OplA: Uma Metodologia para o Desenvolvimento de Sistemas Baseados em Agentes e Objetos

Mellyssa De Martins Schwambach, Juliana Pezzin, Ricardo de Almeida Falbo

Mestrado em Informática - Universidade Federal do Espírito Santo, Vitória - ES
mellyssa_s@hotmail.com, juliana_pezzin@hotmail.com, falbo@inf.ufes.br

Resumo. A pesquisa em Engenharia de Software Orientada a Agentes tem como ponto de partida a possibilidade de se modelar um sistema de software usando abstrações do nível de agentes. Entretanto, há de se considerar que nem todas as abstrações de um sistema podem ser capturadas na forma de agentes. Assim, é importante haver metodologias que combinem agentes com abstrações de outro tipo, tal como objetos. Este artigo apresenta OplA, uma metodologia para desenvolvimento de sistemas baseados em agentes e objetos, e discute, através de um estudo de caso, as fases de análise e projeto propostas.

1 Introdução

Tratar sistemas como sendo compostos de agentes autônomos que interagem entre si tem sido considerada uma promissora abordagem para o desenvolvimento de aplicações em domínios complexos. Essa abordagem introduz novas abstrações e desafios e, portanto, novas metodologias são necessárias, sobretudo no que concerne às atividades de análise e projeto [1]. São muitos os estudos realizados nessa área e, a cada dia, novas soluções são propostas. A maioria dessas soluções, no entanto, está focada somente no desenvolvimento de sistemas baseados em agentes e desconsidera que, na grande maioria das vezes, um sistema não deve ser visto como sendo composto apenas de agentes de software. Ao contrário, agentes atuam em um ambiente e, portanto, é muito importante modelar esse ambiente. Na modelagem do ambiente dos agentes, outros paradigmas, como a orientação a objetos, podem ser mais adequados. Neste contexto, é importante a definição de uma metodologia que apóie o desenvolvimento de sistemas baseados em agentes e objetos. Tal metodologia deve explorar os conceitos relevantes dos dois paradigmas envolvidos e ser de fácil uso, tirando proveito de soluções já conhecidas e, principalmente, usando uma linguagem de modelagem já adotada universalmente, tal como a UML [2].

Este artigo tem como objetivo principal apresentar OplA (*Object plus Agent Methodology*), uma metodologia para o desenvolvimento de sistemas baseados em agentes e objetos. A idéia de se criar OplA nasceu das dificuldades encontradas no desenvolvimento de sistemas multi-agentes usando a metodologia Gaia, na versão definida em [3]. Com base nessas dificuldades e na inspeção de outras abordagens de desenvolvimento orientadas a agentes e orientadas a objetos, identificou-se uma lacuna entre as propostas considerando esses dois paradigmas. Assim, OplA busca

abranger algumas das principais atividades técnicas do processo de software – Especificação de Requisitos, Análise, Projeto, Implementação e Testes – em um desenvolvimento adotando conjuntamente os paradigmas de agentes e de objetos. OplA propõe a realização de certas atividades, define modelos a serem produzidos usando a UML e fornece algumas diretrizes para a condução das atividades e para a elaboração dos modelos prescritos.

Este artigo está estruturado da seguinte forma: a seção 2 discute brevemente alguns dos principais aspectos da Engenharia de Software Orientada a Agentes. A seção 3 apresenta OplA sucintamente, enquanto a seção 4 trata com mais detalhes as principais diretrizes para as fases de análise e projeto através de um estudo de caso. Por fim, as seções 5 e 6 discutem, respectivamente, trabalhos correlatos e as conclusões deste trabalho.

2 Engenharia de Software Orientada a Agentes

À medida que aumenta a complexidade dos sistemas de software a serem desenvolvidos, novas abordagens de Engenharia de Software são propostas, visando resolver os novos problemas que surgem. Em termos das abstrações usadas para gerenciar a complexidade, é possível perceber uma evolução contínua que vai desde a abstração de procedimentos, passando por tipos abstratos de dados, objetos, componentes e, mais recentemente, agentes. Alguns desses estágios, dado o nível das abstrações correspondentes, deram origem a metodologias de análise e projeto, como é o caso dos paradigmas estruturado (calcado nas abstrações relacionadas a procedimentos) e orientado a objetos (calcado nas abstrações envolvidas na orientação a objetos). Outros não originaram metodologias abrangentes envolvendo as várias das fases do desenvolvimento, mas apenas se concentraram em atividades mais técnicas, como projeto, implementação e testes, devido ao nível da abstração ser mais baixo. Este é, por exemplo, o caso de componentes. O desenvolvimento baseado em componentes não propõe novas abstrações para a resolução de problemas, mas apenas abstrações para organização de sistemas, visando um maior nível de reuso. De fato, essa é uma característica marcante que acaba por definir a abrangência de uma nova abordagem de Engenharia de Software: se novas abstrações para a resolução de problemas são introduzidas, novas metodologias de análise e projeto são necessárias. Este é o caso da tecnologia de agente e, portanto, faz-se necessário o desenvolvimento de metodologias adequadas a este paradigma.

Agentes podem ser vistos como entidades de software capazes de realizar ações autônomas em algum ambiente, usando sua própria experiência ou conhecimento sobre o ambiente onde atuam, sempre visando alcançar seus objetivos de projeto. Agentes podem ser úteis como entidades isoladas às quais são delegadas tarefas particulares, ou podem existir em ambientes contendo outros agentes. Neste último caso, eles têm de cooperar, negociar e coordenar ações [1].

A aplicação da tecnologia de agentes se justifica quando observadas algumas características, tais como [4]: (i) o domínio envolve uma inerente distribuição de dados, capacidades para resolução de problemas e responsabilidades; (ii) há a necessidade de se manter a autonomia das partes envolvidas, bem como a estrutura

organizacional; (iii) as interações são razoavelmente sofisticadas, incluindo negociação, compartilhamento de informações e coordenação; e (iv) a solução para o problema não pode ser inteiramente descrita do princípio ao fim, devido às diversas mudanças ambientais que podem ocorrer, além da imprevisibilidade e dinamicidade dos processos de negócio.

