Metodologias Orientada a Agentes

Vera Maria B. Werneck, Luiz Marcio Cysneiros, Abrahão Yehoshua Kano, Andrey Matheus Coppeters, André Luiz Fasando, Leandro Augusto Justian Marzulo, Leila de Oliveira Furtado, Ligia Fortes Pereira, Marco Antonio C. Lopez, Rafael Barros Pereira, Ricardo Augusto Gralhoz, Ródnei Fernando de Andrade Martins, Thiago Schmidt da Silva e Tiago Rocha M. dos Santos

UERJ - Universidade do Estado do Rio de Janeiro
Departamento de Informática e Ciência da Computação
Rua São Francisco Xavier 524, 6o Andar, Bloco B
Maracanã - 20 550-013 - Rio de Janeiro – Brasil
vera@ime.uerj.br

Resumo

A Modelagem Orientada a Agentes vem suprir a demanda pelo desenvolvimento de novas aplicações, que atendam aos requisitos e características das organizações sociais e de seus relacionamentos, de forma autônoma e integrada. O estudo deste novo paradigma é ainda recente, entretanto várias metodologias foram propostas que adotam este conceito para o desenvolvimento de sistemas. Neste trabalho é apresentado um resumo de algumas dessas propostas de modelagem e metodologias orientadas a agentes.

Abstract

Agent-Oriented modeling enriches the growing demand for new application developments, which should sustain the social organizations requirements and characteristics, besides the relationships involved, in an autonomous and integrated way. The study of this new Agent-Oriented design approach is still recent, even though many systems development methodologies that adopt this concept have been proposed so far. This paper presents some of modeling proposals and methodologies agent oriented.

1. Introdução

O crescimento da demanda pelo desenvolvimento de novas aplicações, que atendam aos requisitos e carcaterísticas das organizações sociais e de seus relacionamentos, de forma autônoma e integrada, tem impulsionado a pesquisa por novos *designs patterns* de software que consigam suportar naturalmente estas propriedades. A Modelagem Orientada à Agentes vem suprir muitas destas necessidades, almejando somar-se às demais abordagens existentes, como Orientada à Objetos (OO), na construção de sistemas complexos que tenham na autonomia, na mobilidade e na capacidade de coordenação e adaptação, os aspectos fundamentais de seu funcionamento.

Este trabalho tem como meta fornecer uma visão geral das propostas de modelagem orientada a agentes, descrevendo de forma geral os métodos na seção 2 e apresentando nas seções de 3 a 8 as metodologias GAIA [1], [2], Message [3], [4], MAS-CommonKADS [5], Adelfe [6], Tropos [7] e MaSE [8].

.

2. Métodos e Linguagens de modelagem Orientadas a Agentes

A orientação a agentes é um paradigma ainda em desenvolvimento e vários métodos orientados a agentes têm sido propostos, como por exemplo, Gaia [1], [2], MESSAGE [3], [4], Tropos [7], Adelfe [6], MASE [8], Prometheus [9]. Estes métodos oferecem uma variedade de conceitos, notações, técnicas e diretrizes metodológicas. Algumas extensões de métodos ou linguagens de modelagens consagradas como CommonKADS [10] e UML [11] também têm sido estendidas para atender aos sistemas multi-agentes como é o caso do MAS-CommonKADS [5] e AUML[12].

As metodologias orientadas a agentes possuem várias raízes (Figura 1). Algumas são baseadas na idéia da inteligência artificial oriunda da engenharia de conhecimento (KE), outras são extensões da metodologia orientada a objetos (OO). Há ainda as que utilizam uma mescla de conceitos baseados nessas duas metodologias e algumas são derivadas de outras metodologias também orientadas a agentes.

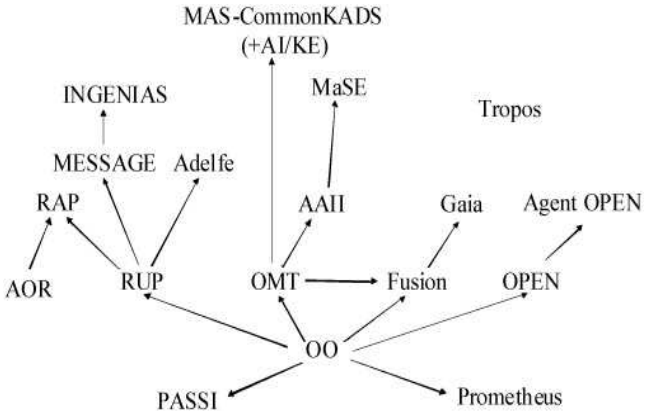

Figura 1: Influências diretas e indiretas de linguagens OO sobre metodologias OA [13]

3. MESSAGE

MESSAGE surgiu no domínio das aplicações de Telecomunicações. Com este propósito, um instituto de pesquisas da área de telecomunicações direcionou um dos seus projetos, o P907 [3], [4], para a definição da metodologia para a análise e design baseados na orientação a agentes.

MESSAGE [3], [4], utiliza a UML (Unified Modeling Language) [10] como ponto de partida, adicionando conceitos de entidades e relacionamentos necessários à modelagem orientada a agentes. Esta modelagem busca descrever a forma como um sistema multi-agentes funciona para a realização de uma meta coletiva, similarmente às organizações humanas e sociedades, além do comportamento cognitivo dos agentes, com o auxílio da Inteligência Artificial.

Tais conceitos lidam com idéias e estruturas em um nível conceitual maior que o da orientação a objetos, referenciado na área de Inteligência Artificial como "Nível de Conhecimento", estabelecendo um contraste entre o conhecimento e dados. A MESSAGE [3], [4] usa a linguagem padrão de Orientação a Objetos, UML (Unified Modeling Language) [10] como padrão de linguagem à "nível de dados", acrescentando os conceitos ao "nível de conhecimento" que estão descritos em seu meta-modelo, que também provê uma interpretação de alguns conceitos da UML [10] usados para descrição do comportamento, dentre os quais a noção de "estado".

