UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

UTILIZAÇÃO DA ENGENHARIA DE SOFTWARE ORIENTADA A AGENTES NA MODELAGEM DE UM SISTEMA DE SELEÇÃO DE PESSOAS

TRABALHO DE GRADUAÇÃO

Liane Santiago Cafarate

Santa Maria, RS, Brasil
2008

UTILIZAÇÃO DA ENGENHARIA DE SOFTWARE ORIENTADA A AGENTES NA MODELAGEM DE UM SISTEMA DE SELEÇÃO DE PESSOAS

por

Liane Santiago Cafarate

Trabalho de Graduação apresentado ao curso de Ciência da Computação – Bacharelado, da Universidade Federal de Santa Maria (UFSM, RS) como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Professor Doutor Marcos Cordeiro d'Ornellas

Trabalho de Graduação nº 263
Santa Maria, RS, Brasil
2008

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada, aprova o Trabalho de Graduação

UTILIZAÇÃO DA ENGENHARIA DE SOFTWARE ORIENTADA A AGENTES NA MODELAGEM DE UM SISTEMA DE SELEÇÃO DE PESSOAS

elaborado por
Liane Santiago Cafarate

como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação

Comissão Examinadora

Professor Dr. Marcos Cordeiro d'Ornellas
(Orientador)

Prof. Dra. Roseclea Duarte Medina

Prof. Oni Reasilvia de O. Sichonany

Santa Maria, 25 de julho de 2008.

"Alguns homens vêem as coisas como são e dizem: Por quê?
Outros sonham com as coisas como nunca foram e se perguntam: Por que não?"
Autor desconhecido.

AGRADECIMENTOS

Agradeço antes de tudo a Deus por estar sempre ao meu lado em todos os momentos que precisei.

Agradeço de todo o coração a minha mãe por ter me apoiado em todos os momentos da minha vida, bem como por ter sido um grande exemplo para mim.

Agradeço a minha família por sempre ter acreditado em mim, me auxiliando da forma que puderam.

Agradeço imensamente ao meu namorado e grande amigo, Henrique Vicentini, que sempre esteve comigo, me ajudando toda as vezes que eu solicitei. Com certeza, essa conquista não teria sido a mesma sem ele.

Agradeço ao meu orientador, por acreditar no meu potencial, bem como pelos momentos de conselho e críticas, que me foram bastante úteis na busca de ser uma pessoa melhor.

Agradeço aos meus professores, pelo conhecimento e experiência compartilhada, por todo o suporte concedido.

Agradeço aos meus amigos e colegas, que me acompanharam nessa caminhada.

Agradeço por fim a todos aqueles que de alguma forma me auxiliaram a alcançar essa conquista. A todos, muito obrigada.

RESUMO

Trabalho de Graduação
Ciência da Computação
Universidade Federal de Santa Maria

UTILIZAÇÃO DA ENGENHARIA DE SOFTWARE ORIENTADA A AGENTES NA MODELAGEM DE UM SISTEMA DE SELEÇÃO DE PESSOAS

Autor: Liane Santiago Cafarate
Orientador: Professor Dr. Marcos Cordeiro d'Ornellas
Local e Data de Defesa: Santa Maria, 25 de julho de 2008

Atualmente, os sistemas estão cada vez maiores e mais complexos, tornando seu desenvolvimento uma atividade difícil. Por isso, novos paradigmas de desenvolvimento de software surgem, buscando solucionar esse problema. Um deles é o paradigma orientado a agentes. Um agente é uma entidade de software que está situada em um ambiente dinâmico, aberto e imprevisível e é capaz de ação autônoma neste ambiente afim de atingir seus objetivos de projeto (WOOLDRIDGE, 2002). Uma variedade de aplicações podem ser caracterizadas dessa forma. Assim, esse paradigma representa um novo meio de analisar, projetar e construir sistemas de softwares complexos, pois oferece um mecanismo que permite um comportamento emergente ao invés de uma arquitetura estática, preocupando-se em mapear as interações entre entidades, da mesma forma que ocorre em nossas vidas. Nesse sentido, várias metodologias têm surgido com o propósito de auxiliar no desenvolvimento de sistemas multi-agentes. Uma delas, a metodologia Tropos, é uma técnica centrada na análise de requisitos que busca englobar as principais fases de desenvolvimento de um software, preocupando-se com as características específicas que sistemas multi-agentes possuem. Dessa forma, o objetivo deste trabalho é aplicar a metodologia Tropos, na modelagem de um sistema multi-agentes para selecionar pessoas para vagas disponíveis dentro de empresas, destacando também a importância de selecionar pessoas adequadas para as organizações e apresentando considerações finais a respeito da aplicação da metodologia.

Palavras-chave: engenharia de software, sistemas multi-agentes, gestão de pessoas, seleção de pessoas, metodologia tropos, agentes

ABSTRACT

Trabalho de Graduação
Computer Science
Universidade Federal de Santa Maria

AGENT-ORIENTED SOFTWARE ENGINEERING APPLICATION IN A PEOPLE SELECTION SYSTEM MODELLING

Author: Liane Santiago Cafarate
Advisor: Prof. Dr. Marcos Cordeiro d'Ornellas
Place and Date of Presentation: Santa Maria, July the 25th, 2008

Nowadays, systems are bigger and more complex, transforming their development in a difficult task. Therefore, new software development paradigms emerge, intending to solve that problem. One of them is the agent-oriented paradigm. An agent is a software entity that is situated in a dinamic, open and unpredictable environment and that is capable of autonomous action in this environment in order to meet its design objectives (WOOLDRIDGE, 2002). A variety of applications can be qualified in this way. Hence, this paradigm represent a new way to analyse, project and build complex software systems, because it offers a mechanism that allows an emergent behaviour instead of a static architecture, concerning about mapping the interactions among entities, the same way that happens in our lives. In this context, several methodologies have appeared in order to help in the multi-agents development systems. One of them, Tropos methodology, is a technique focus on requirement analysis, pursues to include the main software development phases, concerning about the specific characteristics that multi-agents systems own. So, the goal of this essay is model a multi-agents system to select qualified people for the companies using Tropos methodology, stressing also the importance of selecting appropriate people for organizations and presenting final considerations about the application of the methodology.

Keywords: software engineering, multi-agents systems, people management, people selection, tropos methodology, agents

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API – *Application Programming Interface*

AUML – *Agent Unified Modeling Language*

BDI - *Belief-Desire-Intention*

FIPA - *Foundation for Intelligent Physical Agents*

GUI – *Graphical User Interface*

JADE - *Java Agent Development Framework*

RH – *Recursos Humanos*

UML - *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Existem grandes dificuldades no processo de desenvolvimento de um software. Diversas variáveis contribuem na complexidade desse processo, tais como: as estimativas de prazo e de custo freqüentemente são imprecisas; a produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços; a qualidade de software às vezes é menor que a adequada, ocorrendo muito freqüentemente a insatisfação do usuário (PRESSMAN, 1995); entre outros. Atualmente, o hardware deixou de ser o item mais caro na implementação de um sistema, enquanto que o custo relacionado ao software cresceu e se tornou o principal item no orçamento de computação (COSTA, 2001), sendo que estudos demonstram que maior parte dos gastos incorridos durante o ciclo de vida de um software ocorrem durante o processo de manutenção (PIGOSKI, 1997).

Além dessas dificuldades, a complexidade de desenvolvimento de sistemas distribuídos, abertos, dinâmicos, não determinísticos, heterogêneos e com alta interatividade criou a necessidade de novos paradigmas de desenvolvimento que permitam lidar com essas características.

Nesse contexto, o interesse em torno da abordagem orientada a agentes no desenvolvimento de sistemas complexos tem aumentado nos últimos anos (SILVA, 2005). Há diversos motivos para esse interesse, mas o mais importante relaciona-se ao conceito de que um agente, como sistema autônomo, é capaz de interagir com outros sistemas afim de alcançar seus objetivos de projetos, em um ambiente dinâmico, aberto e imprevisível, sendo a forma natural de projetar um software (WOOLDRIDGE, CIANCARINI, 2001).

Assim, surgiram vários metodologias para aplicar a engenharia de software orientada a agentes, tais como, GAIA, TROPOS, AUML, entre outras. Essas metodologias oferecem um nível mais alto de abstração na forma de pensar sobre as características e os comportamentos dos sistemas de software.

Dentre essas metodologias, a metodologia TROPOS visa modelar os sistemas em termos de atores, responsabilidades, tarefas, ao invés de uma forma orientada a implementação (em termos de objetos, estruturas de dados, interfaces). Essa defasagem é um

dos principais fatores para a má qualidade dos softwares e também fracassos nos projetos de desenvolvimento. Assim, a modelagem TROPOS (MYLOPOULOS, CASTRO, 2000) facilita a visão estratégica e intencional dos processos e possibilita um melhor entendimento dos relacionamentos organizacionais e das razões das decisões tomadas.

Entretanto, embora a tecnologia de agentes tenha um grande potencial para se tornar uma solução de engenharia de software amplamente utilizada, o sucesso de sua adoção depende da familiarização e confiança dos desenvolvedores de software, bem como, as organizações em relação a noção de agentes de software.

Ciente dessa realidade, o objetivo desse trabalho é modelar um sistema de auxílio a área de gestão de pessoas, selecionando candidatos a um determinado posto de trabalho. Essa modelagem utilizará o paradigma orientado a agentes, com o uso da metodologia TROPOS. A escolha do tema da modelagem voltado para um sistema que auxilie no processo de recrutamento e seleção de pessoas é devido a importância da escolha da pessoa mais adequada a organização ser impactante no alcance dos objetivos da organização, além da redução dos custos deste processo, que costuma ser relativamente caro.

Esse trabalho está estruturado da seguinte forma: o capítulo 2 compreende o referencial teórico da área de gestão de pessoas; o capítulo 3 apresenta todo o embasamento teórico sobre agentes e sistemas multi-agentes, bem como um breve resumo sobre algumas metodologias orientadas a agente existentes; o capítulo 4 refere-se ao uso da metodologia Tropos na modelagem do sistema em questão; no capítulo 5 apresenta-se as conclusões obtidas durante o desenvolvimento desse trabalho, assim como possíveis trabalhos futuros; por fim, no capítulo 6, apresenta-se as referências bibliográficas utilizadas.

2 GESTÃO DE PESSOAS

O mundo dos negócios é composto por mudanças aceleradas, complexas e significativas em todas as dimensões do ambiente humano-social, econômica, cultural, política religiosa, entre outras. Um fator responsável por essas mudanças é o avanço tecnológico, que modificou esse meio em uma velocidade antes nunca vista, provocando impactos profundos na sociedade como um todo, e, por conseguinte, na gestão das empresas (BRAGA, 2007).

Esse cenário competitivo exige das empresas adaptações rápidas e contínuas para sua sobrevivência e crescimento. Essas adaptações estão fortementes ligadas à tecnologia. Nesse novo ambiente, a tecnologia surge como um fator potencializador e indispensável para o desenvolvimento dessas organizações.

As pessoas e as organizações estão engajadas em uma complexa e incessante interação: as pessoas passam maior parte do tempo nas organizações das quais dependem para viver e as organizações são constituídas de pessoas sem as quais não poderiam existir (CHIAVENATO, 1999).

Nesse novo contexto, a competição entre organizações reflete na concorrência por profissionais que agreguem condições determinantes no diferencial de qualidade das empresas (LIMMONGI-FRANÇA, ARELLANO, 2002).

A eficácia organizacional depende do alcance dos objetivos, por isso é tão importante a área de gestão de pessoas (CHIAVENATO, 1999) e selecionar a pessoa mais adequada para aquela organização.

Na era da informação, as pessoas têm sido consideradas de suma importância nas organizações. É através de suas habilidades, conhecimentos e comportamentos que as empresas têm alcançado um diferencial no mercado competitivo.

Por essa razão, a área de Gestão de Pessoas têm recebido mais destaque e sido responsável por organizações bem sucedidas (CHIAVENATO, 2002). Organizações do mundo inteiro investem para encontrar pessoas qualificadas. Tudo isso porque na era do conhecimento, os bem preparados podem fazer a diferença no rumo ao sucesso. Com o

desenvolvimento tecnológico, a globalização dos negócios, o impacto das mudanças, a busca incansável pela qualidade e produtividade, percebe-se que os colaboradores são um bem intangível das empresas.

Para o sucesso de uma organização é necessário formar e consolidar equipes produtivas e comprometidas com a estratégia e as metas da empresa (KIMURA, 2006). Por isso é tão importante selecionar adequadamente seus colaboradores.

2.1 Provisão de Recursos Humanos

Os processos de provisão de recursos humanos são aqueles que se destinam ao suprimento de pessoas para uma organização. Eles envolvem todas as atividades relacionadas com pesquisa de mercado, recrutamento e seleção de pessoas, bem como sua integração às tarefas organizacionais.

2.2 Recrutamento de Pessoas

Recrutamento é o conjunto de técnicas e procedimentos que visa atrair candidatos potencialmente classificados e capazes de ocupar cargos dentro da organização (CHIAVENATO, 2004).

Segundo (DUTRA, 2002), a captação de pessoas pressupõe a consciência da empresa em relação as suas necessidades. Somente dessa forma, é possível saber quem procurar, onde procurar e que tipo de relação será estabelecida entre a pessoa e a empresa. Esse processo não deve ser encarado só no sentido das empresas para as pessoas, deve ser considerado também o inverso.

O recrutamento é realizado a partir das necessidades presentes e futuras de recursos humanos da organização. Consiste na pesquisa e intervenção sobre as fontes capazes de fornecer à organização um número suficiente de pessoas necessárias à consecução dos seus objetivos.

Esse processo necessita de um planejamento apropriado, que constitui na seqüência de três fases:

- Pesquisa interna de necessidades: O que a organização precisa em termos de pessoas;
- Pesquisa externa de mercado: O que o mercado de RH pode oferecer;
- Técnicas de recrutamento: Definição de quais técnicas de recrutamento serão aplicadas.

2.2.1 Pesquisa Interna das Necessidades

É uma verificação das necessidades da organização em relação às suas carências de recursos humanos, a curto, médio e longo prazo. Esse processo não deve ocorrer ocasionalmente, mas constantemente e deve englobar todas as áreas e níveis da empresa.

Para alcançar os objetivos organizacionais, a organização necessita de pessoas capazes de executar os trabalhos que lhes foram destinados. Isso significa que os administradores tem que estar seguros de que os cargos sob sua responsabilidade estão sendo ocupados por pessoas adequadas. Para isso, é necessário um planejamento pessoal. Existem vários modelos de planejamento, a saber:

- Modelo baseado na procura estimada do produto ou serviço: A quantidade de pessoas a ser contratada depende da demanda por produtos e/ou serviços. Esse modelo utiliza previsões ou extrapolações de dados históricos e está voltado para o nível operacional da organização.
- Modelo baseado em segmentos de cargos: Necessidades totais de pessoal calculadas com base em projeções relacionadas apenas com certos segmentos de cargos de sua força de trabalho que mais apresentem variações.
- Modelo de substituição de postos-chaves: Representação visual de quem substitui quem na eventualidade de alguma possível vaga futura dentro da organização. Essas informações devem provir do sistema de informações gerencial.
- Modelo baseado no fluxo de pessoal: modelo caracteriza o fluxo de pessoas para dentro e para fora da organização. A verificação histórica e o acompanhamento

desse fluxo de entradas, saídas, promoções e transferências internas permitem uma predição a curto prazo das necessidades de pessoal da organização.

- Modelo de planejamento integrado: Modelo mais amplo e abrangente. Considera 4 fatores de influência:
  - Volume de produção planejado;
  - Mudanças tecnológicas que alterem a produtividade do pessoal;
  - Demanda do mercado;
  - Planejamento de carreiras dentro da organização[1];

2.2.2 Pesquisa Externa do Mercado

É uma pesquisa do mercado de RH para segmentá-lo e diferenciá-lo para facilitar sua análise e conseqüente abordagem. Nesse sentido, dois fatores devem ser levados em consideração: a segmentação do mercado de RH e a localização das fontes de recrutamento.

A segmentação refere-se à decomposição do mercado em classes de candidatos com características definidas para atingí-lo de maneira específica. Essa segmentação é feita de acordo com os objetivos da organização.

2.2.3 Técnicas de Recrutamento

O processo de recrutamento varia conforme a organização. Entretanto, de acordo com as diferentes fontes de recursos humanos, existem dois meios de recrutar pessoas, a saber:

- Recrutamento Externo: quando aborda candidatos reais (que estão procurando emprego ou desejam mudar de emprego), potenciais (não estão interessados em procurar emprego), aplicados (trabalhando em alguma empresa) ou disponíveis

1 Planejamento de carreiras: é o conjunto de normas que disciplinam o ingresso e instituem oportunidades e estímulos ao desenvolvimento pessoal e profissional dos trabalhadores de forma a contribuir com a qualificação dos serviços prestados pelos órgãos e instituições, constituindo-se em instrumento de gestão da política de pessoal (FIOCRUZ, 2006).

(desempregados). Sua conseqüência é uma entrada de recursos humanos na empresa.

- Recrutamento Interno: Aborda candidatos reais ou potenciais aplicados unicamente na própria empresa e sua conseqüência é o processo interno de recursos humanos.

2.2.3.1 Recrutamento Externo

Ocorre quando a empresa busca preencher uma falta de recursos humanos com candidatos vindo de fora da empresa. Havendo uma vaga a empresa busca candidatos externos atraídos pelas técnicas de recrutamento (arquivos de candidatos, cartazes, meios de comunicação, etc.).

A empresa pode contatar as fontes de recrutamento de forma direta ou indireta. Na forma indireta, a empresa contata órgãos que recrutam pessoas, como por exemplo, agências, sindicatos, entre outros.

As vantagens desse tipo de recrutamento são:

- Traz pessoas novas, com experiências diferentes para a organização;
- Renova e enriquece os recursos humanos da organização;
- Aproveita os investimentos em treinamento e desenvolvimento feitos por outras empresas ou pelos próprios candidatos.

As desvantagens desse meio de captação são:

- Processo geralmente mais demorado que o recrutamento interno;
- É mais caro e exige inversões e despesas imediatas para a captação de pessoas;
- Em princípio, é menos seguro que o recrutamento interno, já que os candidatos são "desconhecidos";
- Os funcionários podem ver esse tipo de recrutamento como "deslealdade" da empresa com seu pessoal;
- Geralmente afeta a política salarial da empresa e influencia as faixas salariais internas.