Inicialmente, como um novo paradigma, métodos e técnicas adequados a outros paradigmas, sobretudo orientados a objetos, foram empregados no desenvolvimento de sistemas baseados em agentes. Entretanto, ainda que haja similaridades entre agentes e objetos (por exemplo, ambos usam trocas de mensagens para a comunicação e podem usar herança e agregação para definir sua arquitetura), há também diferenças significativas, entre elas: (i) Um objeto só pode exibir autonomia sobre seu estado, ou seja, pode ter controle sobre suas informações (objeto encapsulado), mas um objeto não exibe controle sobre seu comportamento (métodos). Agentes têm controle tanto sobre seu estado quanto sobre seu comportamento. Assim, a troca de mensagens entre agentes não deve ser vista como a invocação de métodos, mas como requisições de ações a serem executadas [5]. (ii) Objetos não são autônomos, no sentido de sua atividade interna só poder ser iniciada através de uma requisição de serviço vinda de uma linha de controle (*thread*) externa [1]. Agentes, por outro lado, têm sua própria linha de controle. (iii) Em aplicações tradicionais de objetos, não há uma concepção explícita de "ambiente", isto é, objetos encapsulam dados em termos de atributos internos ou percebem o mundo apenas em termos de referências a outros objetos [1].

Ainda que haja essas diferenças, deve-se levar em conta que elas estão ficando cada vez menos nítidas nos atuais sistemas de objetos e componentes. Abstrações de objeto tradicionais têm sido enriquecidas com a incorporação de novas características, tais como linhas de controle internas, tratamento de eventos e dependências de contexto. Além disso, na grande maioria das vezes, agentes são implementados em termos de objetos ativos e componentes. Entretanto, isso não quer dizer que agentes não adicionam novas facilidades para ajudar a resolver problemas; muito pelo contrário. O fato de objetos e agentes estarem convergindo sob uma perspectiva tecnológica é uma evidência clara de que as abstrações introduzidas pela tecnologia de agentes são adequadas para o desenvolvimento de sistemas de software complexos. Além disso, há de se considerar que a introdução dessas novas características pode produzir um efeito indesejado no que se refere ao nível de abstração adotado e o nível conceitual no qual os problemas têm de ser resolvidos [1].

Assim, por uma perspectiva principalmente conceitual, é importante considerar que a tecnologia de agentes introduz um novo nível de abstração, que é útil para modelar certas classes de sistemas, independentemente da tecnologia adotada para efetivamente construir agentes. Por outro, é bom levar em conta que as abstrações de objetos e componentes podem ser úteis em níveis mais baixos de abstração, envolvendo o projeto e a implementação de agentes. Logo, a orientação a agentes é um paradigma de engenharia de software e, portanto, requer metodologias adequadas a ele, que podem, dependendo do nível de abstração, estar pautadas em outros paradigmas, sobretudo a orientação a objetos.

Neste cenário, a pesquisa em Engenharia de Software Orientada a Agentes tem como ponto de partida a possibilidade de se modelar um sistema de software usando abstrações do nível de agentes. Neste nível de abstração, a metáfora de organização [1] é bastante empregada e inclui: agentes, ambiente, organização, papéis, objetivos

ou metas, responsabilidades e interações. Sucintamente, um sistema multiagente (SMA) pode ser visto como um conjunto de agentes autônomos, que interagem entre si e com o ambiente, estando imerso e atuando nesse último. Cada agente pode desempenhar um ou mais papéis, possui um conjunto bem definido de responsabilidades e visa atingir objetivos ou metas. Para atingir suas metas, agentes interagem uns com os outros, sendo essa comunicação suportada por uma linguagem de comunicação entre agentes e por uma ontologia usada para associar significado ao conteúdo das mensagens.

Mesmo que o nível de agentes seja o nível natural de abstração para descrever um SMA, a falta de notações diagramáticas aceitas e ferramentas de projeto pode impedir a exploração de seus benefícios. A comunidade de pesquisa de Engenharia de Software Orientada a Agentes vem enfrentando este problema investigando a possibilidade de estender a UML (*Unified Modeling Language*) para suportar os conceitos básicos da orientação a agentes, tais como agentes, ontologias e protocolos de interação [6]. Entretanto, qualquer consideração prática sobre mudanças nas capacidades chave da UML deve levar em conta o fato da UML, na sua forma atual, ser extremamente aceita e suportada por um grande número de ferramentas. De toda forma, a UML deve ser o ponto de partida para a definição de modelos para novas tecnologias [7], tal como a orientação a agentes, principalmente porque disponibiliza alguns mecanismos de extensão.

Além da questão relacionada à linguagem de modelagem, com o objetivo de avançar na área, diversas metodologias têm sido propostas. Para se evitar a proposição de uma metodologia a partir do zero, muitos pesquisadores têm seguido a abordagem de estender metodologias existentes, dentre elas as metodologias orientadas a objetos. Entretanto, há também metodologias inovadoras, no sentido de sugerir seus próprios modelos e linguagem de modelagem.

Uma questão importante no contexto da Engenharia de Software para SMAs diz respeito à modelagem do ambiente. Um SMA está sempre situado em um ambiente e, portanto, um aspecto crucial para o desenvolvimento de SMAs é a modelagem desse ambiente, o que envolve a determinação de todas as entidades e recursos que o SMA pode explorar [1]. Nessa atividade, deve-se caracterizar o que deve ser tratado como parte do ambiente e o que deve ser tratado como agentes. Ainda que no ambiente possam existir entidades ativas, de modo geral, sobretudo em SMAs imersos em ambientes de software, o ambiente é caracterizado basicamente por entidades passivas e, portanto, a modelagem orientada a objetos pode ser empregada. Tendo isso em mente, devemos considerar que nem todas as abstrações de um sistema podem ser capturadas na forma de agentes e que é importante utilizar, de modo casado, os paradigmas orientado a objetos e orientado a agentes. Vale destacar que, neste sentido, a maioria das metodologias existentes na literatura apresenta soluções para o desenvolvimento de SMAs, contemplando apenas as características relevantes aos agentes, não se preocupam em criar pontes com o paradigma orientado a objetos, que pode ser mais interessantes para a modelagem do ambiente no qual o SMA está imerso. Diante deste cenário, foi definida OplA, uma metodologia para o desenvolvimento de sistemas baseados em agentes e objetos, apresentada a seguir.

3 OplA: Uma Metodologia para o Desenvolvimento de Sistemas Baseados em Agentes e Objetos

OplA (*Object plus Agent oriented Methodology*) é uma metodologia que busca prover diretrizes para o desenvolvimento de software utilizando os paradigma orientados a objetos e a agentes. OplA procura tirar proveito de soluções já conhecidas e, assim, adota a UML como linguagem de modelagem.

OplA define atividades técnicas a serem realizadas, artefatos a serem produzidos e técnicas a serem aplicadas. A figura 1 apresenta um modelo de processo contendo as principais atividades propostas por OplA e os artefatos a serem produzidos por elas.