A maior parte dos conceitos de entidades, existentes no nível de conhecimento do MESSAGE [3], [4] podem ser agrupadas em três categorias: Entidade Concreta, Atividade e Entidade de Estado Mental. Na primeira categoria, estão contidas as noções de Agente, Organização, Papel e Recurso. A segunda contém as noções de Tarefa, Interação e Protocolo de Interação. A terceira tem um foco maior no conceito de Meta.

O modelo de análise consiste em uma complexa rede de classes e instâncias inter-relacionadas, derivadas dos conceitos definidos no meta-modelo da metodologia [2], [3], [4].

A Visão Organizacional representa as entidades concretas no sistema e no seu ambiente, além de seus relacionamentos como os de agregação e de conhecimento, este último indicando que há pelo menos uma interação entre as entidades. Estas entidades são agentes, organizações, papéis ou recursos.

A Visão de Meta/Tarefa representa metas, tarefas, estados e as dependências entre eles. É possível representar uma meta como a composição de uma série de sub-metas, de um nível menor, e como as tarefas que propiciam o alcance daquela meta são executadas, além da dependência temporal. Para tal, emprega-se o Diagrama de Atividades da UML.

A Visão de Agente/Papel descreve para cada par agente/papel um diagrama que retrata as metas, os eventos percebidos, os recursos controlados, suas regras de comportamento e quais tarefas são realizadas.

A Visão da Interação tem enfoque na interação entre agentes/papéis. Para cada interação mostra-se o seu iniciador, os colaboradores, o motivador (meta do iniciador), as informações importantes fornecidas e obtidas por cada participante da interação, os eventos que a disparam, dentre outros de seus efeitos relevantes.

A Visão de Domínio retrata as relações e os conceitos específicos do domínio do sistema que está em desenvolvimento.

Na metodologia MESSAGE [3], [4], assume-se que o ponto de partida para a análise é uma definição informal dos requisitos, feita pelo cliente. Emprega uma técnica de análise que consiste em um refinamento, em níveis, na qual estágios subseqüentes do refinamento resultam na criação de modelos nos diferentes níveis, que devem ser consistentes entre si.

O nível mais alto para decomposição é denominado Nível 0, no qual procura-se definir o sistema a ser desenvolvido em relação aos seus interessados e ao seu ambiente, dando uma idéia geral sobre suas funcionalidades gerais. No Nível 1, são definidos o comportamento e a estrutura interna das entidades identificadas, tais como agentes, organização, tarefas e metas. É possível adicionar níveis posteriores para a especificação de requisitos funcionais e não-funcionais como desempenho e segurança, porém, a MESSAGE só engloba os dois primeiros níveis.

Há várias maneiras de se fazer o refinamento dos modelos do Nível 0. A geração de várias visões do sistema, aplicada pela MESSAGE, propicia uma liberdade de escolha ao analista sobre a melhor estratégia

a ser adotada. Porém, usualmente, procura-se utilizar uma combinação destas estratégias de refinamento.

Uma das abordagens é a Centralizada na Organização, que focaliza na análise das propriedades gerais como: a estrutura do sistema, os serviços oferecidos, metas e tarefas gerais, os papéis principais e os recursos. A abordagem Centralizada no Agente tem como foco a definição dos agentes necessários para que as funcionalidades do sistema possam ser providas, sendo que a organização mais apropriada é identificada de acordo com os requisitos do sistema. A abordagem Orientada à Interação prega um refinamento progressivo dos cenários de interação, que caracterizam o comportamento interno e externo da organização e dos agentes. A abordagem por Decomposição de Meta/Tarefa baseia-se na decomposição funcional. Os papéis, metas e tarefas do sistema são sistematicamente analisados para que sejam determinadas as condições de decisão, os métodos para resolução de problemas e tratamento de falhas.

O propósito da fase de projeto é definir as entidades computacionais que representam o sistema multi-agentes (MAS) representado na fase de análise. Estes artefatos são transformados para possibilitar a implementação. A fase de projeto na metodologia MESSAGE [3], [4], é dividida em dois processos: projeto de alto nível e projeto detalhado.

No projeto de alto nível o modelo de análise é refinado produzindo uma versão inicial da arquitetura MAS. O projeto detalhado pressupõe a definição das estruturas de implementação possíveis para o projeto dos agentes, sendo que o MESSAGE define duas abordagens. A primeira é direcionada por uma arquitetura agente e por uma organização multi-agentes, considerando que o agente é uma entidade mais abrangente que uma classe.

A segunda abordagem é mais orientada à plataforma agente, levando-se em conta que cada agente pode ser mapeado para uma classe. Este conceito é extraído dos modelos suportados pela maioria das ferramentas para a implementação de agentes, como a JADE [14] e a FIPA-OS [15], nas quais há uma classe Agente a partir da qual tipos específicos de agentes são derivados.

4. *MAS-CommonKADS*

A metodologia MAS-CommonKADS [5], [16], [17] é uma extensão da metodologia CommonKADS englobando aspectos que são relevantes para sistemas multi-agentes.

CommonKADS tornou-se, principalmente na Europa, uma referência no desenvolvimento de sistemas baseados em conhecimento (SBC). No CommonKADS são definidos vários modelos [10], sendo o Modelo de Experiência o central da metodologia CommonKADS com meta de modelar o conhecimento de resolução de problemas empregado por um agente para realizar uma tarefa. Este modelo divide o conhecimento da aplicação em três subníveis: nível do domínio (conhecimento declarativo sobre o domínio), nível de inferência (uma biblioteca de estruturas genéricas de inferência) e nível de tarefa (ordem das inferências).

Algumas considerações e problemas foram analisados ao se aplicar diretamente CommonKADS ao desenvolvimento de sistemas multi-agentes [16], [17] surgindo a proposta MAS-CommonKADS centrada no Modelo de Agentes. Esta metodologia define sete modelos [16]: Modelo de Agentes, Modelo de Organização, Modelo de Tarefas, Modelo de Experiência, Modelo de Comunicação, Modelo de Coordenação e Modelo de Design (Figura 2).