2.2.3.2 Recrutamento Interno

Ocorre quando a empresa busca preencher uma falta de recursos humanos com seu próprio contingente de colaboradores. Esses colaboradores podem ser promovidos (movimentação vertical), transferidos (movimentação horizontal) ou ainda transferidos com promoção (movimentação diagonal).

O recrutamento interno pode envolver diversos fatores. São eles:

- Transferências de pessoal;
- Promoções de pessoal;
- Transferências com promoções de pessoal;
- Programas de desenvolvimento de pessoal;
- Planos de carreira de pessoal.

Esse tipo de recrutamento exige intensa coordenação do órgão de RH com os demais setores da empresa.

As principais vantagens desse tipo de captação de colaboradores são:

- Mais econômico;
- Mais rápido;
- Maior índice de validade e de segurança;
- Fonte de motivação para os colaboradores;
- Aproveita os investimentos da empresa em treinamento do pessoal;
- Desenvolve espírito sadio de competição entre o pessoal.

Já as principais desvantagens são:

- Exige que os novos empregados tenham potencial de desenvolvimento alguns níveis acima do cargo e motivação suficiente para chegar lá, sendo necessário que a organização ofereça oportunidades para isso ser desenvolvido no momento adequado. Caso contrário, pode ocorrer desmotivação dos funcionários, ou até mesmo, desligamento a fim de procurar oportunidades em outras empresas.
- Pode gerar conflitos de interesses, pois ao oferecer oportunidades de crescimento cria uma atitude negativa nos empregados que não demonstram condições, ou não realizam aquelas oportunidades.
- Quando administrado incorretamente pode ocorrer que um empregado seja

promovido até o máximo de sua competência, estacionando numa posição por não ter capacidade de evoluir.

- Quando efetuado continuamente, leva os empregados a uma progressiva limitação às políticas e diretrizes da organização.
- Não pode ser feito em termos globais da organização. Para não prejudicar o capital humano, o recrutamento interno deve ser feito somente quando os candidatos internos tenham condições de se igualar aos externos.

2.2.3.3 Recrutamento Misto

Na prática, as empresas fazem recrutamento externo e interno ao mesmo tempo, já que os processos se completam e complementam. Quando uma vaga é liberada na empresa e preenchida por algum colaborador, esse deslocamento produz a abertura de outra vaga a ser preenchida pelo recrutamento externo, a não ser que essa vaga seja cancelada. Dessa forma ameniza as vantagens e desvantagens de ambos os processos.

O recrutamento misto pode ser adotado de três formas:

- Inicialmente recrutamento externo seguido de interno caso o primeiro não apresente os resultados desejados. A empresa está mais interessada em candidatos preparados, não busca a transformação do candidato.
- Inicialmente recrutamento interno seguido de externo caso o primeiro não apresente os resultados desejados: A empresa dá prioridade aos seus empregados na disputa das vagas. Não havendo candidatos capacitados, busca no mercado externo.
- Recrutamento externo e interno em conjunto: A empresa está mais preocupada com o preenchimento da vaga existente, não importa o meio. Uma boa política é dar preferência para os candidatos internos, em casos de igualdade entre eles. Com isso a empresa assegura que está com um capital humano de qualidade e ao mesmo tempo motiva os seus funcionários.

2.3 Seleção de Pessoas

A seleção de pessoas faz parte da provisão de recursos humanos para a empresa, mas, diferente do recrutamento, esse processo é uma atividade de escolha, opção, decisão, de filtragem e classificação, portanto restritiva (CHIAVENATO, 2004).

Nessa etapa, o objetivo principal é selecionar dentre os candidatos que possuem os requisitos mínimos preenchidos atraídos pelo recrutamento, aqueles que tenham maiores probabilidades de se ajustar ao cargo vago e desempenhá-lo bem, isto é, aqueles que se adequam as necessidades da organização.

O processo seletivo baseia-se em dados e informações das análises e especificações do cargo a ser preenchido. Essa etapa é um processo de comparação, informando os requisitos indispensáveis ao futuro ocupante do cargo. Por outro lado, como há candidatos diferentes disputando a vaga, a segunda etapa do processo envolve processo de decisão, escolhendo dentre os candidatos que preenchem os requisitos, aquele que melhor se adequa ao cargo e o executa com melhor eficiência e eficácia.

2.3.1 Seleção como Processo de Comparação

O processo de seleção compara duas variáveis: os requisitos do cargo a ser preenchido e o perfil das características dos candidatos que se apresentam. Essa comparação pode ser melhor vista na figura 2.1. A primeira variável é fornecida pela descrição e análise do cargo, enquanto a segunda é obtida por meio de aplicação das técnicas de seleção.

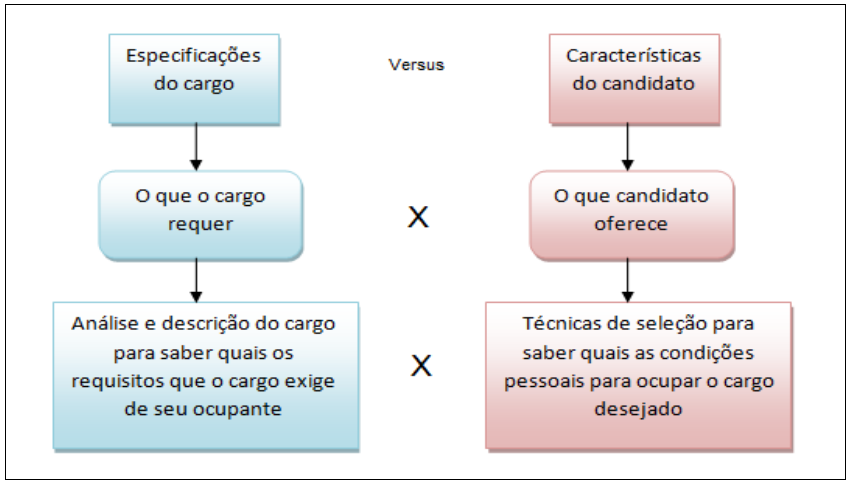

Figura 2.1: Seleção de Pessoal como Comparação

Quando as especificações do cargo são maiores que as características do candidato, esse candidato não está apto a ocupar essa vaga. Quando as variáveis são iguais, o candidato reúne as condições ideais para o cargo. Já quando as características do candidato são superiores as especificações do cargo, dada uma faixa de aceitação, o possível ocupante é superdotado para esse cargo.

2.3.2 Seleção como Processo de Decisão

Uma vez feita a comparação entre as especificações do cargo e as características do candidado, pode ser que vários destes estejam aptos a ocupar o cargo vago. A decisão final de quem ocupará a vaga disponível é uma responsabilidade de linha (de cada chefe) e função de *staff* .

Como um processo de decisão, a seleção do pessoal comporta três modelos de comportamento:

- Modelo de Colocação: Quando não inclui a categoria de rejeição, isto é, há só um candidato para uma vaga.
- Modelo de Seleção: Quando existem vários candidatos para preencher uma vaga.
- Modelo de Classificação: Quando existem vários candidatos para cada vaga e várias vagas para cada candidato. Esse modelo se fundamenta em um conceito

mais amplo, onde a organização considera que um candidato será alocado na vaga que mais se adequa as suas características pessoais. Esse modelo é superior aos demais no quesito aproveitamento dos candidatos, à eficiência dos processos (por envolver a totalidade de cargos vacantes a serem preenchidos) e a redução dos custos envolvidos (por evitar duplicidade de despesas com o processo).

Um breve resumo desses modelos é apresentado na figura 2.2.

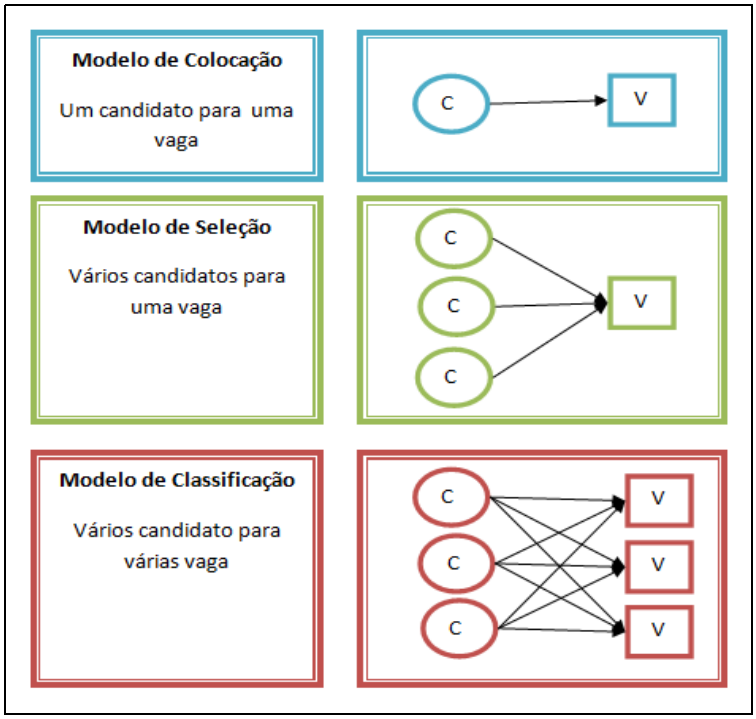

Figura 2.2: Modelos de colocação, seleção e classificação dos candidatos

2.4 Sistemas de Informação de Recursos Humanos

A área de recursos humanos têm sofrido grandes modificações nos últimos tempos. Há pouco tempo atrás o foco dos administradores dessa área estava em realizar atividades burocráticas e de controle, atualmente, a gestão de pessoas se tornou um diferencial estratégico dentro das empresas.

Dessa forma, o foco desse departamento amplia-se. Além de continuar se preocupando

com questões operacionais, o setor de recursos humanos passa a ser responsável por auxiliar as organizações a atrair, reter e desenvolver pessoas que constituem seus negócios.

A área de recursos humanos, com foco estratégico, necessita de informações dos profissionais e do conhecimento gerado pela empresa. Nesse caso, os sistemas de informação ganham importância fundamental, auxiliando como base para o conhecimento, decisão e gerenciamento efetivo das pessoas que participam do negócio.

Um sistema de informação é um conjunto de elementos interdependentes (subsistemas) logicamente associados, para que de sua interação sejam geradas informações necessárias à tomada de decisões (CAUTELA, POLLONI, 1976).

Em virtude da grande importância dos sistemas de informação, os negócios precisam garantir que melhorias ou sistemas auxiliem a reduzir custos, aumentar lucros, melhorar serviços e/ou obter vantagem competitiva (STAIR, REYNOLDS, 2006).

A montagem do sistema de informação de recursos humanos requer análise e avaliação da organização ou de seus subsistemas e de suas respectivas necessidades de informação (CHIAVENATO, 2004).

Um exemplo de uso de um sistema de informação para recursos humanos é ter as habilidades de cada colaborador armazenadas nesse sistema. Assim, caso o gestor necessite de uma habilidade específica, ele pode consultar primeiramente a base de dados da empresa e verificar se já existe alguém com essa característica, possibilitando a tomada de decisão de forma mais ágil.

Os sistemas de informação de recursos humanos, por lidarem com o capital intelectual da empresa, têm papel preponderante para a consolidação das tendências de um mundo globalizado e que invariavelmente trarão conseqüências decisivas de como será realizada a gestão dos negócios empresariais futuramente.

3 AGENTES

O conceito de agente não é padrão. A definição a ser adotada nesse trabalho é a seguinte: “Um agente é uma entidade de software que está situada em algum ambiente e é capaz de ação autônoma neste ambiente afim de atingir seus objetivos de projeto” (WOOLDRIDGE, 2002). Um agente apresenta diversas propriedades que os diferenciam das outras aplicações de software. São algumas delas:

- **Autonomia:** capacidade de agir sem intervenção externa direta, baseando suas ações não só no seu conhecimento embutido, mas também em seu conhecimento sobre o ambiente (percepções).
- **Reatividade:** capacidade de perceber o ambiente em que está inserido e reagir as mudanças desse ambiente. A reatividade do agente pode ser melhor vista na figura 3.1, onde um agente recebe estímulos do ambiente e gera ações que afetam esse ambiente de uma forma contínua e progressiva.

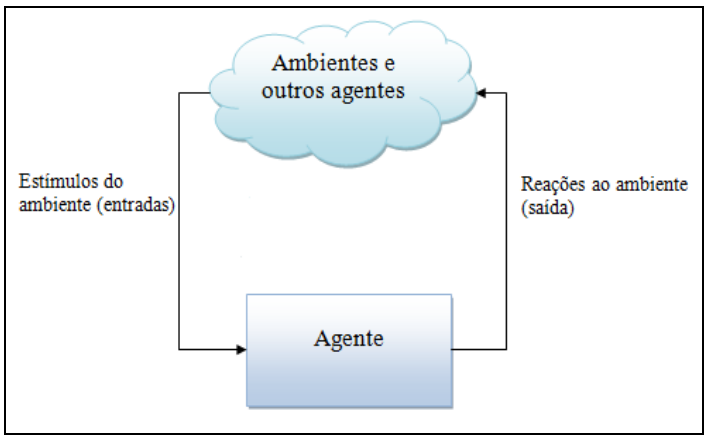

Figura 3.1: Reatividade do Agente

- **Pró-atividade:** apresentação de um comportamento orientado à objetivos, tomando iniciativa quando julgar necessário.
- **Habilidade social:** capacidade de interagir com outros agentes (e humanos), competindo, coordenando ou cooperando, para resolver problemas ou quando lhe for conveniente.

- **Aprendizagem ou capacidade de adaptação:** mudança de comportamento de acordo com experiências prévias. Isso inclui aprendizado de novas situações e não cometer os mesmos erros.
- **Mobilidade:** capacidade de um agente se mover de um ambiente (máquina) para outro.
- **Persistência ou continuidade temporal:** capacidade do agente de manter um estado interno conciso através do tempo, isto é, o agente está continuamente executando um processo.

Para ser um agente, nem todas as propriedades citadas acima precisam estar presentes, mas quanto mais propriedades o agente possuir, em geral, maior será seu grau de inteligência, que também definirá a qual classificação o agente pertence, que pode ser dividida entre a noção forte e fraca de agência. Na noção fraca de agência, os agentes têm vontade própria (autonomia), são capazes de interagir uns com os outros (habilidade social), respondem a estímulos (reatividade) e tomam iniciativa (pró-atividade). Na noção forte de agência, as noções fraca de agência são preservadas com a adição de que os agentes podem se mover de um ambiente para outro (mobilidade), são confiáveis (veracidade), fazem o que é dito para ser feito (benevolência) e operam de maneira ótima para atingir suas metas (racionalidade) (TVEIT, 2001).

3.1 Sistemas Multi-agentes

Um sistema multi-agente é um sistema baseado na idéia na qual um ambiente de trabalho cooperativo incluindo componentes de softwares sinérgicos[2] pode lidar com problemas que seriam difíceis de serem resolvidos usando uma abordagem de computação tradicional centralizada. Agentes de software com as capacidades de autonomia, reatividade, pró-atividade e habilidade social interagem de um modo dinâmico e flexível para resolver problemas de forma mais eficiente (CERVENKA, TRENCANSKY, 2000) da mesma forma que as pessoas fazem em suas vidas (WOOLDRIDGE, 2002).

2 Sinergia é o trabalhou ou esforço coordenado de vários subsistemas na realização de uma tarefa complexa. Fonte: (WIKCIONÁRIO, 2008).

A tecnologia de sistemas multi-agentes está emergindo como uma das mais importantes no desenvolvimento de software. Pesquisadores estão desenvolvendo um número de aplicações baseadas em agentes e sistemas multi-agentes em diversas áreas, como comércio eletrônico, alocação de recursos, controle industrial, sistemas de suporte à decisão, trabalho colaborativo, jogos de computadores, entre outros (SUGUMARAN, 2007).

Existem várias razões que justificam o interesse do uso do paradigma orientado a agentes no desenvolvimento de softwares. São eles (JENNINGS, WOOLDRIDGE, 1998):

- **Sistemas abertos:** um sistema aberto é aquele no qual sua estrutura é capaz de mudar dinamicamente. Disso decorre a dificuldade de afirmar em tempo de projeto quais componentes farão parte do sistema e como eles irão interagir entre si. Um exemplo de sistema aberto é a internet, que se expande em tamanho e complexidade. Para operarem efetivamente, esses sistemas devem ser capazes de decidir de forma autônoma e flexível.
- **Distribuição de dados e controle:** para muitos sistemas, o controle global é realizado em diversos nós de computação, geralmente distribuídos geograficamente. Com o intuito de trabalharem em conjunto alcançando maior eficiência, esses nós devem possuir a habilidade de interagir autonomamente entre eles, ou seja, serem agentes.
- **Agentes como uma metáfora natural:** muitos ambientes (incluindo a maioria das organizações) são naturalmente modeladas como sociedades de agentes, seja cooperando para solucionar problemas ou competindo entre si.
- **Sistemas legados:** uma maneira natural em incorporar sistemas legados em modernos sistemas de informação distribuídos é envolvendo-os em uma camada de agente, permitindo a interação deles com outros agentes.

Outros motivos incentivam o uso de sistemas multi-agentes ao invés de aplicações compostas por um único agente. Esses motivos visam obter um software mais robusto e eficiente (ODELL, 2000). São eles:

- Existe a possibilidade de um só agente ser construído para fazer tudo (onipotente), mas esse agente atingiria uma alta complexidade, representando gargalo em velocidade, manutenibilidade, confiabilidade, etc. Dividir as funcionalidades do sistema entre vários agentes permite uma aplicação modular, flexível, modificável e extensível.

- O conhecimento especializado geralmente não é disponível em apenas um único agente (onisciência). O conhecimento sendo distribuído em vários agentes pode ser integrado para uma maior completude quando for necessária flexibilidade.

3.2 Áreas de Aplicações de Sistemas Multi-agentes

Devido aos fatores citados no tópico anterior, a tecnologia está começando a ser aplicada na resolução de problemas no mundo real. Algumas áreas onde a abordagem de sistemas orientados a agentes é utilizada para o desenvolvimento de aplicações estão citadas abaixo.

Aplicações Industriais:

- Fabricação: projeto e configuração de produtos de fabricação, projeto colaborativo, cronograma e controle de operações de fabricação de um robô e determinação de seqüências de produção para uma fábrica;
- Telecomunicações: controle de rede, transmissão e chaveamento, gerência de serviço e gerência de rede;
- Controle de tráfego aéreo: agentes utilizados para representar os equipamentos de aviação, bem como diversos sistemas de controle de tráfego aéreo em operação;
- Sistemas de transporte: domínio de gerência de tráfego e transporte adequado para uma abordagem baseada em agentes devido a sua natureza geograficamente distribuída.