Devido às limitações de espaço, não é possível apresentar a metodologia completa e em detalhes. Para um exame da proposta completa, recomenda-se consultar [9], onde, além da metodologia, um estudo de caso completo é também apresentado. Para facilitar a apresentação das principais diretrizes de OplA neste artigo, na próxima seção, suas principais atividades – Análise e Projeto – são apresentadas no contexto de uma utilização no desenvolvimento de uma aplicação de alocação de recursos humanos a atividades de um projeto. Na presente seção, essas atividades são comentadas apenas superficialmente, enquanto as demais – Especificação de Requisitos, Implementação e Testes – são apresentadas com um pouco mais de detalhes, já que não são discutidas na seção seguinte.

3.1 As Atividades Técnicas de OplA

Na atividade de *Especificação de Requisitos*, é realizada a captura dos requisitos funcionais do sistema, usando a modelagem de casos de uso. Os requisitos são capturados sob uma perspectiva dos usuários e, portanto, são independentes do paradigma adotado no desenvolvimento. Assim sendo, OplA não apresenta diretrizes adicionais às normalmente estabelecidas por essa técnica.

A atividade de *Análise* é decomposta em três sub-atividades (Análise de Casos de Uso, Análise Estrutural e Análise Comportamental), que são, por sua vez, decompostas em outras sub-atividades contemplando aspectos dos dois paradigmas. Essa atividade é discutida com detalhes na próxima seção, usando um estudo de caso para tornar mais claro o entendimento.

Na atividade de *Projeto*, devem ser definidas as tecnologias envolvidas no projeto - linguagem de programação, infra-estrutura para a construção de agentes, protocolos e linguagens de comunicação etc. Com base nessas definições, os modelos da Análise devem ser revistos para levar em conta aspectos da tecnologia a ser adotada. Além disso, deve-se definir, também, a arquitetura do software em questão. Assim como a atividade de Análise, essa atividade é também discutida com um pouco mais de detalhes na próxima seção.

Na atividade de Implementação é necessário adicionar às decisões de projeto outras que tornarão factível a implementação do sistema. As decisões de projeto formam a base para que a implementação do sistema comece. Assim, na prática, à medida que as principais decisões de projeto forem sendo tomadas, a implementação pode ser iniciada paralelamente.

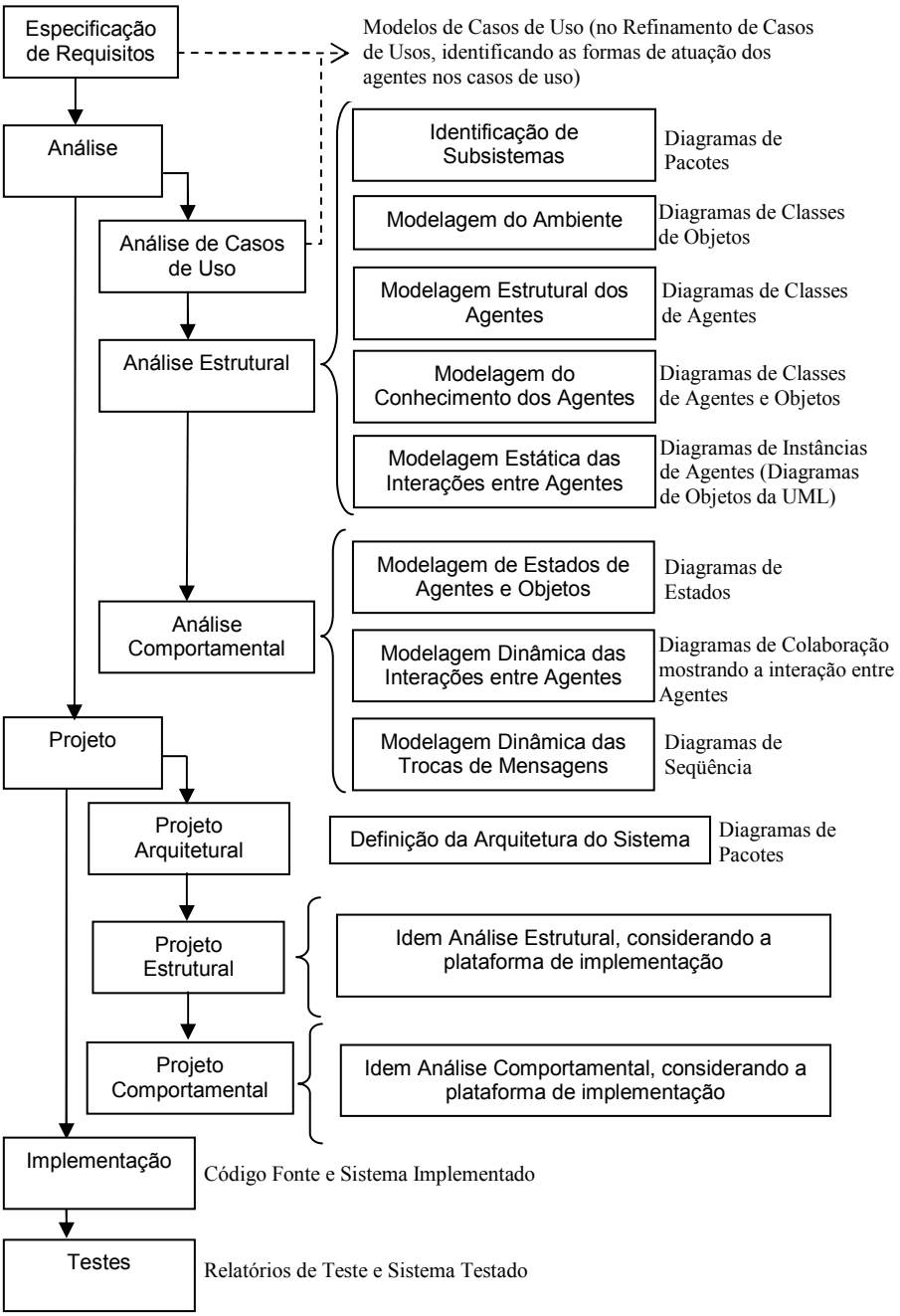

Figura 1 – Principais Atividades de OplA e seus artefatos correspondentes.

Por fim, na atividade de Testes, devem ser realizados os testes para verificar e validar o sistema, considerando, também, os aspectos intrínsecos à tecnologia de agentes, tal como a comunicação entre os agentes, e entre os agentes e o ambiente.