O Modelo de Agentes especifica as características de um agente: sua capacidade de raciocínio, habilidades, serviços, sensores, grupos de agentes a que pertence e classe de agente. Um agente pode ser um agente humano, software, ou qualquer entidade capaz de empregar uma linguagem de comunicação de agentes. O Modelo de Organização é uma ferramenta para analisar a organização humana em que o sistema multi-agente vai ser introduzido e para descrever a organização dos agentes de software e sua relação com o meio. O Modelo de Tarefas descreve as tarefas que os agentes podem realizar, as metas de cada tarefa, sua decomposição e os métodos de resolução de problemas para resolver cada meta. O Modelo de Experiência descreve o conhecimento necessário aos agentes para atingir suas metas. Segue a decomposição de CommonKADS e reutiliza as bibliotecas de tarefas genéricas. O Modelo de Comunicação descreve as interações entre um agente humano e um agente software. Centra-se na consideração de fatores humanos para dita interação. O Modelo de Coordenação descreve as interações entre agentes de software. E finalmente o Modelo de Design descreve a arquitetura e o design do sistema multi-agentes como passo prévio a sua implementação, sendo o único modelo a não tratar da Análise.

O modelo de ciclo de vida para o desenvolvimento de sistemas multi-agentes [5], [16], [17] segue a gestão de projetos de Common-KADS [10] sendo dirigido por riscos e engloba as seguintes fases: Conceituação, Análise, Design, Codificação, Integração e Operação e Manutenção. A fase de Conceituação consiste na tarefa de elicitação para obter uma primeira descrição do problema e a determinação dos casos de uso que podem ajudar a entender os requisitos informais e a testar o sistema. A Análise determina os requisitos do sistema partindo do enunciado do problema. Durante esta fase se desenvolvem os seguintes modelos: organização, tarefas, agentes, comunicação, coordenação e experiência. No Design

define-se como os requisitos da fase de análise podem ser conseguidos mediante o desenvolvimento do modelo de design, determinando-se as arquiteturas tanto da rede multi-agentes como de cada agente. Na Codificação cada agente é implementado e testado e na fase de Integração, o sistema completo é testado.

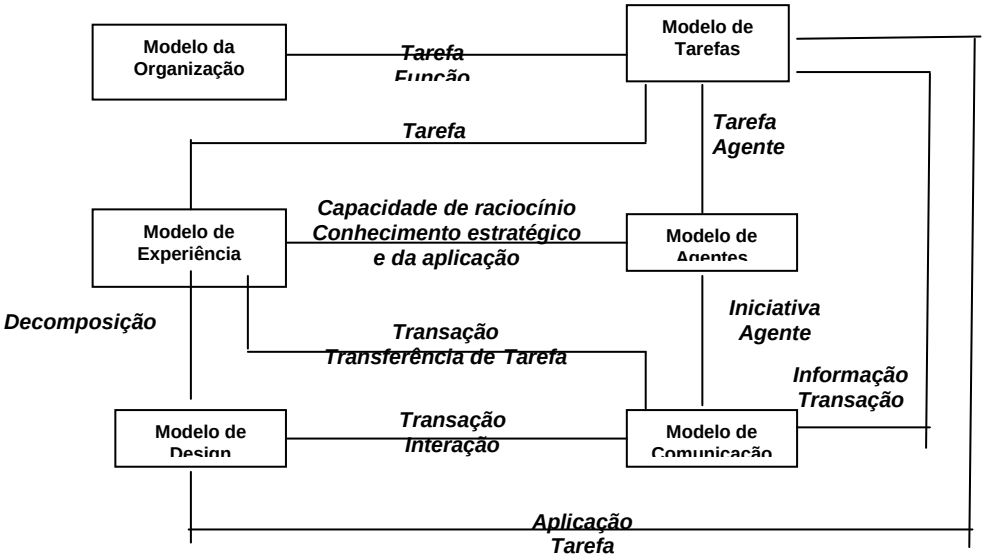

Figura 2. Conjunto dos modelos CommonKADS

5. *GAIA*

A metodologia GAIA foi desenvolvida para análise e projeto de sistemas baseados em agentes, por estes serem flexíveis, terem autonomia em resolver problemas, possuírem interação e pela estrutura organizacional do sistema [1], [2].

A metodologia GAIA permite que o analista passe da especificação de requisitos para o projeto, que é suficientemente detalhado, permitindo que seja implementado diretamente, conforme o analista migra de um conceito abstrato para um mais concreto.

Os principais conceitos dessa metodologia podem ser divididos em duas categorias: abstratos e concretos. Entidades abstratas são aquelas usadas durante a análise representando os conceitos do sistema, mas que não necessariamente tem alguma ligação com o sistema. Entidades concretas, em contraste, são usadas dentro do processo de projeto, e serão direcionadas, em contrapartida, em tempo de execução do sistema.

Gaia possui duas fases distintas [2]: fase de análise e fase de projeto (design). Na fase de análise, são gerados o modelo de ambiente, o modelo preliminar de papéis, o modelo preliminar de interação, e as regras organizacionais do sistema. A fase de projeto subdivide-se em duas: o Projeto de Arquitetura e o Projeto Detalhado. A organização dos modelos do Gaia nestas fases é apresentada na figura 3.

Na fase de Arquitetura, é definida uma estrutura organizacional do sistema, e logo em seguida são completados o modelo de papéis e o modelo de interação. Na fase de Projeto Detalhado, são gerados os modelos de agentes (agent model) e o de serviços (services model).

Na fase de análise, o objetivo é desenvolver um entendimento do sistema e sua estrutura, chamada de organização do sistema, que é tida como uma coleção de papéis, que se relacionam com outros e que fazem parte de padrões sistemáticos e institucionalizados de interação com outros papéis. Assim o sistema é decomposto em papéis que serão desempenhados na organização (meta do modelo de papéis) e definida a maneira como eles interagem de acordo com protocolos específicos (meta do modelo de interação). Esta fase promove um melhor entendimento do sistema e de suas estruturas sem considerar detalhes de implementação [2].

Na fase de projeto, o objetivo é transformar o modelo de análise em modelo de baixo nível de abstração com técnicas de projetos tradicionais (incluindo técnicas de Orientação a Objetos) que possam ser aplicadas de maneira a implementar o agente. Em outras palavras, GAIA se preocupa em como a sociedade de agentes coopera para atingir metas no nível do sistema e o que é solicitado para cada agente. Realmente como um agente realiza estes serviços é além do escopo de GAIA, e dependerá do domínio de uma aplicação particular [2].