Aplicações Comerciais:

- Gerência de informação: mecanismos de busca realizados por agentes, onde essas entidades agem autonomamente para buscar em uma base de dados em benefício de algum usuário. Essa é uma das áreas mais utilizadas nas aplicações de sistemas multi-agentes;
- Comércio eletrônico: agentes têm sido largamente utilizados para automatizar estágios de uma compra na internet, por exemplo, ajudando o usuário a decidir o que comprar, encontrar especificações sobre o produto, apresentar novidades, procurar por ofertas especiais e descontos, entre outros.

- Gerência de processo de negócio: há um grande interesse de empresas em desenvolver sistemas de informação de suporte a gestão nos negócios, no apoio à decisões.

Aplicações de entretenimento:

- Existe um amplo campo de uso de sistemas multi-agentes no desenvolvimento de jogos de computador, cinema interativo e realidade virtual. Para construir um mundo simulado, pode ser criado um agente que provê ilusão de vida. Nesse tipo de aplicação existem vários personagens que podem ser facilmente implementados como agentes.

Aplicações médicas:

- Aplicações de monitoramento de pacientes, plano de saúde, entre outras, podem ser desenvolvidas com sistemas multi-agentes.

3.3 Desafios e Obstáculos no Desenvolvimento de Softwares Orientados a Agentes

Embora haja várias vantagens no desenvolvimento de software orientado a agentes, existem também diversos obstáculos nesse processo. Por exemplo, o fato de que os agentes têm de agir em busca de seus objetivos enquanto mantém uma interação contínua com seu ambiente dificulta o projeto de um software capaz de manter equilíbrio entre os comportamentos pró-ativo e reativo. Para atingir este equilíbrio é necessário que a tomada de decisão seja sensível ao contexto, resultando num grau significante de imprevisibilidade sobre quais objetivos o agente perseguirá, em quais circunstâncias e quais os métodos que serão usados para atingir os objetivos escolhidos.

Como os agentes são autônomos, o seu comportamento e os efeitos de suas ações são imprevisíveis. Por isso, certas interações podem não estar determinadas no início do projeto de um sistema multi-agente. E ainda, como sistema multi-agente, podem ocorrer comportamentos que não são exclusivamente de um único agente, e sim resultado da interação dos agentes do sistema, gerando imprevisibilidade no projeto desse tipo de sistema.

Além de problemas específicos do paradigma orientado a agentes, também existem

vários detalhes a serem considerados no desenvolvimento utilizando esse tipo de abordagem (WOOLDRIDGE, 2002):

- Solução orientada a agentes não é universal: Existem problemas onde outros paradigmas de desenvolvimento de softwares são mais apropriados.
- Acreditar que usar agentes é a única solução (*silver bullet*): a tecnologia de agentes é um paradigma de software novo, emergente e ainda não testado na sua essência. Portanto, ela não pode ser considerada uma solução "mágica". Há bons argumentos que indicam que a tecnologia de agentes levará a melhorias no desenvolvimento de software distribuído complexo, mas estes argumentos ainda não foram quantitativamente comprovados.
- Esquecer que está desenvolvendo software: A falta de técnicas testadas e confiáveis para auxiliar o desenvolvimento desse tipo de abordagem pode fazer os projetistas esquecerem que estão desenvolvendo software. Assim, falhas de projetos podem ser atribuídas pela falta de uso de engenharia de software, e não pelas particularidades dos agentes.
- Entusiasmo excessivo em soluções de agente ou falhar em entender onde agentes podem ser aplicados adequadamente: a tecnologia de agentes representa uma inovação e uma importante maneira de conceituar e implementar software, mas é importante entender suas limitações.
- Esquecer que está desenvolvendo software com múltiplas linhas de controle (*multithreaded*): Por natureza, sistemas multi-agentes tendem a ter múltiplas linhas de controle (tanto num agente quanto numa sociedade de agentes). Assim, na construção de software multi-agentes, é preciso lidar com problemas inerentes aos sistemas concorrentes e distribuídos, tais como sincronização, exclusão mútua para recursos compartilhados e *deadlock*, pois eles não desaparecem apenas porque foi adotada uma abordagem baseada em agentes.
- Não saber porque necessita de uma solução orientada a agentes: pode não haver entendimento de como agentes podem ser usados para melhorar seus produtos nem como eles podem capacitá-los a gerar novas linhas de produto, etc.
- Querer construir soluções genéricas para problemas exclusivos: tipicamente se manifesta no planejamento de uma arquitetura que supostamente possibilita que uma classe completa de sistemas seja construída, quando o que realmente é

solicitado é uma solução feita sob encomenda direcionada para um único problema.

- Criar sua própria arquitetura de agentes: quando se inicia um projeto multi-agentes, tende-se a imaginar que nenhuma arquitetura de agentes existente atende aos requisitos do seu problema. Portanto, há uma tentação para projetar uma arquitetura a partir do zero. Isto é freqüentemente um erro, pois deve-se analisar as várias arquiteturas descritas na literatura antes de construir uma própria.
- Os agentes usam excessiva Inteligência Artificial: ao se focar demais em aspectos de "inteligência", o *framework* de construção de um agente fica muito sobrecarregado com as técnicas de inteligência artificial. Para ser útil sugere-se construir agentes com um mínimo dessas técnicas.
- O projeto não explora concorrência: se há a necessidade de uma única linha de controle num sistema, então a utilidade de uma solução baseada em agentes deve ser seriamente questionada.
- Ver agentes em qualquer lugar: quando se adota este ponto de vista, a tendência é utilizar agentes para tudo. Porém, a sobrecarga gerada para gerenciar agentes e a comunicação entre agentes pesará mais do que os benefícios de uma solução baseada em agentes.
- Ter pouquíssimos agentes: alguns projetistas criam um sistema que completamente falha em explorar o poder oferecido pelo paradigma de agente e desenvolvem uma solução com um número muito pequeno de agentes fazendo todo o trabalho. Tais soluções tendem a não atender o teste de coerência da engenharia de software, que requer que um módulo de software tenha uma única função coerente, isto é, escrever várias funcionalidades em uma única classe pode ser feito, mas o resultado não é bonito.

Além das dificuldades citadas acima, existem obstáculos chave que devem ser superados para que a engenharia de software orientada a agentes se torne comum:

- Organização da relação entre agentes e outros paradigmas: não está claro como o desenvolvimento de sistemas multi-agentes irá coexistir com outros paradigmas de software, tal como o desenvolvimento orientado a objetos.
- Metodologias orientadas a agentes: embora várias metodologias preliminares de análise e projeto orientados a agentes estejam sendo propostas, há

comparativamente pouco consenso entre elas e nenhum acordo sobre os tipos de conceitos que a metodologia deveria suportar.

- Engenharia para sistemas abertos: precisa-se de um melhor entendimento de como fazer engenharia em sistemas abertos. Em tais sistemas, acredita-se que é essencial ser capaz de reagir a eventos imprevistos, explorar oportunidades onde estes eventos surgem, e conseguir dinamicamente acordos com componentes do sistema cuja presença não poderia ser prevista em tempo de projeto.
- Engenharia de escalabilidade: precisa-se de um melhor entendimento de como fazer engenharia de forma segura e previsível em sistemas que consistem de um grande número de agentes interagindo dinamicamente uns com os outros a fim de atingir suas metas.
- Rastreamento de requisitos: a inclusão dessa atividade no processo de desenvolvimento orientado a agentes proporcionará muitos benefícios, pois não apenas permitirá um melhor atendimento aos requisitos do cliente, como também apoiará a avaliação do impacto de mudanças solicitadas (mesmo antes das suas implementações no sistema). Como resultado, ter-se-á uma melhoria na qualidade do software em desenvolvimento em decorrência da melhoria do processo de desenvolvimento o que levará a redução de tempo e custo da manutenção do software orientado a agentes.

Embora o desenvolvimento de sistemas multi-agentes não seja simples, isso não significa que seja mais complexo ou propenso a erros que os paradigmas de desenvolvimento de softwares comumente usados. Nessa seção esse tema é apresentado de forma a reconhecer as dificuldades e buscar o melhor meio de amenizá-las durante o desenvolvimento do sistema, até mesmo porque não existe nenhuma abordagem perfeita.

3.4 Metodologias no Desenvolvimento de Sistemas Multi-agentes

A construção de sistemas multi-agentes não é fácil, pois apresenta todos os problemas de sistemas distribuídos e concorrentes tradicionais, mais as dificuldades adicionais que surgem dos requisitos de flexibilidade e interações sofisticadas. Qualquer metodologia para a

engenharia de software orientada a agentes deve prover abstrações adequadas e ferramentas para modelar não só as tarefas individuais, mas também as tarefas sociais dos agentes. Nesse sentido, várias metodologias para sistemas muti-agentes têm sido propostas nos últimos anos. Nesta seção serão apresentados breves resumos de algumas das principais metodologias existentes orientadas a agentes.

3.4.1 GAIA

Essa metodologia proposta por Wooldridge e outros (WOOLDRIDGE, JENNINGS, KINNY, 2000) tem como idéia chave permitir ao analista examinar sistematicamente dos requisitos ao desenvolvimento de forma suficientemente detalhada a ponto de ser implementada diretamente. A fase de captura dos requerimentos é independente do paradigma usado para a análise e desenvolvimento.

A metodologia GAIA difere dos processos tradicionais, que visam transformar um modelo derivado do processo de análise em modelos de mais baixo nível de abstração que possam ser implementados, objetivando derivar um modelo de análise em um modelo com baixo nível de abstração sobre o qual possam ser aplicadas técnicas de projeto tradicionais (incluindo técnicas OO).

Assim, o modelo de organização GAIA é compreendido de dois outros modelos (Figura 3.2): o modelo de papéis, que identifica os papéis chave no sistema; e o modelo de interações, que identifica as dependências e os relacionamentos entre os vários papéis numa organização multi-agentes.

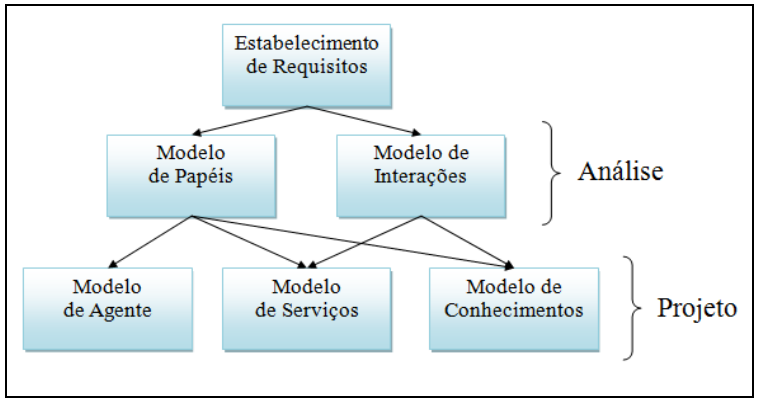

Figura 3.2: Relacionamento entre os modelos GAIA

O principal conceito da metodologia de GAIA pode ser dividido em duas entidades, concretas e abstratas, como mostra a tabela 3.1. As abstratas são aquelas usadas durante a análise e concepção do sistema, mas que não necessariamente têm alguma realização dentro do sistema. Já as concretas são usadas dentro do processo de desenvolvimento e tipicamente terão realização direta no sistema de execução.

Tabela 3.1: Conceitos concretos e abstratos em GAIA

Conceitos Concretos	*Conceitos Abstratos*
Tipos de Agentes	Papéis
Serviços	Permissões
Conhecimentos	Responsabilidades
	Protocolos
	Atividades
	Propriedades Vitais
	Propriedades de Segurança

A fase de análise envolve as entidades abstratas e pretende desenvolver a compreensão do sistema e sua estrutura, sem considerar nenhum detalhe de implementação. Dessa forma, esse entendimento é capturado na organização do sistema. Uma organização é um conjunto de papéis, que se relacionam entre si e que participam de padrões sistemáticos e institucionalizados de interação com outros papéis. Como resultado dessa fase, obtém-se dois modelos: modelo de papéis, que define os principais papéis no sistema e o modelo de interações, que identifica as dependências e os relacionamentos entre os diversos papéis em um sistema multi-agentes.

Ao nível de projeto, GAIA compreende a geração de três modelos: o de agente; o de serviços; e o de conhecimento. O modelo de agente identifica os tipos de agente que vão formar o sistema e as instâncias de agente geradas a partir destes tipos. O modelo de serviços identifica os principais serviços que são requisitados para realizar o papel do agente. Por fim, o modelo de conhecimento documenta as linhas de comunicação entre os diferentes agentes.

A metodologia GAIA, apesar das vantagens que incorpora ao processo de desenvolvimento, apresenta algumas limitações. São elas: limitação quanto a quantidade de agentes; não trata objetivos conflitantes entre os agentes; a habilidade social e as atividades desempenhadas pelos agentes não se modificam em tempo de execução.

Um exemplo do uso da metodologia pode ser visualizado no trabalho desenvolvido

por (MORAITIS, SPANOUDAKIS, 2004), onde integra-se a plataforma JADE[3] e a metodologia GAIA, mostrando o mapeamento que pode ser realizado entre os conceitos básicos propostos por GAIA, para especificação e interações entre agentes, e os conceitos providos por JADE para implementação de agentes.

3.4.2 Agent Unified Modelling Language (AUML)

A AUML é uma metodologia que estende a Linguagem de Modelagem Unificada (UML) para suportar o desenvolvimento de sistemas orientados a agentes. A AUML propõe uma representação em três camadas para os protocolos de interação de agentes. Essas camadas descrevem um padrão de comunicação entre agentes, definindo a seqüência permitida de mensagens e as restrições sobre o conteúdo destas mensagens. Os protocolos de agentes, representados pela abordagem em camadas, definem: modelos e pacotes; diagramas de seqüência e colaboração; e máquinas de estados e diagramas de atividades (ODELL, 1999).

Os diagramas de interação são representados por padrões estruturais de interações entre os agentes. Os diagramas de seqüência e os diagramas de colaboração são subtipos do diagrama de interação, sendo semanticamente equivalentes. A disposição dos elementos gráficos do diagrama de seqüência enfatiza a seqüência cronológica de comunicações, enquanto no diagrama de colaboração são enfatizadas as associações entre agentes, onde a seqüência das interações é representada através da numeração das mensagens.

Os diagramas de atividade e os diagramas de estados dão ênfase ao fluxo de processamento. O diagrama de atividade representa operações e os eventos que ativam estas, diferindo dos diagramas de interação por representar de maneira explícita as linhas de execução dos fluxos de controle, o que é particularmente útil para protocolos de interação complexos envolvendo paralelismo de processamento. Diagramas de estado não são normalmente utilizados para expressar protocolos de interação, pois têm uma visão centrada nos estados, ao invés de uma visão centrada no agente ou no processo. Este tipo de diagrama é

3 JADE (*Java Agent Development Framework*) é framework desenvolvido conforme as especificações FIPA para o desenvolvimento de sistemas multi-agentes implementado na linguagem Java., Maiores informações, consultar a referência (JADE, 1998).

melhor aplicável como um mecanismo de restrições para o protocolo, sendo incorporado aos agentes para que as restrições sejam conhecidas.

A AUML fornece modelos para capturar o aspecto dinâmico das interações inter-agente. mas ainda não fornece extensões para capturar o mapa cognitivo do agente (individual/estrutura estática) ou a organização estrutural de agentes (sistema/estrutura estática).

Por fim, tem-se a metodologia AUML cobrindo apenas a fase de projeto do ciclo de vida de software e suportando apenas a especificação da interação entre os agentes.

Um exemplo de uso dessa metodologia é o trabalho de (PADILHA, JACOMÉ, 2002), onde os autores investigam o uso de modelagem de agentes em sistemas educacionais, comparando duas técnicas, sendo a AUML uma delas.

3.4.3 TROPOS

O projeto TROPOS é uma proposta de desenvolvimento de sistemas orientados a agentes, inspirada na análise de requisitos e fundamentada em conceitos sociais e intencionais, que visa reduzir tanto quanto possível esta incompatibilidade entre o sistema e seu ambiente (MYLOPOULOS, CASTRO, 2000).

A metodologia TROPOS define cinco fases de desenvolvimento de software: Requisitos Iniciais, Requisitos Finais, Projeto Arquitetural, Projeto Detalhado e Implementação. O processo inicia com um modelo do ambiente no qual o sistema que está sendo desenvolvido irá operar. O modelo é descrito em termos de atores, seus objetivos e interdependências. Através de refinamentos incrementais, este modelo é estendido para incluir tanto o sistema a ser desenvolvido quanto aos seus subsistemas, que também são representados como atores a quem foram delegados objetivos para atingir. Também há a definição de planos a serem executados e recursos a serem fornecidos pelos atores.

Um exemplo de aplicação dessa metodologia pode ser verificado no trabalho de (SILVA, 2005), onde a autora desenvolve um sistema de gerenciamento de conteúdo (CMS) que utiliza agentes para simular o processo de publicação de notícias na internet, através da metodologia TROPOS e da plataforma de desenvolvimento JADE. Outro exemplo é a

dissertação de (SILVA, 2003), onde a autora propõe o uso de uma extensão da UML para acomodar os conceitos e características usadas para representar arquiteturas organizacionais em TROPOS.

A metodologia TROPOS foi escolhida para esse trabalho, pois propõe um processo de desenvolvimento de sistemas multi-agentes que engloba desde a análise de requisitos até a implementação. Além disso, o *framework* i*, responsável pela análise de requisitos nessa metodologia, permite uma melhor compreensão dos objetivos e do ambiente de uma organização. Os sistemas de informação geralmente possuem uma impedância[4] em relação ao seu desenvolvimento. Seu ambiente operacional é visto em termos de atores, responsabilidade, objetivos, tarefas e recursos enquanto o sistema de informação em si é concebido com uma coleção de módulos, entidades (por exemplos objetos), estruturas de dados e interfaces. Essa defasagem é um dos principais fatores para a má qualidade dos softwares e também fracassos nos projetos de desenvolvimento (CASTRO, KOLP, MYLOPOULOS, 2002).

O *framework* i* propõe exatamente essa visão estratégica e intencional dos processos, possibilitando um melhor entendimento dos relacionamentos organizacionais e das razões das decisões tomadas e ilustra várias características encontradas na fase inicial da engenharia de requisitos.

A metodologia TROPOS será descrita detalhadamente no próximo capítulo.