Os testes devem ser realizados em três etapas, a saber: testes de unidade, de integração e de validação. Os Testes de Unidade são aqueles focados na menor parte do sistema desenvolvido e podem ser considerados como parte da atividade de implementação. No caso dos sistemas desenvolvidos segundo OplA, tem-se como unidades classes de objetos e de agentes. Para testar essas unidades, OplA sugere que, inicialmente, sejam testadas as classes que não são clientes de outras classes, ou seja, aquelas que têm apenas o papel de classes servidoras. No caso dos objetos, essas classes são aquelas que não enviam mensagens para outras classes de objetos. No caso dos agentes, pode ser difícil se deparar com tal situação, uma vez que, na maioria das vezes, a base de conhecimento de um agente é composta de objetos. Assim, é interessante testar, primeiro, as classes de objetos que compõem a base de conhecimento de um agente, para então testar o agente. Contudo, vale destacar que, entre as classes de agentes, é interessante testar, primeiro, aquelas que atuam somente sobre a sua base de conhecimento, sem interagirem com outros agentes.

Uma vez testadas as unidades do sistema em separado, devem ser realizados os testes de integração. OplA sugere uma estratégia *bottom-up* baseada em casos de uso para os testes de integração. Inicialmente, deve-se testar as funcionalidades do sistema separadamente, caso de uso a caso de uso, sem considerar, ainda, a atuação dos agentes. Uma vez testados os casos de uso sem a atuação dos agentes, deve-se integrar as formas de atuação (os agentes) e prosseguir com os testes. Quando a forma de atuação exigir a presença de mais de um agente, deve ser testada a integração entre os agentes e se a comunicação entre eles está ocorrendo corretamente. Neste caso, deve-se começar testando a integração dos agentes aos pares, até que seja possível realizar o caso de uso com a atuação de todos os agentes.

À medida que os casos de uso vão sendo testados individualmente, eles podem ser integrados, até que todo o sistema seja testado. Deve-se realçar que os testes de integração, em nível de subsistema ou acima, devem ser feitos por uma equipe de testes independente. Esses testes devem ser realizados, preferencialmente, no ambiente alvo, isto é, no ambiente onde o sistema executará quando em operação.

Finalmente, uma vez totalmente integrado o sistema, os Testes de Validação devem ser realizados. Esses testes são responsáveis por descobrir os erros relacionados ao não atendimento de requisitos do cliente.

4 As Atividades de Análise e Projeto em OplA através de um Estudo de Caso - Alocação de Recursos a um Projeto

A alocação de um recurso é uma atividade de extrema importância e saber quem deve realizar determinada parte de um trabalho não é uma tarefa trivial. O gerente deve conhecer as competências de cada um dos membros da sua equipe para, então, poder fazer a melhor escolha. Importante saber, também, que nem sempre o especialista da área é a melhor opção num determinado momento e conhecer as atuações dos membros da equipe em outros projetos, sobretudo similares, pode auxiliar na decisão de quem alocar.

Levando em consideração estes aspectos, foi desenvolvida uma aplicação de alocação de recursos humanos para um Ambiente de Desenvolvimento de Software,

utilizando a tecnologia de agentes. A seguir, são discutidos alguns aspectos das fases de Análise e Projeto, apresentando os modelos desenvolvidos no contexto dessa aplicação. Vale destacar que, em OplA, os mecanismos de extensão da UML são utilizados para permitir a modelagem de sistemas envolvendo objetos e agentes, sem que seja necessário criar nenhum novo modelo.

4.1 A Fase de Análise de OplA

A primeira atividade na fase de Análise consiste em refinar o modelo de casos de uso definido na Especificação de Requisitos, para se começar a estabelecer o que será considerado ambiente e o que será tratado usando agentes. Para isso, um refinamento do modelo de casos de uso proposto deve ser realizado com vistas a identificar potenciais atuações de agentes. Um novo modelo de casos de uso deve ser gerado, agora contendo as formas de atuação dos agentes, apontadas como casos de uso estereotipados com <<atuação do agente>>. Estes casos de uso tipicamente estendem os originais apontados pela modelagem convencional de casos de uso, como mostra a figura 2. Assim, essa figura indica que, durante a realização do caso de uso "Alocar Recurso Humano a Atividade", um agente estará atuando, sugerindo os melhores recursos a serem alocados.

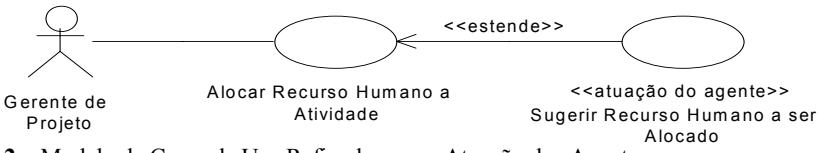

Figura 2 – Modelo de Casos de Uso Refinado com a Atuação dos Agentes

Uma forma de atuação representa um papel exercido pelo agente na execução do caso de uso. Um agente pode atuar de diversas formas num mesmo caso de uso ou em casos de uso distintos. No documento de especificação de requisitos revisado, o agente deve ser identificado por um nome e deve-se descrever sua(s) forma(s) de atuação, como mostra a tabela 1.

Tabela 1 – Definindo as formas de atuação de um agente em OplA.

Nome	Forma de Atuação	Descrição da Forma de Atuação
AgConselheiroAlocacaoRH	Sugerir recursos humanos a ser alocados a uma atividade	Com base nas especialidades de cada um dos recursos humanos que compõem a equipe do projeto e no conhecimento sobre a alocação desses recursos humanos em projetos similares, o agente sugere quais deles desempenhariam melhor a atividade em questão.

Uma vez refinados os modelos de casos de uso, o passo seguinte, Análise Estrutural, consiste em modelar o ambiente e os agentes nele atuando. Nesta etapa, diagramas de pacotes podem ser produzidos para organizar as classes em pacotes,

procurando ajudar no tratamento da complexidade. Mas os modelos mais importantes a serem produzidos nesta etapa são: (i) os diagramas de classes de objetos, usados para modelar o ambiente, mostrando as classes de objetos, seus atributos e associações, como na modelagem orientada a objetos tradicional, (ii) os diagramas de classes de agentes, usados para modelar os agentes, suas relações estruturais e suas dependências, (iii) os diagramas de classes de agentes e objetos, usados para modelar as associações entre agentes e objetos, buscando definir a parte factual do conhecimento que os agentes têm sobre o ambiente, e (iv) os diagramas de instâncias de objetos (diagramas de objetos da UML), usados para mostrar as interações entre agentes em um dado momento.

