Projeto Arquitetural de Sistema Multi-Agente sob o Conceito de *Groupware*

Stefan Oliveira[1]*, **João Carlos Gluz**[1]

[1]Universidade do Vale do Rio dos Sinos – UNISINOS
Av. Unisinos, São Leopoldo, RS.

stefanoliver@gmail.com, jcgluz@unisinos.br

Abstract. *The work presented in this paper is part of a project in development that intends to implement an application that helps Project Managers at selection of professionals to integrate software development teams. The creation of Multi-Agent Systems involves the selecting of architectures that best fit the context and its restrictions. Since some different methodologies to develop these systems exist, it seems to be quite interesting the integration of practices and techniques to obtain a better result at implementation. To concerning systems that have the concept of collaboration involved in the process of integration between the agents is necessary to create a specific architecture, which in this paper it happens from the integration between practices of TROPOS methodology and methods obtained from developing Intelligent Tutoring Systems.*

Resumo. *O conteúdo apresentado neste trabalho é parte de um projeto em desenvolvimento, cujo objetivo é implementar um aplicativo que auxilie Gerentes de Projeto na seleção de profissionais para integrar equipes de desenvolvimento de software. A criação de Sistemas Multi-Agente (MAS - Multi-Agent System) envolve a seleção de arquiteturas que melhor se encaixam ao contexto e suas restrições. Sabendo que existem diferentes metodologias de desenvolvimento MAS, parece ser bastante interessante que práticas e técnicas adequadas sejam integradas a fim de obter um melhor resultado na aplicação. O trabalho mostra a integração entre práticas da metodologia TROPOS e métodos derivados do desenvolvimento Sistemas Tutores Inteligentes que resultam na elaboração de uma arquitetura específica ao domínio, abordando o conceito de colaboração envolvido no processo de integração entre os agentes.*

1. Introdução

Quando uma dada tarefa ou projeto requer a formação de uma equipe de profissionais para sua conclusão, então, naturalmente há um processo seletivo para escolha destes profissionais. Este processo consiste basicamente de avaliações e inferências com base nas características (conhecimentos, habilidades e atitudes) dos profissionais que estão concorrendo a uma vaga na equipe. Em organizações ou empresas de desenvolvimento de software essa realidade não seria diferente.

Com o propósito de automatizar e ao mesmo tempo agregar aspectos mais formais à formação de equipes, este trabalho propõe aplicar conceitos de agentes cognitivos e sistemas baseados em agentes para auxiliar neste processo. A utilização de tais conceitos e técnicas de AOSE (*Agent-Oriented Software Engineering*) tem se mostrado justificável pela possibilidade de considerar características psicológicas como desejos, intenções e

*Supported by CAPES.

crenças no projeto do sistema de auxílio ao processo de seleção, tornando este sistema mais realista no que diz respeito a aspectos humanos de interesse. Portanto, para que seja possível representar tais características (desejos, intenções e crenças) em um sistema, é necessário que uma arquitetura multi-agente seja elaborada levando em consideração aspectos de cooperação existente dentro do ambiente organizacional.

O trabalho está dividido em três seções, primeiramente uma introdução (seção atual), seguida da base teórica e finalizando com a seção de modelagem, que mostra detalhes da utilização das técnicas apresentadas na seção 2..

2. Base Teórica

Uma vez que se deseja construir um (*Multi-Agent System*) - MAS é preciso tomar como iniciativa a escolha de práticas e métricas para guiar as atividades de desenvolvimento deste sistema. Existem várias metodologias ou estruturações de métodos para auxiliar desenvolvedores no processo de engenharia de software de sistemas multi-agente. Tais metodologias, normalmente, definem uma seqüência lógica de ações a serem executadas que envolvem desde a representação do domínio do problema à elicitação de requisitos, elaboração de arquitetura, escolha de tecnologias e etc. Considerando os propósitos deste trabalho é possível perceber que existem duas estruturações bem elaboradas de práticas para guiar o desenvolvimento de MAS bastante relevantes ao processo de interesse.

A Metodologia TROPOS [TroposProject, 2007] num primeiro momento nos parece ser muito útil nas atividades de compreensão e representação do domínio de aplicação. Assim como os métodos extraídos de experiências com o desenvolvimento de *Intelligent Tutoring Systems* - ITS [Vicari and Gluz, 2007] que fornecem alguns critérios de aplicabilidade para que o desenvolvedor se certifique de que o paradigma orientado a agentes é o mais adequado ao sistema a ser desenvolvido.

