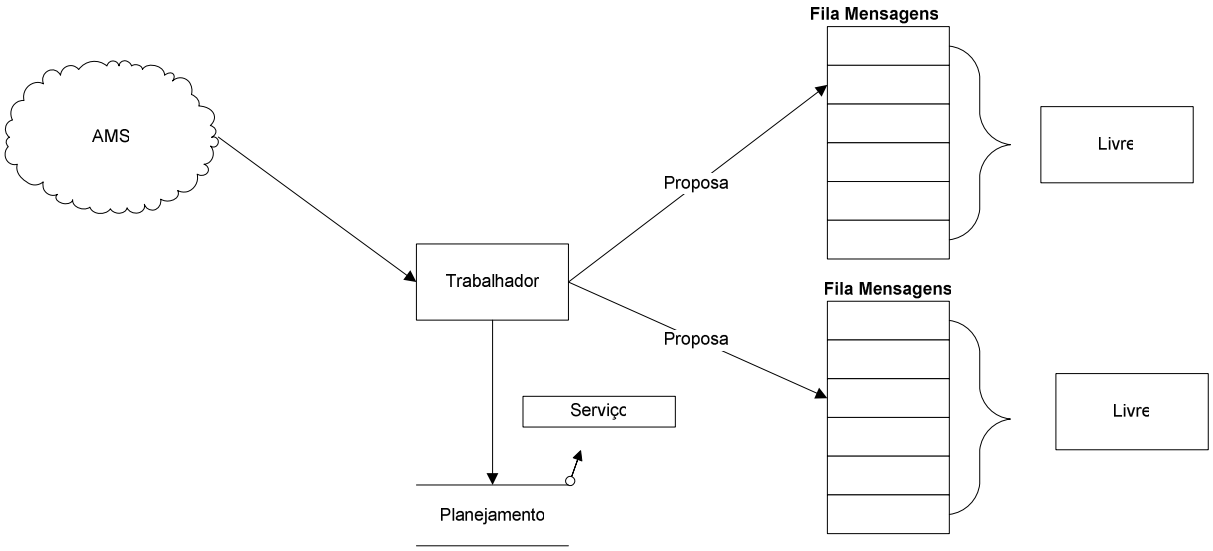


Figura 4-3 - O aspecto SMA da proposta com o Agente Broker e Papéis

Porém, antes de se explicar como isso ocorre convém apresentar a interação entre agentes na sociedade, uma vez que daí advém a proposta de estudo e aplicação de um padrão de **comunicação** para garantir a interoperabilidade entre os agentes e prover uma maneira de mantê-la. Por essa razão foram estudados métodos de comunicação como os citados em Ito (ITO, 2006): Hunhs & Stephens, FIPA, JADE (BELLIFEMINE, 2003) e Retsina. A proposta organizacional adotada foi o FIPA Contract-Net para possibilitar formação de coalisões: “chamamos de coalisão esta noção de estrutura organizacional que é formada dinamicamente pelos agentes, num procedimento bottom-up” (ITO, 1999, p.13).

Assim em intervalos regulares os agentes executam uma verificação de parceiros através da implementação da classe `TickedService`, que se responsabiliza por encontrar um parceiro que ofereça o serviço buscado. Uma vez que esse comportamento é comum a qualquer agente do *framework*, todos os agentes atuam em busca de formação de coalisões dinâmicas, conforme o quadro 4-1:

```
 if(ticket.sellerAgents != null  && ticket.sellerAgents.length > 0)
 {
      myAgent.inform(this,"Encontrei    um    agente    desocupado:    "    +
ticket.sellerAgents[0]);
```

Quadro 4-1 - Formação de coalisão entre os agentes do *framework*

Quando esse esforço por parte de um agente para encontrar um parceiro não obtém sucesso, o agente então solicita ao *broker* para que sejam criados novos agentes na sociedade, pois uma requisição se encontra pendente.

Isso ocorre através da criação de agentes livres pelo *Broker* que possui uma estrutura especial cujo planejamento é encontrado somente na classe BrokerRoler, pois ele é o único que pode ter acesso aos mecanismos da plataforma para poder criar novos agentes.

Independente de quais papéis precisam ser executados, e qual a demanda por eles o *broker* é responsável por garantir que toda a requisição por um serviço é atendida através da criação de novos agentes para posteriormente se organizarem na sociedade e assumir um papel.

Uma vez que o parceiro enfim é encontrado, o agente então envia automaticamente uma proposta contendo o serviço que desejaria que ele prestasse para que ele possa escolher, (ou não) prestar o serviço e ou assumir um novo papel para atendê-lo, como ilustrado no quadro 4-2.

```
      ACLMessage order = new ACLMessage(ACLMessage.PROPOSE);

      order.addReceiver(ticket.sellerAgents[0]);
                                  order.setContent(super.question);
                                  order.setConversationId(question);
                                  order.setReplyWith("order"             +
System.currentTimeMillis());
                                  myAgent.send(order);
                                  this.block(20000);  //aguardar  o  outro
agente receber a proposta
```

Quadro 4-2 - Envio de proposta de trabalho entre os agentes do *framework*

Portanto, toda vez que um agente é criado na sociedade ele não possui um papel definido, estando à disposição para assumir um papel da forma explicada no capítulo anterior. Como todo agente criado na arquitetura é criado sem um papel específico ele não é considerado um trabalhador na sociedade e possui uma lógica de raciocínio limitada a essa compreensão de ajudar ou não demais agentes e assumir papéis.

Assim, a classe `ServerRole` baseada em papéis limita-se a conhecer os diversos planos existentes a serem alocados a diferentes agentes de acordo com a necessidade da sociedade. Esta classe fica visível a cada agente da sociedade, tendo ele um trabalho ou não, e tem as características do quadro 4-3:

ServerRole	
+createRoles() +action() +ServerRole(in a : Agent)	1. ServerRole é um comportamento, e é executado a partir do seu método *action* herdado de *behaviour* no JADE 2.Quando o action é executado o ServerRole atualiza sua base de papéis

Quadro 4-3 - Detalhamento do servidor de papéis (comportamento ServerRole)

O comprometimento com o papel é realizado especificamente pela classe `ServerRole, que` resume um mecanismo a partir do qual podem ser criados diversos agentes com diversos papéis com a mesma facilidade. A escolha por papéis pelos agentes é realizada como ilustrado na figura 4-4:

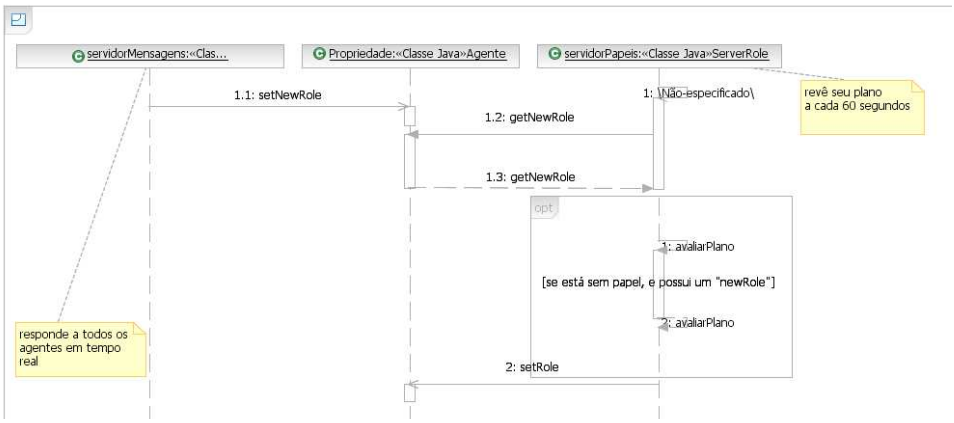


Figura 4-4 - Escolha pela troca de papéis dos agentes propostos

O `ServerRole` têm seu funcionamento viabilizado pelos atributos `newService` e `newRole` (visto na figura 4-4) que representam as decisões de um agente em ter aceito um novo serviço ou um novo papel. As decisões são tomadas em conjunto com a um mecanimos análogo a uma caixa de entrada de e-mail (`ServerIncomming`) que recebe diversas solicitações de serviços e baseando-se em um critério próprio define um `newService` para o agente.

4.3. CRIAÇÃO E EXECUÇÃO DE PLANOS

Na **racionalização** está o ponto central da arquitetura SMA proposta, pois é o mecanismo de atuação dos agentes do modelo proposto. A proposta realizada foi concentrar a racionalização acerca de conhecimentos específicos em papéis que são assumidos por agentes "livres", ou seja, que apesar de serem agentes completos e ativos na sociedade são benevolentes e possuem na sua racionalização a capacidade de poder assumir algum papel específico e um novo trabalho, sem se preocupar o quanto assumir esse papel vai agradá-lo ou não. Tal característica do *framework* é apoiada pela existência de um agente especial denominado **broker** cuja responsabilidade é monitorar o desempenho da organização como um todo e disponibilizar novos agentes.

No que diz respeito à racionalização toda ela é baseada no *framework* em um aspecto conversacional. Agentes conversacionais são compostos por algumas habilidades que lhe garantem o aspecto conversacional: *compreensão da linguagem natural, análise sintática de frases, co-referência, representação em forma lógica, conhecimento acerca dos fatos, crenças a respeito do mundo (dos outros e de si mesmo também), racionalização sobre intenções e planos* (Mackassy, 1996). Apesar de serem grandiosos sistemas que apresentem tais características, este trabalho tornar-se-ia demasiadamente complexo e extenso se objetivasse

também compreender todos os critérios que tangem o processamento de linguagem natural, ou PLN. Por esta razão, apenas os quatro últimos tópicos serão tratados, os relativos ao planejamento dos agentes: *representação em forma lógica*, *conhecimento acerca dos fatos* e *racionalização sobre intenções e planos* e aquele relativo ao ambiente do agente: *crenças a respeito do mundo (dos outros e de si mesmo também).*

Existem propostas como Cabri (2004) que explica uma abordagem de papéis evolutivos baseado em "requisitos", "capacidades" e "comportamento" que são analisados pelos próprios requisitos e capacidades do agente, e usados “para expandir as capacidades dos agentes” (figura 4-5).

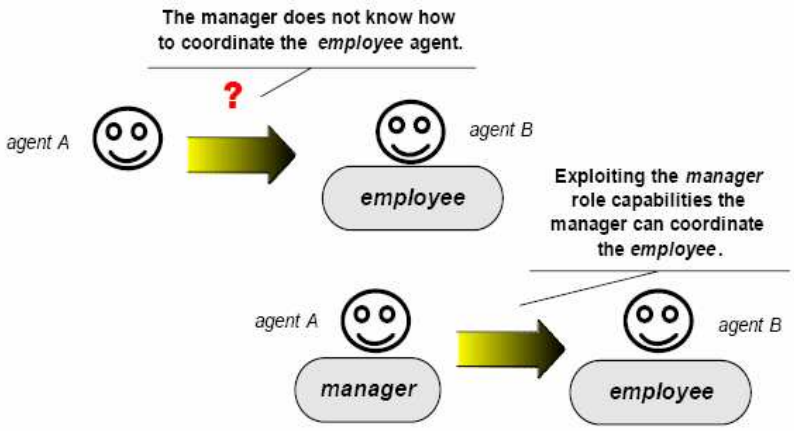


Figura 4-5 - Capacidades dos agentes se alteram após assumir um papel (CABRI, 2004)

À medida que aumenta as capacidades dos agentes, já que aumenta as crenças (beliefs) dos agentes e que passaria a permiti-los julgar um *goal*, aumenta também a lógica para implementar um agente. Mas uma das premissas do *framework* é que a introdução dos conceitos para os agentes deve ser feita de maneira simples, e o aspecto conversacional não exige a utilização de *beliefs* formalmente representados na arquitetura excluindo o *belief* do processo de racionalização com o objetivo de reduzir toda complexidade adicional, sob a pena de não permitir que os agentes evoluam seus *beliefs* (crenças), voltando-se diretamente aos *goals* e às ações.

A exclusão da representação do *belief* do processo de racionalização no *framework* foi tomada também com base na argumentação de Wooldridge (WOOLDRIDGE, 2002) de que a deliberação com base em seus *belief* deve ser comedida, pois os recursos computacionais são limitados e o agente não pode deliberar eternamente (WOOLDRIDGE, 2002, p. 68): “As intenções não podem persistir por muito tempo” e principalmente a colocação de Martha Pollack de que um agente deve acima de tudo agir, e não só deliberar: “nós queremos construir atores inteligentes, e não pensadores inteligentes” (POLLACK, 1991, p. 2, tradução nossa).

Assim passa-se a definir os *goals* como ponto de partida para o raciocínio dos agentes propostos. Ou seja, os agentes são incapazes de racionalizar sobre seus *beliefs (crenças)* e somente sobre seus *goals* chegando a *intentions (ações)*.

Esta técnica de exclusão dos *beliefs* permite racionalização com conhecimento parcial sem possuir um conhecimento completo do ambiente (ou seja, otimizada), pois como se pode evidenciar em Nute (1994) a lógica de raciocínio pode até ser realizada diretamente sobre as *intentions* (ações).

Sabe-se que o “*Pratical Reasoning Agents*” (WOOLDRIDGE, 2002) trata que a *deliberação* ocorre sobre os *beliefs* e baseia-se nos *desires* (*como* gostaria que o ambiente ficasse) que implicam em *goals*. Uma vez definidos os *goals* o Agente racionaliza sobre os eles com o objetivo de gerar seu plano. Sendo assim, como existe um consenso em AI de que racionalizar a respeito do planejamento resulta em um plano (WOOLDRIDGE, 2002) e “a saída de um algoritmo de planejamento é um ‘plano’” (WOOLDRIDGE, 2002, p. 66), passa-se a entender também *goals* como planejamento.

Portanto por não possuir *beliefs* no *framework* proposto, não existem mecanismos para julgar um *goal* e descartá-lo no curso de um planejamento. Logo, se representa apenas os

objetivos (*goals)* e ações (*intentions)* havendo para cada agente um ou mais objetivos (*goals)* e para cada objetivo (*goal)* uma porção de ações (*intentions).*

Com relação às *intentions* de acordo com Bratman (1987, pp. 8, 29, tradução nossa) “planos são intenções detalhadas”. Ou seja, apesar de não existir uma distinção clara entre intenções e planos Wooldridge (2000, p. 45, tradução nossa) ressalta que “os planos são seqüências lineares de ações”. Sendo assim, no *framework* proposto o mecanismo BDI apresentado tratará os planos como seqüência lineares de ação.