Quando analisando as entidades do domínio do problema, OplA sugere que as entidades ativas, isto é, aquelas que, de forma autônoma, produzem ou consomem recursos de informação, sejam consideradas candidatas a agentes. As demais devem ser modeladas como objetos pertencendo ao ambiente no qual os agentes atuarão. A figura 3 mostra parte do diagrama de classes de objetos, modelando os principais elementos passivos envolvidos na alocação de recursos humanos a atividades de um projeto, que correspondem ao ambiente no qual os agentes atuarão. A modelagem do ambiente segue a modelagem OO tradicional, já que, em OplA, o ambiente não deve possuir entidades ativas

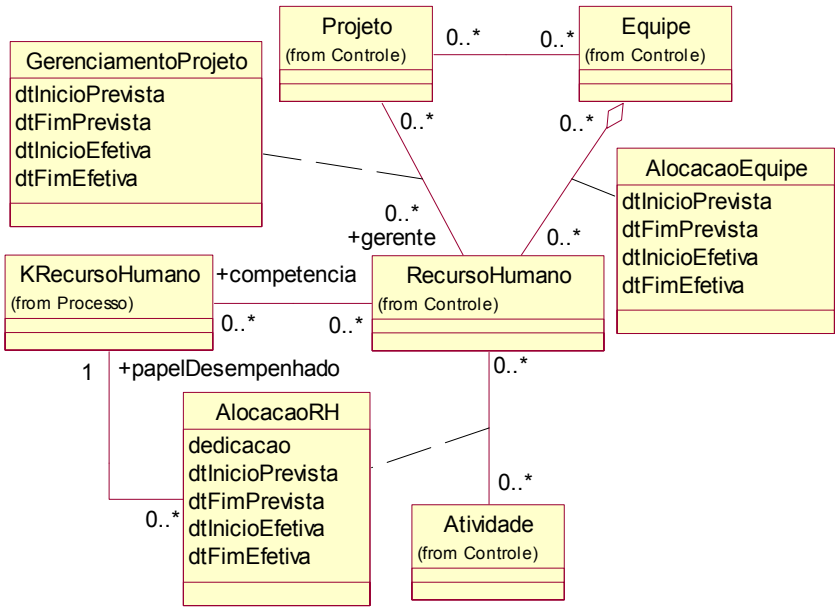

Figura 3 – Diagrama de Classes de Objetos para o Sistema Proposto.

Os agentes, por sua vez, são modelados em OplA como classes ativas [2] estereotipadas com <<agente>>. Essas classes se distinguem das classes tradicionais por possuírem uma borda externa mais escura, mas possuem todos os compartimentos usuais para nome, atributos e operações. Uma vez que um agente é definido também pelas suas responsabilidades, um compartimento extra é usado para listar as

responsabilidades de um agente. Além disso, são utilizados estereótipos para melhorar a identificação e a separação de operações, protocolos e responsabilidades, como mostra a figura 4, que apresenta parte do diagrama de classes de agentes para o sistema proposto. Nesse diagrama há dois agentes: *AgConselheiroAlocacaoRH*, que tem como objetivo sugerir recursos humanos a serem alocados a uma atividade, e *AgIdentificadorProjetosSimilares*, que é responsável por identificar projetos similares ao projeto corrente. A relação de dependência entre eles indica que *AgConselheiroAlocacaoRH* solicita uma ação de *AgIdentificadorProjetosSimilares*, no caso, pedindo que informe os projetos similares.

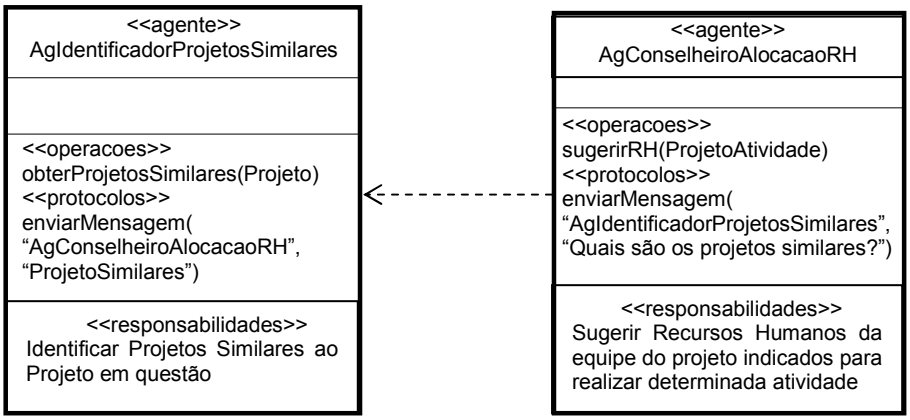

Figura 4 – Exemplo de Diagramas de Classes de Agentes em OplA.

Após terem sido definidos os diagramas de classes de agentes e os diagramas de objetos, devem ser elaborados os diagrama de classes de agentes e objetos. OplA sugere que, para cada agente identificado, um diagrama desse tipo seja elaborado, representando, dessa forma, a parte factual da base de conhecimento de cada um dos agentes em termos dos objetos que modelam o ambiente. A figura 5 mostra parte do diagrama de classes e objetos elaborado para o *AgConselheiroAlocacaoRH*. Deve-se realçar que outras classes de objetos (*Equipe*, *AlocacaoEquipe*, *KRecursoHumano* e *KAtividade*), mostradas na figura 2, também são usadas por esse agente. Contudo, ele não precisa ter uma referência explícita a esses objetos e, por isso, não foram mostrados na figura 5.

Ao fazer a modelagem de classes, sejam classes de agentes ou de objetos, deve-se ter em mente que todas as classes devem corresponder a alguma abstração tangível ou conceitual no domínio da aplicação. Assim, uma classe de agentes ou objetos bem-estruturada: (i) proporciona uma abstração clara de algo definido a partir do vocabulário do domínio do problema ou do domínio da solução; (ii) incorpora um conjunto pequeno e bem-definido de responsabilidades e é capaz de executá-lo de modo eficiente; (iii) fornece nítida separação entre a especificação da abstração e sua implementação; (iv) é compreensível e simples, além de extensível e adaptável.