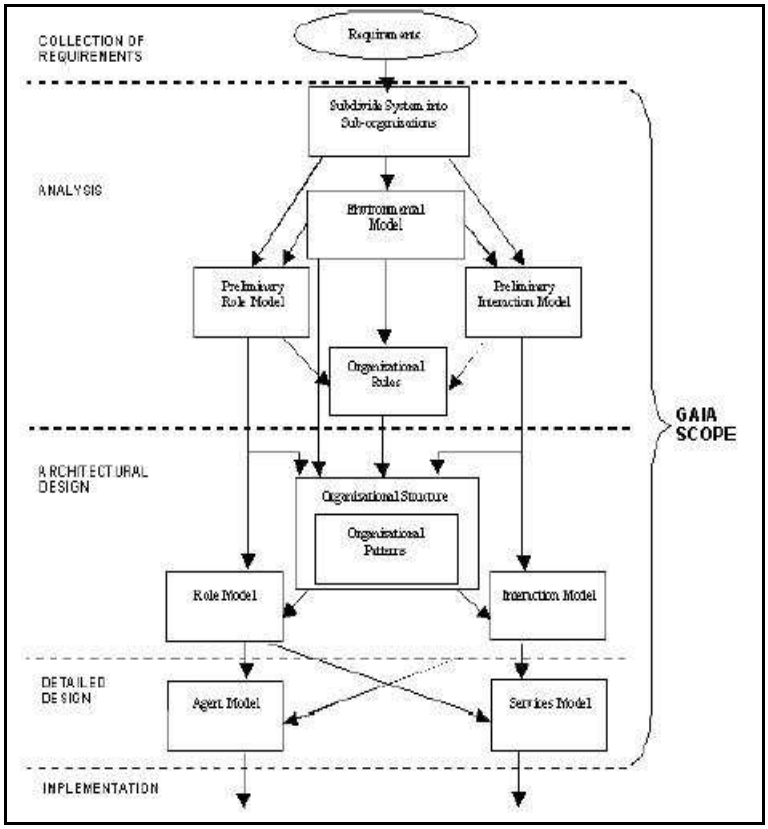

Figura.3: Organização dos Modelos do Gaia

6. *Adelfe*

A metodologia ADELFE foi desenvolvida com propósito de trabalhar alguns aspectos ainda não considerados pelas metodologias orientadas a agentes previamente utilizadas e para consolidar a teoria de AMAS (Adaptative Multi-Agent Systems). A sigla ADELFE é um acrônimo que, traduzido do francês, significa "framework para desenvolvimento de software com funcionalidades emergentes" [6].

ADELFE trabalha baseado num processo específico adaptado de uma interpretação do RUP, Rational Unified Process, de acordo com o Projeto Netuno (http://www.neptune.irit.fr). Alguns acréscimos foram feitos para levar em consideração especificidades da teoria própria dos AMAS, como, por exemplo, a caracterização do ambiente do sistema, a identificação de falhas de cooperação. Nesse tipo de sistema, cada usuário e fornecedor de serviços têm uma meta individual que é a execução da solicitação requerida e não conhece a função global do sistema. Este sistema é fortemente indefinido, devido a eventual entrada ou saída de usuários e ou fornecedores de serviços. Mais que isso, não existe um algoritmo prévio para descrever o comportamento do sistema [18].

ADELFE além do UML e do RUP [10] também utiliza os princípios de AUML [12], [19] para expressar protocolos de interação entre agentes e utiliza duas ferramentas integradas ao framework ADELFE: OpenTool e ferramenta interativa [2]. OpenTool é uma ferramenta gráfica para modelagem que dá suporte à notação UML e protocolos de interação AUML. A ferramenta interativa descreve o processo de modelagem e ajuda o desenvolvedor a aplicá-lo.

O objetivo principal da metodologia ADELFE é cobrir todas as fases do processo de desenvolvimento clássico de um software, desde os requisitos até a implantação com base no processo RUP adaptado para AMAS. Apenas as *work definitions* de requisitos, análise e design é que requerem modificações a fim de serem adaptadas a um AMAS, o restante do RUP aplica-se sem maiores problemas [18].

Na fase de análise a atividade "Caracterizar o Ambiente" foi adicionada ao RUP para caracterizar o ambiente do sistema, e constitui-se das seguintes tarefas: determinar as entidades, definir o contexto e caracterizar o ambiente. Essa caracterização tem início na identificação das entidades que interagem com o sistema e

nas restrições sob essas interações. Uma entidade é um ator no mesmo sentido que em UML, podendo ser classificada em ADELFE como ativa ou passiva.

A definição do contexto analisa o ambiente por meio das interações entre as entidades e o sistema, definindo os diagramas de seqüência e colaboração da UML. Os fluxos de informações entre entidades passivas e o sistema são expressos por meio de Diagramas de Colaboração, enquanto que interações entre entidades ativas e o sistema são descritas através de Diagramas de Seqüência. De acordo com UML, o diagrama de Colaboração deriva-se do diagrama de Classes. Entretanto, a metodologia ADELFE não exige tais pré-requisitos sendo desenvolvidos baseados na definição prévia das entidades considerando-se o conjunto de palavras-chave definidas anteriormente.

Por fim, o analista tem que descrever o ambiente sob termos desenvolvidos por Russel e Norvig (1995 em [2]). Assim, o ambiente pode ser descrito como: acessível (em contraste com "inacessível"), contínuo (em contraste a "discreto"), determinístico (em contraste a "não determinístico"), ou dinâmico (em contraste a "estático").

ADELFE está interessada apenas em "agentes cooperativos" que possibilitam a construção de um AMAS. O desenvolvedor, após definir os casos de uso, deve pensar já em eventos potencialmente "nocivos" ou inesperados no sistema, tendo em mente que estas podem levar a Situações Não Cooperativas no nível do agente. Tais "falhas de cooperação" podem ser vistas como uma espécie de exceção. Para levar isto em consideração, na confecção dos casos de uso é incrementada as falhas de cooperação utilizando-se de notação específica [2],[18\.