Resumindo a abordagem do *framework* proposto: para cada *papel* de raciocínio de agentes existente, haverá um ou mais *planejamento* e para cada *planejamento* um conjunto de *planos*. A figura 4-6 demonstra em detalhes o funcionamento dessa idéia na arquitetura proposta: uma vez que um agente assume um papel na sociedade, ele só irá revê-lo após terminar todo o seu plano. Além disso, de acordo com Bratman, Israel e Pollack (1988), o fato de um agente computar para qual caminho que vai satisfazer mais seus objetivos e pelo qual deve seguir antes de executar o plano pode fazer com que o ambiente mude a respeito de deliberações que ele já fez. Por isso introduzimos o “*choice*” no planejamento. O choice é capaz de redirecionar a execução para qualquer lugar do plano dependendo do andamento da interação com o usuário.

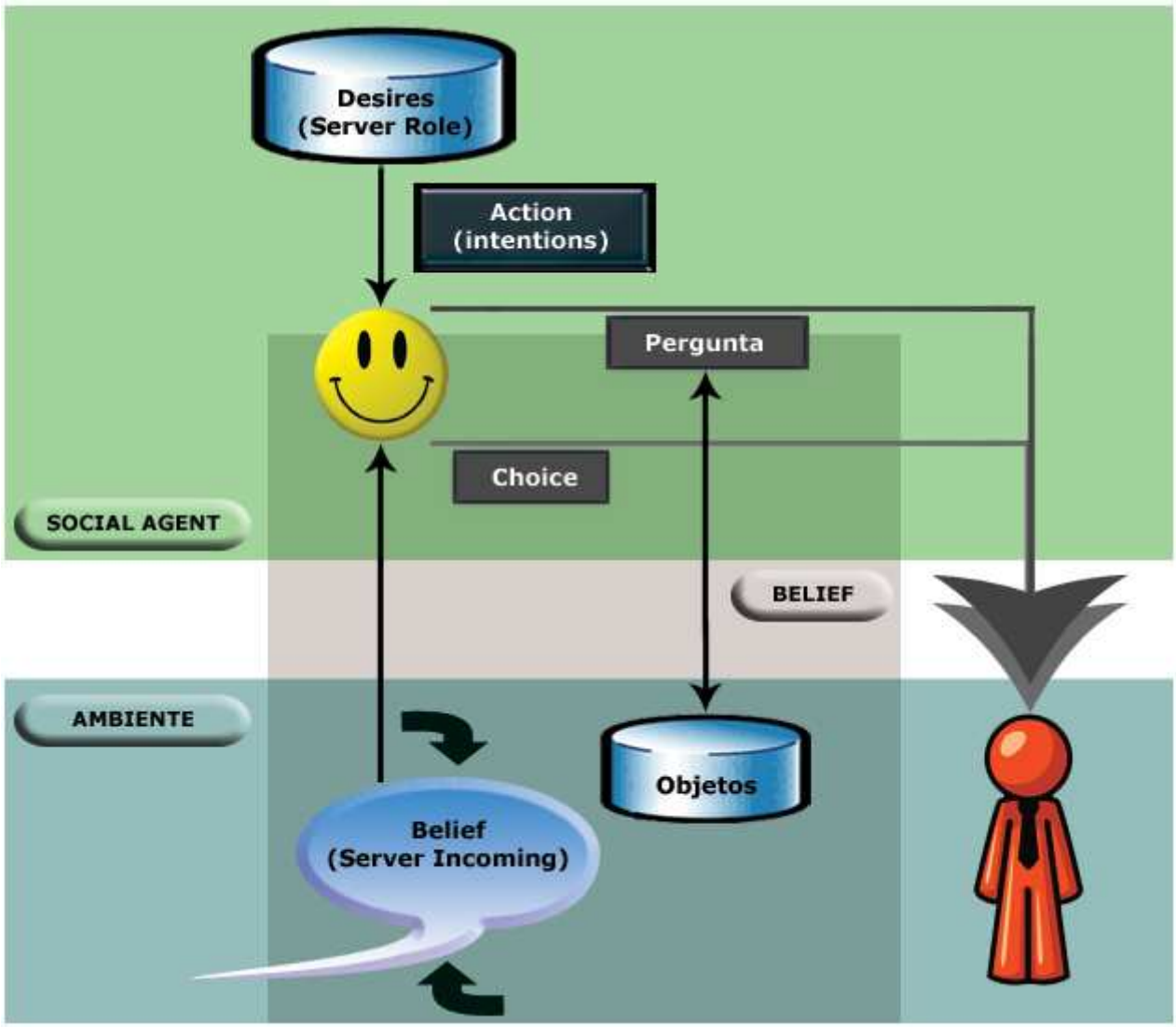


Figura 4-6 - Arquitetura completa do *framework*, ações, ambiente, crenças e o aspecto que torna cada agente inserido tanto em um universo SMA com em um universo RDP

Antes de entrar nos detalhes da execução de planos detalha-se como foi desenvolvida a implementação da racionalização. Em primeiro lugar, o usuário deve se preocupar basicamente com duas questões: quais são as atividades (ou ações, ou intentions como já foram discutidas no capítulo 3) que um determinado papel pode executar e quais os planos válidos (*goals* como também já discutido anteriormente) para execução dessas ações. Primeiro tratamos da criação das ações no *framework* para depois tratarmos de seu encadeamento lógico.

A proposta de Amant e Zettlemoyer (2000), por exemplo, relata como um agente pode estar inserido em um ambiente unicamente gráfico, e responsável pelo gerenciamento da interface com o usuário ou outros agentes. Apesar de não ser esse o enfoque desta proposta, é importante ressaltar a pesquisa e desenvolvimento de agentes voltados em interfaces e como essas interfaces podem sofrer alterações e serem evoluídas a partir de agentes inteligentes. Assim, não é de se estranhar que parte dos objetivos do *framework* estejam intimamente ligados às interações com usuário. Seguindo a definição de ambiente de Sistemas MltiAgentes dada por Wooldridge (2002) de que os agentes estão inseridos em um ambiente com o qual podem interagir, retirando e avaliando informações e atuando nesse mesmo ambiente, pode-se definir que o usuário compõe o ambiente do *framework.*

Na proposta, o usuário interage com o sistema via interface gráfica, e utilizando sua linguagem natural. Macskassy (1996) fala da criação de um agente capaz de receber entradas de dados do usuário a partir de um formato texto ou qualquer outro formato proposto, e em seguida decidir ao que se refere essa entrada do usuário baseando-se em algum algoritmo. Assim ele se torna capaz de perceber se faz algum sentido para si e para a pessoa que está falando, e emitir uma resposta ao usuário. (MASSARO, 2001).

Toda interação com o usuário, é a princípio um questionamento e aguarda uma resposta daquele. Portanto, *a priori*, as ações do *framework* são perguntas textuais.

Para criação de ações o lado cliente (ou seja, o agente) apenas precisa utilizar um *hashtable*. Esse cadastramento é feito em uma base de dados que é mapeada para o sistema através do Hibernate, e deve obedecer ao quadro 4-4 na criação de novas ações:

```
Texto   x 99
  \/    || ||
ação    \/ \/
        1 2  (qualificadores)
```

Quadro 4-4– Representação de uma ação no *framework*

No quadro 4-4 o item “texto” representa justamente a ação, pergunta, a ser emitida ao usuário. Obviamente, somente a interação com o usuário em forma de pergunta não seria suficiente para construir um SMA e por isso surgiram o 1º e 2º qualificadores de ação.

O 1º qualificador de ação modifica o sentido original da pergunta conferindo-lhe recursos adicionais, conforme tabela 4-1:

Símbolo	“?”	“.”	“01”	“$”
Significado	Uma pergunta emitida ao usuário que permite mudar o plano original	Ação final. Indica que o plano chegou ao fim e o objetivo atingido	Identifica o usuário. Será discutido no capítulo sobre ambiente	Proposta de trabalho para algum agente

Tabela 4-1 - Lista de qualificadores de ação do *framework*

Mas uma vez criadas as ações e definidos seus qualificadores, quando criado um novo papel o mesmo deve possuir um agrupamento de planejamentos para atender a *goals* que lhe sejam convenientes. Dessa forma, um *goal* terá um conjunto de ações numeradas que serão executadas na seqüência crescente determinada pelo usuário. Assim, o 2º qualificador diz respeito a essa seqüência e quando essas ações serão executadas em um plano. E como nem sempre as seqüências de ações são lineares, quando o projetista cria uma ação contendo o 1º qualificador igual a “?” é permitido ao usuário informar como resposta a essa ação o código da próxima ação a ser executada, criando assim planos dinâmicos que se alteram de acordo com a interação com o usuário. No quadro 4-5 demonstramos um exemplo simples desta estrutura adaptada do BDI:

```
      perguntas.put("01 ObjetoDoAmbiente", "Descrição da identificação");
      perguntas.put("02 $", "ServiçoProcurado");
      Planning planejamentoAtendimento = new Planning(myAgent,perguntas);
      planejamentoAtendimento.setBehaviourName("ServiçoPrestado");
      roles.get("MeuPapel").addSubBehaviour(planejamentoAtendimento);
```

Quadro 4-5 – Mecanismo de racionalização dos agentes propostos

Os agentes agindo de forma conversacional e colaborativa conseguem relacionar-se e gerar uma resposta ao usuário. O exemplo passado no quadro 4-5 ilustra um agente sob um papel genérico. Abaixo exemplificamos outros dois papéis reais, no quadro 4-6 o papel de agente "diabetologista" que oferece os serviços de "medir-glicemia", e um segundo papel "enfermeiro" no quadro 4-7 que necessita deste tipo de serviço:

```
      TreeMap<String, String> perguntas = new TreeMap<String, String>();

      perguntas.put("01 Glicemia", "Em que dia o paciente mediu a sua
glicemia?");
      perguntas.put("11 ?", "O valor de sua glicemia é > 100 ? (digite 20 para
sim, 12 para não.)");
      perguntas.put("12 .", "Sua glicemia está normal.");
      perguntas.put("20 .", "Procure um hospital");

      Planning planejamentoMedirGlicemia= new Planning(myAgent,perguntas);
      planejamentoMedirGlicemia.setBehaviourName("medir-glicemia");
     roles.get("diabetologista").addSubBehaviour(planejamentoMedirGlicemia);
```

Quadro 4-6 – Um papel de agente auto-suficiente

```
      TreeMap<String, String> perguntas = new TreeMap<String, String>();

      perguntas.put("00 ?", "O que o paciente deseja fazer? (01-Registrar-se/"+
      "/10-Medir a Glicemia/0-Finalizar:");

      perguntas.put("01 Paciente", "Digite o nome, peso e idade separados por
';'");
      perguntas.put("02 Endereco", "Informe seu endereço");
      perguntas.put("03 Telefone", "Informe seu telefone");
      perguntas.put("04 ?", "Deseja se consultar com um diabetologista? Digite
10!");



      perguntas.put("10 $", "medir-glicemia");

      Planning planejamentoTriagem = new Planning(myAgent,perguntas);
      planejamentoTriagem.setBehaviourName("triagem");

      roles.put("enfermeiro", new Goals());
      roles.get("enfermeiro").addSubBehaviour(planejamentoTriagem);
```

Quadro 4-7 – Um papel de agente que precisa de parceiro

No que diz respeito à racionalização, o mecanismo quando em execução se comporta como ilustrado no diagrama de atividades da figura 4-7 a seguir:

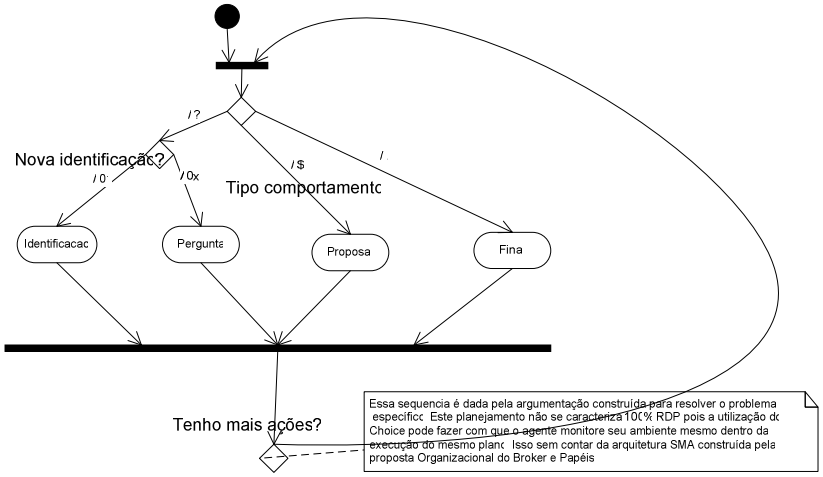


Figura 4-7 – Detalhamento do aspecto RDP: Segmento da arquitetura que possibilita a inserção de um simulacro RDP no sistema SMA proposto.

4.4. ATUAÇÃO DO AGENTE NO AMBIENTE

Segundo (WOOLDRIDGE, 2002) os agentes estão inseridos em um ambiente com o qual podem interagir, retirando e avaliando informações e atuando nesse mesmo ambiente. No *framework* tem-se a necessidade de manter o estado conversacional do ambiente com os agentes.

Existem diversos modelos de interação com o ambiente, sejam a respeito da atuação no ambiente ou ainda para percepção de alteração do ambiente. (WEYNS, 2005)

Um modelo de percepção de alteração do ambiente propositalmente ainda não tratado neste trabalho é proposto por Weyns, Steegmans e Holvoet (WEYNS, 2004): um modelo genérico de percepção ativa (***model for active perception)*** que foca em agentes situados em um ambiente virtual. Percepção ativa habilita um agente a direcionar sua percepção ao aspecto mais relevante do ambiente de acordo com sua tarefa atual, facilitando a manter o

processamento de dados percebidos sob controle. Dentre as características relevantes ao modelo, algumas se tornaram necessárias ao pensarmos em um sistema SMA que poderia ser aplicado a qualquer domínio e destacamos:

- É independente de domínio específico;
- Abstrações reutilizáveis para percepção ativa;
- suporte a modelo específico de percepção.

Algumas abordagens existem para análise e projeto de sistemas orientados a Agentes, tais como Gaia, Message ou AUML. No entanto, apesar da existência dos conceitos de interação com o ambiente pouco tem sido feito com relação à análise, modelagem e projeto do ambiente com o qual os agentes se comunicam (SCHWAMBACH, 2004).

Complementa ainda Weyns et. al (WEYNS, 2005):

> Explorar alternativas arquiteturais para relacionar os agentes aos seus ambientes oferece um amplo escopo para uma nova disciplina na engenharia de software. Assim, enquanto pesquisas em relação ao ambiente não forem consideradas importantes para o desenvolvimento de sistemas orientados a agentes, o potencial será do ambiente nesses sistemas não revelados. (WEYNS, 2005, tradução nossa).