2.1. TROPOS

De acordo com [Bresciani and Mylopoulous, 2004], TROPOS é uma metodologia de desenvolvimento de sistemas orientados a agentes. Possui como principais características a idéia de agência, objetivo, planos e vários outros níveis de conhecimento utilizados em todas as fases de desenvolvimento, desde a fase inicial de análise até a implementação. Outra característica chave de TROPOS é o fato de dar grande importância à fase inicial de especificação e análise de requisitos, permitindo assim uma compreensão mais detalhada do ambiente de implantação do sistema e dos possíveis tipos de interação que podem ocorrer entre o sistema e seus usuários.

A metodologia TROPOS define cinco fases para o processo de desenvolvimento de MAS. As duas primeiras fases: Requisitos Iniciais (*Early Requirements*) e Requisitos Finais (*Late Requirements*), tratam basicamente da elicitação e análise de requisitos e especificação do domínio. A terceira, Projeto Arquitetural (*Architecture Design*), trata de aspectos estruturais, que envolvem a escolha de um estilo arquitetural e padrões sociais para o sistema. A quarta fase, Projeto Detalhado (*Detailed Design*), define questões referentes à comunicação e comportamento de cada componente do sistema. E por fim, na quinta e última fase, Implementação (*Implementation*), é feito um mapeamento entre os conceitos de TROPOS e os elementos de uma plataforma de implementação de agentes [Silva, 2005].

2.2. Framework I*

Segundo [Yu and Mylopoulos, 1994], i* tem como objetivo permitir a representação de aspectos organizacionais envolvidos com processos, descrevendo, por meio de modelos,

motivações e aspectos de intencionalidade que envolvam atores em um ambiente organizacional. Dentre as muitas técnicas de modelagem de ambientes organizacionais, i* tem se destacado por possibilitar uma melhor expressão dos ”porquês” associados a práticas e estruturas organizacionais existentes. Podendo ser destacados os objetivos gerais da organização, intenções dos atores da organização em relação a sistemas pretendidos, detalhamento das razões associadas com atores para atingir determinados objetivos e descrição de requisitos não funcionais. Desta forma, possibilita aos desenvolvedores investigar dependências estratégicas entre atores, bem como as razões estratégicas destes atores. Para descrever o ambiente organizacional, i* propõe dois modelos: o *Strategic Dependencies Model* - SD e o *Strategic Reasons Model* - SR.

O modelo SD é composto graficamente por nós e ligações. Os nós são representações de atores no ambiente e as ligações são representações das dependências entre os atores. Ator é o nome dado a uma entidade que realiza ações para obter objetivos no contexto do ambiente. Atores têm relações de dependência entre si para obterem objetivos. Ao ator que depende de alguma forma de outro é dado o nome Depender e o ator que atende e satisfaz a dependência recebe o nome Dependee. O elemento de dependência entre Depender e Dependee é denominado de Dependum. Sendo assim, há uma relação do tipo: $Depender \rightarrow Dependum \rightarrow Dependee$.

As dependências apresentadas no SD podem ser de diferentes tipos, tendo como base o tipo do Dependum. Existem quatro tipos de dependências, que poder ser: dependência de objetivo, de tarefa, de recurso ou de objetivo-soft. Diferentemente do SD, o SR permite observar e representar de forma gráfica as razões associadas a cada ator e suas dependências. Diferentemente do SD onde são modelados somente os relacionamentos externos entre atores, o SR permite maior compreensão das razões estratégicas de atores do ambiente em relação a processos da organização e como os mesmos são representados. Este modelo pode ser desenvolvido a partir da observação das razões associadas a cada ator em relação às dependências com outros atores. Tais dependências podem ser mais facilmente decompostas através da observação de como os Dependees podem satisfazer os Dependums associados aos mesmos e, a partir deste ponto, observar e decompor as intenções e razões organizacionais estratégicas como um todo.

2.3. Métodos AOSE Derivados de ITS

O conjunto de métodos de engenharia de software proposto em [4] fornece vários critérios de aplicabilidade, princípios de projeto e guias de desenvolvimento que podem ser úteis para analisar, projetar e desenvolver um sistema multi-agente BDI. Estes métodos foram abstraidos a partir da observação empírica de como vários sistemas tutores inteligentes foram construídos. No entanto, eles ainda não foram testados em domínios e aplicações não relacionadas com ITS e ambientes inteligentes de aprendizagem. O trabalho apresentado aqui é uma das primeiras aplicações destes métodos fora da área de ITS. De acordo com [4], as atuais metodologias AOSE (possivelmente com exceção de TROPOS) não consideram abstrações cognitivas de agentes desde o início do processo de desenvolvimento do sistema, incluindo a fase de engenharia de requisitos. Abstrações de cognição tais como crenças, objetivos, intenções e relações sociais baseadas em modelos cognitivos, deveriam constituir a base para a extração de propriedades de domínios alto-nível, o que não somente pode ser intuitivamente compreendido, mas constituirá a base de especificação de requisitos da aplicação.