Na fase de Análise são incorporadas as atividades de verificar a adequação a AMAS, Identificar agentes e estudar as Interações entre as entidades. ADELFE adiciona a atividade de Verificar a adequação do sistema em questão à tecnologia AMAS, sendo esta realizada através da Ferramenta de Adequação. A atividade de Identificar Agentes visa identificar o que é considerado como agente no sistema. Na ADELFE, agentes não são conhecidos de antemão, e o analista deve identificá-los, estudando as entidades definidas a priori. Entidades que demonstram propriedades como autonomia, meta local a se atingir, existência de interações com outras entidades, visão parcial do sistema e habilidade de negociação são aquelas que devem ser consideradas como agentes em potencial. Isso não se aplica, obrigatoriamente, a todas as entidades ativas identificadas, pois algumas permanecem como objetos comuns.

A construção de um sistema AMAS se interessa por agentes que se comportam cooperativamente devendo ser oriundos das entidades definidas anteriormente na atividade na fase de requisitos e das classes elaboradas na atividade análise. Os agentes cooperativos são entidades que satisfazem a pelo menos os requisitos de autonomia, meta local e interação com outras entidades. Ao terminar esse estudo de identificação dos agentes, as respectivas classes são marcadas com o estereótipo de "agente cooperativo".

A atividade de estudar as interações entre entidades também foi incorporada ao processo podendo estudar interações entre entidades ativas/passivas, entre entidades ativas e entre agentes. Assim é possível validar as interações entre entidades ativas e passivas e as interações entre entidades ativas previamente definidas representadas por diagramas de colaboração e seqüência, respectivamente. O último estágio desta atividade, é realizada através de um novo diagrama para descrever as relações entre agentes: diagrama de protocolo.

A fase de design tem um papel importante porque trata-se de uma fase de refinamento do projeto, onde pode ser possível inclusive identificar candidatos a agentes que passaram desapercebidos nas fases anteriores. Esta fase é dividida de em 5 etapas a saber: estudo da arquitetura detalhada e do modelo do Multi-Agente, estudo das linguagens de interação, refinamento dos agentes do projeto, teste e finalização dos diagramas do Projeto [18].

7. *TROPOS*

Tropos é uma metodologia orientada a agentes, caracterizada por três aspectos fundamentais: (i) noção de agente e as noções intelectuais relacionadas, da arquitetura BDI (*belief, desire, intention*); (ii) atenção especial para as atividades que precedem a especificação dos requisitos prescritos, como o entendimento de como e porquê o sistema atenderia aos propósitos da organização; (iii) idéia de construir-se um modelo do que deverá ser o sistema, que será refinado e estendido, incrementalmente, desde um nível conceitual até artefatos executáveis, por uma seqüência de passos de transformação [7], [13], [20].

A idéia de se focar nas atividades que antecedem a especificação dos requisitos de software, tal que se entenda como o sistema desejado auxiliará a organização a atingir suas metas. Esta proposta é baseada no framework de modelagem i* [21], que oferece os conceitos primitivos de atores, que podem ser agentes (organizacional, humano ou software), posições ou papéis, metas concretas (g*oals*) e dependências sociais (tais como metas flexíveis (*softgoals)*, tarefas e recursos) para definição das obrigações de cada ator em relação aos outros [7], [13], [20].

Várias atividades contribuem para elaboração de uma primeira modelagem dos requisitos, e subseqüentemente seu refinamento e evolução: Modelagem dos Atores, Modelagem de Dependências, Modelagem de Metas Modelagem de planos e Modelagem de Competências.

O Modelo dos Atores consiste na identificação e análise dos atores de um ambiente, dos agentes e atores do sistema, e seu foco dependerá da fase de desenvolvimento. Esta modelagem é representada no Diagrama de Atores.

A Modelagem de Dependências consiste na identificação de atores que possuem um relacionamento de dependência, sendo também representada no Diagrama de Atores.

A Modelagem de Metas baseia-se no estudo das metas de um ator, a partir do seu ponto de vista, utilizando técnicas como: (i) a análise do tipo *means-end* (meio-fim) que visa identificar planos, recursos e metas fexíveis que provêm meios para o alcance de uma meta concreta; (ii) a análise de contribuição, que aponta as metas que podem contribuir positiva ou negativamente para que uma meta flexível analisada seja atingida; (iii) a decomposição AND/OR, na qual uma meta maior é decomposta em sub-metas. A representação gráfica desta modelagem é feita pelo Diagrama de Metas.

A Modelagem de Planos é similar à Modelagem de Metas, empregando as mesmas técnicas, analogamente, e também é representada graficamente pelo Diagrama de Metas.

A Modelagem de Competências é realizada na fase de projeto da arquitetura, sendo baseada na atribuição de competências aos sub-atores especificados, representadas graficamente pelos Diagramas de Plano e de Competências. Com este intuito, são usados também os Diagramas de Atividades da UML [10] e os Diagramas de Interação da AUML [12].

A metodologia Tropos tem como meta suportar todas as atividades de análise e projeto no processo de desenvolvimento de software, com a concepção de construir-se um modelo do que deverá ser o sistema e o seu ambiente, que será aperfeiçoado e estendido, provendo uma interface comum para as várias atividades de desenvolvimento de software, assim como para sua documentação e evolução.

As cinco principais fases de desenvolvimento de software que compõem a metodologia são: Early Requirements e Late Requirements (análise de requisitos), Projeto Arquitetural, Projeto Detalhado, e Implementação.

Geralmente, a análise de requisitos é a fase inicial nas metodologias existentes na engenharia de software, tendo como função levantar um conjunto de requisitos funcionais e não funcionais do que deverá ser o sistema. Na Tropos, esta fase é dividida em duas etapas, Early Requirements e Late Requirements, que compartilham a mesma abordagem conceitual.