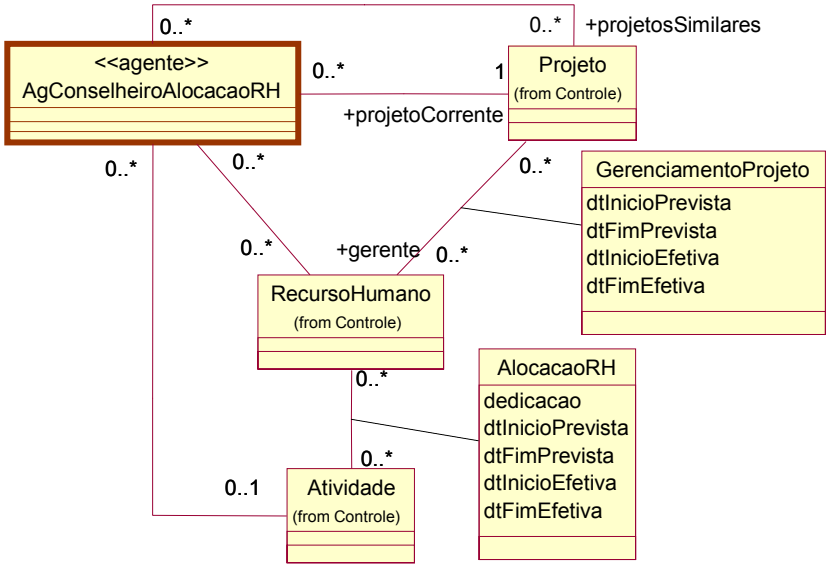

Figura 5 – Diagrama de Classes de Agentes e Objetos para o *AgConselheiroAlocacaoRH*.

Finalizando a análise estática da aplicação, OplA sugere a modelagem estática da interação entre os agentes, usando diagramas de objetos. Através desses diagramas, é possível visualizar quantas instâncias de um mesmo agente existem no sistema em um dado momento e como se dá a sua interação com as demais instâncias de agentes, mostrando um retrato da comunidade de agentes. A figura 6 mostra um exemplo no contexto do sistema proposto, indicando que uma instância de *AgConselheiroAlocacaoRH*, em um determinado momento, estará interagindo com uma e somente uma instância de *AgIdentificadorProjetosSimilares*.

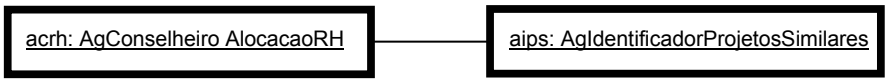

Figura 6 – Diagrama de Objetos – Interação dos Agentes

Além da modelagem dos aspectos estáticos de um sistema, devem ser modelados, também, os seus aspectos dinâmicos, objeto da atividade de Análise Comportamental. Para esta atividade, OplA sugere o uso de diagramas de estados e diagramas de interação, esses últimos em suas duas formas: diagramas de seqüência, representando os cursos normais e alternativos dos casos de uso, e os diagramas de colaboração, para mostrar a interação existente entre os agentes durante sua existência no sistema. Os diagramas de estado, por sua vez, além de serem utilizados para modelar os estados dos objetos, podem ser usados para modelar os possíveis estados de agentes.

Uma vez que os diagramas de seqüência são diagramas de interação que dão ênfase à ordenação temporal das mensagens, eles são utilizados em OplA para modelar aspectos dinâmicos do sistema relacionados diretamente à realização de casos de uso e é através deles que aspectos de negociação devem ser explicitados. A figura 7

mostra um exemplo da interação entre os elementos de modelagem (agentes e objetos) envolvidos no caso de uso "Sugerir Recurso Humano a ser Alocado" (figura 1), mostrando que o agente *AgConselheiroAlocacaoRH* interage com o agente *AgIdentificadorProjetosSimilares* para saber os projetos similares ao projeto corrente e, depois, com objetos do ambiente para poder sugerir os melhores recursos a serem alocados para o projeto em questão. Vale destacar que a figura 7 apresenta apenas parte da interação necessária. Além disso, é importante lembrar que a seta pontilhada, segundo a UML, representa um retorno. Ou seja, a mensagem *"Quais são os projetos similares?"* é resultado da ação *enviarMsg* do *AgConselheiroAlocacaoRH*, enquanto a mensagem *"Projetos Similares"* é o resultado da ação *enviarMsg* do *AgIdentificadorProjetosSimilares*.

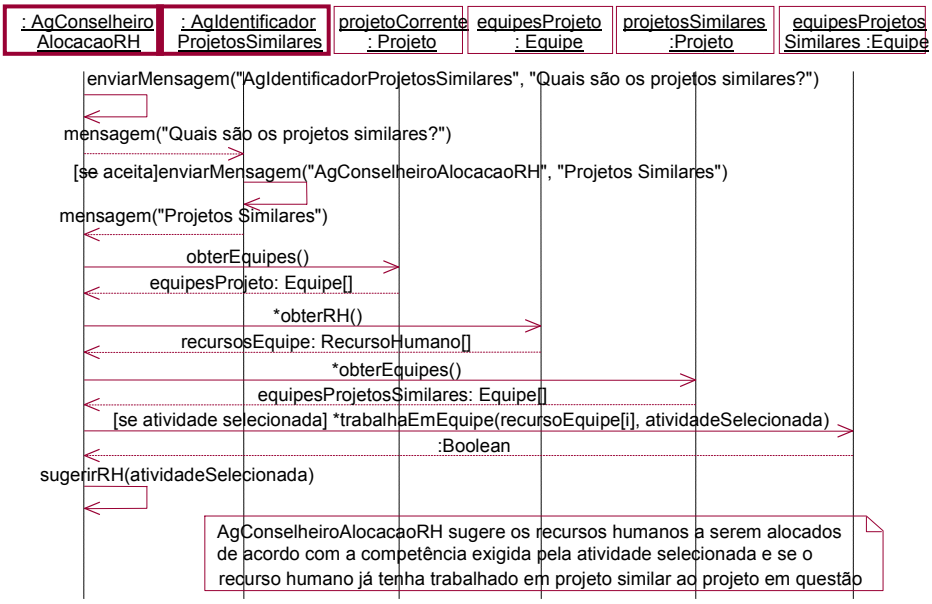

Figura 7 – Parte de um Diagrama de Seqüência.

Um diagrama de colaboração, por sua vez, é um diagrama de interação que dá ênfase à organização estrutural dos elementos de modelagem (objetos e agentes) que enviam e recebem mensagens e, portanto, é bastante indicado para modelar a colaboração entre os agentes, como mostra a figura 8. Em OplA, esse diagrama é usado para mostrar somente as instâncias dos agentes se comunicando. No exemplo, o agente *a1: AgConselheiroAlocacaoRH* pergunta ao agente *a2:AgIdentificadorProjetosSimilares* quais os projetos similares ao projeto corrente. Se *a2* aceita o pedido de *a1*, ele retorna a informação desejada.