Os métodos propostos em [4] são baseados em uma abordagem descendente para a análise, projeto e desenvolvimento de aplicações multi-agente. O primeiro passo é verificar se vários critérios de aplicabilidade são satisfeitos na fase de elicitação de requisitos do processo de engenharia de software. A idéia principal por detrás destes critérios é o

de assegurar que os princípios de projeto e guias de desenvolvimento propostos para o projeto de MAS-BDI poderão ser aplicados após a fase de análise. Tais critérios forçam engenheiros de requisitos a considerar as abstrações de agente desde o início do processo de análise de domínio e elicitação de requisitos. Se a aplicação satisfizer os critérios propostos, a especificação irá naturalmente incorporar os conceitos e abstrações de agentes.

Os critérios de aplicabilidade compreendem seis diferentes aspectos que devem ser considerados quando se deseja desenvolver um sistema multi-agente. Primeiramente, é necessário verificar se o domínio de aplicação realmente inclui entidades que podem ser entendidas como agentes trabalhando juntos em um sistema organizado. Os passos seguintes estabelecem critérios sobre a estruturação de crenças de agentes, a utilização de modelos cognitivos pelos agentes, como interações de comunicação devem ocorrer, e o estabelecimento de relações sociais entre os agentes. Eles ainda estabelecem que requisitos de aplicação atribuídos a agentes devem ser claramente indicados por uma especificação que determina qual conhecimento (extraído de uma base de crenças) é necessário para satisfazer tais exigências.

Após os critérios terem sido aplicados com êxito, vários princípios de projetos são propostos, mostrando como o domínio de aplicação deve ser dividido, em pelo menos, em três subdomínios distintos: o subdomínio *Problem Solving* (PS) contém conhecimento sobre os problemas específicos que a aplicação se destina a resolver (e não estão contidos em outros subdomínios); o subdomínio *Users and Agents Modelling* (UAM) contém conhecimento sobre os usuários (ou agentes externos) que interagem com a aplicação; e o subdomínio *Social Mediated Interactions* (SMI) contém o conhecimento sobre mecanismos de interação social necessários para mediar a comunicação entre a aplicação e os seus usuários (ou agentes externos).

Uma arquitetura específica MAS (que pode ser chamada de "Triad Architecture"), é então proposta para lidar com estes domínios. Esta arquitetura é composta por três tipos de agentes: *PS Agentes* que trabalham no subdomínio PS e solucionam problemas na aplicação; *UAM Agentes* que fornecem serviços de interface à aplicação e fazem modelos cognitivos dos seus usuários e agentes externos; e *SMI Agentes* que incorporam mecanismos de interação social. Os subdomínios não precisam estar completamente separados. É possível (e até necessário) que existam intercessões não-nulas entre subdomínios, contendo conhecimento que relacione conceitos em ambos os subdomínios, conforme mostrado na Figura 1.

Guias de desenvolvimento que devem ser aplicados durante o processo de implementação e teste de sistema são também apresentados em [4]. Estes guias provêem idéias úteis sobre como transformar a arquitetura da aplicação projetada de acordo com os princípios de projeto propostos em sistemas eficazes.

2.4. Agentes Cognitivos

Agentes cognitivos são estruturas de software cuja principal característica é o fato de possuírem estados internos. Estes estados internos de agentes seriam correspondentes aos estados mentais humanos, que apresentam um vínculo com o mundo em termos de sua existência e significância. Portanto, os estados internos de um agente cognitivo se relacionam com estados do ambiente com o qual o agente interage a fim de realizar ações diversas.