Na etapa de Early Requirements, a preocupação é com a compreensão de um problema através do estudo da configuração organizacional existente. Identifica-se os participantes do domínio, que são modelados como atores sociais, os quais dependem uns dos outros para o alcance de metas, realização de planos e fornecimento de recursos. Os atores em uma organização caracterizam-se por possuírem metas e intenções que seriam atingíveis, ou talvez não tão facilmente alcançados, de forma isolada, retratando uma dependência que torna possível verificar como a implementação final satisfaz as necessidades iniciais, ao descrever o "porquê" das funcionalidades do sistema. Uma vez que são identificados os atores sociais, suas metas e dependências, gera-se o Diagrama de Atores desta fase. Em seguida, é feito um enriquecimento do modelo ao analisar-se a lógica de cada meta, através de uma decomposição em sub-metas e planos, gerando o Diagrama de Metas. Neste, as metas mais abaixo na hierarquia são mais específicas e motivadas pelos que se encontram mais acima. Identifica-se também quais as metas que contribuem positivamente ou negativamente para as metas flexíveis.

Na etapa de Late Requirements o modelo conceitual é estendido. O enfoque passa a ser o que deverá ser o sistema, o qual é representado como um ator, que tem uma série de dependências com os demais atores da organização. Tais dependências definem os requisitos funcionais e não funcionais do sistema, representados no Diagrama de Atores desta fase. No Diagrama de Metas, cada meta é analisada do ponto de vista do sistema, havendo um detalhamento sobre os planos necessários a sua execução e os recursos envolvidos. O sistema é descrito dentro de seu ambiente operacional, com suas qualidades e funções relevantes.

O Projeto Arquitetural é composto de três passos e define a arquitetura geral do sistema em termos de sub-sistemas, representados como atores, interligados por fluxos de dados e controles, representados como dependências. O primeiro passo abrange a inclusão de novos atores, conforme o padrão arquitetural escolhido, e a sua decomposição em sub-atores aos quais são delegadas sub-metas, baseando-se em uma análise sobre as metas do sistema. O resultado deste passo é um Diagrama de Atores Estendido. No segundo passo, identifica-se quais as competências necessárias aos atores para a realização de suas metas e planos, que podem ser obtidos de uma análise sobre o Diagrama de Atores Estendido. O terceiro e último passo estabelece um conjunto de tipos de agentes, atribuindo a cada um destes uma ou mais competências, de acordo com a escolha do designer após uma análise cuidadosa do Diagrama de Atores Estendido e a sua visão sobre o sistema em termos de agentes.

O enfoque na fase de Projeto Detalhado é a especificação de cada componente arquitetural, em maiores detalhes. É feita uma análise dos agentes a um nível micro, na qual suas metas, crenças e competências,

além da comunicação com os demais agentes, são especificadas detalhadamente, estando fortemente relacionada com as escolhas de implementação. São empregados os Diagramas de Atividades da UML [10] para a representação dos planos e competências, em Diagramas de Competência e Diagramas de Plano, permitindo a modelagem de uma competência sob o ponto de vista de um determinado agente e o detalhamento de cada plano contido no Diagrama de Competência. Nos Diagramas de Interação de Agentes, os Diagramas de Seqüência da AUML [12] são explorados, onde agentes correspondem a objetos cuja linha do tempo de vida é independente da interação a ser modelada e a comunicação entre agentes corresponde à arcos de mensagem assíncronos.

A programação orientada à agentes é motivada principalmente pela necessidade de se ter arquiteturas abertas, que mudam continuamente e evoluem para suportar mais componentes e atender a novos requisitos. Para algumas aplicações a programação orientada à agentes parece ser muito adequada, como: o comércio eletrônico, o planejamento de recursos da empresa, sistemas de controle de tráfego aéreo, assistentes pessoais (PDA's) e assim por diante.

Como exemplo de plataforma de programação, pode-se citar a JACK Intelligent Agents [], um ambiente de desenvolvimento orientado a agentes construído para estender a linguagem Java pelo modelo de agente BDI (Belief, Desire, Intention), muito usado na área de inteligência artificial e de ciências cognitivas. Os agentes JACK [22] podem ser vistos como componentes de software autônomos que possuem metas e desejos a serem atingidos, sendo programados com um conjunto de planos, no qual cada plano descreve como atingir uma meta sob determinadas circunstâncias. Para suportar este modelo, JACK [22] oferece cinco construções de linguagens principais: agentes, competências, relações de bases de dados, eventos, e planos.

8. MaSE

A metodologia MaSE (Multiagent System Engineering) é independente de qualquer arquitetura de agentes, linguagem de programação, ou framework de comunicação. A MaSE vê os agentes como uma abstração mais profunda do paradigma de orientação a objeto, onde os agentes são especializações de objetos. Em vez de simplesmente objetos, com métodos que podem ser invocados por outros objetos, os agentes conversam entre si e agem proativamente de modo a alcançar uma meta individual ou do sistema [7].

MaSE tem como objetivo principal ajudar o engenheiro de software a elicitar um conjunto inicial de requerimentos e analisar, projetar e implementar um sistema multiagente.

A metodologia MaSE é uma especialização das metodologias de engenharia de software mais tradicionais e segue as fases e passos mostrados na figura 4 [7].

O objetivo da fase de Análise da MaSE é definir um conjunto de papéis que podem ser usados para alcançar as metas dos níveis do sistema. Estes papéis são definidos por uma série de tarefas, que por sua vez são descritas por modelos de estado finitos. Este processo é alcançado em três passos: Capturando Metas, Aplicando Casos de Uso e Refinando Papéis [7], [23].

Na fase Capturando Metas as metas do sistema são capturados a partir da análise do conjunto de requisitos do sistema. Metas são o que o sistema está tentando alcançar e eles geralmente se mantêm constantes por todo o resto do processo de projeto e análise. Para estruturar as metas, o analista estuda sua importância e inter-relações. O primeiro passo é identificar a principal meta do sistema. Essa meta é colocada no topo do Diagrama de Hierarquia de Metas. As metas deverão ser decompostas em sub-metas até que quaisquer mais decomposições dessas metas resultem em funções e não sub-metas [7], [23].

Na fase Aplicando Casos de Uso, as metas e sub-metas são traduzidas em casos de uso que contêm os cenários obtidos no passo anterior. Os cenários são descritos de forma textual e são transformados em uma série de diagramas de seqüência similares aos diagramas de seqüência da UML [6]. A principal diferença entre um diagrama e outro é o fato de que no diagrama de seqüência da MaSE as seqüências de eventos ocorrem entre papéis enquanto que no diagrama da UML [6] as seqüências de eventos ocorrem entre objetos [7], [23].