Schwambach, Pezzin, Falbo (SCHWAMBACH, 2004), notando esta lacuna, propõem em seu trabalho a utilização da tecnologia de orientação a objetos para implementar o ambiente dos agentes, a OplA (*Object plus Agent*), de forma a suprir a falta gerada pela ausência de uma metodologia unificada de agentes e objetos. Em termos de modelos, tal metodologia não introduziu muitas mudanças, agindo principalmente como uma reunião de modelos já existentes e orientação de como utilizá-los (nível de processo), mas segundo os próprios criadores da metodologia ela inova a medida que introduz o conceito de se modelar o ambiente dos agentes com um padrão já afirmado e com diversas soluções existentes, que é o modelo de análise e projeto Orientado a Objetos.

Assim sendo, propõe-se que o ambiente de execução deve utilizar os conceitos de Orientação a Agentes, ou SMA, conforme explicado anteriormente. Os Agentes são

responsáveis pela prestação do serviço e assim não faz sentido colocar as informações dentro do Agente, ficando estas nos objetos para não sobrecarregar o Agente, que serve também para possíveis futuras integrações com sistemas “legados”.

A partir do planejamento, todas as interações que o agente tem com o ambiente são mapeadas automaticamente para uma arquitetura orientada a objetos. A classe que realiza esse interfaceamento é a classe *pergunta.*

Tendo em vista o que foi discutido anteriormente, o ambiente de um agente é qualquer elemento externo ao agente que interage com ele direta ou indiretamente (WOOLDRIDGE, 2000). Porém, para nós serão consideradas as relações intra-agentes de uma maneira separada, que como explicado anteriormente, será entendido como o processo de comunicação e organização da sociedade. Portanto a relação Agente x Ambiente ficaria por conta da interação proveniente de seu aspecto conversacional, ou seja, com os usuários. No caso do SMA/GRPC o **ambiente que não se refere aos demais agentes** é representado por Objetos e se relaciona com a lógica racional dos agentes explicada através de uma interface única que automatiza o armazenamento de qualquer informação proveniente do contato com usuário, com outros agentes ou qualquer outro envolvido. Os Agentes são responsáveis pela prestação do serviço, e assim não faria sentido colocar as informações dentro da estrutura do Agente: as informações ficam nos objetos para não sobrecarregar o Agente (ITO, 2006).

Para tanto, foi necessária a criação de alguns mecanismos de controle e implementação que proporcionassem este tipo de funcionalidade. Toda vez que perguntas são emitidas para o usuário, as respostas passam a fazer parte do ambiente do agente, e tudo isso é criado dinamicamente pelas classes `Pergunta`, `Identificação` e `Escolha` sendo utilizadas pela classe `Planning`. A classe `Pergunta` armazena respostas obtidas do mundo exterior ao agente e é uma classe de comportamento de interação com o usuário. O agente informa para seu plano qual é a entidade do ambiente relacionada com a pergunta, e o plano cria

automaticamente essa entidade e armazena no ambiente do agente. A classe `Pergunta` é responsável pelo relacionamento entre o ambiente e o processo de racionalização dos agentes. Afinal, é ela quem conhece o objeto.

Já a Classe Identificação é um tipo especial de pergunta. O Agente a identifica pelo status 01 na pergunta, e o plano sabe que todas as perguntas que seguem uma identificação passam a integrar a coleção de respostas identificadas por ela. A identificação fica ativa e encabeça todas as respostas que se seguem até que uma nova chave seja encontrada.

5. Simulações de casos específicos

5.1. ACOMPANHAMENTO DE PACIENTES CRÔNICOS COM SISTEMA MULTI-AGENTES

Diversos segmentos da sociedade têm sido influenciados na forma de trabalho com o advento da tecnologia da informação (CARDOSO, 1999) e até mesmo a saúde como afirma a OMS (OMS, 2006), criando oportunidade de aplicação para os sistemas em informática médica que vêm ganhando grande importância e gerando demanda (CELLER; LOVELL; BASILAKIS, 2003; CHAU; HU, 2004; LASTRES; ALBAGLI, 1999; MEA, 2001).

Uma abordagem indicada para sistemas de *telemedicina* são sistemas orientados a agentes, segundo Mea (2001) e Ito (2006), devido sua capacidade de resolver problemas complexos.

Mas por outro lado conforme Jennings (1999) tais sistemas possuem uma complexidade arquitetural intrínseca, e ao término dessa proposta o resultado esperado é viabilizar tecnicamente a expansão do modelo GRPC com o sistema piloto TeleDM, e com isso imagina-se que teremos um sistema de solução de Telemedicina capaz de resolver problemas complexos tanto com a introdução de novos agentes em sua sociedade quanto na facilidade de manutenção de sua base de conhecimento quando novos requisitos assim o requererem.

Além do sistema TeleDM, a arquitetura CENINT sobre a qual o sistema está implementado poderá também ser utilizada para outros domínios dentro do mesmo modelo GRPC ou até mesmo para outros modelos de softwares aproveitando-se das melhorias tanto na arquitetura como no ambiente de comunicação proporcionados pela adaptação do sistema TeleDM.

Atualmente, muito se tem discutido a respeito das estratégias de intervenção para o controle da doença que se adaptem melhor ao paciente uma vez que, segundo Ito (2006, p.19):

> o modelo tradicional de acompanhamento do paciente crônico através de consultas em clínicas, ambulatórios e postos de saúde, pode ser otimizado para melhor atender à comunidade através da maximização da utilização de recursos e de médicos e minimização das despesas em clínicas e hospitais.

Por outro lado, as mais variadas áreas do conhecimento têm se beneficiado dos recursos que as novas tecnologias podem lhe oferecer (CASTELLS, 2003; SIMON, 1996), e no que se refere à área de saúde, organizações como a OMS, afirmam que a tecnologia da informação tem papel-chave no aumento da qualidade do sistema de saúde (OMS, 2006) e assim estão interessadas em utilizar essas novas tecnologias para apoiar, expandir ou aumentar a qualidade dos cuidados e serviços aos pacientes (CHAU; HU, 2004; LASTRES; ALBAGLI, 1999).

No contexto da informática médica, segundo Mea (MEA, 2001)

> os sistemas de telemedicina são compostos por complexos sub-sistemas com componentes heterogêneos gerenciando dados e recursos compartilhados e eventualmente necessitam de integração com sistemas legados.

Um dos problemas na atualidade com relação à prestação de serviços de saúde está relacionado à falta de percepção de qualidade por parte dos pacientes (PORTER, 2007). Uma possível causa é apontada em Porter (2007) ao considerar que muitas operadoras no ramo tendem a achar que a redução de custos nos casos gerais pode criar um sistema de saúde de baixo investimento e bom valor agregado. Mas tal afirmativa é refutada quando se analisa os dados de retorno de investimento sobre as aplicações em saúde, que são de aproximadamente U$ 2,40 a U$3,00 para cada dólar investido no setor.

Observando ainda que outras áreas têm se beneficiado dos recursos que as novas tecnologias podem oferecer (CASTELLS, 2003), o investimento em inovação no setor é ainda baixo comparado com outros setores da economia, deixando sub-representadas as pesquisas

em estrutura, organização, processos e mensuração da prestação de serviços de saúde, explicando até mesmo porque erros na área da saúde são maiores se comparados a outros setores (PORTER, 2007).

Além disso, organizações como a OMS afirmam que a tecnologia da informação tem papel-chave neste aumento da qualidade do sistema de saúde (OMS, 2006) e estão interessadas em utilizá-la para apoiar, expandir ou aumentar a qualidade dos cuidados e serviços aos pacientes, difundindo conceitos como (Branko, 2003):

- Telemedicina – serviço de saúde no qual o médico examina o paciente através do uso de uma tecnologia de telecomunicação;
- E-saúde - serviço baseado na tecnologia da informação e que incorpora algumas atividades relacionadas à saúde, como: educação ao paciente e seu cuidador, serviços administrativos e cuidados com o paciente;
- *Home care* - serviço de atendimento ao paciente em sua própria casa,
- ou *home telecare,* utilizando os recursos da tecnologia da informação.

Uma das áreas de investimento fomentada por tais organizações relacionadas com a saúde se referem às doenças crônicas, uma vez que com o crescente aumento da incidência de doenças crônicas, como a hipertensão arterial e o *diabetes mellitus,* a OMS vem se preocupando com as questões relacionadas ao acompanhamento e ao monitoramento de pacientes crônicos, identificando inclusive seus impactos econômicos e sociais (ITO, 2006; OMS, 2006). A necessidade de inovação, neste setor, é latente quando se ressalta que o modelo tradicional de acompanhamento do paciente crônico, por meio de consultas em clínicas, ambulatórios e postos de saúde, pode ser otimizado para melhor atender a comunidade, maximizando a utilização de recursos e de médicos e minimizando as despesas em clínicas e hospitais (ITO, 2006). Isso também faz com que seja percebida a melhora nas condições de saúde durante todo o ciclo do tratamento (PORTER, 2007).

Desta forma, a qualidade da prestação de serviços passa a ser um valor sensível aos pacientes, sendo muito bem vindas abordagens inovadoras que, apesar de mais caras, aumentam a expectativa de vida do paciente ou fazem com que o mesmo consiga conviver de forma mais harmoniosa com a doença. (PORTER, 2007).

Em se tratando de propostas já realizadas com o objetivo de melhorar a qualidade de vida de pacientes crônicos através de sistemas de telemedicina, o modelo GRPC (ITO, 2006) (Gestão do Relacionamento com o Paciente Crônico) foi criado com o intuito de aumentar a qualidade de vida de pacientes crônicos e outros envolvidos, como médicos e familiares, garantindo um acompanhamento efetivo e adequado a todas as camadas populacionais. O modelo propõe a criação de diversas centrais de relacionamento com o paciente crônico, inspirando-se no modelo CRM (*Customer Relationship Management*).

O modelo GRPC (Gestão de Relacionamento do Paciente Crônico) (ITO, 2006) adapta os conceitos do CRM (*Customer Relationship Management*) no acompanhamento e monitoramento dos pacientes, propondo aumentar a aderência dos mesmos ao seu tratamento. Os componentes do modelo são uma adaptação das estratégias do CRM e estão ilustradas na figura 5-1.

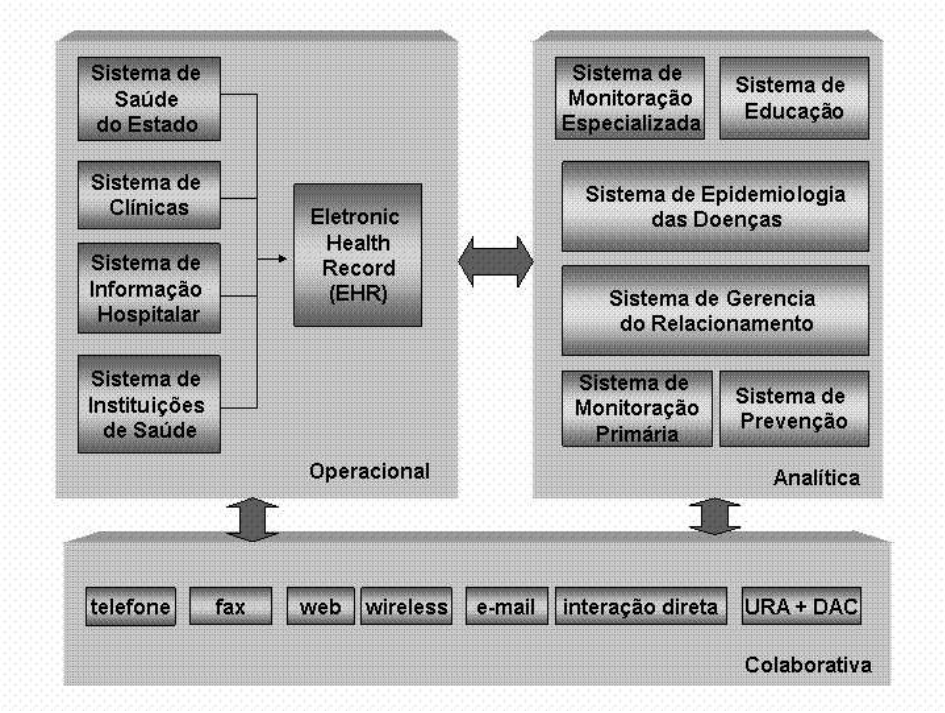


Figura 5-1 – Componentes adaptados do CRM para o modelo GRPC (ITO, 2006)

O *componente operacional* é responsável pela consolidação dos dados coletados nos diversos níveis de interação com o paciente, enquanto que o *componente colaborativo* engloba os canais de comunicação em que é possível uma interação direta. Finalmente, o *componente analítico* realiza a análise dos dados através de funções nos níveis estratégicos, táticos e de inteligência do atendimento personalizado ao paciente (ITO, 2006).

De forma a operacionalizar o modelo GRPC, Ito (ITO, 2006) propõem uma Central de Relacionamento com Pacientes Crônicos (CRC) que é composta pelos três componentes citados anteriormente. De forma a viabilizar o projeto foi proposto uma Central de Monitoração de Paciente Diabético (CMD) com a definição de processos para implementação do modelo GRPC para diabéticos e apresentados na Figura 5-2.

Dentre estes processos, Ito (2006) elegeu o processo “Monitorar a Glicemia” para iniciar o desenvolvimento do sistema de informação que fornece o suporte a CMD. Para o desenvolvimento do sistema de informação (TeleDM) do modelo GRPC Ito (ITO, 2006) ressaltada a necessidade do uso da Tecnologia de Agentes para a sua implementação uma vez

que considera complexo o domínio de negócio do sistema, além da necessidade de um sistema de apoio à decisão no processo “Monitorar a Glicemia”.

Segundo Janca (1995) a Tecnologia de Agentes surgiu da evolução da IAD que surgiu a partir da solução de problemas. Apesar da simplicidade do conceito, não existe um consenso entre os autores sobre a definição de agente. Para a arquitetura proposta neste trabalho utilizaremos a seguinte definição de Wooldridge (WOOLDRIDGE, 2002, p. 15): “Um agente é um sistema computacional que está situado em algum *ambiente* e que é capaz de *ações autônomas* neste ambiente de modo a alcançar os seus objetivos”.