Para representar a arquitetura cognitiva de agentes, implementa-se modelos formais cognitivos, chamados modelos BDI (*Belief, Desire and Intention*). Estes modelos são baseados em estados mentais, e tem sua origem no modelo de raciocínio prático humano [Bratman, 1988]. O nome atribuído ao modelo é justificado pelos seus estados men-

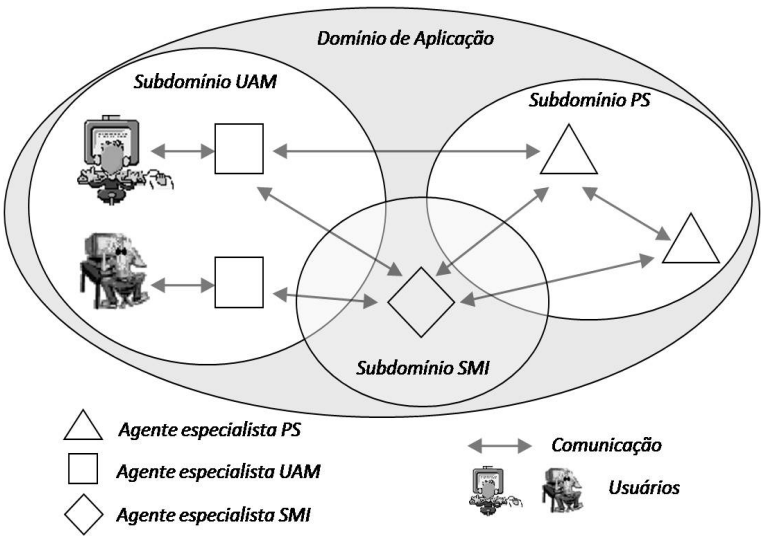

Figure 1: Arquitetura Triad, [Vicari and Gluz, 2007].

tais: crença, desejo e intenção. Uma arquitetura baseada no modelo BDI representa seus processos internos através dos estados mentais acima citados, e define um mecanismo de controle que seleciona de maneira racional o curso das ações.

2.5. Trabalho Cooperativo Apoiado por Computador

Computer Supported Cooperative Work (CSCW)[Grudin, 1994] (leia-se em português, Trabalho Cooperativo Apoiado por Computador) é definido como sendo um campo de pesquisa que diz respeito a projetos de sistemas que têm o computador como base. Tais sistemas devem dar suporte e melhoria ao trabalho em grupo, principalmente na realização de tarefas e objetivos comuns.

Podem ser encontradas diferentes interpretações para CSCW, em que normalmente seus focos alternam entre aspectos que abrangem a equipe, o individual ou o cooperativo. Este campo de estudo não está restrito a tópicos como: cooperação ou trabalho, mas estuda também conceitos como: competitividade, socialização e participação, e envolve a todos que estejam interessados no projeto de software e comportamentos sociais e organizacionais. Incluindo pessoas do próprio negócio, cientistas da computação, psicólogos organizacionais, pesquisadores da comunicação e antropologistas, por exemplo.

Ao ambiente computacional que implementa os processos de apoio à cooperação/colaboração, e assim possibilita o trabalho, a produção em conjunto e a troca de informações, denomina-se *Groupware* ou Sistema de Trabalho Cooperativo Apoiado por Computador. Conforme [Greif, 1988], *groupware* é uma classe de softwares projetados para considerar o trabalho em grupo de maneira integral. Estes produtos vão desde ferramentas de coordenação como correio eletrônico, a sistemas de gestão e controle de atividades. Tais sistemas têm em comum o fato de deixar sob o controle dos membros do grupo a coordenação da tecnologia (*groupware*), dando-lhes acesso aos aspectos positivos de coordenação, não apenas prevenindo conflitos, mas permitindo que haja colaboração.

3. Modelagem

Nesta seção é feita uma análise da perspectiva de que a combinação entre conceitos de TROPOS e métodos de ITS seja útil à análise do domínio da aplicação HRCSystem, e da representação de modelos cognitivos de atores do domínio. A seção apresenta a aplicação

parcial da metodologia TROPOS, dando foco especificamente às fases *Initial Requirements*, *Final Requirements* e *Architectural Project*. Apresenta também a aplicação dos então chamados Critérios de Aplicabilidade e Princípios de Projeto derivados de ITS, na definição da arquitetura do sistema HRC (*Human Resource Consultant*). Neste contexto, TROPOS e métodos de ITS são aplicados de forma integrada, e critérios de aplicabilidade e princípios de projeto fazem papel de "ponte" entre a especificação de requisitos de TROPOS e a arquitetura MAS do sistema HRC.

3.1. Domínio de Aplicação e Tipos de Atores

O domínio é composto por cinco categorias de atores, os quais realizam comunicação entre si para atingir um objetivo em comum, ter um produto de software desenvolvido e entregue ao cliente.