4 Impedância: medida da oposição ao fluxo de dados e/ou informações. Fonte: (WIKIPEDIA, 2008).

4 APLICAÇÃO DA METODOLOGIA TROPOS

Esse capítulo destina-se a aplicar a metodologia TROPOS na modelagem do sistema. Ele está dividido em duas seções principais: a seção 4.1, que aborda os requisitos do sistema em questão e a seção 4.2, que detalha a modelagem do sistema explicando e utilizando essa metodologia.

4.1 Sobre o sistema

O sistema a ser modelado tem por objetivo procurar a pessoa mais qualificada para ocupar uma determinada vaga dentro de uma empresa. Ele se baseia no modelo de classificação (quando existem vários candidatos para cada vaga e várias vagas para cada candidato).

Para a definição mais clara do sistema, é necessário que algumas nomenclaturas sejam definidas. São elas:

- Cargo: Função em uma dada organização, por exemplo o cargo de programador.
- Posição: Uma ocupação de uma pessoa em um cargo, por exemplo o cargo programador possui três posições ocupadas, ou seja três programadores.
- Vaga: Toda a posição ainda não preenchida.

O sistema possuirá 4 subsistemas:

- Cadastro de empresas;
- Cadastro de cargos;
- Cadastro de pessoas (currículos);
- Cadastro de posições.

Quando uma posição for sinalizada como não preenchida, deverá ser procurado entre os candidatos disponíveis o melhor para ocupar determinada vaga.

O sistema utilizará o recrutamento misto, dando preferência para o recrutamento

interno inicialmente, e caso não encontre a pessoa adequada, passará a usar o recrutamento externo.

Quando houver conflito, isto é, quando mais de uma vaga desejar o mesmo candidato, o critério de desempate será o faturamento da empresa na qual a vaga pertence. Se as vagas forem da mesma empresa, será levado em consideração a importância do cargo na hierarquia da organização como critério de desempate.

4.2 Sobre a metodologia TROPOS

TROPOS é uma metodologia de engenharia de software orientada a agentes caracterizada por dois aspectos (BRESCIANI, SANNICOLÒ, 2002):

- Preocupa-se em entender como e porque o sistema pretendido alcança os objetivos organizacionais;
- Lida com as fases de análise de requerimentos do sistema e as fases de projeto e implementação baseada em noções como atores, objetivos, planos, recursos, dependências;

A metodologia reside na idéia de construir um modelo do sistema que será incrementalmente refinado e estendido do nível conceitual ao executável, ou seja, numa seqüência de passos modificáveis.

Essa metodologia adota o *framework* de modelagem organizacional i* em suas fases iniciais de modelagem e utiliza seus conceitos intencionais[5] durante todas as fases de desenvolvimento (CASTRO, KOLP, MYLOPOULOS, 2002).

4.2.1 *Framework* i*

O *framework* i* (simboliza "intencionalmente distribuído") modela contextos

5 Conceitos Intencionais: conceitos provenientes da Inteligência Artificial, tais como objetivos, crenças, habilidades e comprometimento.

organizacionais baseado nos relacionamentos de dependência entre os atores. Para realizar esta análise, os engenheiros de requisitos modelam os *stakeholders*[6] como atores e suas intenções como metas (SILVA, 2005). Cada meta é analisada do ponto de vista do seu ator, resultando em um conjunto de dependências entre pares de atores. Assim, o ambiente do sistema e o sistema em si são vistos como organizações de atores, que dependem da ajuda de outros atores para que suas metas sejam cumpridas.

A utilização desse *framework* possibilita a compreensão das razões internas dos atores, uma vez que as mesmas são expressas explicitamente, auxiliando na escolha de alternativas durante a etapa de modelagem do software.

O *framework* é dividido em dois modelos:

- Modelo de Dependência Estratégica(SD): fornece uma descrição intencional de um processo em termos de uma rede de relacionamentos de dependência entre atores relevantes;
- Modelo de Razão Estratégica(SR): apresenta uma descrição estratégica do processo, em termos de elementos do processo e das razões que estão por detrás deles, ou seja, fornece uma análise meios-fins de como as metas podem ser cumpridas através das contribuições dos demais atores.

4.2.1.1 Modelo de Dependência Estratégica

O modelo de dependência estratégica é um gráfico envolvendo atores que possuem dependências estratégicas entre eles (CASTRO, KOLP, MYLOPOULOS, 2002). Para explicar o que é uma dependência estratégica, primeiro é necessário definir alguns conceitos. São eles:

- Ator: entidade ativa que realiza ações para atingir metas, exercitando seu *know-how*. O termo ator é utilizado para referenciar genericamente qualquer unidade para a qual dependências intencionais possam ser atribuídas. Atores podem ser considerados intencionais (possuem motivações, intenções e razões por trás de

6 *Stakeholders*: são pessoas ou organizações que serão afetadas pelo sistema e que possuem influência, direta ou indireta, sobre os requisitos do sistema: usuários finais, gerentes, engenheiros responsáveis pelo desenvolvimento e manutenção do sistema, clientes da organização que usarão o sistema, etc

suas ações) e estratégicos(não focam apenas o seu objetivo imediato, também se preocupam com as implicações de seu relacionamento estrutural com outros atores);

- Meta ou objetivo: uma condição ou estado do mundo que um ator gostaria de alcançar. Geralmente uma meta é expressa como um desejo;
- Meta flexível: também representa um desejo a ser satisfeito, exceto pelo fato de que não há um critério claramente definido de verificação se a condição desejada foi atingida. A sua satisfação depende do julgamento subjetivo e interpretação dos *stakeholders*;
- Tarefa: define uma maneira específica de se fazer algo. Tarefas podem ser vistas como soluções que provêem operações, processos, representações de dados, estruturação, restrições e agentes para atender às necessidades estabelecidas nas metas (flexíveis ou não);
- Recurso: é uma entidade física ou de informação. Entende-se como recurso o produto final de alguma ação, em um processo, que estará ou não disponível para o ator dependente.

A figura 4.1 apresenta a simbologia utilizada pela metodologia dos termos citados acima.

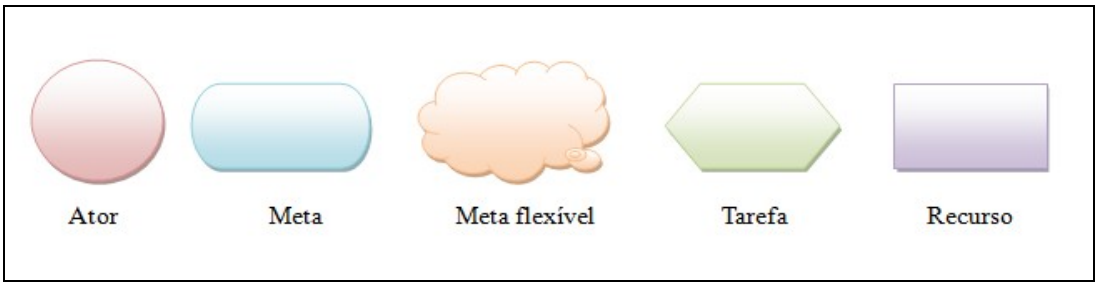

Figura 4.1: Elementos de modelagem do *framework* i*

Para ilustrar os conceitos acima, pode-se citar a vaga como um exemplo de ator do sistema em questão, que tem como meta buscar uma pessoa dentre as cadastradas no sistema para preenchê-la e como meta flexível encontrar a pessoa mais qualificada para ocupá-la.

Depois de expostas as definições de ator, meta, meta flexível, tarefa e recurso, é possível conceituar dependência estratégica de uma maneira mais clara. Dependência estratégica é uma espécie de acordo (chamada *dependum*) entre dois atores: o *depender* (ator dependente) e o *dependee* (ator que o *depender* depende). A natureza do acordo caracteriza o

tipo de dependência (*dependum*) que podem ser:

- Dependências de meta (*goal*): são usadas para representar delegação de responsabilidade para o cumprimento de uma meta;
- Dependências de meta flexível (*softgoal*): são semelhantes às dependências de meta, mas seu cumprimento não pode ser definido com precisão (por exemplo, a satisfação da meta é subjetiva, ou o cumprimento pode ocorrer apenas para um determinado grau, isto é, um nível de satisfação razoavelmente aceitável);
- Dependências de tarefa: são usadas em situações onde o *dependee* é necessário para executar alguma atividade;
- Dependência de recurso: ocorre quando um ator (*depender*) necessita do outro (*dependee*) para que uma entidade (física, computacional) seja disponibilizada.

A figura 4.2 apresenta alguns tipos de relacionamentos de dependência estratégica.

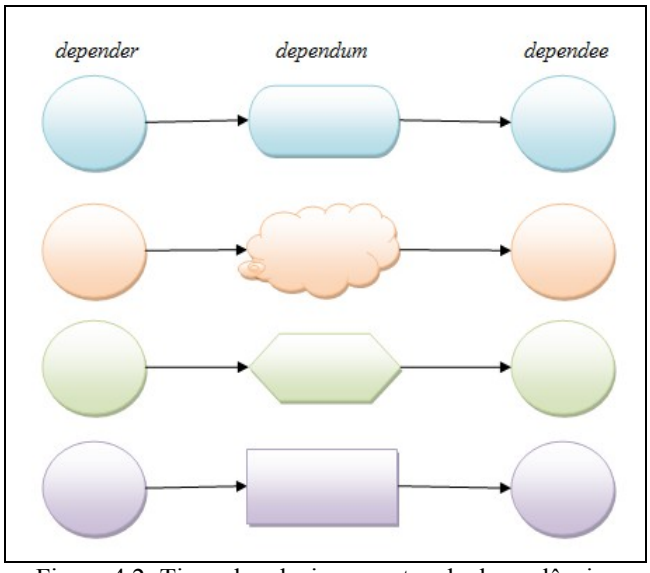

Figura 4.2: Tipos de relacionamentos de dependências

Um exemplo de dependência estratégica em relação ao sistema desse trabalho é o ator vaga depender do ator pessoa para alcançar a meta de encontrar currículos compatíveis para preencher a vaga. Esse relacionamento é ilustrado na figura 4.3. Os modelos completos de dependência estratégica são apresentados nas seções de requisitos iniciais e finais.

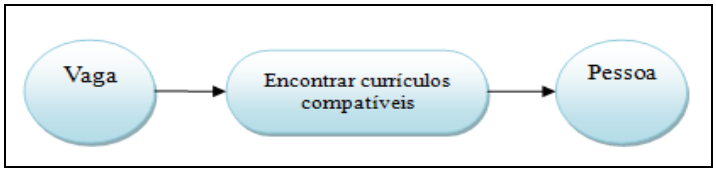

Figura 4.3: Exemplo de relacionamento de dependência estratégica

Na dependência estratégica, pode-se definir três tipos de atores: agentes, papéis e posições. Os mapeamentos de agente, papel e posição são úteis para a modelagem do agrupamento de um grande número de dependências que agentes no mundo real geralmente possuem. Um agente executa um determinado papel e ocupa uma determinada posição que cobre um papel. Os tipos de atores são:

- Agente: é um ator que possui manifestações físicas concretas. Pode-se referir a um humano ou agentes de *software* ou *hardware*;
- Posição: entidade intermediária entre um agente e um papel. É o conjunto de papéis tipicamente ocupados por um agente, ou seja, representa uma posição dentro da organização onde o agente pode desempenhar várias funções;
- Papel: caracteriza abstratamente o comportamento de um ator dentro de determinados contextos sociais ou domínio de informação. Apresenta as funções que podem ser exercidas por um agente dentro da organização.

A figura 4.4 ilustra a representação gráfica e as possíveis associações entre agentes, papéis e posições.

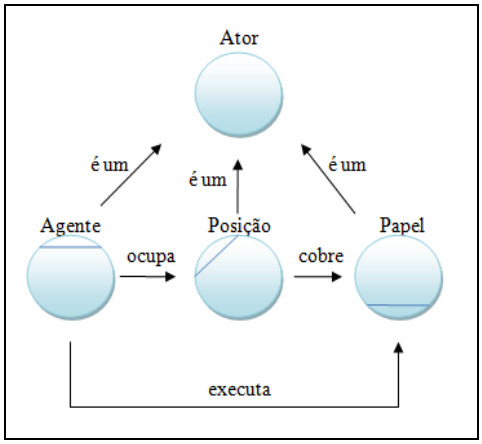

Figura 4.4: Agentes, posições e papéis

4.2.1.2 Modelo de Razão Estratégica

O modelo de dependência estratégica (SD) preocupa-se apenas com os relacionamentos externos entre os atores. Por outro lado, o modelo de razão estratégica (SR)

trata da descrição dos interesses, preocupações e motivações dos atores participantes de um processo. Ele permite a avaliação das possíveis alternativas de definição do processo, investigando mais detalhadamente as razões existentes por trás das dependências entre os atores.

Esse modelo é formado por nós e elos, que juntos provêem uma estrutura para exprimir as razões envolvidas no processo. Ele também usa quatro espécies de nós que se baseiam nos tipos de dependências do modelo SD: meta, meta flexível, tarefa e recurso. Nesse modelo são acrescidos outros dois tipos de relacionamento que interconectam os nós. São eles:

- Ligações de meio-fim: define um relacionamento entre um nó fim (que pode compor uma meta a ser alcançada, uma tarefa a ser realizada, um recurso a ser produzido, ou uma meta flexível a ser satisfeita) e um meio para alcançá-lo. Esse tipo de ligação sugere alternativas existentes para se atingir um determinado fim. Os meios geralmente são expressos na forma de tarefas enquanto os fins podem ser metas, metas flexíveis, recursos ou tarefas. Existe a possibilidade também de o meio e o fim serem ambos metas ou metas flexíveis. As ligações de meio-fim podem ser visualizadas na figura 4.5.

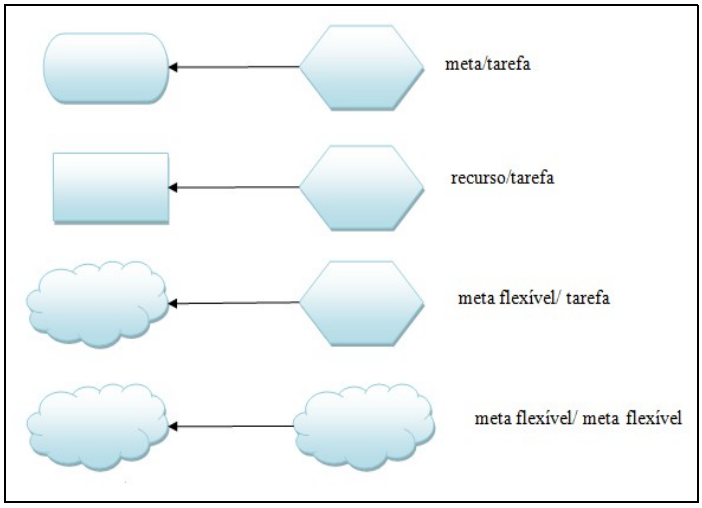

Figura 4.5: Ligações de meio-fim

Uma ligação meio-fim pode ser explicada através do seguinte exemplo: a meta flexível "ter os melhores funcionários" pertencente ao ator empresa é um fim e tem como meio a tarefa "selecionar o currículo que mais se encaixa na vaga dentre os pré-selecionados". A figura 4.6 ilustra esse exemplo.

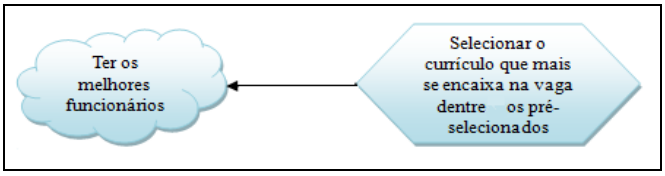

Figura 4.6: Exemplo de ligação meio-fim

- Ligações de decomposição de tarefas: unem um nó de tarefa a seus nós componentes, que podem ser outras tarefas, metas, metas flexíveis ou recursos. Uma tarefa decomposta só será considerada realizada quando todos os seus nós componentes são cumpridos. As ligações de decomposição de tarefas são apresentadas na figura 4.7.

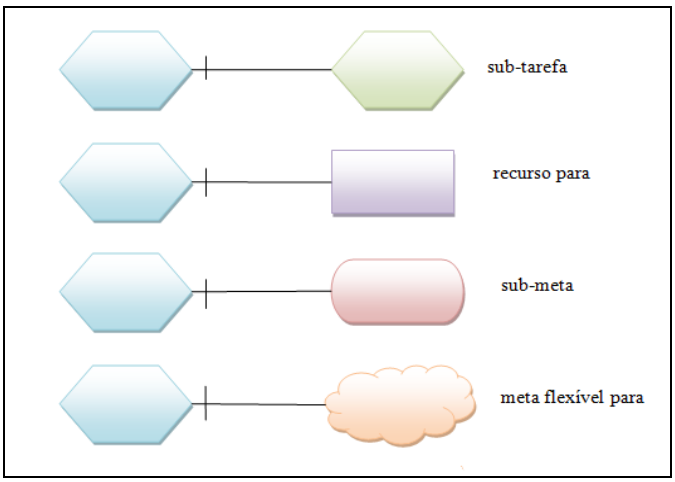

Figura 4.7: Ligações de decomposição de tarefas

Uma ligação de decomposição de tarefas pode ser exemplificada através da tarefa "selecionar o currículo que mais se encaixa na vaga dentre os pré-selecionados" que pode ser decomposta na sub-tarefa "comparar currículos com os pré-requisitos da vaga". Esse relacionamento é ilustrado na figura 4.8.

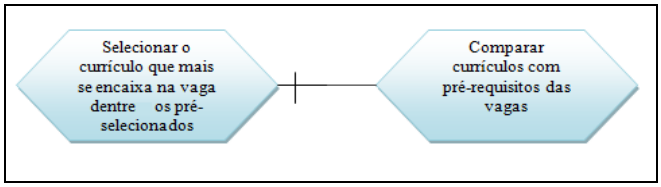

Figura 4.8: Exemplo de ligação de decomposição de tarefas

A construção do modelo SR ocorre a partir de um detalhamento decorrente do modelo SD, de alguns atores envolvidos no processo. Nesse modelo é possível identificar o comportamento interno dos atores detalhados e as ligações existentes (de decomposição de

tarefas e do tipo meio-fim).

Os modelos completos de razão estratégica são apresentados nas seções de requisitos iniciais e finais.

4.2.2 Fases da metodologia Tropos

A metodologia Tropos é composta das seguintes fases: requisitos iniciais, requisitos finais, projeto arquitetural e projeto detalhado. A análise de requisitos representa a etapa inicial do desenvolvimento de software e é abordada nas fases de requisitos iniciais e finais utilizando o *framework* i*.