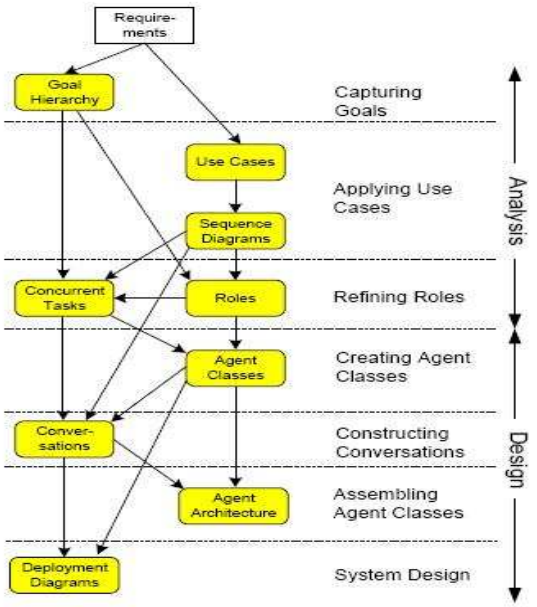

Figura 4. Fases da MaSE

Quando os casos de uso são transformados em diagramas de seqüência, os papéis que foram identificados se tornando o conjunto inicial de papéis que serão usados no próximo passo, que é o refinamento dos papéis. O analista deverá representar os requerimentos críticos, sendo eles positivos ou negativos, pois os dois são importantes para definir os papéis do sistema.

O objetivo da fase Refinando Papéis é transformar as metas estruturados e os diagramas de seqüência em papéis e suas tarefas associadas. Papéis são a base para a definição de classes de agentes e representam as metas do sistema durante a fase de desenho.

Na MaSE as metas do sistema são atingidos se cada meta consegue ser mapeada em um papel, e cada papel é usado por no mínimo uma classe de agentes. Em geral, o mapeamento das metas em papéis é realizado de maneira 1 para 1. Entretanto, o desenvolvedor pode permitir que um papel seja responsável por múltiplos metas. O desenvolvedor poderá também combinar papéis, simplificando, assim, o desenho geral.

Na MaSE, o refinamento das metas é capturado em um Modelo de Papéis. onde os papéis são decompostos em um conjunto de tarefas, as tarefas individuais são designadas a alcançar as metas pelo qual o papel é responsável [7], [23].

O Modelo de Tarefas Concorrentes associa as tarefas com cada papel que descreve o comportamento que um papel precisa ter para alcançar a sua meta. Em geral, um único papel pode possuir múltiplas tarefas concorrentes que definem o comportamento esperado do papel. Especificamos o comportamento dos papéis como um conjunto de n tarefas concorrentes. Cada tarefa especifica uma única linha de controle que define um comportamento particular que o papel pode ter. Tarefas concorrentes são representadas graficamente usando um autômato de estados finitos. Esse autômato é chamado Diagrama de Tarefas concorrentes.

Atividades são utilizadas para especificar funções do papel e são realizadas dentro dos estados das tarefas. Enquanto essas tarefas executam concorrentemente elas podem ser coordenadas usando eventos internos. Eventos internos são passados de uma tarefa para outra e são especificados nas transições entre estados. Para a comunicação com outros papéis, as mensagens externas podem ser enviadas e recebidas. Essas mensagens externas são especificadas através do uso de eventos de send e recieve. Esses eventos recuperam as mensagens do componente responsável por tratar as mensagens do papel. As tarefas, além de se comunicarem com outros papéis, podem também interagir com o ambiente. Essa interação é tipicamente capturada em funções definidas nos estados.

Na fase de análise, um conjunto de metas foi detalhado e usado para criar um conjunto de casos de uso e diagramas de seqüência que descreviam o comportamento básico do sistema. Esses modelos são usados para desenvolver um conjunto de papéis e tarefas que mostravam como as metas foram alcançadas. O propósito da fase de projeto é usar esses papéis e tarefas e convertê-los em uma forma mais adequada à implementação. Existem quatro passos na fase de projeto da MaSE. O primeiro passo é Criando Classes de Agentes, no qual o desenvolvedor designa papéis para tipos de agentes específicos. No segundo passo, Construindo Conversações, as conversações entre as classes de agentes são definidas enquanto que no terceiro passo, Montando Classes de Agentes, a arquitetura interna e os processos de raciocínio das classes de agentes são desenhados. Finalmente, no último passo, Projeto do Sistema, o desenvolvedor define o número e a localização dos agentes no sistema desenvolvido.

O Diagrama de Desenvolvimento é usado para mostrar momentos de possíveis configurações do sistema em um sistema multi-agentes, no qual os agentes se movem, ou são criados e destruídos [7], [23].

9. Conclusão

A tecnologia de agentes não é um padrão isolado para o desenvolvimento de aplicações, neste caso, baseando-se em entidades de software autônomas, interativas e adaptativas. A orientação a agentes deve ser integrada com as demais metodologias já existentes, visando explorar aspectos que estas não retratavam, como as características presentes nas organizações humanas.

Esta abordagem se adapta ao desenvolvimento de aplicações inteligentes e distribuídas, na execução de tarefas ou disponibilização de serviços. Em relação à orientação a objetos, tecnologia mais usada atualmente, difere pela autonomia. Os agentes dispensam uma intervenção externa para agir, diferentemente dos objetos, que precisam ter seus métodos invocados para efetuar qualquer processamento.

Este novo paradigma ainda está restrito à área de pesquisa com alguns relatos sobre casos de uso comercial. Além do fato de ser recente, outros fatores contribuem para que a tecnologia não desponte no mercado, tais como a falta de um consenso sobre o suporte à agentes e a existência de várias metodologias com processos de análise e design distintos.

Este trabalho revela a importância do desenvolvimento da tecnologia de agentes e os aspectos relevantes a serem discutidos, a medida que surgem novas necessidades na área de tecnologia de informação. A demanda por sistemas mais inteligentes e flexíveis impulsionará o amadurecimento das metodologias apresentadas,

acelerando sua evolução e propiciando a sua utilização na criação de soluções reais para o mercado.

Este trabalho está inserido num projeto de pesquisa de avaliação de metodologias Orientadas a Agentes [24], [25], [26], [27], [28], [29] com base no framework de avaliação proposto por Yu e Cysneiros [30].