Sempre que necessário, normalmente quando há uma nova solicitação de desenvolvimento, é feito um levantamento das características dos colaboradores vinculados ao processo de desenvolvimento de software. Este levantamento tem o propósito de relacionar todos os profissionais que atendam a determinadas características, as quais são inerentes à nova solicitação de desenvolvimento, e também obter informações a respeito da disponibilidade de participação em outros projetos, considerando alguns fatores como: a prioridade do projeto no qual o recurso está alocado e por quanto tempo o recurso continuará alocado, adicionalmente pode ser verificado se o recurso já possui solicitação para futuras alocações.

Os atores existentes no domínio são representações de papéis normalmente encontrados em ambientes de desenvolvimento de software. Neste domínio são considerados os atores *Gerente de Projeto*, *Líder de Equipe*, *Desenvolvedor*, *Consultor de RH (Recursos Humanos)* e *Cliente*.

O Gerente de Projetos é quem tem contato direto com o Cliente. Por isso trata de questões administrativas relacionadas ao projeto, além de ser responsável por receber novas solicitações de desenvolvimento de software. O Gerente de Projetos também estrutura e relaciona o elenco inicial das equipes de desenvolvimento, acompanha o andamento dos projetos da empresa e delega responsabilidades aos Líderes de Equipe. Quando necessário, faz consultas ao Consultor de RH para obter informações a respeito dos recursos humanos a sua disposição.

O Líder de Equipe é quem controla e define as características e recursos envolvidos em um projeto. Tais projetos possuem uma equipe elaborada especificamente para implementar uma solução. Este ator delega atribuições aos Desenvolvedores que participam da equipe sob sua responsabilidade e faz consultas ao Consultor de RH sempre que houver necessidade de obter informações sobre recursos humanos, como por exemplo, se um determinado recurso se encontra disponível para alocação. Um mesmo Líder de Equipe pode ser integrante de um ou mais projetos de desenvolvimento, onde não necessariamente, o profissional assume papel de líder em todos os projetos os quais participa.

O ator Desenvolvedor realiza atividades delegadas pelo Líder de Equipe do projeto do qual participa. Para que seja integrante de um novo projeto, o Desenvolvedor deve atender a algumas características as quais são necessárias para o bom andamento das atividades de desenvolvimento. As características requisitadas a estes profissionais variam de acordo com o domínio de aplicação do software a ser desenvolvido. Naturalmente, um Desenvolvedor pode melhorar seu currículo sempre que participa de um curso de formação ou demonstra novas habilidades e atitudes. Havendo melhorias no currículo, o ator terá cada vez mais possibilidades de participar de novos projetos. Tais melhorias podem ser sugeridas pelo Consultor de RH sempre que um Desenvolvedor não

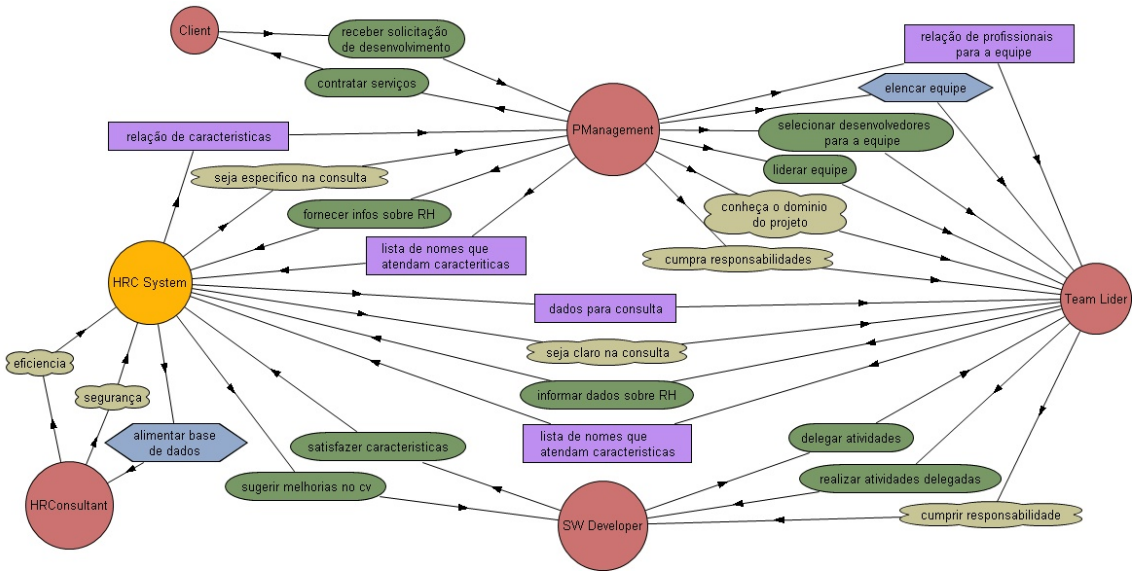

Figure 2: Modelo de Dependências Estratégicas da fase 2 de TROPOS.

possuir competências suficientes para ser relacionado em um levantamento realizado para formação de nova equipe de desenvolvimento.