4.2.2.1 Requisitos Iniciais

Essa fase preocupa-se com a compreensão de um contexto organizacional no qual o sistema a ser desenvolvido será implantado, ou que servirá de modelo para este sistema. O resultado dessa fase são dois modelos (BRESCIANI et al., 2000):

- Modelo de Dependência Estratégica (SD): modelo organizacional que captura atores relevantes, seus objetivos e suas respectivas dependências. Os principais atores desse sistema são: as empresas, as vagas, o administrador do sistema e as pessoas. A figura 4.9 apresenta o modelo i* de dependência estratégica da fase de requisitos iniciais deste trabalho. Todos os atores dependem do administrador de sistema para que dados sejam cadastrados. As empresas necessitam que o administrador cadastre seus dados no sistema, bem como as pessoas precisam ter seus currículos cadastrados e as vagas precisam ter seus detalhes cadastrados. A empresa tem como meta preencher seu quadro de funcionários e como meta flexível que esses funcionários sejam os melhores. A vaga deseja encontrar currículos compatíveis, buscando a meta flexível de achar o melhor currículo compatível.

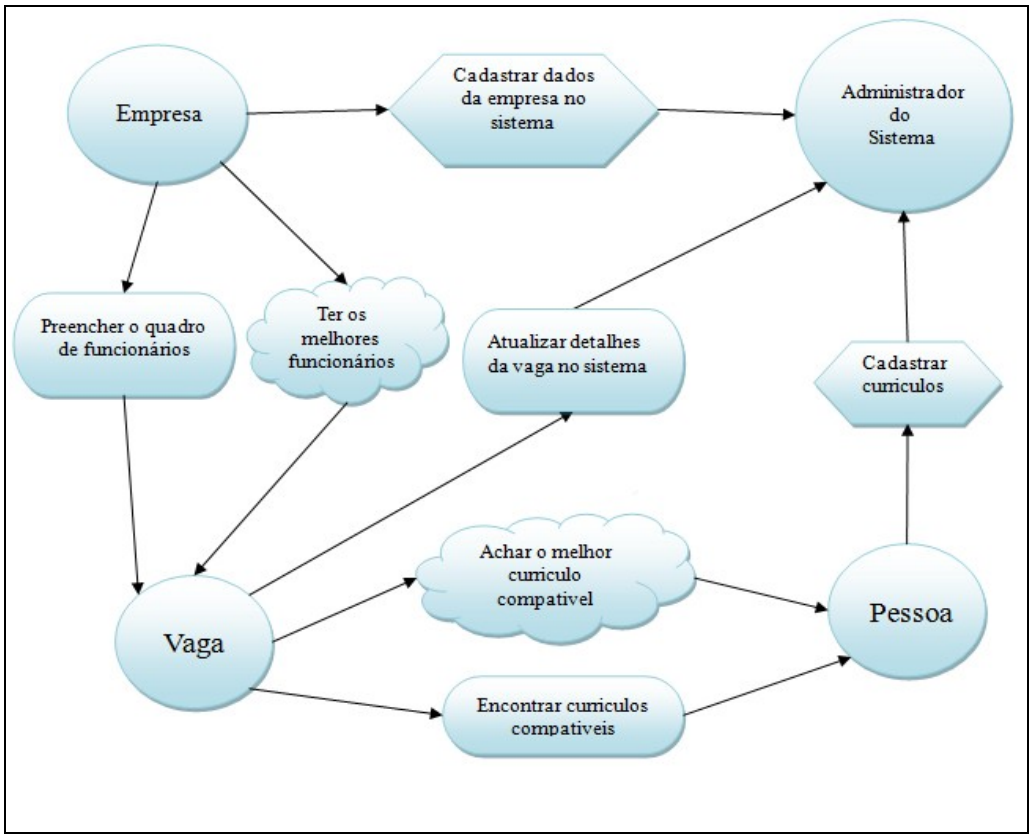

Figura 4.9: Modelo de dependência estratégica da fase de requisitos iniciais

- Modelo de Razão Estratégica (SR): esse modelo é um refinamento do modelo SD, apresentando uma descrição estratégica dos processos, detalhando os relacionamentos intencionais, que são internos aos atores. Isso é feito através de análises meio-fim, que mostram como os objetivos podem ser alcançados através da ajuda dos outros atores. No modelo em questão, o ator vaga foi detalhado em duas principais relações meio-fim. A primeira refere-se ao fim "preencher o quadro de funcionários" que pode ser alcançado através dos meios "cadastrar detalhes da vaga" e "comparar currículos com pré-requisitos das vagas". A segunda refere-se ao fim "ter os melhores funcionários", onde o meio para alcançá-lo é "selecionar o currículo que mais se encaixa na vaga dentre os selecionados". Além das ligações meio-fim, existem também ligações de decomposição de tarefas. Uma das ligações refere-se a tarefa "selecionar o currículo que mais se encaixa na vaga dentre os selecionados" que é decomposta na sub-tarefa "comparar currículos com pré-requisitos das vagas". Essa última

tarefa também é decomposta em sub-tarefas. São elas: “buscar currículos no sistema” e “buscar detalhes da vaga no sistema”. Esse modelo é apresentado na figura 4.10.

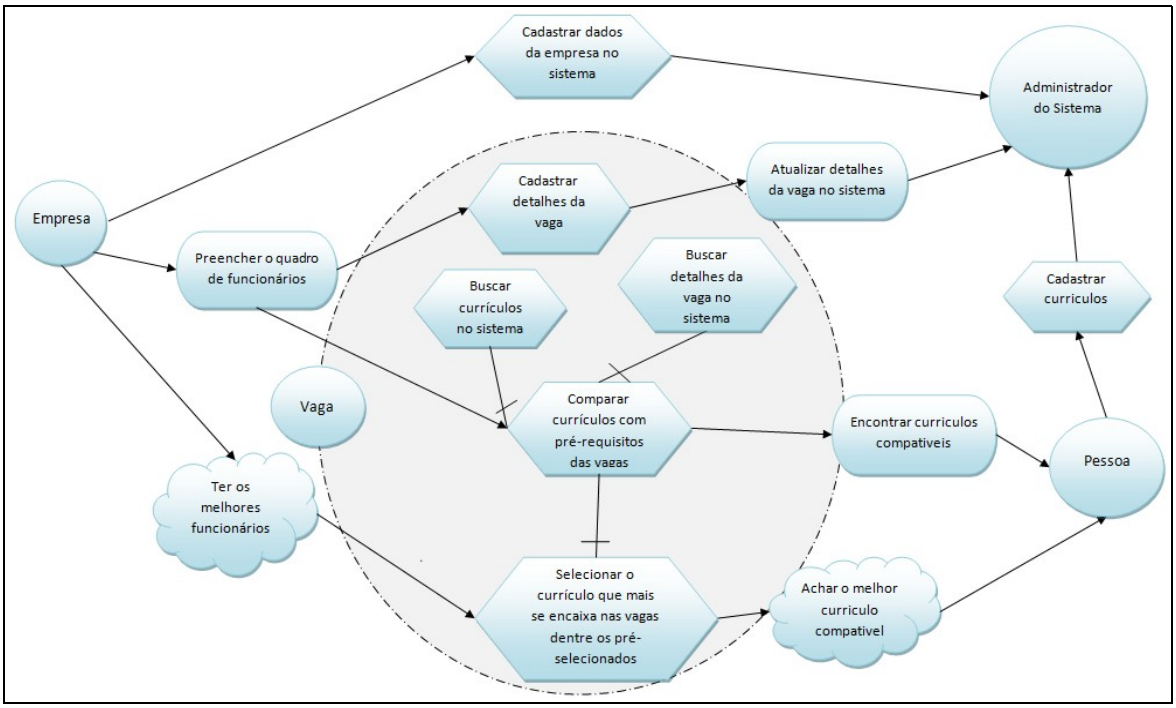

Figura 4.10: Modelo de razão estratégica da fase de requisitos iniciais

4.2.2.2 Requisitos Finais

A fase de requisitos finais é um refinamento dos modelos SD e SR desenvolvidos na fase de requisitos iniciais. Nessa fase, os modelos conceituais desenvolvidos na fase anterior são revisados e estendidos para incluir novos atores, que representam tanto o sistema a ser desenvolvido quanto os seus subsistemas.

Para dar origem ao modelo SD dessa fase, esses atores devem ser relacionados em termos de novos pares de dependências, aos atores do ambiente operacional do sistema que foram previamente identificados. Todas as dependências em relação ao sistema refletem as responsabilidades atribuídas a ele, de forma a contribuir com as necessidades dos

stakeholders. Essas dependências descrevem os requisitos funcionais[7] e não funcionais[8] do sistema a ser desenvolvido.

Para construção do modelo SR desta fase, o ator que representa o sistema a ser desenvolvido deve ser expandido para apresentar as razões que estão por trás de suas dependências. Suas tarefas e metas precisam ser revisadas, analisadas e detalhadas através de ligações de decomposição e de meios-fins e irão eventualmente levar a revisar e adicionar novas dependências com um subconjunto de atores sociais (os usuários). Se necessário decompõe-se o sistema que representa o ator em vários sub-atores[9] e se revisa os modelos de razão e dependência estratégica. Além disso, pode haver a atribuição de responsabilidades aos seus subsistemas. O objetivo desta etapa é dar suporte ao processo de sugestão, exploração e avaliação de soluções alternativas para o sistema.

No modelo de dependência estratégica, ilustrado na figura 4.11, foi acrescentado o ator cargo, do qual os atores empresa e vaga necessitam para alcançar suas metas. O ator empresa necessita do ator cargo para que as tarefas da empresa sejam executadas, bem como a empresa depende do cargo para que essas tarefas sejam executadas da maneira mais eficiente possível. O ator vaga depende do cargo para obter os pré-requisitos necessários das pessoas a ocuparem a posição disponível liberada pelo cargo.

Já o modelo de razão estratégica, apresentado na figura 4.12, detalha as razões por trás das dependências, onde há uma ligação meio-fim entre a meta “obter os pré-requisitos necessários” e a tarefa “comparar os currículos com os pré-requisitos das vagas”.

7 Requisitos funcionais: condições necessárias para o obtenção de certo objetivo, neste caso, a definição das funções que um sistema ou componente deve possuir.

8 Requisitos não-funcionais: também conhecidos como atributos de qualidade, referem-se a requisitos gerais que fazem parte do desenvolvimento de um sistema, tais como custo, desempenho, confiabilidade, manutenibilidade, portabilidade, entre outros.

9 Sub-atores: novos atores que recebem responsabilidades que foram adicionadas ao sistema em uma nova fase.

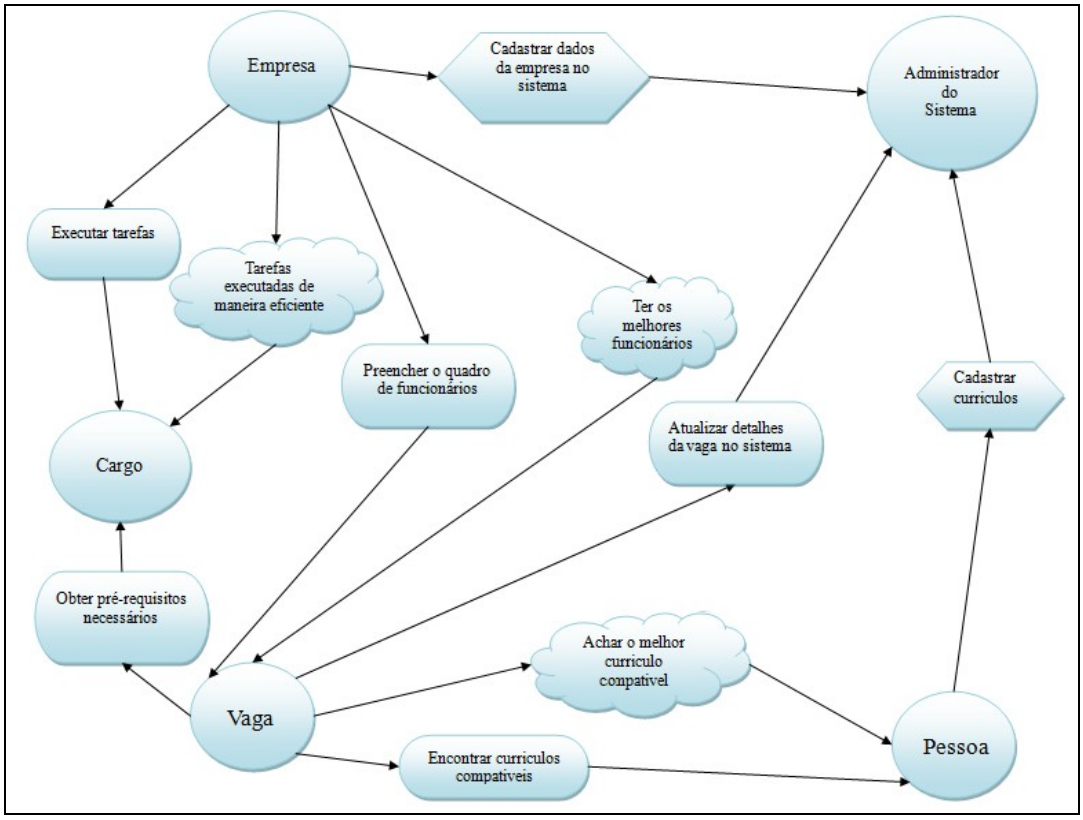

Figura 4.11: Modelo de dependência estratégica da fase de requisitos finais

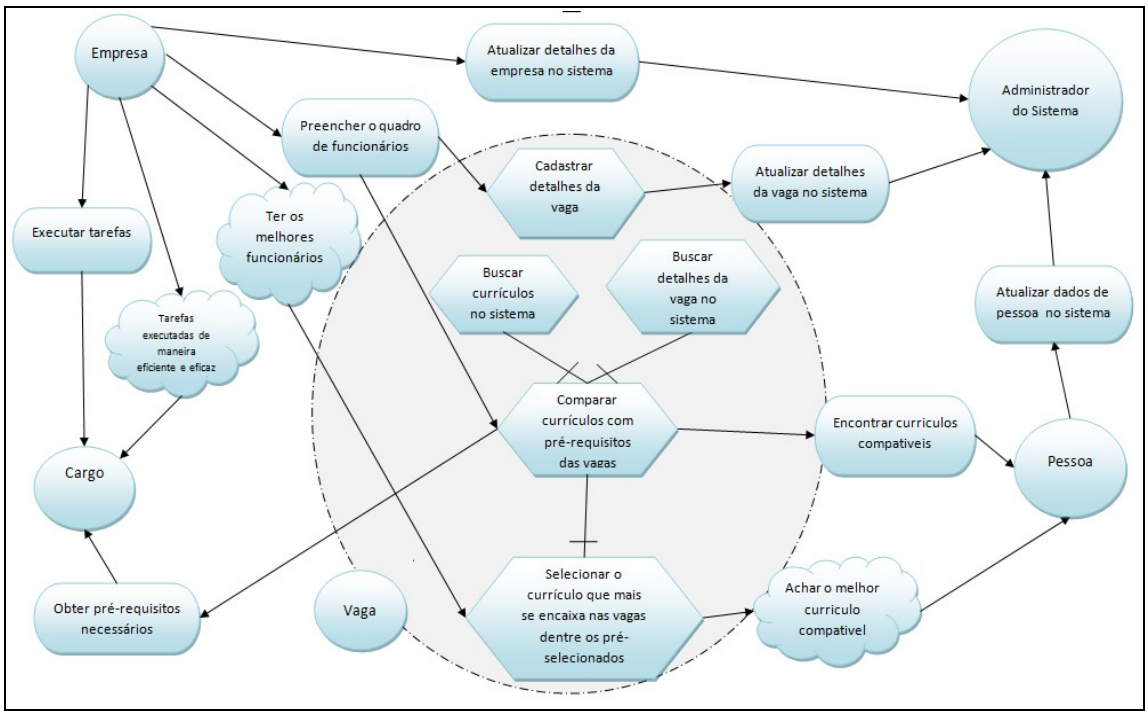

Figura 4.12: Modelo de razão estratégica da fase de requisitos finais

4.2.2.3 Projeto Arquitetural

Essa fase define uma arquitetura global do sistema em termos de subsistemas, interconectados através de dados e fluxos de controle. A arquitetura de um sistema compõe um modelo de estrutura que define como componentes do sistema trabalham juntos. Subsistemas são representados como atores e interconexões de dado/controle são representados como dependências de ator.

Essa fase consiste de três passos. São eles:

- Passo 1 – Escolha da arquitetura: Selecionar o estilo arquitetural, utilizando como critério as qualidades desejadas que foram identificadas na fase anterior e incluir novos atores, se necessário, de acordo com a escolha de um estilo arquitetural específico de agente. Para realizar a escolha dessa arquitetura TROPOS utiliza o *framework* NFR[10] e o catálogo de correlação entre os estilos arquiteturais organizacionais e requisitos não-funcionais, ou atributos de qualidade (KOLP, GIORGINI, MYLOPOULOS). A Tabela 4.1 apresenta os estilos arquiteturais organizacionais utilizados em TROPOS e a tabela 4.2 apresenta as correlações desses estilos com os atributos de qualidade de um sistema.

  Os atributos de qualidade a serem levados em consideração nesse sistema são:

  - Coordenação: componentes autônomos não são úteis se eles não forem capazes de interagir com outros componentes (KOLP, CASTRO, MYLOPOULOS, 2001). Isso pode ser feito de duas formas:
    - Cooperação: Esses componentes devem ser capazes de cooperar com outras entidades para alcançar propósitos comuns.
    - Competição: O sucesso de um componente implica na falha de outros.

10 *Framework* NFR (*Non-functional Requirements*): utiliza os requisitos não-funcionais(vistos como metas flexíveis) para guiar o processo de escolha da arquitetura e oferece uma estrutura (Gráfico de interdeprendência de Softgoals – SIG) para representar e armazenar os passos e raciocínios do projeto. Maiores detalhes, consultar (CHUNG et al., 2000).