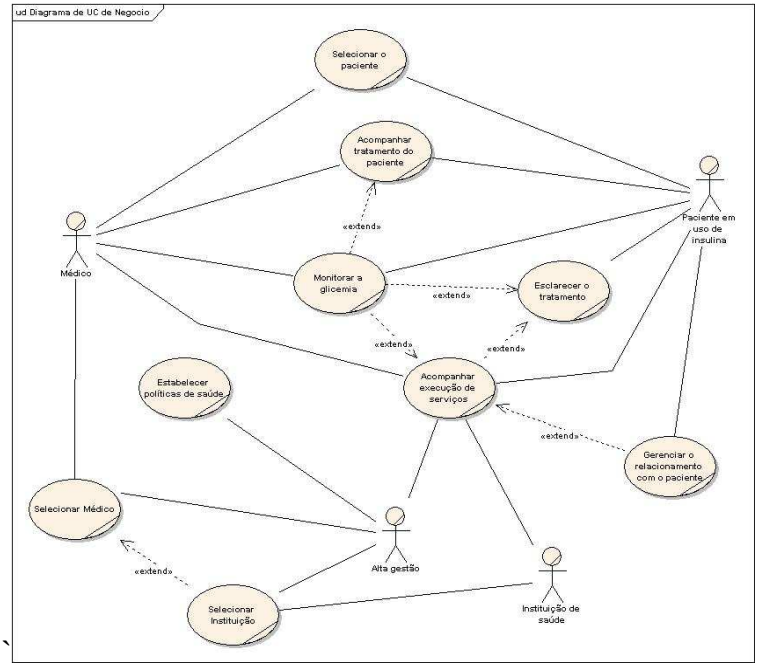


Figura 5-2 – Central de relacionamento com pacientes crônicos:funcionalidades(ITO, 2006)

Ito (ITO, 2006) definiu objetivos e planos para cada um desses objetivos e os atribuiu aos agentes. Tais planos detalhados na publicação de Ito (ITO, 2006) foram largamente utilizados nesse trabalho como prova de conceito de suporte à CMD do modelo GRPC, no entanto a definição de responsabilidades entre agentes, comunicação, ativação de planos, racionalização e outras características dos agentes são específicas deste trabalho.

5.2. SIMULAÇÕES PARA VALIDAR O SISTEMA

Com a arquitetura definida os agentes dos sistemas foram criados: o diabetologista, o atendente e o enfermeiro. O agente atendente é responsável por atender um indivíduo que entra em contato com a central, definir o tipo de atendimento e buscar pelo agente que realiza o serviço solicitado. O agente enfermeiro é responsável por coletar o valor da glicemia, requisitar o serviço de análise do resultado a outro agente e informar o resultado e a orientação fornecida. O agente diabetologista, com as informações fornecidas pelo agente enfermeiro, analisa-as e fornece as orientações a serem dadas ao paciente. Se necessário, interage diretamente com o usuário do sistema para coletar dados adicionais. Sendo o ambiente destes agentes composta por classes da orientação a objetos, as seguintes classes foram implementadas:

o Contato – Registra os contatos feitos por alguém junto à central;

o Endereço – Representa o endereço completo da pessoa. Uma pessoa pode encontrar-se em mais de um endereço; um médico, por exemplo, que pode ter como referências a clínica e o hospital em que atende;

o Lista Paciente – Contém a lista de pacientes que são atendidos pela central;

o Médico – Controla as informações relacionadas ao médico do paciente;

o Glicemia – Registra os dados de monitoração da glicemia do paciente;

o Paciente – Controla as informações do paciente;

o Telefone – Representa os números de telefones para contato

Para as simulações foram utilizados os *goals* necessários para implementação da funcionalidade Monitorar Níveis Glicêmicos da CMD do modelo GRPC, figura 5-3.

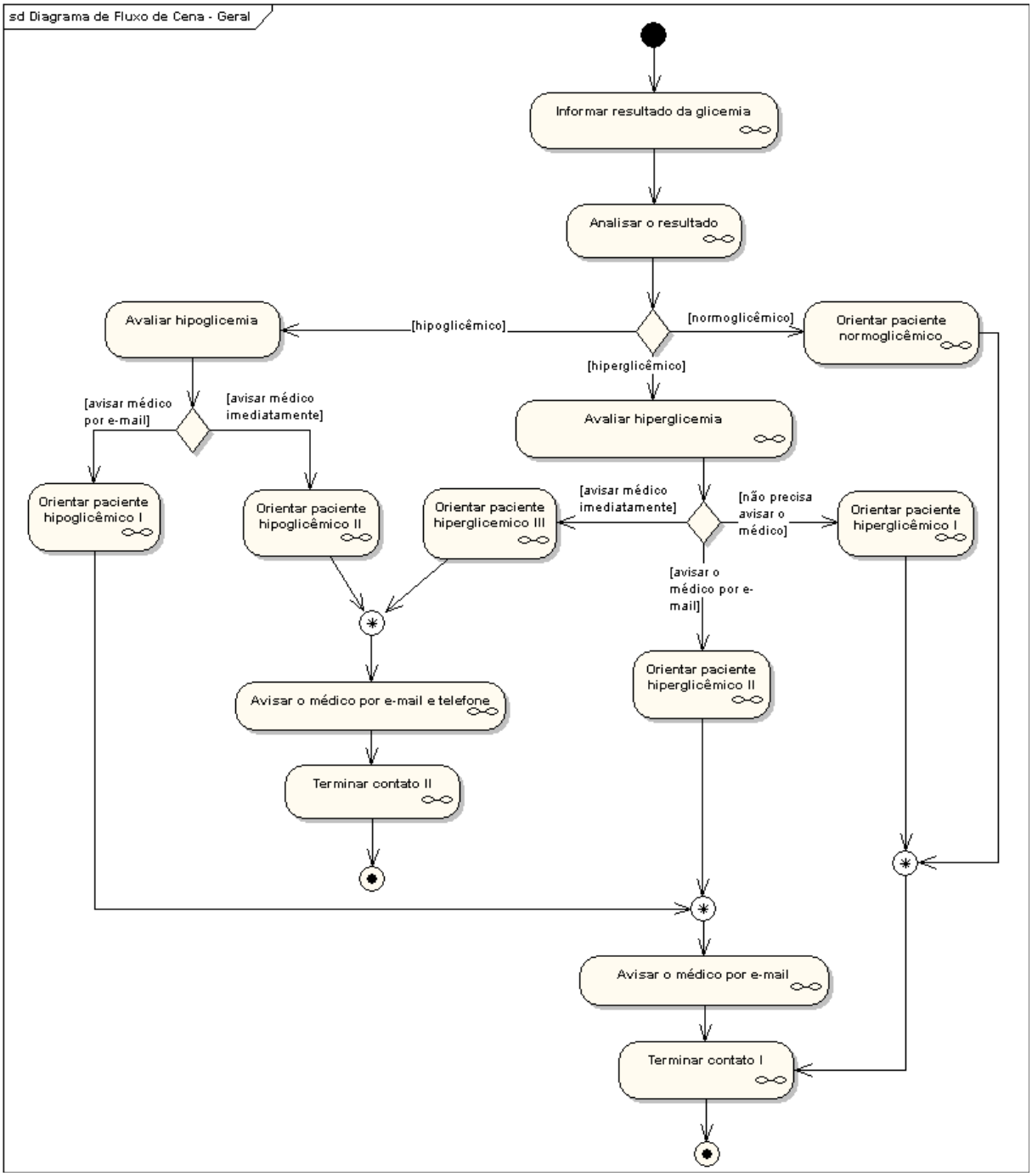


Figura 5-3 – Cenas do caso de uso "monitorar níveis glicêmicos" (ITO,2006)

No entanto, como o objetivo não é comprovar o modelo GRPC nem a funcionalidade "Monitorar níveis glicêmicos", foram implementadas somente as duas primeiras cenas "Informar resultado glicemia" e "analisar resultado" que por si só envolvem três agentes em negociação. Na tabela 4 (ITO, 2006) se encontram as ações referentes a estas duas cenas:

Cena	Agente	Objetivo	Plano/ação
Informar resultado glicemia	Atendente	Monitorar glicemia	▪ Coletar glicemia ▪ Apresentar menu principal
	Enfermeiro	Coletar glicemia	▪ Abrir ocorrência ▪ Perguntar data medição ▪ Perguntar hora medição ▪ Perguntar valor medição
		Orientar paciente	▪ Analisar resultado ▪ Anotar resultado ▪ Mostrar resultado
Analisar o resultado	Diabetologista	Analisar resultado	▪ Analisar glicemia ▪ Emitir laudo

Tabela 5-1 – Análise das cenas encontrando os agentes, os obejtivos, os planos e as ações no caso de uso "monitorar níveis glicêmicos"

Baseando-se nestes modelos de agentes, foram assim projetados os três agentes com as seguintes funcionalidades:

- Atendente: encaminhar para "Coletar Glicemia" e mostrar o menu principal;
- Enfermeiro: "Coletar glicemia": Abrir ocorrência, perguntar data da medição, perguntar hora da medição, perguntar valor da medição e encaminhar para "Analisar o resultado";
- Diabetologista: "Analisar resultado": Analisar glicemia e emitir laudo.

Para aplicação dos testes algumas adaptações foram necessárias para tornar uma comparação da proposta viável. O princípio básico é que para podermos realizar uma comparação efetiva da proposta esta deveria ser confrontada com algum modelo estabelecido. O modelo estabelecido, no caso, é um modelo onde não existe a criação dinâmica de agentes e a conseqüente racionalização sobre papéis a serem assumidos de acordo com a necessidade da sociedade.

Assim, foi necessária a configuração do *framework* para simular sua inexistência. Ou seja, os mesmos testes realizados com o *framework* puderam ser feitos com uma plataforma equivalente que, porém, não gerencia a criação de novos agentes e nem sua distribuição,

denominando-o de *framework inativo*. Para poder realizar os testes, as perguntas da racionalização foram dispostas de forma que o agente simula a resposta dos usuários sem interferir no papel dos agentes.

Os testes foram realizados em dois microcomputadores com algumas características distintas. Devido à finalidade desta simulação (que é averiguar a eficácia da distribuição automática de papéis) e a constante evolução de hardware e criação de novos dispositivos, as configurações exatas das máquinas tornam-se irrelevantes e não serão reveladas em sua totalidade. Passamos a chamá-las de N e P de forma que a máquina P possui o dobro de memória e o dobro de processadores da máquina N.

5.2.1. Simulações com o framework inativo

Primeiramente foi realizada uma simulação para compreender os efeitos de muitos usuários simultâneos e o impacto na organização do tempo de resposta destes usuários, ou seja, o tempo que um usuário leva entre ler uma pergunta, digitar e enviar a resposta.

Número de atendimentos	Número de agentes	Tempo total de atendimento	Resposta usuário
15	3	**2s**	1ms
15	3	**34s**	1000ms
200	3	**12s**	1ms
200	3	**5m42s**	1000ms

Tabela 5-2 – Resultados obtidos na simulação com o *framework* inativo e três agentes

Essa simulação confirmou o fato de que quanto maior o número de atendimentos ou o tempo de resposta do usuário, maior o esforço por parte de todos os agentes. Ainda assim, foram realizados alguns testes também comparando a máquina N e P para certificar-se que esse aumento não era eventualmente ocasionado pelo computador utilizado:

Número de atendimentos	Número de agentes	Tempo total de atendimento	*Framework* ativo	Resposta usuário	Máquina
15	3	**25s**	Não	**1ms**	**N**
15	3	**2s**	Não	**1ms**	**P**
200	3	4 min 32s	Não	1ms	N
200	3	12s	Não	1ms	P
200	3	**>15 min**	Não	**1000ms**	**N**
200	3	**5 min 42s**	Não	**1000ms**	**P**

Tabela 5-3 – Resultados obtidos na simulação com o *framework* inativo e três agentes executando em máquinas de diferentes capacidades de processamento e memória

Com isso constatou-se que a suspeita o funcionamento da organização estava diretamente relacionado ao número de usuários e de sua interação individual com os agentes.

5.2.2. Simulações com o framework ativo

Em uma outra simulação (detalhada na tabela 7), dessa vez já com o *framework ativo* o mesmo problema foi exposto para a plataforma que criou autonomamente uma organização:

Número de atendimentos	Número de agentes	Tempo total de atendimento	Resposta usuário
200	13	11s	1ms
500	17	13s	1ms
15	11	10s	1000ms
200	55	27s	1000ms

Tabela 5-4 – Resultados obtidos na simulação com o *framework* ativo

Como pode ser visto na tabela 7, o teste foi realizado com o *framework ativo* onde os agentes se organizaram no número de 13 agentes realizando os 200 atendimentos em 11s ou no caso do usuário demorar 1000ms (ou um segundo) para responder, os agentes se organizaram no número de 55 agentes realizando os mesmos 200 atendimentos em 27s.

Foram realizadas algumas simulações para validar a capacidade de máquina e certificar que os tempos eram factíveis, relatadas na tabela 8:

Número de atendimentos	Número de agentes	Tempo total de atendimento	*Framework* ativo	Resposta usuário	Máquina
200	29	1min 13s	Sim	1000ms	N
200	55	27s	Sim	1000ms	P
200	12	41s	Sim	1ms	N
200	12	7s	Sim	1ms	P

Tabela 5-5 – Resultados obtidos na simulação com o *framework* ativo executando em máquinas de diferentes capacidades de processamento e memória

Apesar do aumento de tempo significativo em uma máquina de menor capacidade, continua crescendo o número de agentes caso os usuários demorem mais a responder.

5.2.3. Simulações comparativas

A partir dos testes individuais realizados tanto com o *framework ativo* como com o *framework inativo*, um resultado chamou muito a atenção: o teste realizado para 200 atendimentos com 1 segundo de tempo de resposta por cada pergunta emitida ao usuário. O *framework ativo* realizou o mesmo número de atendimentos em 8% do tempo do *framework inativo* porém com quase vinte vezes o número de agentes deste.

Assim, criou-se a suspeita de que a possibilidade de existir um número maior de agentes de acordo com a demanda (aumento no tempo de resposta do usuário ou no número de usuários) levou o *framework ativo* a realizar os atendimentos de forma otimizada.

Sendo assim, foram geradas diversas simulações que foram repetidas entre si, já que os valores diferiam, e então os valores mais recorrentes para cada simulação foram registrados neste trabalho.

Convém ressaltar que conforme pode ser observado na tabela 9 o *framework* não possui constância com relação aos tempos de testes, razão pela qual devemos ressaltar que os tempos discriminados neste capítulo não devem ser considerados como valores absolutos para capacidade de processamento de SMAs.