O Consultor de RH fornece informações sobre características profissionais e pessoais de cada um dos colaboradores da empresa. As informações fornecidas são coletadas com auxílio de algumas técnicas, que normalmente são baseadas em documentos e formulários com respostas fornecidas pelos próprios colaboradores, levando em consideração algumas características específicas relacionadas a tecnologias ou competências.

3.2. Requisitos do Domínio

Requisitos Iniciais (*Early Requirements*) é a primeira fase de TROPOS e faz uso da técnica i* para representar o domínio. Nesta fase são criados dois modelos, um SD e um SR, em que o domínio é representado respeitando a realidade do ambiente organizacional onde o sistema será implantado. Até este momento estão representados somente os atores e suas relações de dependência.

Requisitos Finais (*Last Requirement*) é a segunda fase de TROPOS e assim como a fase anterior, faz uso da técnica i* para representar o domínio e também são criados dois modelos, um SD e um SR. A diferença entre as duas fases é basicamente o foco dado à existência do futuro sistema no domínio, que neste momento, passa a ser considerado nos modelos, tendo representadas suas relações de dependência com os demais atores. Também pode ocorrer algum tipo de refinamento nas relações de dependência em virtude da representação do sistema, que no modelo mostrado pela Figura 2 aparece na cor amarela. A partir de uma macro-visão é possível identificar os outros elementos do modelo: em vermelho estão representados os atores, em verde os objetivos, em azul as tarefas, em lilás os recursos e em pardo estão representados os objetivos-soft.

3.3. Critérios de Aplicabilidade

Os critérios de aplicabilidade [Vicari and Gluz, 2007] devem ser considerados durante o processo de análise e elicitação de requisitos da aplicação. Eles devem ser especificamente considerados quando exigências funcionais da arquitetura do sistema estão sendo analisadas. Sob a perspectiva da metodologia TROPOS estes critérios funcionam como uma *checklist* que deve ser verificada entre as fases de elicitação de requisitos do domínio e projeto arquitetural.

O primeiro critério (AC.1) propõe que o domínio da aplicação deve conter entidades que são melhor compreendidas como sendo agentes e a aplicação deve ser conceitualmente entendida como um sistema composto de agentes trabalhando juntos. No nosso caso, é possível perceber claramente que os papéis (atores) apresentados no domínio do problema, podem ser mais bem compreendidos como agentes, do que se fossem implementados como objeto, por exemplo. Isto pode ser justificado considerando a necessidade de troca de informações (conhecimentos) entre os papéis (agentes) do sistema para atingirem seus objetivos. O sistema HRC, que aparece na fase *late requirements* de TROPOS Figura 2, pode ser imaginado como uma sociedade de agentes que representará o inter-relacionamento dos papéis (atores) do domínio. Sendo assim, a aplicação pode ser entendida como um sistema composto de agentes que representam os papéis Gerente de Projetos, Líder de Equipe, Desenvolvedor e Consultor de RH, de forma que todos trabalhem juntos para satisfazer as dependências existentes entre si.

O critério (AC.2) propõe que todas as possíveis crenças de agentes a cerca de um domínio devem ser divididas em crenças sobre agentes e crenças sobre entidades não-agentes. Em termos do sistema HRC, o conhecimento relacionado ao modelo de agente representará características profissionais e pessoais dos colaboradores baseado em suas competências e qualificações. Este critério simplesmente determina que este tipo de conhecimento pode ser claramente distinguido, por exemplo, do conhecimento de como projetos são estruturados e gerenciados.

O próximo critério (AC.3) é facilmente satisfeito por todos os atores do modelo SR, incluindo HRCSystem. Este simplesmente requer que a comunicação entre os agentes seja simbólica e ocorra no nível de conhecimento.