10. Referências Bibliográficas

[1] Wooldridge, M., Jennings, N.R. and Kinny, D., The Gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multi-Agent Systems, 3(3), 2000, pp 285-312.

[2] Zambonelli, F., Jennings, N. R. and Wooldridge, M., Developing Multiagent Systems: The Gaia Methodology, In ACM Transaction on Software Engineering and Methodology, Vol. 12, No. 3, July 2003, pp 317-370.

[3] Caire, G., Coulier, W., Garijo, F., Gómez-Sanz, J., Pavón, J., Kearney, P.and Massonet. P., Message: A Methodology for Development of Agent-Based Applications, In: Methodologies And Software Engineering For Agent Systems, edited by Federico Bergenti, Marie-Pierre Gleizes and Franco Zambonelli, to be published by Kluwer Academic Publishing (New York), 2004.

[4] Message, http://www.eurescom.de/~public-webspace/P900-series/P907/index.htm, May 23, 2000.

[5] Iglesias, C.A. e Garijo, M. (2005) "The Agent Oriented Methodology MAS-CommonKADS; IN Agented-Oriented Methodologies", Brian Henderson-Sellers e Paolo Giorgini (ed.), IDEA Group Publishing, p. 46-78.

[6] ADELFE (Atelier de Développement de Logiciels à Fonctionnalité Emergente) at http://www.irit.fr/ADELFE.

[7] Tropos at BRESCIANI, Paolo et al. Tropos: An Agent-Oriented Software Development Methodology, Kluwer Academic Publishers, 2004.

[8] MaSE, at http://macr.cis.ksu.edu/projects/mase.htm

[9] Prometheus at http://www.cs.rmit.edu.au/agents/SAC2/methodology.html

[10] Schreiber, G et al, "Knowledge Engineering and Management: The CommonKADS Methodology", Cambridge, MA,1999.

[11] Rumbaugh, J., Jacobson, I. E Booch,G., The Unified Modeling Language Reference Manual, Second edition, Addison,-Wesley, 2004.

[12] AUML at http://www.auml.org/auml/

Castro, J., Kolp M. and Mylopoulos. J., Towards Requirements-Driven Information Systems Engineering: The Tropos Project. In Information Systems, Elsevier, Amsterdam, The Netherlands, 2002.

[13] Brian Henderson-Sellers e Paolo Giorgini (ed.), Agent-Oriented Methodologies, IDEA Group Publishing, 2005.

[14] JADE: ,Java Agent DEvelopment Framework, available at <http://jade.cselt.it/>, 2004.

[15] FIPA-OS Developers Guide, avaiable at <http://fipa-os.sourceforge.net/>, 2004.

[16] Iglesias, C. Á., "Definición de una Metodología para el Desarrollo de Sistemas Multiagente", Tese de Doutorado, Departamento de Engenharia de Sistemas de Telecomunicação, Universidade Politécnica de Madri. Madri, 1998, 322p.

[17] Iglesias, C.A. e Garijo, M., "UER Technique: Conceptualisation for Agent-Oriented Development", In Proceeding of 5th International Conference on Information Systems Analysis and Synthesis (ISAS'99), Vol 5, 1999, p. 535-540.

[18] Bernon, C., Camps, V., Gleizes, M. P. and Piscard, G., "Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology"; IN: Agent-Oriented Methodologies, Brian Henderson-Sellers e Paolo Giorgini (ed.), IDEA Group Pu blishing, 2005, pp. 172-202.

[19] Odell, J., Parunak, H. and Bauer, B., "Representing agent interaction protocols in UML," In Agent-Oriented Software Engineering, First International Workshop (AOSE 2000), Ciancarini, P., Wooldridge, M., Eds., LNCS 1957 Springer, Limerick, Ireland, 2001, pp. 121-140.

[20] Tropos, at http://www.troposproject.org/

[21] E. Yu, "Modelling Strategic Relationships for Process Reengineering", PhD Thesis, Graduate Department of Computer Science, University of Toronto, Canada, 1995.

[22] Coburn, M., "Jack Intellignet Agents: User Guide version 2.0," AOS Pty Ltd, 2000.

[23] Scott A. DeLoach. Multiagent Systems Engineering of Organization-based Multiagent Systems. 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'05). May 15-16, 2005, St. Louis, MO. Springer LNCS Vol 3914, Apr 2006, pp 109 - 125.

[24] Cysneiros, L. M., Werneck, V. M. B.; Yu, Eric, "Evaluating Methodologies: A Requirements Engineering Approach Through the Use of an Exemplar". Journal of Computer Science & Technology, USA, v. 5, n. 2, 205, pp. 71-79.

[25] Cysneiros, L. M., Werneck, V. M. B., Amaral, J. and Yu, E., "Agent/Goal Orientation versus Object Orientation for Requirements Engineering: A Practical Evaluation Using an Exemplar", In: Proc. of VIII Workshop in Requirements Engineering, Porto, 2005, pp.123-134, ISBN 972-752-079-0.

[27] Coppieters, A. M., Marzulo, L.A.J., Kinder, E. e Werneck, V.M., "Modelagem Orientada a Agentes utilizando MESSAGE", Cadernos do IME-Série Informática, Rio de Janeiro, v.18, 2005, pp. 38-46.

[28] Werneck, V. M.; Pereira, L. F.; Silva, T. S.; Almentero, E. K.; Cysneiros, L. M.; . "Uma Avaliação da Metodologia MAS-CommonKADS", In: Proceedings of the Second Workshop on Software Engineering for Agent-oriented Systems, (SEAS´06), Florianópolis, 2006, pp. 13-24.

[29] Werneck, Vera M. B. ; Cysneiros, Luiz Marcio ; KANO, A. Y.. Evaluating ADELFE Methodology in the Requirements Identification. In: Proceedings Of 10th Workshop On Requirements Engineering. Toronto: York University Printing Services, 2007. V. 1. P. 13-24.

[30] Yu,E., Cysneiros.L.M., "Agent-Oriented Methodologies-Towards a Challenge Exemplar" in Proc of the 4th Intl. Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 2002) Toronto May 2002.