Tabela 4.1: Estilos arquiteturais organizacionais definidos em Tropos

Estilo Arquitetural	***Principais Características***
Estrutura Plana (*Flat Structure*)	Não possui estrutura fixa e assume que nenhum ator tenha controle sobre outro. Suporta autonomia, distribuição e evolução da arquitetura de um ator.
Estrutura em 5 (*Structure in 5*)	Consiste dos componentes típicos (estratégicos e logísticos) geralmente encontrados em várias organizações.
Pirâmide (*Pyramid*)	Estrutura de autoridade hierárquica exercida com limites organizacionais, onde atores nos níveis mais baixos dependem daqueles nos níveis mais altos.
União Estratégica (*Join Venture*)	Envolve acordos entre dois ou mais parceiros principais, relacionados comum, gerente comum e com parceiros secundários.
Oferta (*Bidding*)	Abrange mecanismos de competitividade e os atores se comportam como se estivessem tomando parte em um leilão.
Tomada de Controle (*Takeover*)	Envolve a delegação total de autoridade e gerenciamento de dois ou mais parceiros para um único ator, semelhante ao estilo União Estratégica.
Comprimento do Braço (*Arm's length*)	Implica em acordos entre atores independentes e competitivos, porém parceiros. Não há delegação de autoridade.
Contratação Hierárquica (*Hierarchical Contracting*)	Identifica mecanismos de coordenação que combinam características do acordo Comprimento de Braço com aspectos de autoridade Pirâmide.
Integração Vertical (*Vertical Integration*)	Consiste de atores comprometidos em atingir metas ou realizar tarefas relacionadas em estágios diferentes de um processo de produção.
Apropriação (*Co-optation*)	Envolve a incorporação de agentes externas na estrutura ou no comportamento tomador de decisão ou conselheiro de uma organização iniciante.

Para a escolha arquitetural, deve-se analisar a tabela 4.2, verificando a melhor combinação dos atributos de qualidade. O *framework* NFR define as notações *help*, *make*, *hurt* e *break*, para modelar, respectivamente, contribuições parcial/positiva (+), suficiente/positiva (++), parcial/negativa (-) e suficiente/negativa (--) entre as metas do sistema. Analisando os atributos de qualidade desejados para o sistema, o estilo arquitetural escolhido é o de Oferta, que possui as contribuições positiva para os requisito não-funcional de cooperação e parcial/negativa para requisito coordenação. Esse estilo envolve mecanismos de

competitividade e os atores se comportam como se eles estivessem tomando parte em um leilão (Figura 4.13). O ator *Leiloeiro* faz a mostra, anuncia o leilão exposto pelo *Expositor* do leilão, recebe ofertas dos atores *Comprador* e garante a comunicação e o *feedback* com o *Expositor* do leilão. O Leiloeiro deve ser um ator do sistema que meramente organiza e opera o leilão e seus mecanismos. Ele também pode ser um dos compradores (por exemplo, vendendo um item que todos os outros compradores estão interessados). O *Expositor* do leilão é responsável por expor a oferta. Este estilo implica em rápido tempo de resposta e adaptabilidade para o sistema (SILVA, 2003).

Tabela 4.2: Catálogo de correlação entre estilos arquiteturais e atributos de qualidade

Correlação	*Previsibilidade*	*Segurança*	*Adaptabilidade*	*Coordenação*	*Cooperação*	*Disponibilidade*	*Integridade*	*Modularidade*	*Agregabilidade*
Estrutura Plana	BREAK	BREAK	HURT			HELP	HELP	MAKE	HURT
Estrutura em 5	HELP	HELP		HELP	HURT	HELP	MAKE	MAKE	MAKE
Pirâmide	MAKE	MAKE	HELP	MAKE	HURT	HELP	BREAK	HURT	
União Estratégica	HELP	HELP	MAKE	HELP	HURT	MAKE		HELP	MAKE
Oferta	BREAK	BREAK	MAKE	HURT	MAKE	HURT	BREAK	MAKE	
Tomada de Controle	MAKE	MAKE	HURT	MAKE	BREAK	HELP		HELP	HELP
Comprimento de Braço	HURT	BREAK	HELP	HURT	MAKE	BREAK	MAKE	HELP	
Contratação Hierárquica			HELP	HELP	HELP	HELP		HELP	HELP
Integração Vertical	HELP	HELP	HURT	HELP	HURT	HELP	BREAK	BREAK	BREAK
Apropriação	HURT	HURT	MAKE	MAKE	HELP	BREAK	HURT	BREAK	

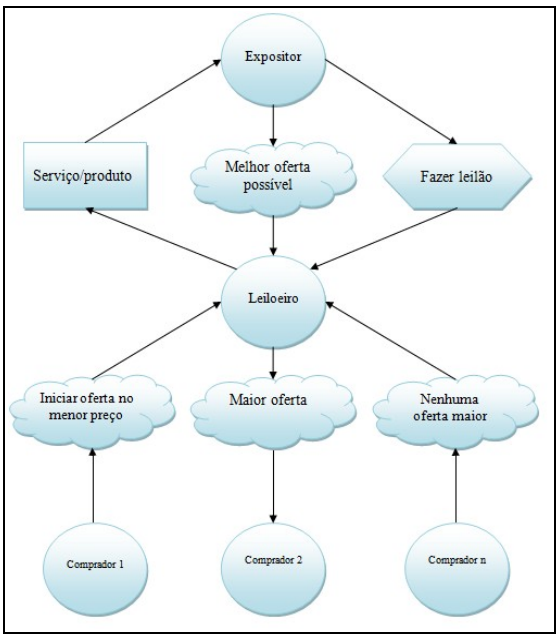

Figura 4.13: Estilo arquitetural Oferta

- Passo 2 – Modelagem da arquitetura: Incluir novos atores e dependências, assim como decompor os atores e as dependências existentes em sub-atores e sub-dependências. Revisar os modelos de dependência e razão estratégica. As capacidades de ator são identificadas da análise das dependências indo e vindo do ator, bem como das metas e planos que o ator irá executar a fim de cumprir requisitos funcionais e não-funcionais. A arquitetura modelada é ilustrada nas figuras 4.14 e 4.15.

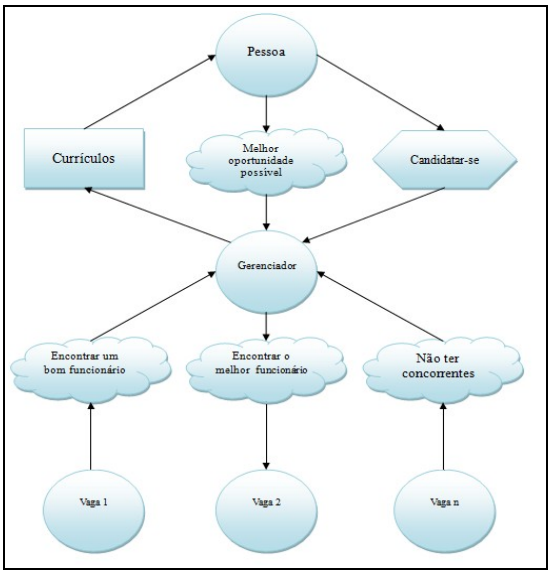

Figura 4.14: Modelagem utilizando o estilo Oferta

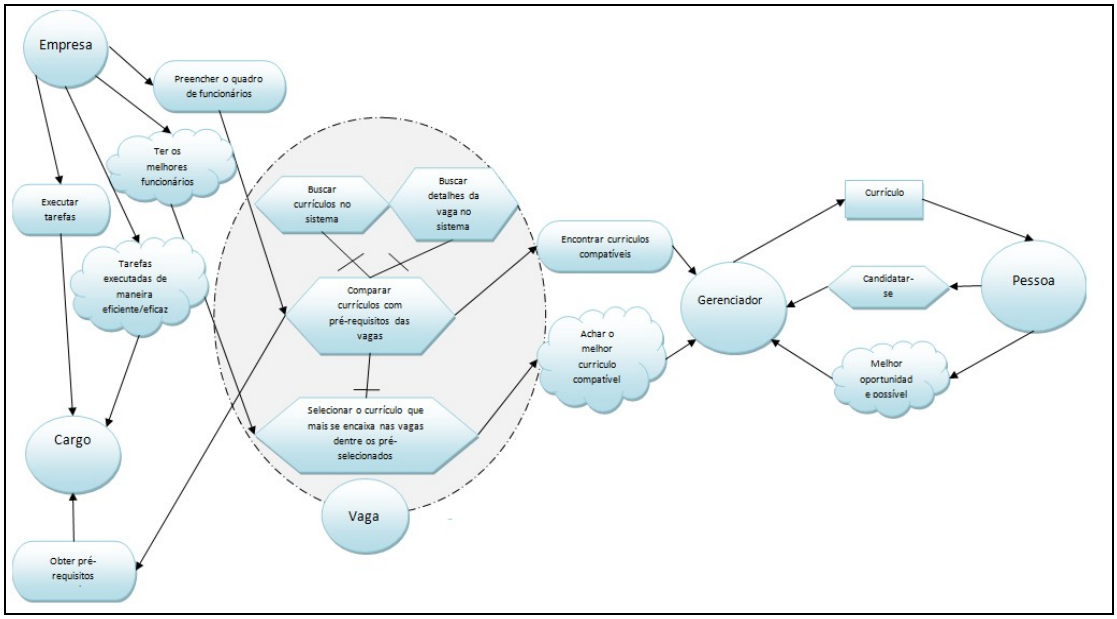

Figura 4.15: Modelagem da arquitetura detalhada utilizando o estilo arquitetural Oferta

- Passo 3 – Aplicação dos padrões sociais: Definir como as metas associadas a cada ator são cumpridas por agentes com respeito a padrões sociais. Eles são usados para solucionar uma meta específica que foi definida no nível arquitetural através da identificação de estilos organizacionais e atributos de qualidade relevantes (metas flexíveis). Os padrões sociais em TROPOS são classificados em duas categorias (KOLP et al., 2005): padrões entre pares e padrões de mediação. Como exemplos de padrões entre pares podemos citar: *booking*, *call for-proposal*, *subscription* e *bidding*. Estes padrões sociais descrevem interações diretas entre agentes negociadores. Por sua vez, os padrões de mediação são compostos por agentes intermediários que ajudam outros agentes a atingirem um acordo ou realizar troca de serviços. São eles: *monitor*, *broker*, *matchmaker*, *mediator*, *embassy*, e *wrapper.* Nesse trabalho, o agente gerenciador utiliza o padrão *Bidding*. Esse padrão envolve um iniciador e um número de participantes. O iniciador organiza e conduz o processo de oferta, e recebe propostas. A cada iteração, o iniciador publica a oferta atual; ele pode aceitar um pedido, aumentar a oferta ou cancelar o processo (CASTRO et al., 2005). Esse padrão é mostrado na figura 4.16.

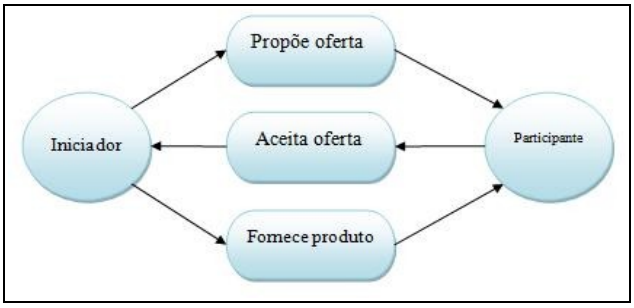

Figura 4.16: Padrão Bidding

Uma análise detalhada de cada padrão social permite definir um conjunto de capacidades associadas com os agentes envolvidos no padrão. Uma capacidade estabelece que um ator é capaz de agir a fim de atingir uma dada meta. Em particular, para cada capacidade o ator tem um conjunto de planos que pode aplicar em diferentes situações. Um plano descreve a seqüência de ações a executar e as condições sob o qual o plano é aplicável. Capacidades são coletadas num catálogo e associadas ao padrão. Isto permite definir os papéis e capacidade do ator que são adequados para um domínio particular. A tabela de capacidades será descrita na próxima seção.

4.2.2.4 Projeto Detalhado

Essa fase lida com a especificação do micro nível dos agentes, preocupando-se em apresentar detalhes adicionais para cada componente arquitetural do sistema. Está incluso nessa etapa os aspectos dos agentes que envolvem a comunicação e seus comportamentos.

Nessa fase geralmente existe uma forte dependência entre a modelagem do projeto detalhado e determinadas características das linguagens de programação de agentes e a plataforma de desenvolvimento adotada, o que relaciona estritamente o projeto detalhado com as escolhas de implementação. Por exemplo, utilizando a plataforma JADE[11] na implementação, deve-se preocupar com quatro pontos principais: agentes, comportamentos de

11 JADE (*Java Agent Development Framework*) é um *framework* desenvolvido conforme as especificações FIPA para o desenvolvimento de sistemas multi-agentes implementado na linguagem Java. Maiores informações, consultar (JADE, 1998).

agentes, mensagens e ontologias, enquanto utilizando o modelo BDI[12] a preocupação refere-se a agentes, crenças, desejos e intenções. Esse trabalho utilizará o *framework* de (KOLP et al., 2005), que é baseado no modelo BDI, para modelar e investigar os padrões sociais identificados na fase anterior. Esse *framework* é baseado em cinco dimensões complementares de modelagem de Tropos. A tabela 4.3 resume o relacionamento entre essas dimensões e suas representações.

Tabela 4.3: Dimensões de modelagem de padrões de projeto em TROPOS

Dimensão	***Pergunta***	***Representação***	***Notação***
Social	Quais os agentes, suas necessidades e dependências?	Modelo SD do i*	i*
Intencional	O que cada serviço faz?	Tabelas	-
Estrutural	Como cada serviço é operacionalizado?	Diagrama de classes adaptado	UML
Comunicativa	Como é realizada a comunicação entre os agentes?	Diagrama de seqüência adaptado	UML
Dinâmica	Como são representadas as capacidades de um agente?	Diagrama de atividades adaptado	UML

Cada dimensão reflete um aspecto particular da arquitetura de um sistema multi-agentes. A dimensão social identifica os agentes relevantes e suas interdependências intencionais. A dimensão intencional identifica e formaliza serviços providos por agentes para realizar as intenções identificadas pela dimensão social, independentemente dos planos que implementarão esses serviços. A dimensão estrutural preocupa-se com a operacionalização dos serviços identificados pela dimensão intencional em termos de conceitos orientados a agentes BDI, tais como crenças, eventos, planos e seus relacionamentos. A dimensão comunicativa modela as trocas temporárias de eventos entre os agentes, enquanto que a dimensão dinâmica modela os mecanismos de sincronização entre eventos e planos. As dimensões social e intencional são específicas de sistemas multi-agentes. As três últimas dimensões da arquitetura, também são relevantes para sistemas tradicionais (não orientados a agentes), mas elas foram adaptadas e entendidas para os conceitos da orientação a agentes.

Nas próximas seções serão detalhadas as dimensões intencional, estrutural,

12 Modelo BDI (*Belief-Desire-Intention*): é uma forma de modelagem para agentes cognitivos (agentes de software com capacidade de raciocínio) baseado em crenças, desejos e intenções como atitudes mentais que geram a ação humana. Maiores informações, consultar (BRATMAN, 1987).

comunicativa e dinâmica, visto que a social já foi descrita anteriormente pelo *framework* i*.

Dimensão Intencional

Esta dimensão está preocupada com a identificação dos serviços providos pelos agentes e apresenta as intenções identificadas na dimensão social. Cada serviço pertence a um agente. Definições de serviços podem ser formalizadas como intenções que descrevem a condição de preenchimento desse serviço. A coleção de serviços de um agente define seu comportamento. A tabela 4.4 dá uma informal definição para os vários serviços do padrão Bidding.

Tabela 4.4: Serviços do padrão Bidding

Nome do serviço	***Definição informal***	***Agente***
publicaCandidatos	Disponibiliza uma lista dos candidatos disponíveis para preencher a vaga.	Gerenciador
respondePedido	Informa a vaga se o seu pedido de selecionar um candidato foi aceito ou não.	Gerenciador
buscaCandidatoIdeal	Busca o candidato ideal para preencher a vaga e se achar faz o pedido de um candidato para o gerenciador.	Vaga
gerenciaCompetição	Caso haja duas (ou mais) vagas competindo pelo mesmo candidato, o gerenciador define qual das vagas ficará com o candidato primeiro verificando se as vagas são da mesma empresa (cooperação: onde o gerenciador define a troca de candidatos de acordo com o critério cargo mais importante na hierarquia) ou de empresas diferentes (competição: critério empresa de maior faturamento).	Gerenciador

Dimensão Estrutural

A dimensão estrutural procura explicar como cada serviço é operacionalizado. Serviços são operacionalizados como planos, isto é, seqüência de ações. Essa dimensão é modelada utilizando um estilo de diagrama de classes estendido para a engenharia de sistemas multi-agentes. Esse modelo é apresentado na figura 4.17, onde são descritos os conceitos e relacionamentos necessários para construir a dimensão estrutural. Cada conceito define um *template* comum para classes de sistemas multi-agentes (DO, KOLP, PIROTTE, 2003). Os conceitos são:

- Crença: descreve o conhecimento que um agente possui a respeito de si próprio e do ambiente. Uma crença é uma tupla composta por uma chave e campos de valores.
- Evento: descreve um estímulo, emitido por agentes ou automaticamente gerados, no qual os agentes devem tomar alguma atitude. A figura 4.17 descreve a estrutura de um evento, que é composta por três partes: declaração de atributos do evento, declaração de métodos para criar um evento e declaração das crenças e condições usadas para um evento automático. Eventos podem ser classificados em externos ou internos, onde externo é um evento onde o agente envia para outro agente (*sent*) e interno é um evento que o agente posta para si mesmo (*post*). Ainda, eventos podem ser classificados em normal ou evento tipo BDI, onde normal é aquele que o agente só tem uma plano para responder a esse evento e um evento do tipo BDI o agente tem planos alternativos. E por último, os eventos podem ser classificados em automáticos e não automáticos, onde o primeiro é criado quando certo estado de crença surge.
- Plano: descreve uma seqüência de ações que um agente pode realizar quando um evento ocorre. Na figura 4.17, o plano é apresentado como estruturado em três partes: a parte do evento, a parte da crença e a parte do método. A parte do evento declara os eventos que iniciam a execução de um plano e eventos que os planos produzem. A parte da crença declara crenças que o plano "lê" e aquelas que ele "modifica". A parte do método descreve o próprio plano, isso é, as ações desempenhadas quando o plano é executada.

- Agente: define o comportamento de um agente, incluindo o tipo de eventos, os planos utilizados para responder a esses eventos e as crenças que possuem seu conhecimento. A estrutura do agente é composta por cinco partes: declaração dos atributos, eventos, planos, crenças e métodos.