De acordo com o critério (AC.4) agentes usados no sistema precisam ser explicitamente modelados por outros agentes e estes modelos devem ser modelos cognitivos (BDI). Portanto, quando um modelo de um agente é mencionado, o sujeito atual é a representação de suas características, como: crenças, desejos e intenções. Para o caso específico do ator SWDeveloper, será especificamente necessário que agentes dentro do sistema HRC criem um modelo para este ator que represente as características as quais se deseja mensurar e analisar. Sendo assim, por exemplo, para o caso do ator SWDeveloper existirão agentes dentro do sistema HRC que representam este tipo de ator, e criam um modelo cognitivo sobre esta entidade externa por meio de suas percepções sobre o comportamento observável de SWDevelopers (por exemplo, quais tecnologias o profissional normalmente utiliza, qual grau de criticidade o profissional possui em relação a um projeto, como é a relação de comunicação com outros Agentes SWDevelopers, quais tipos de experiência profissional e quanto tempo de empresa o profissional possui).

O critério (AC.5) requer que relacionamentos e interações sociais que ocorram no sistema sejam baseados em modelos cognitivos de agentes requeridos por (AC.4). No modelo SR do domínio, existe grande quantidade de relacionamentos e interações sociais entre os agentes, como, por exemplo, o principal relacionamento social entre HRConsultant e SWDeveloper é como selecionar o desenvolvedor para integrar uma equipe de desenvolvimento. Para que um relacionamento seja estabelecido com sucesso, depende de modelos cognitivos que os agentes dentro do sistema terão construído sobre o desenvolvedor.

O último critério (AC.6) determina que os requisitos de aplicação designados aos agentes devem ser claramente relacionados em uma especificação de requisitos. Partindo de modelos SR é possível relacionar mais facilmente qual agente irá satisfazer determinado requisito de aplicação, e adicionalmente, após uma análise mais detalhada é possível relacionar qual conhecimento é necessário para satisfazer tal requisito. O propósito geral

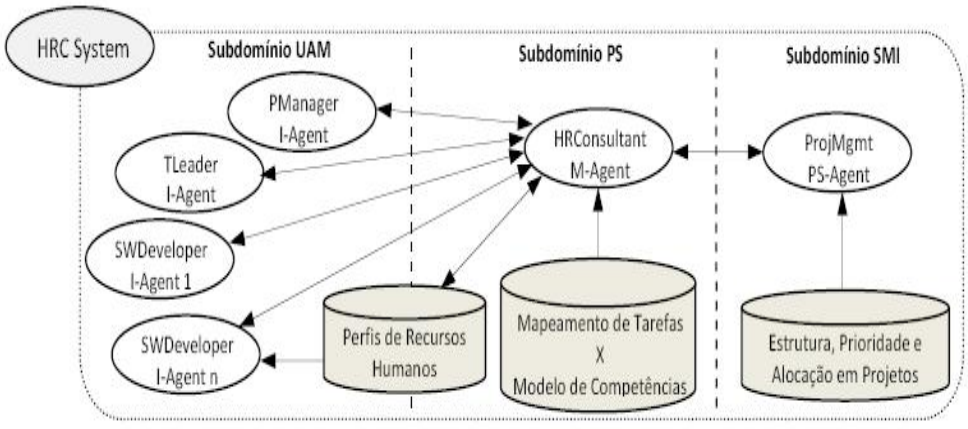

Figure 3: Arquitetura Triad - Mapeamento de agentes para subdomínios.

do sistema está incorporado ao ator Consultor de RH (agente HRConsultant), cuja principal funcionalidade é auxiliar aos atores Gerente de Projetos (agente PManager) e Líderes de Equipe (agente TLeader) a escolher atores Desenvolvedores (agente SWDeveloper) que atentam a especificações de características afins ao domínio da aplicação.

3.4. Projeto Arquitetural

A metodologia TROPOS sugere muitas arquiteturas para a implementação de soluções MAS. Contudo, no caso do sistema HRC, parece ser mais natural utilizar a arquitetura MAS proposta em [Vicari and Gluz, 2007], que pode ser chamada de "*Triad Architecture*". Nesta arquitetura é possível classificar os atores do sistema de acordo com subdomínios com características bem definidas, como mostrado na Figura 3.