	Número de atendimentos	Número de agentes	Tempo total de atendimento	*Framework* ativo	Resposta usuário	Máquina
1	**15**	3	**34s**	Não	1000ms	P
2	**15**	3	**34s**	Não	1000ms	P
3	**15**	3	**32s**	Não	1000ms	P
4	**15**	3	**30s**	Não	1000ms	P
5	**15**	3	**28s**	Não	1000ms	P
6	**15**	3	**28s**	Não	1000ms	P
7	**15**	3	**30s**	Não	1000ms	P
8	**15**	3	**30s**	Não	1000ms	P

Tabela 5-6 – Resultados obtidos na simulação indicando a variação média de tempo de execução do *framework*

5.2.3.1. Seqüência de simulação 1: Tempo de atendimento

Inicialmente foram realizados alguns testes na máquina P, que possui o dobro de processadores, mas os resultados não foram conclusivos conforme gráfico 5-1 seguinte:

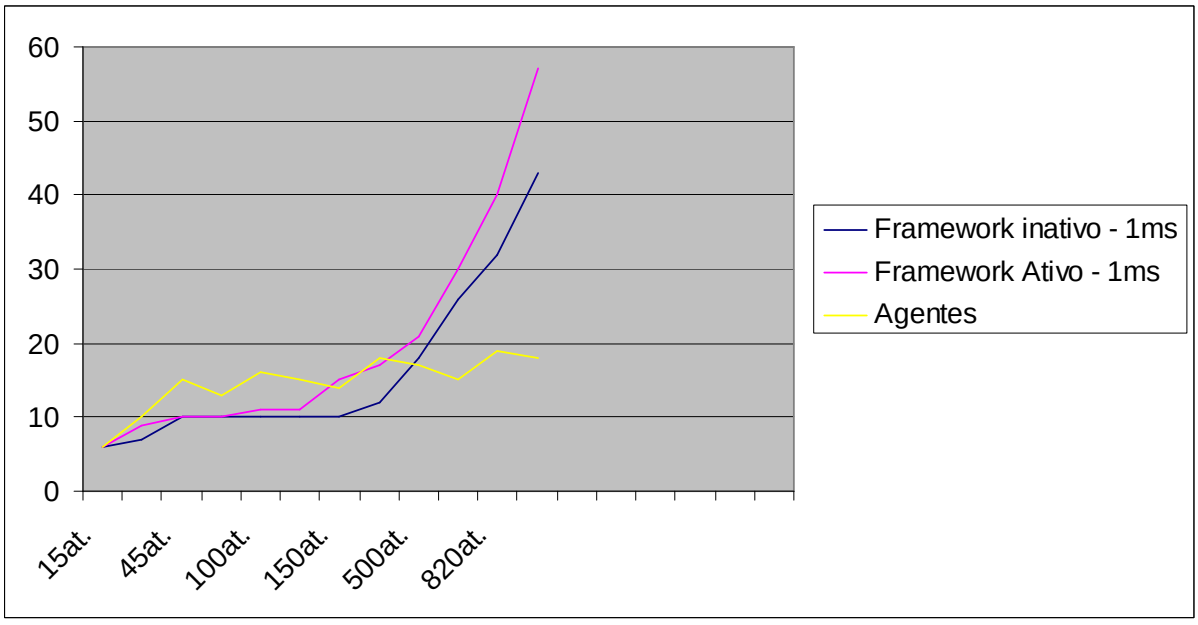


Gráfico 5-1 - Evolução de tempo de atendimento comparativa entre a simulação no *framework* ativo e no *framework* inativo (respostas do usuário em tempo real)

Acreditou-se que a possível causa se dava devido à capacidade de processamento paralelo, e que as respostas no tempo de 1ms (tempo real) depende exclusivamente da capacidade de simulação da máquina, ou no máximo, da interação com outros sistemas de agentes que executem no mesmo computador.

As simulações foram realizadas então considerando 1 segundo para o usuário ler uma pergunta e emitir uma resposta. Então, o *framework ativo* foi apresentado para este problema, e em seguida o *framework inativo* com 3 agentes (um diabetologista, um enfermeiro e um atendente). Posteriormente, mais duas simulações foram realizadas: com o *framework inativo* utilizando 6 e 12 agentes respectivamente distribuídos igualmente entre si, gerando o gráfico 5-2:

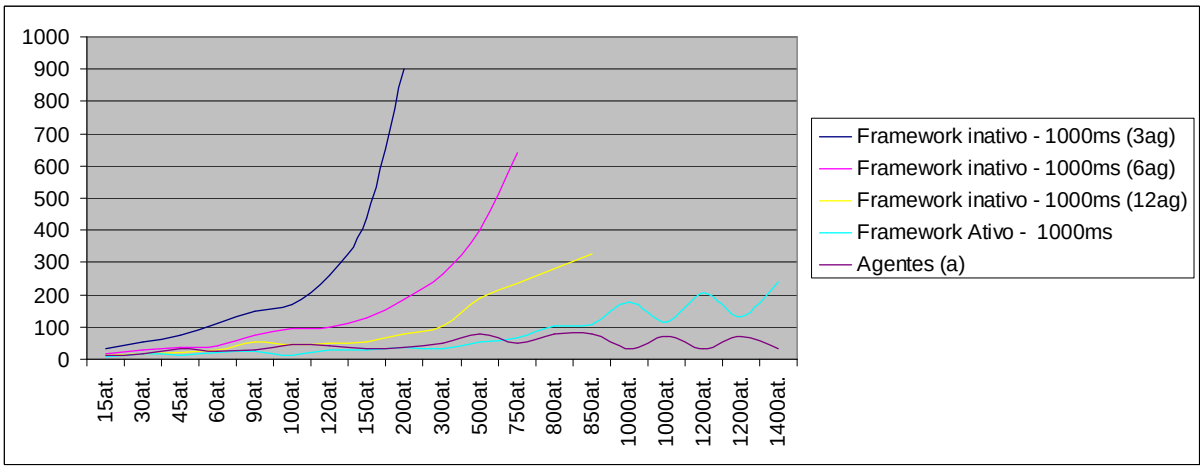


Gráfico 5-2 - Evolução de tempo de atendimento comparativa entre a simulação no *framework* ativo e no *framework* inativo (respostas do usuário em um segundo)

Para se concretizar a suspeita de que um número maior de agentes otimizava a execução, foi realizado um teste na máquina N que possui um desempenho menor e a diferença seria mais ressaltada, caso a suspeita fosse real. Realizamos o teste com o *framework inativo* e realizamos a distribuição de papéis ao mesmo número de agentes identificado pelo *framework ativo*. Os papéis foram igualmente divididos entre os agentes, no

seguinte formato: 4 atendentes, 4 enfermeiros e 4 diabetologistas. E isso proporcionou um ganho sensível de tempo, discriminado na tabela 10 seguinte:

Número de atendimentos	Número de agentes	Tempo total de atendimento	*Framework* ativo	Tempo resposta	Máquina
200	3	4min 32s	Não	1ms	N
200	12	1min 24s	Não	1ms	N

Tabela 5-7 – Impacto do número de agentes na organização

Os testes então foram repetidos para a execução exaustiva a partir do *framework ativo* comparativamente ao *framework inativo*, conforme gráfico 5-3:

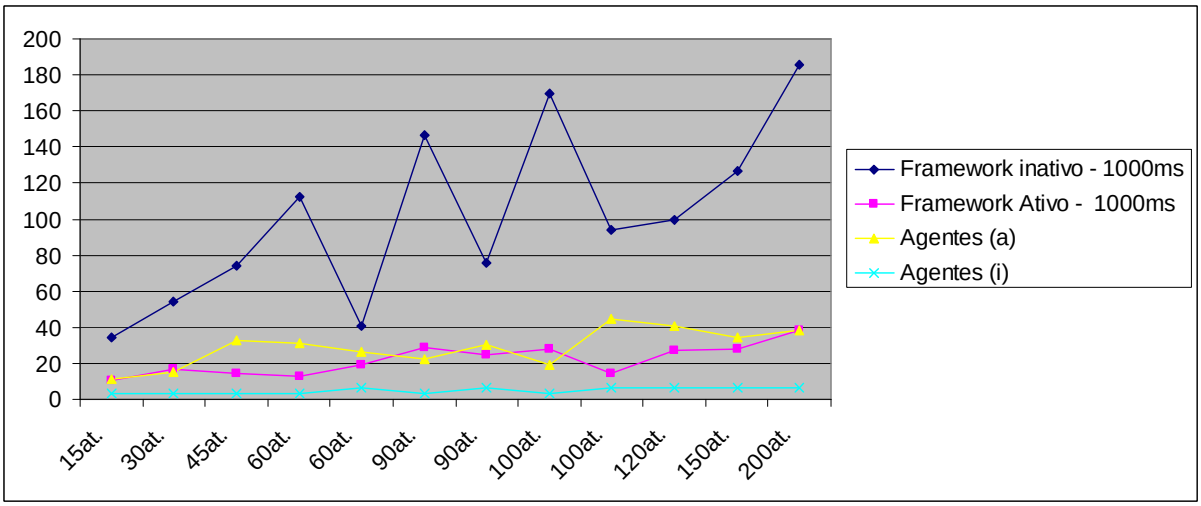


Gráfico 5-3 - Evolução de tempo de atendimento comparativa entre a simulação no *framework* ativo e no *framework* inativo (incrementando manualmente os agentes no *framework* inativo)

Como pode ser visto no gráfico 5-3 anteriormente apresentado, em dados momentos da simulação com o *framework inativo* teve seu número de agentes incrementado para evidenciar a otimização no tempo de atendimento contendo um número maior de agentes.

5.2.3.2. Seqüência de simulação 2:Número de agentes

Porém, o teste seguinte ainda não era conclusivo se não apresentasse um estudo mais aprofundado a cerca do número de agentes criado pelo *framewor ativo.* Um teste então contendo o mesmo número de agentes foi aplicado ao *framework* inativo e ao *framework* ativo a partir da máquina N (que por ser mais lenta evidencia mais os resultados) – tabela 11:

Número de atendimentos	Número de agentes	Tempo total de atendimento	*Framework* ativo	Tempo resposta	Máquina
200	12	41s	Sim	1ms	N
200	12	1min 24s	Não	1ms	N

Tabela 5-8 – Comparativo entre o *framework* ativo e o *framework* inativo (máquina de médio desempenho)

O mais interessante é que esse teste demonstrou uma economia de 50% do tempo de execução total da sociedade quando os agentes assumem papéis dinâmicos comparados a uma distribuição manual dos papéis de forma igual entre o mesmo número de agentes.

Foram realizados mais testes na máquina P (mais veloz) para comparar se essa economia de 50% de tempo é constante independente do número de agentes – tabela 12:

Número de atendimentos	Número de agentes	Tempo total de atendimento	*Framework* ativo	Tempo resposta	Máquina
200	3	>15min	Sim	1ms	P
200	3	12s	Não	1ms	P
200	13	11s	Sim	1ms	P
200	12	13s	Não	1ms	P
200	22	43s	Sim	1000ms	P
200	21	45s	Não	1000ms	P

Tabela 5-9 – Comparativo entre o *framework* ativo e o *framework* inativo (máquina de melhor desempenho)

Os resultados dos testes não apresentaram a economia de 50% de tempo esperada caso fosse feita manualmente a mesma distribuição no *framework inativo* que foi realizada automaticamente *framework ativo.*

Assim, independentemente da redução de 50% deve-se ressaltar que os testes foram feitos no *framework inativo* sabendo-se o número de agentes de antemão obtidos de testes com o *framework* ativo e fazendo a distribuição manual dos papéis entre eles. Pudemos deixar claro com isso que a distribuição de papéis automática se demonstrou muito útil para se identificar o número ótimo de agentes na sociedade e para que eles organizem-se entre si de forma autônoma, sem intervenção humana.

Mais testes foram então realizados sobre o *framework* ativo para esclarecer o mecanismo de criação de agentes e troca de papéis para atender à sociedade – gráfico 5-4:

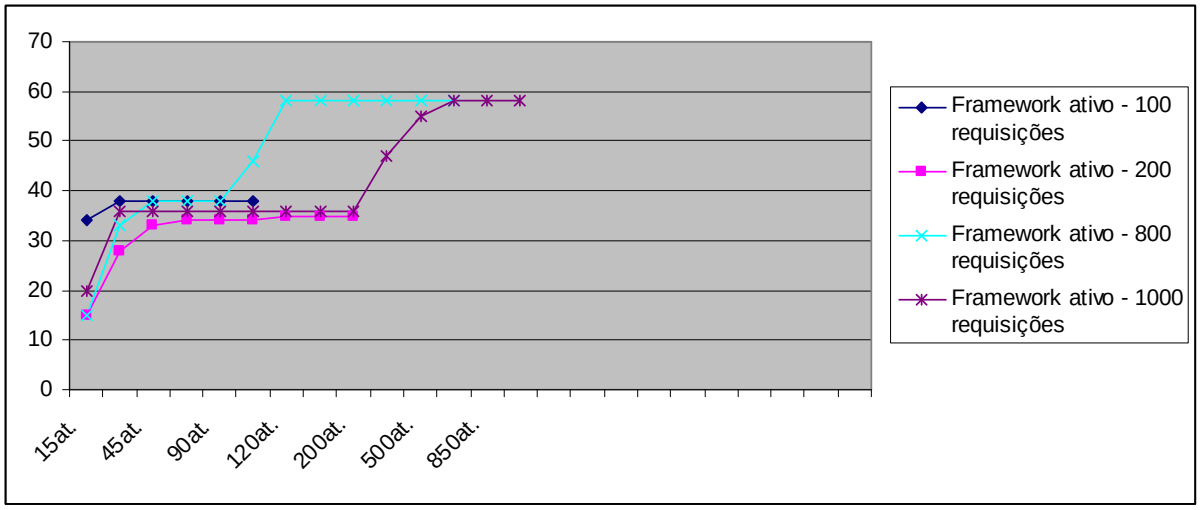


Gráfico 5-4 - Evolução do número de agentes no *framework* ativo de acordo com o número de requisições

A existência de um padrão em todas as simulações chamou a atenção: independente da simulação realizada, foi recorrente em todas as simulações o fato de que por um período de tempo o número de agentes permanece inalterado, quando de repente realiza um salto significativo. Estes dados foram analisados cautelosamente, e identificou-se que realmente a sociedade só se encarrega de criar novos agentes para assumir novos papéis quando todos os agentes estão ocupados procurando trabalhando e não há novos trabalhadores disponíveis para assumir papéis, como demonstrado nos gráficos 5-5 e 5-6:

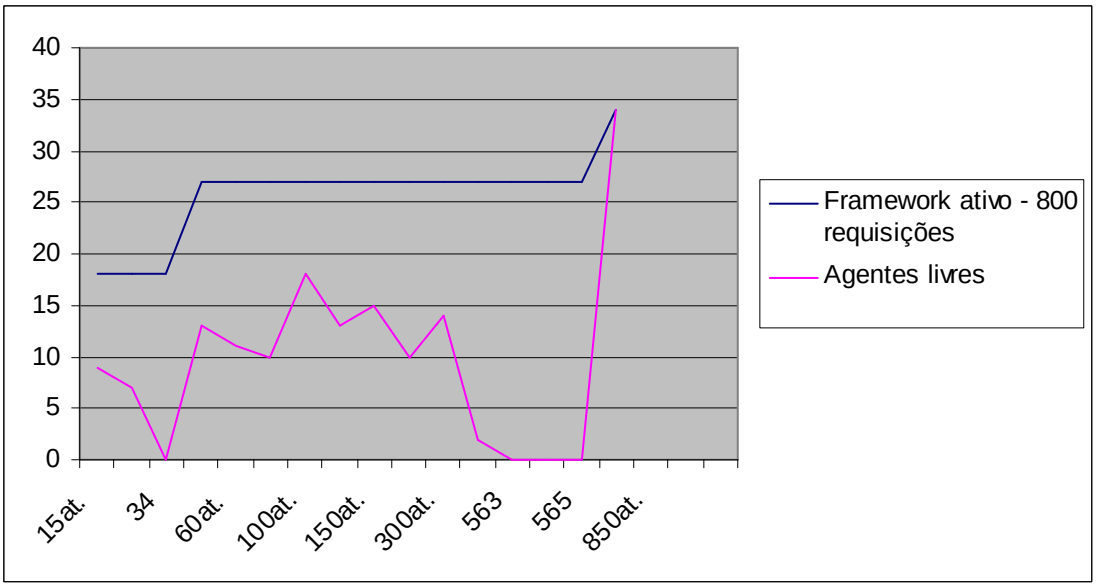


Gráfico 5-5 - Gerenciamento do número de agentes livres no *framework* ativo (com 800 requisições na fila)

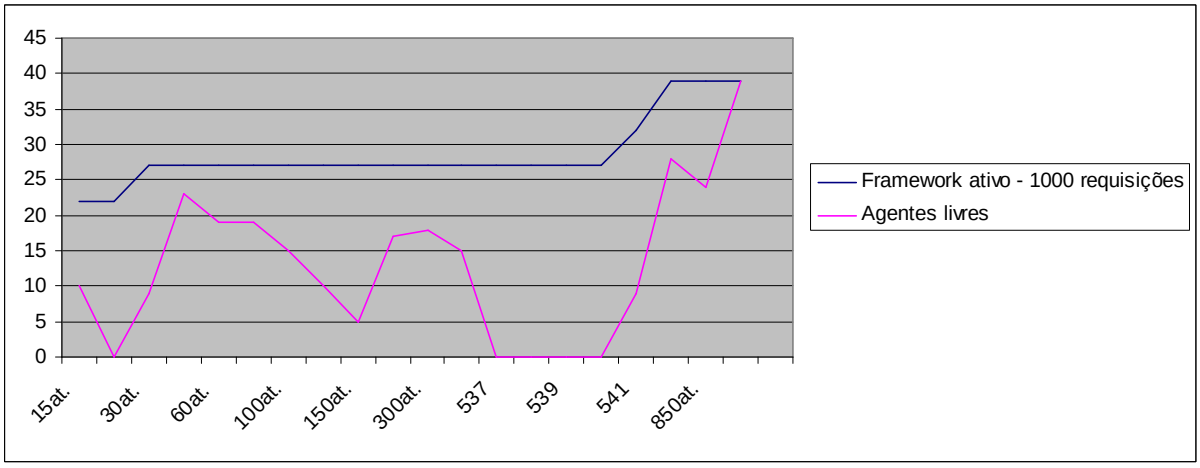


Gráfico 5-6 - Gerenciamento do número de agentes livres no *framework* ativo (com 1000 requisições na fila)

Porém os gráficos ainda nos trazem uma informação importante: ambos apresentam uma queda inicial de agentes livres que é imediatamente suprida, mas apresentam um vale extenso quando há um número maior de agentes trabalhando na sociedade. Ou seja, durante um período maior não existem trabalhadores disponíveis suficientes na sociedade. Isso nos leva a crer que a maior demanda por recursos de máquina impulsionada por um maior número de agentes trabalhando faz com que o agente *broker* tenha dificuldades na criação de novos agentes para ajudar a sociedade.

6. Conclusões e discussão

O sistema proposto para o modelo se trata de uma *Plataforma de Agentes* (FIPA, 2008b), onde é suportada a característica de inteligência distribuída orientada a agentes. A arquitetura proposta implementa as características de Agentes definidas por Wooldridge (2002, p.23), e seus maiores esforços estão no seu sistema de racionalização que se baseia no formato de papéis (possibilita que novos papéis sejam inseridos mantendo o sistema em operação) e principalmente pela distribuição dos papéis que se dá pela negociação social apoiada por um gerente.

6.1. SUPORTE AO MODELO GRPC

Nota-se que para viabilizar o componente operacional no modelo GRPC é necessário criar uma infra-estrutura para permitir a coleta de dados do paciente, pois segundo Ito (2006) a fragmentação sobre dados clínicos do paciente em diversas instituições dificulta o trabalho dos médicos durante o diagnóstico.

Foi proposta a melhoria da comunicação entre as unidades de saúde utilizando a tecnologia da informação baseada no modelo GRPC e Sistemas MultiAgentes (SILVA; ITO, 2007a).

Segundo Silva e Ito (2007a) são vislumbrados grandes ganhos tanto para pacientes como para prestadores de serviços como resultado de tal integração. Nisso a contribuição do *framework* é latente à medida que possibilita um acesso externo aos agentes da central, tanto através da integração com outros Sistemas Multi-agentes como através da disponibilidade das ações dos diversos papéis dos agentes cadastrados e a serem cadastrados no formado de serviços prestados pelos agentes a qualquer tipo de sistema (SILVA; ITO, 2007b).

6.2. GERENCIAMENTO DE PAPÉIS E DESEMPENHO

Os testes indicam uma tendência de um tempo otimizado quando os agentes compartilham papéis e assumem responsabilidades distintas de acordo com a necessidade da sociedade, porém não foi possível encontrar um número exato que representasse a eficácia dessa abordagem sobre a existência de papéis fixos igualmente distribuídos uma vez que com a abordagem utilizada para os testes todos os papéis eram igualmente necessários (proporção Atendente, Enfermeiro, Diabetologista 1:1:1), e consequentemente nos testes comparativos foram distribuídos igualmente. No entanto, ficou evidente a necessidade de um *framework* que gerencie a existência de um número ótimo de agentes dado o ambiente nos quais estão inseridos. E sendo assim, torna-se necessário que os agentes que compõe este *framework* consigam em tempo de execução ter o discernimento para saber qual a o papel que devem assumir, para não termos agentes inúteis à organização. O *framework* proposto se demonstrou capaz de atender a esse propósito, sem ferir os aspectos de SMAs de autonomia, racionalização e negociação.

Provavelmente a maior contribuição do trabalho é a criação de agentes inspirados na criação simples de ações conversacionais e também na possibilidade de atingir picos de performance na organização, criando então um mecanismo de racionalização de gerenciamento de carga auxiliado por um agente gerente *broker*.

Apesar de outros trabalhos já terem se preocupados com a idéia de performance em sistemas de agentes, esta proposta inova à medida que se preocupa principalmente com o aspecto organizacional dos agentes e como possibilitar a sua utilização de forma facilitada, diferenciando-se testes de performance como (CHMIEL; TOMIAK; GAWINECKI; KARCZMAREK; SZYMCZAK; PAPRZYCKI, 2004) que apesar de falar do Jade, que também foi utilizado no *framework* proposto, faz um exaustivo teste de performance na plataforma mas sob um aspecto mais geral sem entrar nos méritos de como a organização de

agentes ou o processo de racionalização pode interferir na performance. Ou até mesmo a análise de um outro artigo (KINNY; GEORGE, 1991) que entra na questão de performance em agentes e usa uma plataforma de teste de performance de agente denominada "Tileworld", mas cujo resultado é propor um processo de medição de performance e escalabilidade de agentes.

Segundo Kolp (2001) uma arquitetura onde não existe controle de um ator sobre o outro é denominada Flat Structure. Apesar de possuirmos um gerente da sociedade que monitora a interação entre todos os agentes trabalhando como um facilitador da sociedade, a estrutura *flat* é a mais marcante na proposta apresentada, uma vez que os agentes apresentam um grande número de comunicação e negociação entre si “[a estrutura flat] apresenta um alto número de racionalização e de comunicação por cada ator participante” (KOLP, 2001, p. 6, tradução nossa). Apesar dessa desvantagem, ainda de acordo com o mesmo estudo este tipo de estrutura facilita a ininterrupção dos serviços oferecidos, e a tolerância a falhas uma vez que “[na estrutura flat] a falha de um ator não implica na falha de todo sistema” (KOLP, 2001, p.22, tradução nossa).

Vale ainda ressaltar o estudo realizado em Ito (1999) sobre o volume de troca de mensagens em sistemas baseado em busca não informada de parceiro (Rede Contractual – RC) *versus* busca informada de um parceiro (Coalisão Baseada em Dependências – CBD). O estudo demonstra que no modelo CBD onde o agente já conhece de antemão seus parceiros antes de enviar a mensagem, de uma forma geral, a sociedade possui uma eficiência melhor:

> numa sociedade em que seus integrantes são agentes que geralmente conseguem formar coalisões, o modelo CBD se torna mais eficiente em termos de fluxo de comunicação quando obtém respostas afirmativas na primeira proposta (ITO, 1999, p.141).

Assim, com a utilização do JADE e a proposta de criação de papéis do *framework* foi permitido criar um catálogo de agentes e serviços onde mesmo baseando-se no modelo

Contract-net (RC), conseguimos realizar o proposto em Ito (1999, p.43) onde “os anúncios podem ser enviados a todos os agentes disponíveis, para um pequeno grupo de agentes ou a um único agente” possivelmente assemelhando-se ao fluxo de mensagens do modelo CDB, algo não comprovado por não fazer parte do objetivo de estudo.

6.3. CONTINUIDADE E ESTUDOS FUTUROS

Como devido às suas características cada agente é capaz de atuar independentemente, a orientação a Agentes é uma grande candidata para prover implementação prática de diversos domínios uma vez que seu modelo de representação de serviços e modelo de componentes preza pela padronização e independência de arquitetura.

Assim, com este trabalho algumas abordagens de pesquisa são vislumbradas, tais como o análise do fluxo de mensagens entre os agentes – a exemplo do elaborado em (ITO,1999), o gerenciamento do desempenho da organização a partir do desempenho total do *hardware* - tendo em vista as colocações de Wooldridge (2002) ao colocar SMAs como um subitem de pesquisa em sistemas distribuídos, a integração desta proposta com alguma espécie de representação do conhecimento, a aplicação e testes em outros domínios além do acompanhamento de pacientes crônicos, a expansão da sua racionalização para além do aspecto conversacional para, por exemplo, o aspecto dimensional em aplicativos GED (de referência geográfica) e também o trabalho de usabilidade do sistema sob a ótica do usuário.

7. Referências Bibliográficas

ALLEN, J. F.; HENDLER, J.; TATE, A. **Readings in Planning**. San Mateo, CA: Morgan Kaufmann, 1990.

AMANT, R.St.; ZETTLEMOYER, L.S. THE USER INTERFACE AS AN AGENT ENVIRONMENT. In: International Conference on Autonomous Agents, 2000**. Proceedings of the fourth international conference on Autonomous agents**, ISBN:1-58113-230-1. Barcelona, Spain, 2000. p. 483 - 490 Year of Publication: 2000

ÁLVARES, L.O.; SICHMAN, J, S. Introdução aos Sistemas Multiagentes. In: JORNADAS DE ATUALIZAÇÃO EM INFORMÁTICA, XVI JAI. XVII Congresso da SBC, Brasília, 2-8 agosto 1997. **Anais do XVII Congresso da SBC.** 1997

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software architecture in practice**. SEI – Series in Software Engineering. 2. ed. Addison Wesley, 2003. 528 p.

BALDONI, M.; BOELLA, G.; Genovese,V.; GRENNA, R.; van der TORRE L.; How to Program Organizations and Roles in the JADE Framework. In: GERMAN CONFERENCE ON MULTI-AGENT SYSTEM TECHNOLOGIES, MATES 2008, September 2008, Germany. R. Bergmann, G. Lindemann, S. Kirn, and M. Pechoucek, eds, **Proceedings of the Multiagent System Technologies**, volume 5244, Germany, 2008. p. 25-36

BELLIFEMINE, F.; POGGI, A.; RIMASSA, G. Jade, a FIPA-compliant agent framework. In: INTERNATIONAL CONFERENCE ON PRACTICAL APPLICATION OF INTELLIGENT AGENTS AND MULTI-AGENT TECHNOLOGY, 4, 1999, London, **Proceedings of International Conference On Practical Application Of Intelligent Agents And Multi-Agent Technology.** 1999. p. 97-108.

BELLIFEMINE F.; CAIRE, G.; TRUCCO, T.; RIMASSA, G**. JADE Programmer's Guide, 2003**. Disponível em http://sharon.cselt.it/projects/jade/doc/programmersguide.pdf

BOELLA, G.; TORRE, L. van der. Attributing mental attitudes to roles: the agent metaphor applied to e-trade organizations. In: 6TH INTERNATIONAL CONFERENCE ON ELECTRONIC COMMERCE. Session: Multi-agent systems and social behavior table of contents. Delft, The Netherlands. **ACM International Conference Proceeding Series**. Vol. 60. ISBN:1-58113-930-6. Università di Torino, Italy. p. 130 - 137. 2004.

BOELLA, G.; TORRE, L. van der. Regulative and constitutive norms in normative multiagent systems. In: INTERNATIONAL CONFERENCE ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING (KR'04), 9, 2004, Whistler, Canada. **Proceedings of KR'04**. AAAI Press, 2004. p. 255—265.

BOELLA, G.; TORRE, L. van der. Organizations as Socially Constructed Agents in the Agent Oriented Paradigm. Engineering Societies in the Agents World V 1-13. Book SeriesLecture Notes in Computer Science PublisherSpringer Berlin / Heidelberg ISSN0302-9743 (Print) 1611-3349 (Online) Volume 3451/2005 Book Copyright2005

BOOCH, G. **Object-oriented analysis and design with applications.** Addison Wesley, 1994.

BRATMAN, M.E. **Intention, Plans, and Practical Reason**. Harvard University Press: Cambridge, MA, 1987

BRATMAN, M.E.; ISRAEL, D.J.; POLLACK, M.E. PLANS AND RESOURCE-BOUNDED PRACTICAL REASONING**.** In: **Computational Intelligence.** v 4, 1988. p. 349 – 355.