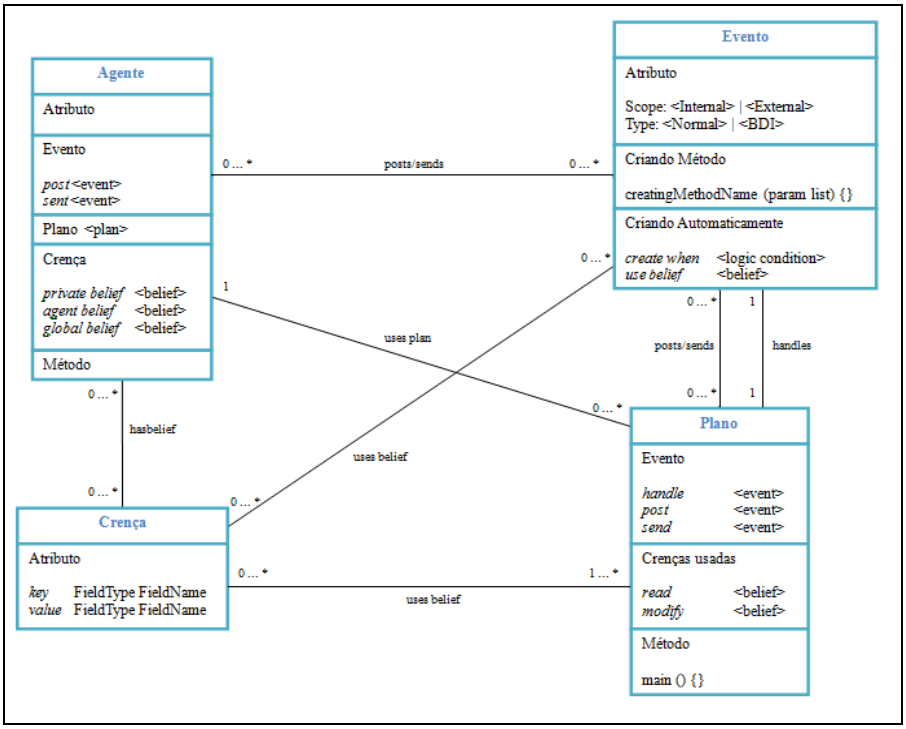

Figura 4.17: *Template* do diagrama estrutural

O agente gerenciador possui os serviços *publicaCandidato*, *respondePedido* e *gerenciaCompetição.* Na dimensão estrutural esses serviços são vistos como planos. O conhecimento sobre o mundo que o agente gerenciador possui (suas crenças) são: os candidatos existentes (*listaCandidatos*), sendo esse uma crença global (que todos os agentes conhecem); os pedidos feitos pelos agentes vaga (*listaPedidos*), sendo essa uma crença privada (só o agente gerenciador sabe sobre ela); os pedidos em conflito (*pedidosConflito*), isto é, os candidatos desejados por mais de um agente; e as vagas existentes no sistema (*listaVagas*). Além disso, existem eventos lançados de acordo com certos acontecimentos, como por exemplo, quando o gerenciador aceita o pedido de uma vaga para ceder o candidato, ele deve avisar a mesma de que o processo está resolvido. O diagrama de dimensão estrutural do gerenciador vaga pode ser visualizado na figura 4.18.

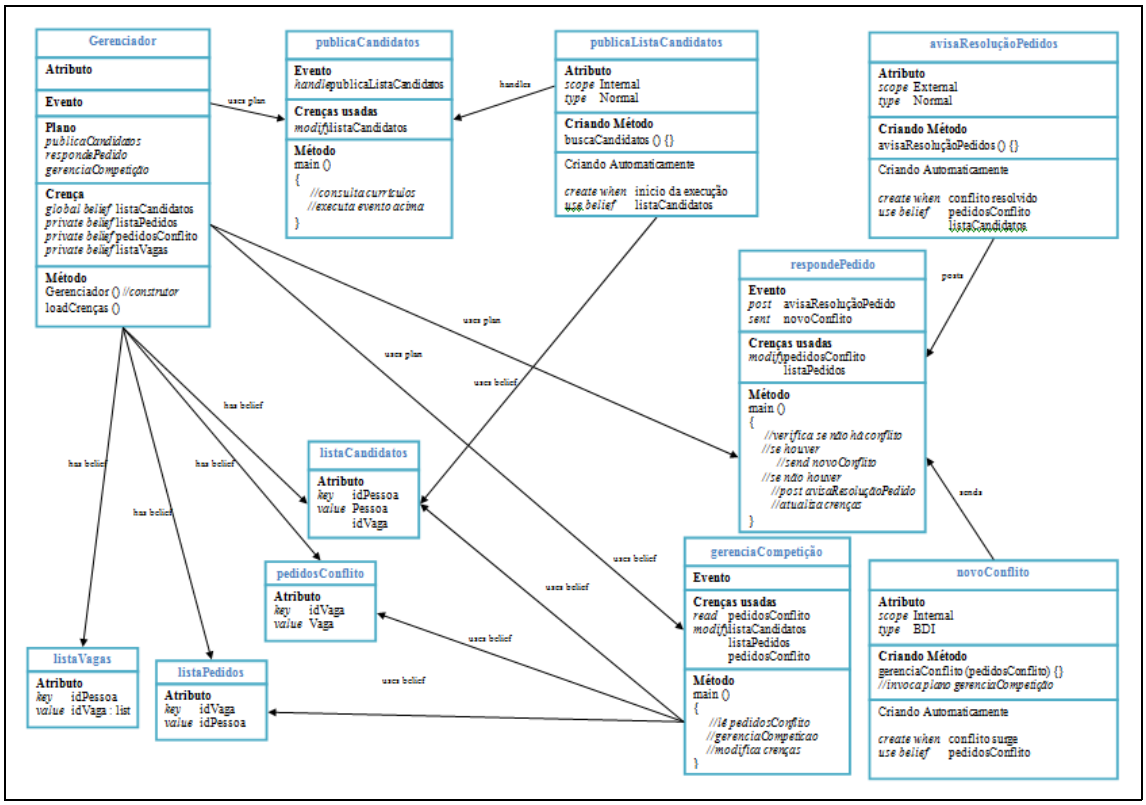

Figura 4.18: Dimensão estrutural do agente gerenciador

Já o agente vaga possui o plano *buscaCandidatoIdeal* e apenas tem como crença a lista de candidatos existentes no sistema. No início da execução do agente vaga, ele envia um evento para si mesmo (*internal*) onde ele busca o candidato adequado através do plano buscaCandidatoIdeal, onde ele acessa sua crença *listaCandidatos* e checa de acordo com seus atributos (detalhes do cargo) e com os atributos do candidato (Pessoa) quais são os currículos que estão de acordo com a vaga. No momento em que ele acha alguém adequado, ele faz um pedido ao agente gerenciador para ter o candidato para si e aguarda a resolução do mesmo. O diagrama estrutural do agente vaga pode ser visualizado na figura 4.19.

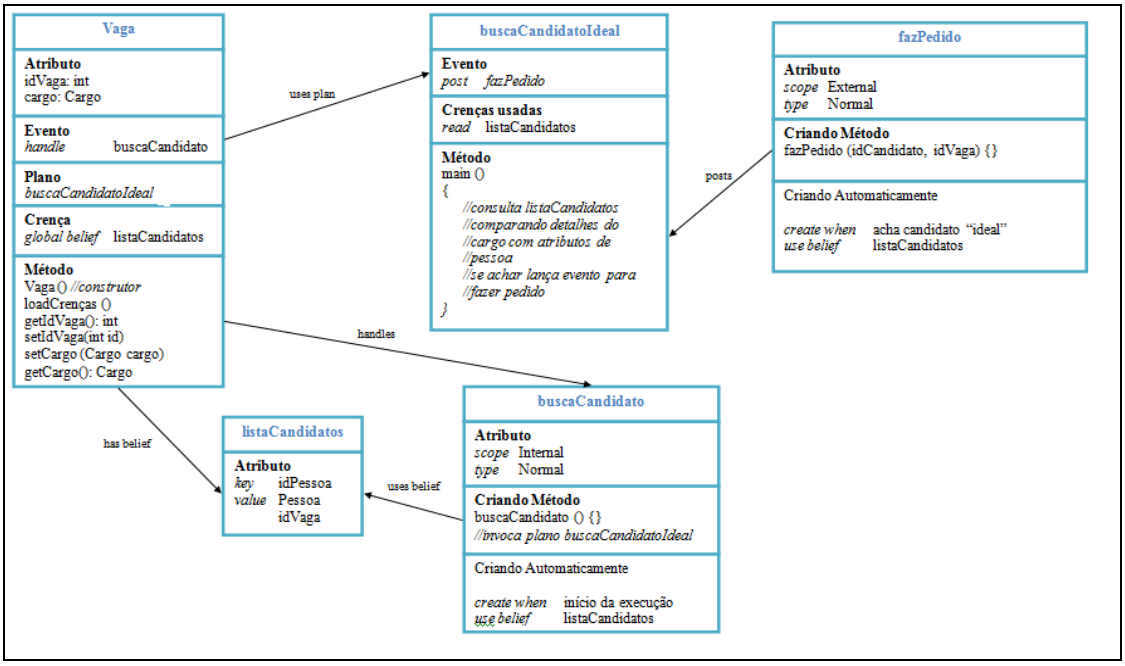

Figura 4.19: Dimensão estrutural do agente vaga

Dimensão Comunicativa

A dimensão comunicativa visa responder como é realizada a comunicação entre os agentes. Agentes interagem uns com os outros trocando eventos. Essa dimensão representa, de maneira temporal, os eventos relacionados a troca de mensagens no sistema (DO, KOLP, PIROTTE, 2003).

Para representar a dimensão comunicativa, é utilizado o diagrama de seqüências proposto pela AUML, pois o diagrama UML não provê base suficiente para modelar agentes e sistemas multi-agentes. Isso porque comparado a objetos, agentes são ativos porque eles podem ter iniciativa e controle como eles processam pedidos externos e porque eles não agem isoladamente, mas cooperando e competindo com outros agentes (BAUER, MÜLLER, ODELL, 2001).

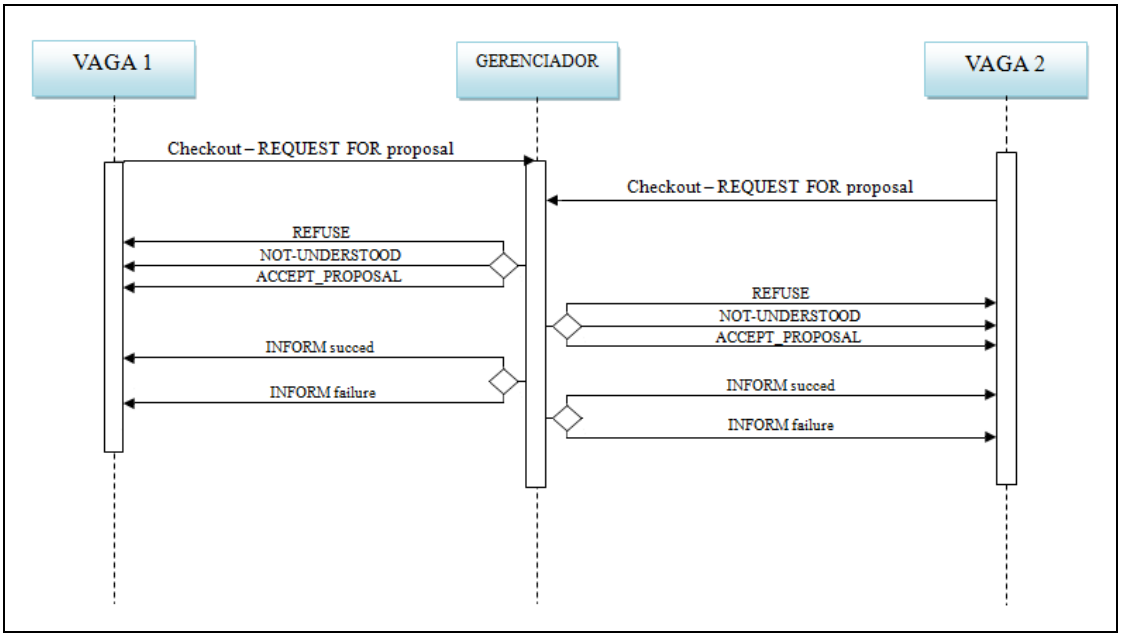

Figura 4.20: Dimensão comunicativa

O diagrama da dimensão comunicativa apresentado na figura 4.20, captura algumas das interações entre o agente vaga e o agente gerenciador. O agente vaga solicita ao gerenciador a posse de um determinado candidato. O gerenciador pode recusar (REFUSE) se houver algum problema, por exemplo, o candidato não existir mais por algum motivo; o gerenciador pode enviar uma mensagem identificando que a requisição não foi entendida (NOT-UNDERSTOOD), caso tenha algum erro na mensagem por exemplo; ou, o gerenciador pode aceitar o requisição (ACCEPT_PROPOSAL). Sendo aceita essa requisição, o gerenciador verificará se não existem conflitos, ou se existir ele resolverá, informando no final se o pedido foi aceito e a vaga pode ter a posse do candidato ou não.

Dimensão Dinâmica

Como descrito anteriormente, um plano pode ser invocado por um evento que executa e cria novos eventos. Relacionamentos entre planos e eventos podem se tornar complexos rapidamente. Para lidar com esse problema, a dimensão comunicativa busca modelar a sincronização e os relacionamentos entre planos e eventos com diagramas de atividades estendidos para sistemas orientados a agentes (BAUER, MÜLLER, ODELL, 2001). Esses diagramas especificam os eventos que são criados em paralelo, as condições nas quais um

evento é criado, qual plano lida com qual evento e assim por diante. Um diagrama de atividade pode ser considerado como um diagrama de interação cujo interior é revelado (BOOCH, RUMBAUGH, JACOBSON, 2000).

Um evento interno é representado por uma seta tracejada e um evento externo por uma seta sólida. Como mencionado anteriormente, um evento BDI pode ser tratado por planos alternativos. Esse evento é ilustrado como uma caixa de cantos arredondados assim como o plano. Em um nível inferior, cada plano pode ser também modelado por um diagrama de atividades se necessário mais detalhes (DO, KOLP, PIROTTE, 2003).

Na figura 4.21 é apresentado o diagrama da dimensão comunicativa para esse trabalho, onde o processo inicia com o agente vaga que lança o evento interno *buscaCandidato*. Esse evento invoca o plano *buscaCandidatoIdeal*, que por sua vez lança o evento externo *fazPedido* se comunicando com o agente gerenciador. Dessa forma, dispara o plano *respondePedido* onde se não houver conflito, o evento externo *avisaResoluçãoPedido* informa ao agente vaga uma mensagem de pedido aceito ou alguma falha devido a má-comunicação, conforme citado no diagrama da dimensão comunicativa. Se houver conflito, é disparado o evento interno *novoConflito.* Esse evento é um evento BDI, que pode executar vários planos. O plano previsto nesse trabalho é o *gerenciaCompetição*, onde será resolvido o conflito de disputa de vagas. Após a resolução, será novamente lançado o evento *avisaResoluçãoPedido* informando ao agente vaga a solução do processo. Na figura 4.22 é possível visualizar detalhadamente o plano *gerenciaCompetição*.

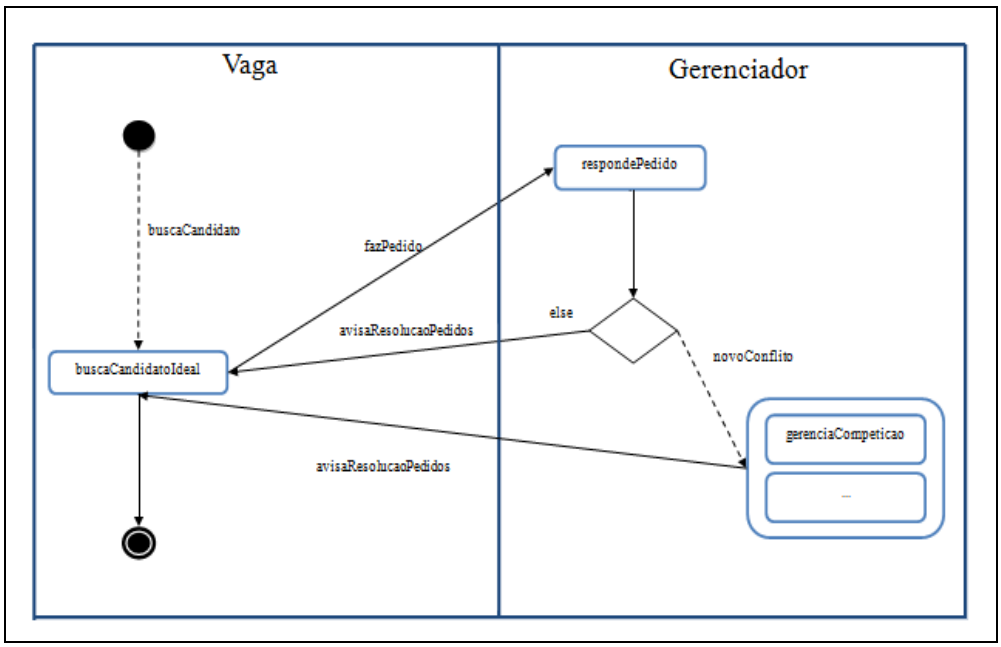

Figura 4.21: Dimensão dinâmica

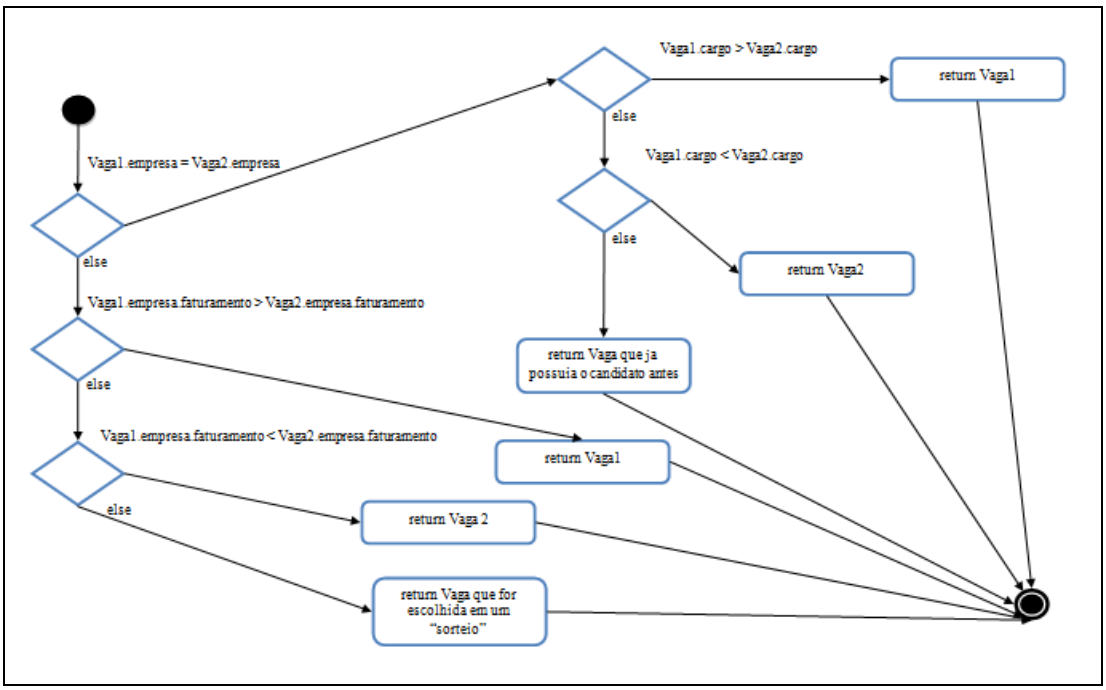

Figura 4.22: Diagrama de atividades de *gerenciaCompetição*

A fase de implementação segue a especificação do projeto detalhado para a geração de um esqueleto para a implementação dos sistemas multi-agentes. Isso é feito através de um mapeamento entre os conceitos de Tropos e os elementos de uma plataforma de implementação de agente, tal como JACK (JACK, 2006) ou JADE (JADE, 1998). Esse mapeamento serve para guiar o processo de implementação em determinada plataforma. O código do sistema deve complementar a estrutura básica de implementação, oferecida pelo esqueleto do sistema através do mapeamento usando a linguagem de programação suportada pela plataforma de implementação de agentes escolhida. Por exemplo, utilizando JACK, os conceitos atores, recursos, metas, metas flexíveis e tarefas do *framework* i* seriam mapeados para o modelo BDI como agentes, crenças, desejos e intenções respectivamente. Então, um agente BDI mapeado como agente de JACK, seria uma crença declarada (ou retirada) como uma relação da base de dados, um desejo seria publicado como *BDIGoalEvent* (objetivo que um agente deseja alcançar) e manipulado como um plano e uma intenção é implementada como um plano. Por fim, uma dependência seria realizada através de um *BDIMessageEvent* Utilizando a plataforma JACK, o código do sistema deve ser implementado em JAVA.