Ao final da fase de análise, o sistema terá sido modelado tanto de uma perspectiva de agentes, quanto do ambiente onde eles vão atuar, este último modelado segundo o paradigma de objetos. O documento de especificação produzido deverá, então, servir de base para a fase de projeto, discutida brevemente a seguir.

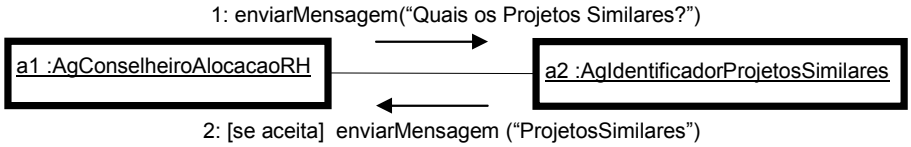

Figura 8 – Diagrama de Colaboração

4.2 A Fase de Projeto de OplA

A fase de projeto envolve importantes decisões a serem tomadas no que concerne à definição da plataforma de implementação a ser usada, o que inclui definições sobre a linguagem de programação, o mecanismo de persistência de dados, o protocolo de comunicação entre agentes etc. A decisão de se utilizar uma infra-estrutura para apoiar a construção de agentes também deve ser incluída na fase de projeto. Com base nessas definições, deve ser estabelecida a arquitetura do sistema e os modelos estruturais e comportamentais desenvolvidos durante a fase de análise devem ser revistos e adequados ao domínio da solução.

Em OplA, a fase de projeto está dividida em três partes: Projeto Arquitetural, Projeto Estrutural e Projeto Comportamental. A etapa de projeto arquitetural visa definir a organização do sistema em pacotes e componentes.

O projeto estrutural tem por base os modelos gerados na atividade de análise estrutural. A partir das tecnologias que serão utilizadas, os diversos diagramas de classes devem ser refinados, detalhando com maior rigor as operações, protocolos, atributos e responsabilidades das classes envolvidas. Tipicamente, para cada um dos diagramas de classes gerados na fase de análise, um novo diagrama de classe deve ser produzido. Esses diagramas contêm características ligadas diretamente às tecnologias que estão sendo utilizadas. Por exemplo, caso se defina usar KQML [10] como linguagem de comunicação entre agentes, as classes ativas que representam os agentes terão seus protocolos definidos e representados segundo as performativas de KQML. Se a linguagem de programação utilizada for Java, então eventuais modelos usando herança múltipla devem ser revistos, já que Java não suporta esse recurso. Ainda, se foi escolhida uma infra-estrutura para apoiar a construção de agentes, tal como AgeODE [11], então os diagramas de classe têm de estar compatíveis com as exigências de tal infra-estrutura. A figura 9 mostra o diagrama de classes de agentes revisado, levando em conta que a plataforma de implementação do sistema proposto inclui a infra-estrutura AgeODE, a linguagem de programação Java e a linguagem de comunicação KQML.

AgeODE [11] provê alguns tipos de agentes e sua comunicação é baseada em um modelo cliente-servidor, no qual agentes clientes se comunicam sempre através de um agente roteador. Em relação aos tipos dos agentes da infra-estrutura AgeODE, *AgConselheiroAlocacaoRH* se comporta como sendo dos tipos de agentes *AgInterface* e *AgInformacao*, sendo o segundo o seu tipo principal. Uma vez que Java não suporta herança múltipla, *AgConselheiroAlocacaoRH* herda do segundo e implementa a interface do primeiro.

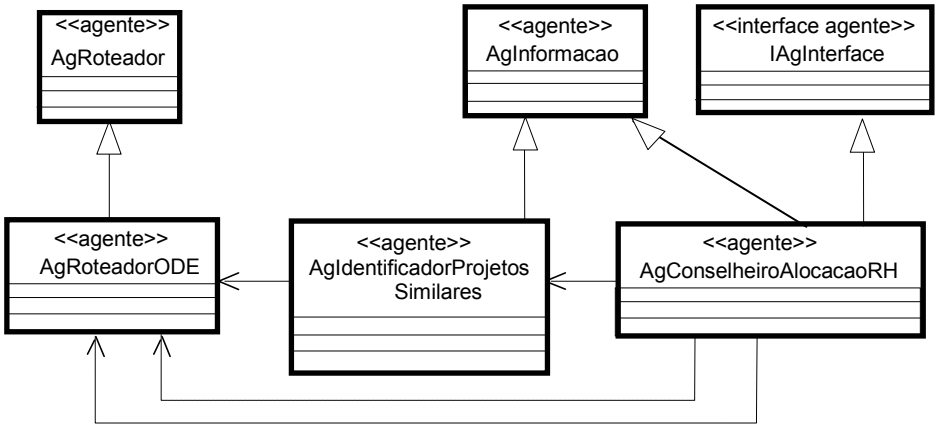

Figura 9 – Exemplo de Diagramas de Classes de Agentes da fase de Projeto.

De maneira análoga ao projeto estrutural, o projeto comportamental envolve a remodelagem dos aspectos comportamentais do sistema, levando-se em consideração as tecnologias envolvidas e requisitos não funcionais. Essa atividade consiste, basicamente, de um refinamento dos modelos gerados na análise comportamental. Por exemplo, na fase de projeto, os diagramas de colaboração deverão ter a interação entre os agentes explicitada com base no protocolo de comunicação adotado.

Assim como o diagrama mostrado na figura 9, os demais diagramas apresentados anteriormente também têm suas contrapartidas na fase de projeto. Entretanto, por limitações de espaço, eles não foram apresentados. Além disso, vale realçar que apenas os diagramas que sofrerem alterações importantes para guiar a implementação devem ser revistos, evitando-se esforço desnecessário na remodelagem de todos os diagramas propostos na fase de análise.

5. Trabalhos Correlatos

Conforme citado na seção 2, há diversos trabalhos descritos na literatura voltados para o paradigma orientado a agentes (OA). OplA se propõem a ser uma metodologia de segunda geração, no sentido de buscar reunir em uma única proposta as características consideradas mais relevantes encontradas tanto no paradigma OA quanto no paradigma OO. Assim, OplA foi inspirada em diversos trabalhos, dentre eles Gaia [1, 3], MESSAGE [7], UML [2] e extensões para agentes [6], incluindo a AUML [12].