Branko, G.C.; Nigel, H.L.; Jim, B. **Using information technology to improve the management of chronic disease**. Medical Journal Australian, (s. l.), v. 179, N. 5, p. 242-246. September, 2003.

Cabri, G.; Ferrari, L.; Leonardo, L. **Rethinking agent roles: extending the role definition in the BRAIN framework**. Systems, Man and Cybernetics, 2004 IEEE International Conference on Volume 6, Issue , 10-13 Oct. 2004 Page(s): 5455 - 5460 vol.6 Digital Object Identifier 10.1109/ICSMC.2004.1401061

CARDOSO, F.H. Prefácio. In: CASTELLS, M. (autor). **A sociedade em rede**. 1ª. Edição. São Paulo: Editora Paz e Terra. 1999. Prefácio, p. 35-37. (A era da informação: economia, sociedade e cultura – volume 1).

CASTELLS, M. **A sociedade em rede**. Tradução: Roneide Venancio Majer com a colaboração de Klauss Brandini Gerhardt. 7ª. Edição. São Paulo: Editora Paz e Terra. 2003. (A era da informação: economia, sociedade e cultura – volume 1).

CELLER, B. G.; LOVELL, N. H.; BASILAKIS, J. **Using information technology to improve the management of chronic disease**. Medical Journal Australian, (s. l.), v. 179, N. 5, p. 242-246. September, 2003.

CHMIEL, K.; TOMIAK, D.; GAWINECKI, M.; KARCZMAREK, P.; SZYMCZAK, M.; PAPRZYCKI, M. Testing the Efficiency of JADE Agent Platform. In: THIRD INTERNATIONAL WORKSHOP ON ALGORITHMS, MODELS AND TOOLS FOR PARALLEL COMPUTING ON HETEROGENEOUS NETWORKS. **Proceedings of the Third International Symposium on Parallel and Distributed Computing**. ISBN:0-7695-2210-6. 2004. p. 49 - 56.

CHAU, P. Y. K.; HU, P. J. H. Techonology implementation for telemedicine programs. Communication of the ACM, New York, v. 47, n. 2, p. 87-92. February 2004.

CLEMENTS, P.; BACHMANN, F.; BASS, L.; GARLAN, D.; IVERS, J.; LITTLE, R.; a NORD, R.; STAFFORD, J. **Documenting software architectures: views and**

beyond. 1. edição. (s. l.): Addison Wesley, 2003. 512 p. (SEI – Series in Software Engineering).

COEN, M.H. SodaBot: A Software Agent Environment and Construction System. In: THIRD INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, CIKM 1994. **Proceedings of the CIKM Workshop on Intelligent Information Agents**,

COHEN, S.; SHABO, A.; **Electronic Health Record (EHR) Standards Survey**. IBM Haifa Research Lab., Haifa, 2001.

DURFEE, E. H.; ROSENSCHEIN, J. S. Distributed Problem Solving and Multi-Agent Systems: Comparions and Examples. In: 13TH INTERNATIONAL WORKSHOP ON DISTRIBUTED ARTIFICIAL INTELLIGENCE, IWDAI-94. Seattle, 1994. **Proceedings**. Seattle, AAII, 1994. p. 46-55.

FERBER, J.; MÜLLER, J.P. Influences and Reaction: a Model of Situated Multiagent Systems. In: 2TH INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS. **Proceedings of the 2th International Conference on Multi-agent Systems.** AAAI Press, Nara, Japan, 1996. p. 72–79.

FIPA. **Fipa Abstract Architecture Specification**. Disponível em: http://www.fipa.org/specs/fipa00001/index.html. Acessado em: 10/08/2008.

FIPA. **Fipa Agent Management Specification**. Disponível em: http://www.fipa.org/repository/managementspecs.php3. Acessado em: 05/11/2008.

(FRANKLIN, 1996) FRANKLIN, S.; GRAESSERM A. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In: 3RD. INTERNATIONAL WORKSHOP ON AGENT THEORIES, ARCHITECTURES, AND LANGUAGES. **Proceedings of the 3rd. International Workshop on Agent Theories, Architectures, and Languages**. Springer-Verlag, 1996.

GROSOF, B.N. Building Commercial Agents: An IBM Research Perspective (Invited Talk)" (Apr. 21, 1997). In: SECOND INTERNATIONAL CONFERENCE ON THE PRACTICAL APPLICATIONS OF INTELLIGENT AGENTS AND MULTI-AGENT TECHNOLOGY (PAAM97)**. Proceedings**. Barry Crabtree ed. The Practical Applications Company, Blackpool, Lancashire, UK. Held London, UK, April 21-23, 1997.

HAIT, F. B. **Agent oriented programming with Ada'95: application to financial markets**. Volume XX , Issue 1 (March 2000) table of contents. Pages: 67 – 80. Year of Publication: 2000. ISSN:1094-3641. Université Paris XII., 61, avenue du GAL de Gaulle. 94010 Creteil Cedex Publisher ACM Press New York, NY, USA

HANNOUN, M.; BOISSIER, O.; SICHMAN, J.S.; SAYETTAT, C. MOISE: An organizational model for multi-agent systems. In: BRAZILIAN SYMPOSIUM ON AI (IBERAMIA-SBIA 2000), International joint conference No15, Atibaia, BRASIL,

(19/11/2000). **Proceedings of International joint conference.** ISBN 3-540-41276-X. Atibaia, BRASIL, 2000, vol. 1952, pp. 156-165[Note(s) : XV, 498 p.,] (14 ref.)

HAYES-ROTH, B. "An Architecture for Adaptive Intelligent Systems," Artificial Intelligence: Special Issue on Agents and Interactivity, 72, 329-365. 1995

HUBNER, J.F.; SICHMAN, J.S. Organização de Sistemas Multiagentes In II Jornada de Atualização em Inteligência Artificial (in Portuguese), JAIA 2003, XXI Congresso da Sociedade Brazileira de Computação, SBC'2003, em CD, 40 pages, Campinas, Brazil, August 2003. Disponível em http://www.lti.pcs.usp.br/moise/doc/orgSMA-jaia-2003.pdf

ITO, Márcia. Um modelo de gestão de paciente crônico baseado nos conceitos de relacionamento com o cliente. 2006. p. 19 (Motivações e resultados esperados), p. 30 (Principais sistemas de monitoramento de pacientes crônicos). Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2006.

ITO, M. Uma Análise do Fluxo de Comunição em Organizações Dinâmicas de Agentes. São Paulo, 1999. 154 p. Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Elétrica.

JANCA, P. C. "**Pragmatic application of information agents**". BIS Strategic Report, 1995

JENNINGS, N. R. Agent-Oriented Software Engineering. In: GARIJO, F. J.; BOMAN, M. (eds.). Multi agent system engineering: proceedings of the ninth european workshop on modelling autonomous agents in a multi-agent world, 1999. pp. 1-7. (Lectures Notes in Artificial Inteligence,1647).

Macskassy, S.. **A conversational agent**. Master Essay, Rutgers University, 1996

KINNY, D.; GEORGE, M. COMMITMENT AND EFECTIVENESS OF SITUATED AGENTS. In: **Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)**, Sydney, Australia, 1991. p. 82-88.

KOLP, M. Organizational Styles and Agent Patterns.Tropos Meeting, University of Toronto, Department of Computer Science January 18 2001

LASTRES, H. M. M.; ALBAGLI, S. (Org.). **Informação e globalização na era do conhecimento**. 1ª. edição. Rio de Janeiro: Editora Campus. 1999. 318 p.

MASSARO, W.D.; COHEN, M.M.; BESKOW, J.; COLE, R.A. Developing and Evaluating Conversational Agents. In **Embodied conversational agents**. ISBN:0-262-03278-3. Pages: 287 – 318. 2001.

MEA, V. D. Agents acting and moving in healthcare scenario: a paradigm for telemedical collaboration. IEEE Transactions on Information Technology in Biomedicine, (s. l.), v. 5, no. 1, p. 10-13. March 2001.

Moore, R.C. A formal theory of knowledge and action. In Readings in Planning (eds J. F. Allen, J. Hendler and A. Tate), pp. 480-519. Morgan Kaufman, San Mateo, CA. 1990.

NWANA, H. S. SOFTWARE AGENTS: AN OVERVIEW. In: **Knowledge Engineering Review**. v. 11, n 3. Cambridge University Press, Outubro/Novembro 1996, pp. 205-244. Disponível em: http://agents.umbc.edu/introduction/ao/5.shtml. Acessado em 10/06/2008

NUTE, D. Defeasible Logic. In: **Handbook of Logic in Artificial Intelligence and Logic Programming.** Oxford University Press, 1994. p. 353-395.

OMS (Organização Mundial de Saúde). Information Technology in Suport of Health Care. Site da Instituição. Disponível em: <http://www.who.int/entity/eht/en/InformationTech.pdf>. Acesso em: 04 Jun. 2006.

POLLACK, M. E. The Uses of Plans. In: 12th International Joint Conference on Artificial Intelligence. **Computers and Thought Award at the 12th International Joint Conference on Artificial Intelligence**. Sydney, Australia, 26 de Agosto de 1991.

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

PRESSMAN, R. S. **Engenharia de software**. Tradução de Mônica Maria G. Travieso, Revisão técnica de Paulo César Masiero, José Carlos Maldonado, Fernão Stella R. Germano. 5ª. Edição. Rio de Janeiro: Editora McGraw Hill, 2002. 802 p.

RUSSELL, S. J.; NORVIG, P.; CANNY, J. F. **Artificial Intelligence: A Modern Approach**. Prentice Hall, 2003. 1080 p.

ROSENSCHEIN, J.S.; GENESERETH, M.R. Deals among rationals agents. In: 9TH INTERNATIONAL JOINT CONFERENCE ON AI (IJCAI 1985), 9, Los Angeles, California, USA, 1985**. Proceedings of the 9th International Joint Conference on AI**. Los Angeles, CA: Morgan Kaufmann,1985. p. 91-99.

SCHWAMBACH, M. M.; PEZZIN, J.; FALBO, R. A. OplA: uma metodologia para o desenvolvimento de sistemas baseados em agentes e objetos. In: JORNADA IBEROAMERICANA DE ENGENHARIA DE SOFTWARE E ENGENHARIA DO CONHECIMENTO, 4, Madrid, 2004. **Proceedings**.

SILVA, C. T. L. L.; CASTRO, J. F. B. Modeling Organizational Architectural Styles in UML: The Tropos Case. In: V WORKSHOP ON REQUIREMENTS ENGINEERING – WER'02. Valencia, Espanha, Novembro de 2002. **Proceedings**, 2002.

SILVA, L. R. ; ITO, M. Proposta de Integração das Unidades de Saúde Utilizando um Sistema Multiagente para Acompanhamento de Pacientes Crônicos. In: VII WORKSHOP DE INFORMÁTICA MÉDICA (WIM´2007), 2007, Porto de Galinhas,

PE, Brasil. VI Simpósio Brasileiro da Qualidade de Software (SBQS 2007), 2007. **Anais do VI Simpósio Brasileiro de Qualidade de Software.**

SILVA, L. R. ; ITO, M. . Utilizando a tecnologia de orientação a agentes para implementar a arquitetura orientada a serviços. In: 2º WORKSHOP DE PÓS-GRADUAÇÃO E PESQUISA DO CENTRO PAULA SOUZA, 2007, São Paulo. **Anais do 2º Workshop de Pós-Graduação e Pesquisa do Centro Paula Souza**, 2007.

SIMON, H. A. "**The sciences of the artificial**". MIT Press, 1996.

SMITH, D. C.; CYPHER, A.; SPOHRER, J. "KidSim: Programming Agents Without a Programming Language," **Communications of the ACM**, 37, 7, 55-67. 1994

THOMASON, R. H. Desires and Defaults: A Framework for Planning with Inferred Goals. In: INTERNATIONAL CONFERENCE ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING (KR2000), 7, 2000, Breckenridge, Colorado, USA. **Proceedings of Seventh International Conference on Principles of Knowledge Representation and Reasoning**. Morgan Kaufmann, San Francisco, pages 702-713, year 2000.

(URUGUAY; HIRATA, 2006) URUGUAY, A., HIRATA, C. Using IDEF0 to Enhance Functional Analysis in MOISE+ Organizational Modeling. Brazilian Artificial Intelligence Symposium, 2006

(VENKATESAN, 2008) VENKATESAN, V.P. Arquitetura para orquestração de serviços com agente BDI. White Paper, Microsoft. Disponível em: http://www.microsoft.com/brasil/msdn/arquitetura/20080428/Arquitetura_agente_BDI.mspx. Acessado em 15/10/2008.

(WEYNS, 2004) [Weyns2004](117) WEYNS, D.; STEEGMANS, E.; HOLVOET, T. Toward active perceptions in situated multi-agent systems. Journal on Applied Artificial Intelligence, (s. l.), v. 18, p. 9-10. 2004.

WEYNS, D.; PARUNAK, V. D.; MICHEL, F.; HOLVOET, T.; FERBER, J. Environments for multiagent systems state-of-the-art and research challegens. In: ENVIRONMENTS FOR MULTI-AGENT SYSTEMS INTERNATIONAL WORKSHOP, 1, New York, 2004. **Revised Selected Papers of Environments For Multi-Agent Systems International Workshop**. (s. l.): Spring-Verlag, Vol. 3374. 2005.

WOOLDRIDGE, M.; JENNINGS, N. R. "**Agent Theories, Architectures, and Languages: a Survey**". In Wooldridge and Jennings Eds., Intelligent Agents, Berlin: Springer-Verlag, 1995. p. 1-22.

WOOLDRIDGE, M. Agent-based software engineering. IEE. **Proc. on Software Engineering**, 144 (1) 26-37. 1997

WOOLDRIDGE, M. **Reasoning about Rational Agents**. The MIT Press (Cambridge, Massachussetts/London, England). In: *Intelligent robotics and autonomous agents* series. ISBN 0-262-23213-8. June 2000. 240pp;

WOOLDRIDGE, M. J. **An introduction to multiagent systems**. 1a. edição. England: John Wiley & Sons, 2002. 348 p.

A. Apêndice I - A Arquitetura Jade

JADE é uma plataforma de agentes, AP que segundo FIPA (2008b, p. 6, tradução nossa) :

> Provê a infra-estrutura física onde os agentes podem ser configurados. Uma plataforma de agentes (AP – *agent plataform*) consiste de máquinas, sistemas operacionais, softwares de suporte a agentes, componentes de gerenciamento de agentes FIPA (DF, MAS e MTS) e agentes.