Assim como existem ferramentas geradoras de código para o uso de UML por exemplo, a ferramenta SkwyRL (DO, KOLP, PIROTTE, 2003) produz código genérico para os padrões sociais descritos nesse trabalho, em formato aceito pela plataforma JACK,

facilitando assim o processo de desenvolvimento. Já em relação a plataforma JADE, ainda não existe um gerador de código que auxilie esse processo.

4.3 Considerações Finais

A metodologia Tropos apresenta um *framework* que suporta as fases de desenvolvimento de software de análise de requisitos (requisitos iniciais e finais), projeto arquitetural e projeto detalhado. A parte de implementação é orientada para a plataforma JACK, embora a metodologia busque detalhar tanto quanto possível a especificação da estrutura, comunicação e comportamento dos agentes para que possibilite a implementação em qualquer plataforma disponível.

A vantagem é que essa metodologia provê um nível mais alto de modelagem, onde há uma melhor compreensão dos objetivos e do ambiente de uma organização, já que o modelo é visto em termos de atores, responsabilidades, objetivos, tarefas, e não orientado a implementação.

Como desvantagens, além de TROPOS não possuir um conjunto de ferramentas que suporte todas as fases da metodologia, a metodologia não possui algumas atividades importantes no desenvolvimento de software, tais como testes, distribuição, gerência de configuração, entre outras. É necessária a criação de mecanismos de estruturação adequados que permitam construir modelos simplificados e fáceis de entender quando aplicados a domínios complexos (SILVA, 2005).

5 CONCLUSÕES

O paradigma orientado a agentes tem um grande potencial de se tornar um solução de engenharia de software popular. Entretanto, para que isso aconteça, é preciso que essa tecnologia seja mais difundida, que os possíveis usuários sintam-se mais confiantes e familiarizados com a noção de agentes e suas aplicações.

Pela tecnologia não ser tão conhecida, há carências em relação à ausência de metodologias sistêmicas e ferramentas de suporte que facilitem a especificação e estruturação de aplicações complexas como sistemas orientados a agentes.

Outra barreira de aceitação desse paradigma é entender quando é apropriado usá-lo, e como estabelecer uma melhor relação entre a tecnologia de agentes e uma tecnologia antecedente (por exemplo, o paradigma orientado a objetos) de forma que o paradigma orientado a agentes seja uma extensão incremental de métodos conhecidos e confiáveis.

Todavia, mesmo com os argumentos apresentados acima, a abordagem orientada a agentes pode melhorar significantemente o processo de desenvolvimento de software, para certos tipos de aplicação citados anteriormente (JENNINGS, BUSSMANN, 2003).

Com esse trabalho foi possível estudar sobre esse paradigma e aplicar o uso de uma metodologia, a metodologia Tropos na modelagem de um sistema de seleção de pessoas para vagas disponíveis em empresa. Pôde-se perceber que, no uso de agentes, uma metodologia como a orientada a objetos dificulta o desenvolvimento, visto que, não considera certas características dos mesmos, como a habilidade social, por exemplo.

Uma vantagem a ser destacada na metodologia Tropos é o uso de conceitos organizacionais na análise de requisitos. O sistema é visualizado através de metas, tarefas, recursos, atores e não de uma forma orientada a implementação. Assim, essa abordagem possui um alto nível de abstração com relação as peculiaridades e comportamentos de sistemas de software complexos.

É importante salientar que, no projeto detalhado, onde preocupa-se em criar um esqueleto para a implementação, a metodologia Tropos direciona para o uso da plataforma JACK. Isso limita sua utilização de certa forma, visto que a plataforma JACK é paga. Então, o

trabalho de (SILVA, 2005) buscou modificar algumas fases para orientar a implementação para a plataforma JADE, que é software livre. Ainda assim, esse trabalho seguiu as especificações iniciais da metodologia, pois estavam mais claras e possuíam mais referências bibliográficas.

Como trabalhos futuros, o principal a ser feito é implementar esse sistema, seja direcionando para a plataforma JACK ou JADE, com as modificações necessárias.

6 REFERÊNCIAS BIBLIOGRÁFICAS

Agent Oriented Software Group. **JACK Development Environment**. Disponível em: <http://www.agent-software.com/shared/products/index.html>. Acesso em: 17 julho 2008.

BAUER, B.; MÜLLER, J. P.; ODELL, J., **Agent UML: A Formalism for Specifying Multiagent Interaction** in Workshop on Agent-Oriented Software Engineering. Limerick, Ireland, 2001. Disponível em: <http://citeseer.ist.psu.edu/cache/papers/cs/22633/http:zSzzSzjamesodell.comzSzAOSE-Bauer.pdf/agent-uml-a-formalism.pdf>. Acesso em: 17 julho 2008.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML - Guia do Usuário**. São Paulo: Campus, 2000.

BRAGA, F. **O contexto actual do mundo dos negócios e a estratégia de Recursos Humanos**. In: Capital Intelectual: Fator crítico de sucesso. Portugal, 2007. Disponível em: <http://recursoshumanos.blog.com/1643954/>. Acesso em: 2 maio 2008.

BRATMAN, M. E. **Intention, Plans, and Practical Reason**. Cambridge: CSLI Publications, 1987.

BRESCIANI, P. et al., **A Knowledge Level Software Engineering Methodology for Agent Oriented Programming** in Autonomous Agent. Montreal, 2000. Disponível em: <http://coblitz.codeen.org:3125/citeseer.ist.psu.edu/cache/papers/cs/25590/http:zSzzSzsra.itc.itzSztrzSzPGMzPz00.pdf/perini00knowledge.pdf>. Acesso em: 16 maio 2008.

BRESCIANI, P.; SANNICOLÒ, F., **Requirements Analysis in Tropos: a self referencing example** in Net.ObjectDays. Erfurt, 2002. Disponível em: <http://citeseer.ist.psu.edu/rd/59936630%2C552359%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/26838/http:zSzzSzwww.netobjectdays.orgzSznode02zSzdezSzConfzSzpublishzSz..zSz..zSz..zSz..zSzpdfzSz02zSzpaperszSzws-ageszSz0800.pdf/bresciani02requirements.pdf>. Acesso em: 16 maio 2008.

CASTRO, J.; KOLP, M.; MYLOPOULOS, J., **Towards Requirements-Driven Information Systems Engineering** in Information Systems. Amsterdam, 2002. Disponível em: <http://citeseer.ist.psu.edu/rd/59936630%2C444227%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/22305/http:zSzzSzwww.cs.toronto.eduzSz%7EmkolpzSztroposzSzelsvex.pdf/castro02towards.pdf>. Acesso em: 16 maio 2008.

CASTRO, J.; SILVA, C.; TEDESCO, P.; SILVA, I., **Describing Agent-Oriented Design Patterns in Tropos** in Proceedings of the 2006 international workshop on Software engineering for large-scale multi-agent systems. Shanghai, China, 2005. Disponível em:

<http://www.sbbd-sbes2005.ufu.br/arquivos/01-%209556.pdf>. Acesso em: 13 julho 2008.

CAUTELA, A.; POLLONI, E. F. G. **Sistemas de informação na administração de empresas**. São Paulo: Editora Atlas, 1976.

CERVENKA, R.; TRENCANSKY, I. **AML The Agent Modeling Language**. Berlin: Birkäuser, 2000.

CHIAVENATO, I. **Recursos Humanos: O Capital Humano das Organizações**. São Paulo: Editora Atlas, 2004.

CHIAVENATO, I. **Gestão de Pessoas: O novo papel dos recursos humanos nas organizações**. Rio de Janeiro: Editora Campus, 2002.

CHIAVENATO, I. **Administração de Recursos Humanos: Fundamentos Básicos**. São Paulo: Editora Atlas, 1999.

CHUNG, L.; NIXON, B. A.; YU, E., MYLOPOULOS, J. **Non-Funcional Requirements in Software Engineering**. Boston: Kluwer Academic Publishers, 2000.

COSTA, C. G. A. **Desenvolvimento e Avaliação Tecnológica de um Sistema de Prontuário Eletrônico do Paciente, Baseado nos Paradigmasda World Wide Web e da Engenharia de Software**. 2001. 268 f. Dissertação (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação, Campinas, SP, 2001.

DO, T. T.; KOLP, M.; PIROTTE, A., **Social Patterns for Designing Multiagent Systems** in 15th Int. Conf. on Software Engineering and Knowledge Engineering. San Francisco, 2003. Disponível em: <http://citeseer.ist.psu.edu/rd/31923020%2C573065%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/27491/http:zSzzSzwww.cs.toronto.eduzSz%7EmkolpzSzseke03.pdf/do03social.pdf>. Acesso em: 16 julho 2008.

DUTRA, J. S. **Gestão de Pessoas: modelos, processos, tendências e perspectivas**. São Paulo: Editora Atlas, 2002.

FIOCRUZ. **Plano de Carreira**. Disponível em: <http://www.direh.fiocruz.br/planofiocruz/conceitos.htm>. Acesso em: 2 maio 2008.

Java Agent DEvelopment Framework. **CSELT**. Disponível em: <http://jade.tilab.com>. Acesso em: 11 julho 2008.

JENNINGS, N. R.; WOOLDRIDGE, M., **Applications of Intelligent Agents** in Agent Technology: Foundations, Applications, and Markets (eds. N. R. Jennings and M. Wooldridge. London, 1998. Disponível em: <http://agents.umbc.edu/introduction/jennings98.pdf>. Acesso em: 2 maio 2008.

JENNINGS, N.; BUSSMANN, S. **Agent-based control systems**. In: , 2003, Berlin,

Germany**.** **Anais...** : , 18 julho 2008. Disponível em: <http://eprints.ecs.soton.ac.uk/8563/1/ieeeCS.pdf>. Acesso em: .

KIMURA, E. S. **A importância da gestão de pessoas nas organizações em mudança**. In: ARTIGOS.COM. São Paulo, 2006. Disponível em: <http://www.artigos.com/artigos/sociais/administracao/recursos-humanos/gestao-de-pessoas-1068/artigo/>. Acesso em: 12 julho 2008.

KOLP, M.; CASTRO, J.; MYLOPOULOS, J., **A social organization perspective on software architectures** in Proc. of the 1st Int. Workshop From Software Requirements to Architectures. Toronto, 2001. Disponível em: <http://citeseer.ist.psu.edu/cache/papers/cs/22305/http:zSzzSzwww.cs.toronto.eduzSz~mkolpzSztroposzSzstraw.pdf/kolp01social.pdf>. Acesso em: 12 julho 2008.

KOLP, M.; DO, T. T.; FAULKNER, S,; PIROTE, A., **Introspecting Agent-Oriented Design Patterns** in Advances in Software Engineering and Knowledge Engineering. Pittsburgh, 2005. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.7529&rep=rep1&type=pdf>. Acesso em: 13 julho 2008.

KOLP, M.; GIORGINI, P.; MYLOPOULOS, J. **Information Systems Development through Social Structures**. In: 14th International Conference on Software Engineering andKnowledge Engineering (SEKE'02), 2002, Ishia**. Anais...** Italy: University of Louvain, 2002. Disponível em: <http://citeseer.ist.psu.edu/rd/0%2C515509%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/26047/http:zSzzSzwww.cs.toronto.eduzSz%7EmkolpzSzseke02infosoc.pdf/kolp02information.pdf>. Acesso em: 26 maio 2008.

LIMMONGI-FRANÇA, A. C.; ARELLANO, E. B. **As pessoas nas organizações: Os processos de recrutamento e seleção**. São Paulo: Editora Gente, 2002.

MORAITIS, P.; SPANOUDAKIS, N. I. , **Combining Gaia and JADE for Multi-Agent Systems Development** in Fourth International Symposium "From Agent Theory to AgentImplementation" (AT2AI'04). Viena, 2004. Disponível em: <http://jade.tilab.com/papers/AT2AI4_Moraitis_final.pdf>. Acesso em: 18 julho 2008.

MYLOPOULOS, J.; CASTRO, J., **Tropos: A Framework for Requirements-Driven Software Development** in Information Systems Engineering: State of the Art and Research Themes. London, 2000. Disponível em: <http://www.cs.toronto.edu/~mkolp/tropos1.pdf>. Acesso em: 4 maio 2008.

ODELL, J., **Representing Agent Interaction Protocols in UML** in AOSE. Limerick, 1999. Disponível em: <http://citeseer.ist.psu.edu/rd/59936630%2C672152%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/32308/http:zSzzSzwww.ca.sandia.govzSzFIPAPDMzSzodell.pdf/odell99representing.pdf>. Acesso em: 4 maio 2008.

ODELL, J., **Agent Technology - Green Paper** in Agent Working Group OMG Document. Needham, 2000. Disponível em: <http://citeseer.ist.psu.edu/cache/papers/cs/16218/http:zSzzSzjamesodell.comzSzec2000-08-0

1.pdf/group00agent.pdf>. Acesso em: 02 maio 2008.

PADILHA, T. P. P.; JACOMÉ, T. F., **O Uso de Técnicas de Modelagem de Agente em Ambientes Educacionais** in VI Congresso Iberoamericano de Informática Educativa. Vigo, Espanha, 2002. Disponível em: <http://lsm.dei.uc.pt/ribie/docfiles/txt2003729192912paper-226.pdf>. Acesso em: 11 julho 2008.

PIGOSKI, T. M. **Practical Software Maintenance - Best Practices for Managing your Investiment**. USA: John Wiley & Sons, 1997.

PRESSMAN, R. S. **Engenharia de Software**. São Paulo: Makron Books, 1995.

SILVA, C. T. L. L. **Detalhando o projeto arquitetural no desenvolvimento de software orientado a agentes: O caso Tropos**. 2003. 168 f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Pernambuco, Recife, 2003.

SILVA, I. G. L. **Projeto e Implementação de Sistemas Multi-Agentes: O Caso Tropos**. 2005. 108 f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Pernambuco, Recife, 2005.

STAIR, R. M.; REYNOLDS, G. W. **Princípios de Sistemas de Informação, Uma abordagem Gerencial**. São Paulo: Editora Thomsom, 2006.

SUGUMARAN, V. **Applications of Agents and Intelligent Information Technologies**. London: Idea Group Publishing, 2007.

TVEIT, A. **A survey of Agent-Oriented Software Engineering**. In: Proc. of the First NTNU CSGS Conference, 2001, Trondheim**. Anais...** Trondheim: Norwegian University of Science and Technology, 2001. Disponível em: <http://citeseer.ist.psu.edu/cache/papers/cs/22000/http:zSzzSzcavenan.idi.ntnu.nozSzamundzSzpublicationszSz2001zSzASurveyOfAgentOrientedSoftwareEngineering.pdf/tveit01survey.pdf>. Acesso em: 02 maio 2008.

WIKCIONÁRIO. **Conceito de Sinergia**. Disponível em: <http://pt.wiktionary.org/wiki/sinergia>. Acesso em: 21 julho 2008.

WIKIPEDIA. **Conceito de Impedância**. Disponível em: <http://pt.wikipedia.org/wiki/Imped%C3%A2ncia>. Acesso em: 17 julho 2008.

WOOLDRIDGE, M. **An Introduction to Multiagents Systems**. Chichester: John Wiley & Sons LTD, 2002.

WOOLDRIDGE, M.; CIANCARINI, P. **Agent-Oriented Software Engineering:The State of the Art**. In: First Int. Workshop on Agent-Oriented Software Engineering, 2001, Limerick**. Anais...** Liverpool, Bologna: University of Liverpool; University of Bologna, 2001. Disponível em: <http://www.csc.liv.ac.uk/~mjw/pubs/aose2000a.pdf>. Acesso em: 2

maio 2008.

WOOLDRIDGE, M.; JENNINGS, N. R.; KINNY, D., **The Gaia Methodology for Agent-Oriented Analysis and Design** in Journal of Autonomous Agents and Multi-Agent Systems. Netherlands, 2000. Disponível em: <http://citeseer.ist.psu.edu/rd/59936630%2C513675%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/26098/http:zSzzSzwww.ecs.soton.ac.ukzSz%7EnrjzSzdownload-fileszSzjaamas2000.pdf/wooldridge00gaia.pdf>. Acesso em: 03 maio 2008.