Gaia [3, 1] trata das fases de análise, projeto arquitetural e projeto detalhado de SMAs, definindo seus próprios modelos e sua própria linguagem de modelagem. Em Gaia, as fases são dispostas como um processo de desenvolvimento que parte de modelos conceituais em direção a modelos mais concretos, que possam ser implementados usando objetos.

MESSAGE (*Methodology for Engineering Systems of Software Agents*) [7] é uma metodologia de engenharia de software orientada a agentes, cobrindo as fases de

análise e projeto de SMAs. MESSAGE estende a UML, introduzindo conceitos no nível de conhecimento dos agentes e propondo diagramas com notações próprias para a visão de agentes, a partir de diagramas de classe e de atividade da UML.

No que se refere a extensões da UML, Bergenti et al. [6] atacam o problema da definição de uma notação diagramática para suportar o projeto do nível de agente explorando uma notação baseada na UML. Quatro diagramas são propostos: diagramas de ontologias, diagramas de arquitetura, diagramas de protocolos e diagramas de papel. Já no contexto da AUML (*Agent* UML) [12, 13], são propostas revisões nos diagramas de classes, de seqüência e de atividades da UML, dentre outros, para que seja possível modelar os agentes e seus comportamentos.

OplA se baseou fortemente nos trabalhos anteriormente citados. Apenas para ilustrar, a Modelagem dos Agentes usando Diagrama de Classes de Agentes tem correlatos em [6], onde diagramas de arquitetura, um tipo de diagrama de classes, são utilizados para modelar agentes e a interação entre eles, e em [12], onde se propõe uma revisão nos diagramas de classes da UML para que eles modelem os agentes e seus comportamentos. Além disso, várias metodologias propõem a modelagem de classes de agentes, ainda que algumas, como Gaia [1], proponham notações próprias.

Entretanto, OplA não é apenas uma coleção de idéias da literatura. Ela também tem propostas próprias, tais como a modelagem das formas de atuação dos agentes como casos de uso de extensão e a proposta de se modelar o ambiente usando o paradigma OO, em uma abordagem explícita de casamento dos paradigmas OO e OA.

6. Conclusões

O progresso na Engenharia de Software nas últimas duas décadas tem se dado através do desenvolvimento de abstrações de nível mais alto, poderosas e naturais, por meio das quais sistemas complexos são modelados e desenvolvidos. Abstração procedimental, tipos abstratos de dados e, mais recentemente, objetos e componentes são exemplos de tais abstrações. Os agentes representam um avanço similar no nível de abstração: eles podem ser usados para entender, modelar e desenvolver mais naturalmente sistemas complexos.

Se agentes têm potencial como um paradigma de engenharia de software, então é necessário desenvolver abordagens especificamente adequadas a eles. Questões como linguagem de modelagem e metodologias estão presentes nos principais estudos na área. Algumas metodologias propostas definem diretrizes que vão desde a fase da análise até a implementação. Outras apóiam algumas atividades técnicas e deixam a critério do engenheiro de software definir as demais. Vale ressaltar, contudo, que a grande maioria das metodologias encontradas na literatura está focada apenas em definir como modelar os agentes, não dando a devida ênfase à modelagem conjunta do ambiente onde eles vão atuar.

Partindo do pressuposto que o ambiente onde os agentes vão atuar é composto apenas por entidades passivas, OplA advoga o uso de objetos para modelar essa perspectiva. Assim, para o desenvolvimento de um sistema completo (incluindo agentes e seu ambiente), OplA combina técnicas, modelos e diretrizes para se efetuar um desenvolvimento de software baseado em objetos e agentes. OplA utiliza a UML

em sua forma padrão, com algumas extensões propostas usando apenas os mecanismos de extensão oferecidos pela própria UML, tal como estereótipos.

OplA tem sido utilizada com sucesso no desenvolvimento de sistemas multiagente imersos em ambientes de software. Em seu estágio atual, contudo, OplA só contempla o processo de desenvolvimento e, assim, precisa evoluir para contemplar outros processos de ciclo de vida de software, tal como Garantia da Qualidade.

Agradecimentos

Este trabalho foi realizado com o apoio do CNPq, uma entidade do Governo Brasileiro dedicada ao desenvolvimento científico e tecnológico.

Referências

1. F. Zambonelli, N.R. Jennings, M. Wooldridge. "Developing Multiagent Systems: The Gaia Methodology", ACM Transactions on Software Engineering and Methodology, vol. 12, No. 3, July 2003, pp. 317-370.
2. G. Booch, J. Rumbaugh, I. Jacobson. UML - Guia do Usuário. Editora Campus, 2000.
3. M. Wooldridge, N.R. Jennings, D. Kinny. "The Gaia Methodology for Agent-Oriented Analysis and Design", 2000.
4 M. Juchem, M.R. Bastos. "Projetando Sistemas Multiagentes em Organizações Empresariais". XVI Simpósio Brasileiro de Engenharia de Software, Brasil 2002.
5 M. Wooldridge. "Intelligent Agents". In: Weiss, G., Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence, London, The MIT Press, 1999, p. 27-77.
6 F. Bergenti, A. Poggi. "Exploiting UML in the Design of Multi-Agent Systems", ESAW Workshop at ECAI 2000.
7 Caire, G., Leal, F., Chainho, P., Evans, R., Garijo, F., Gomez J., Pavon, J., Kearney, P., Stark, J., Massonet, P.: Agent Oriented Analysis using MESSAGE/UML. AOSE 2001.
8 Schwambach, M.M. OplA: Uma Metodologia para Desenvolvimento de Sistemas Orientados a Objetos e Agentes. Dissertação de Mestrado, Brasil, 2004 (em preparação).
9 Finin, T., Mckay, D., Fritizson, R.: "An Overview of KQML: A Knowledge Query and Manipulation Language". [online]. Disponível: http://www.cs.umbc.edu/~finin/papers/.
10 Pezzin, J., Falbo, R.A., "AgeODE: Uma Infra-estrutura para Apoiar a Construção de Agentes para Atuarem no Ambiente de Desenvolvimento de Software ODE", 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering, JIISIC'2004, Madrid, Spain, November 2004.
11 Odell, J., Parunak, H.V.D., Bauer, B., "Representing Agent Interaction Protocols in UML", *Agent-Oriented Software Engineering*, Paolo Ciancarini and Michael Wooldridge eds., Springer, Berlin, pp. 121-140, 2001.
12 Bauer, B., "UML Class Diagrams: Revisited in the Context of Agent-Based Systems", Proceedings of Agent-Oriented Software Engineering (AOSE'2001), Agents 2001, pp. 1-8, Montreal, Canada, 2001.