O JADE pode ser compreendido como um sistema construído em JAVA que implementa mecanismos de sistemas orientados a agentes. O propósito do JADE é servir como uma implementação base para outras implementações de aplicações Multi-agentes.

Ao contrário do que discutimos a respeito de tipos de agentes e racionalização, o Jade não especifica quais tipos de agentes podem ser construídos a partir dele. Por outro lado, disponibiliza alguns mecanismos da maioria das teorias de agentes atuais:

- Passagens de mensagens usando objetos *ACLMessage*;
- Suporte ao ciclo de vida FIPA dos agentes;
- Execução de múltiplas atividades concorrentes;
- Facilidade para gerenciamento da sociedade *Directory Facilitator* (ou *yellow pages*);
- e Interface gráfica para gerenciamento da sociedade (figura 8-1)

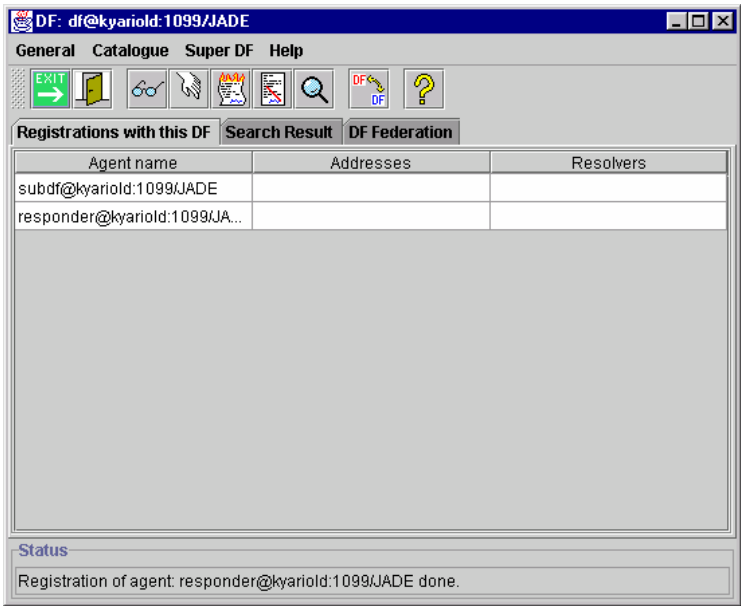


Figura A-1 – Gerenciamento do JADE por Interface gráfica (BELLIFEMINE, 2003)

Além disso o próprio JADE disponibiliza mecanismos de mobilidade entre agentes. O JADE utiliza o conceito de "container" de agentes, a partir do qual tem-se a relação direta de um container executando sobre uma JVM do JAVA. Definindo-se um container padrão, e dizendo quais são os seus *containers* subordinados, os agentes passam a ter a possibilidade de migrar (de acordo com sua própria vontade) para um outro container que pode estar em uma máquina distante, visível apenas por uma rede local, *intranet* ou internet.

Alem disso, os agentes podem ainda migrar de um container principal, para qualquer outro container principal. Isso possibilita sistemas de organizações distintas se integrarem através dos agentes, definidas as regas e políticas para isso [Artigo WIM 2007]. A figura 8-2 representa 3 *containers* principais, sendo que dois deles possuem *containers* subordinados:

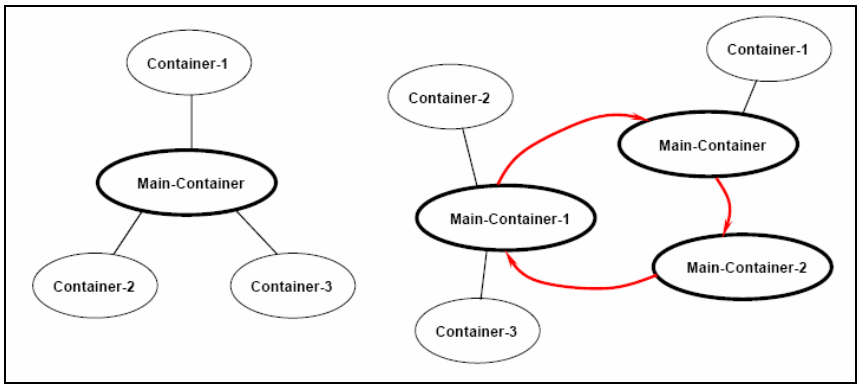


Figura A-2 - Agentes migrando entre diversas plataformas (BELLIFEMINE, 2003)

Como os agentes definem suas regras de migração para outras plataformas e a requerem para o container aos quais estão subordinados, o JADE disponibiliza uma linguagem comum para que os agentes possam "conversar" sobre mobilidade. Em uma situação exemplo, um agente em outro container autoriza sua entrada baseado em uma comunicação utilizando a linguagem disponibilizada pelo JADE no pacote *jade.domain.mobility*.

A Figura 8-3 representa a descrição da arquitetura interna de um agente genérico em JADE:

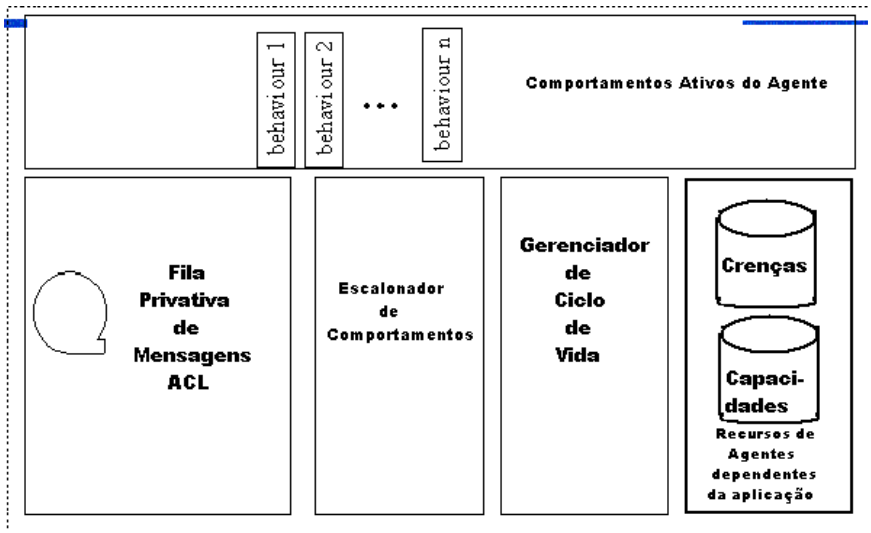


Figura A-3 - Arquitetura interna de um agente genérico em JADE (BELLIFEMINE, 2003)

Da figura 8-4, apenas o quadro inferior direito é implementado pelo desenvolvedor de agentes na plataforma JADE. E para isso, o JADE disponibiliza alguns mecanismos para facilitar a implementação de comportamento na plataforma:

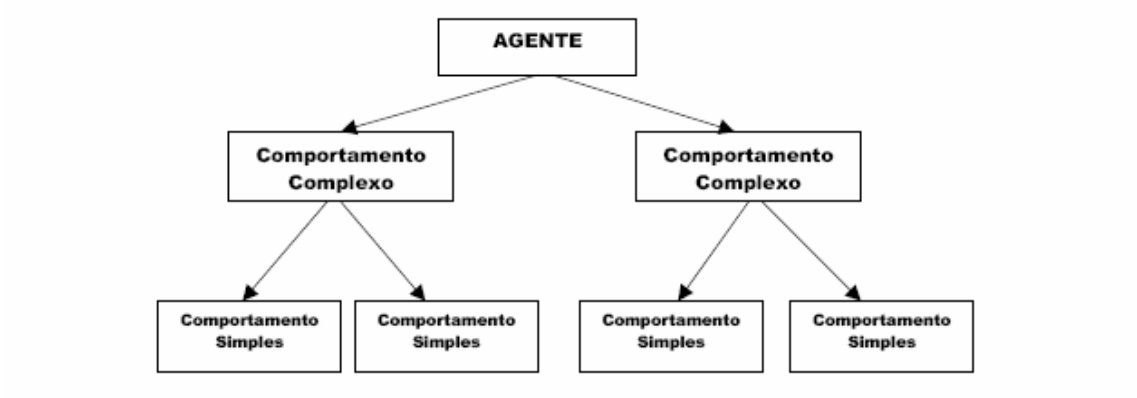


Figura A-4 - Estrutura do comportamento no JADE (BELLIFEMINE, 2003)

B. Apêndice II – Implementação da classe ServerRole

A estrutura básica da proposta é apresentada na figura 23 e a representação formal de agentes e seus papéis é visto no relacionamento composto entre a tríade `Agente, ServerRole e Goals.` As três classes juntas representam a estrutura básica de um Agente no *framework*, pois os agentes "livres" são criados e podem começar a trabalhar em qualquer papel definido na sociedade.

Resumidamente, o conceito que pode ser representado por um "Role Schema" em Gaia é exibido no Quadro 9-1. Um agente assim que criado se torna "livre", ou sem algum papel determinado. Para que possam assumir papéis, existe um mecanismo `ServerRole` que é disponível a todos agentes.

FREE = Setup. (ServerRole) SERVERROLE = (ReceiveBrokerACLProposal. (AnyRole | SubscribeFree) . DFRegister)$^{\omega}$ GOALS = (Pergunta | ServerIncoming) | Identificacao | (Choice.[RequestPerformer]) PROPOSAL = (LookFree. (InformNone | InformeThose) . [SendMessage]) + SERVERINCOMING= ReceiveMessage*

Quadro B-1 - Role Schema do *framework* em Gaia

Esta classe é representada na arquitetura como um *comportamento* do agente, e esse comportamento é explicado no quadro 9-2:

```
  public void action() {

       if(myAgent.newRole != null)
       {
         atualizaPapeis();
         atribuiPapelAgente();
         myAgent.newRole = null;
       }
       else if(myAgent.newService != null)
      {
       atualizaPapeis();

       for ( Goals papel_goals : roles.values() )
       {
         for(int i = 0 ; i < papel_goals.getChildren().size(); i++)
         {
           Planning planejamento = (Planning)papel_goals.getChildren().toArray()[i];
          if( planejamento.getBehaviourName().equals(myAgent.newService) )
           {
                   myAgent.newRole = papel_goals.getRoleName();
                   atribuiPapelAgente();
            }
          }
        }

        if(myAgent.newRole != null)
              myAgent.newService = null;
        else
        {
         myAgent.inform(this,"Não é possível encontrar um papel que atenda
ao serviço de: " + myAgent.newService);
         for ( Goals papel_goals : roles.values() )
              for(int i = 0 ; i < papel_goals.getChildren().size(); i++)
                    myAgent.inform(this, "Serviços disponíveis: " +
((Planning)papel_goals.getChildren().toArray()[i]).getBehaviourName());

        }
        myAgent.newRole = null;
     }
     else if(myAgent.myRole == null)
     {
        myAgent.myRole = "sou-vagal";

        DFAgentDescription dfd = new DFAgentDescription();
        dfd.setName(myAgent.getAID());
        ServiceDescription sd = new ServiceDescription();

        sd.setType("free");
        sd.setName("free ");
        dfd.addServices(sd);
        try {
            DFService.register(myAgent, dfd);
        }
        catch (FIPAException fe) {
             fe.printStackTrace();
        }
        myAgent.inform(this,"Acabei de me candidatar aos serviços de " +
sd.getName());}
```

Quadro B-2 - Comprometimento com o papel dos agentes propostos

C. Apêndice III – Redirecionamento de planos

As classes envolvidas são demonstradas na figura 10-1 e explicadas a seguir:

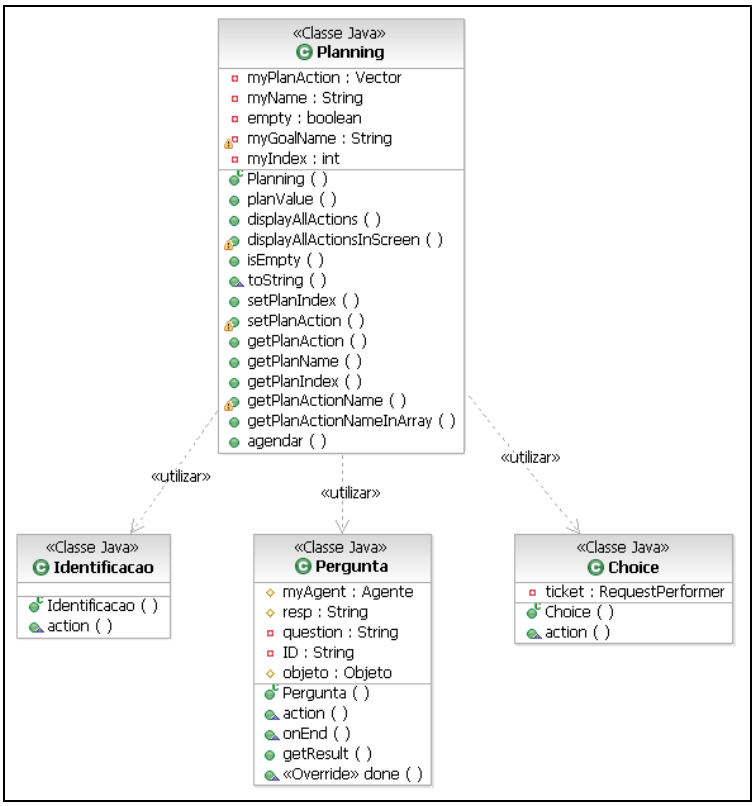


Figura C-1 – A estrutura do mecanismo de racionalização

O “choice” passa a ser um elemento para caracterizar os beliefs, e é capaz de redirecionar a execução do plano. Ou seja, mesmo definindo o plano corretamente, o criador do plano pode transformá-lo em um planejamento criando pontos de redirecionamento onde a resposta do usuário com uma representação lógica é capaz de mudar a execução do plano definido através da seqüência numérica das ações. Para isso, basta utilizar o primeiro qualificador “?”.

Quando um plano é então iniciado para a execução a interface entre a organização de agentes citada utiliza o planejamento criado e atribuído a um objeto do tipo `Goals` e adiciona tudo isso como um comportamento do agente. A classe `Goals` é utilizada como papel_goals pelo `ServerRole`, e possui um registro de `Goals` para cada role no `hashtable` do

`ServerRole`. Os planejamentos (`Planning`) cadastrados anteriormente na criação dos papéis irão integrar os `Goals` do papel para os quais foram determinados, de forma que cada `Goals` (`papel_goals`) terá diversos planejamentos (`Planning`), um para cada serviço prestado.

O plano é alocado no *setup* de cada agente. É uma instância da classe `Planning` que passa a incorporar o `ParalellBehaviour` (`Goal`) juntamente com os sensores e atuadores responsáveis pelo recebimento e envio de mensagens respectivamente (negociação CNET).