Baseado no modelo SR da fase *Last Requirements* de TROPOS, o ator que representa o sistema sofre uma "eclosão" sendo subdividido em um sistema de várias entidades (agentes) distintas. Seguindo os princípios de projeto DP.1 ao DP.3 da arquitetura triad proposto em [Vicari and Gluz, 2007], este sistema consiste de um MAS composto de agentes trabalhando em três subdomínios distintos. Assim, de um mapeamento do modelo SR para o projeto arquitetural, estes aspectos são classificados como segue:

- HRConsultant (M-Agent): pertence ao subdomínio SMI e desempenha o papel de Mediador. Conhece o modelo *Competence Management* (CM) sendo capaz de compreender informações sobre tarefas, papéis e referenciais de desempenho. Também é responsável por identificar os Desenvolvedores (SWDevelopers) que atendam às necessidades de Gerentes de Projeto (PManagers) e Líderes de Equipe (TLeaders). Este agente estabelece e reforça (por meio de mediação), o principal relacionamento social buscado por HRCSystem e seus usuários.
- ProjMgmt (PS-Agent): pertence ao subdomínio PS e desempenha o papel de Solucionador. Conhece o que é um projeto e quais são os projetos da organização, e é responsável por solucionar questões relacionadas a prioridades/importância de tarefas/projetos.
- SWDeveloper, PManager, and TLeader (I-Agents): pertencem ao subdomínio UAM e desempenham o papel de Interfaceador. São responsáveis por interagir e construir modelos cognitivos de usuários do HRCSystem, e realizar a interface entre o sistema e seus usuários.

As bases de conhecimento Perfis de Recursos Humanos, Mapeamento de Tarefa x Modelo de Comptências, e Estrutura, Prioridade e Alocação em Projetos estão sendo

analisadas e estruturadas conforme o princípio DP.4. Elas são organizadas em bases de crenças que devem representar:

- Entidades relevantes ao subdomínio e suas principais propriedades;
- A identificação de suas habilidades básicas, e possíveis ações e percepções de agentes em relação a estas entidades;
- Planos (algoritmos e heurísticas) para solucionar problemas, necessários para atingir desejos (objetivos) relacionados a estas entidades.

4. Conclusões

Este trabalho apresenta uma proposta de integração entre diferentes práticas de projeto e desenvolvimento de sistemas multi-agente a fim de obter uma aplicação que, através do atendimento a especificações de relacionamentos sociais em um ambiente organizacional, possa contribuir com o processo de formação de equipes nesta organização. E para isto, parte-se da certificação de viabilidade de aplicação deste ambiente organizacional às características inerentes a sistemas multi-agente como, por exemplo, pressupor que o domínio da aplicação contenha entidades que sejam melhores compreendidas como agentes e a aplicação possa ser conceitualmente entendida como um sistema multi-agente.

Parece-nos bastante interessante a possibilidade de representar agentes de software, identificados por meio de mapeamento, a partir da representação de atores do domínio pelos modelos organizacionais desenvolvidos com o framework i*.

Interessante também é a possibilidade de utilizar uma arquitetura em que haja subdomínios de classificação para agentes do sistema, cujas funcionalidades já estão definidas. E internamente a estes subdomínios, a existência de bases de conhecimento utilizadas para a troca de informações entre agentes de diferentes subdomínios, colaborando entre si para alcançarem um objetivo em comum, auxiliar Gerentes de Projeto na seleção de profissionais para integrar uma equipe de desenvolvimento de software.

References

Bratman, M. E., D. J. I. e. a. (1988). Plans and resource-bounded practical reasoning. 4(3):349–355. Computational Intelligence.

Bresciani, P.; Perini, A. G. P. G. F. and Mylopoulous, J. (2004). Tropos: An agent-oriented software development methodology. 8:203–236. Autonomous Agents and Multi-Agent Sytems.

Greif, I. (1988). Computer-supported cooperative work: A book of readings. pages 5–9.

Grudin, J. (1994). Cscw: History and focus. pages 19–26. IEEE Computer.

Silva, I. G. L. (2005). Projeto e implementação de sistemas multi-agentes: O caso tropos. Master's thesis, Projeto e Implementação de Sistemas Multi-Agentes: O Caso Tropos, Pós-Graduação em Ciências da Computação.

TroposProject (2007). "TROPOS: Requirements-Driven Development for Agent Software", *http://www.troposproject.org/*, Acesso em Dezembro.

Vicari, R. M. and Gluz, J. C. (2007). An its view on aose. 1(3/4):295–333. International Journal of Agent-Oriented Software Engineering.

Yu, E. and Mylopoulos, J. (1994). Towards modelling strategic actor relationships for information systems development - with examples from business process reengineering. Proceedings of the 4th Workshop on Information Technologies and Systems.