Integrando Agentes no Ambiente de Desenvolvimento de Software ODE.

Conference Paper · January 2006

Source: DBLP

CITATIONS
0

READS
19

2 authors, including:

Ricardo de Almeida Falbo
Universidade Federal do Espírito Santo
172 PUBLICATIONS **1,661** CITATIONS

SEE PROFILE

Integrando Agentes no Ambiente de Desenvolvimento de Software ODE

Juliana Pezzin, Ricardo de Almeida Falbo
Mestrado em Informática, Universidade Federal do Espírito Santo
Av. Fernando Ferrari, CEP 29060-900, Vitória - ES – Brasil
juliana_pezzin@hotmail.com, falbo@inf.ufes.br

Abstract

The agent technology has been pointed as an interesting approach for developing complex software systems. This is the case of Software Engineering Environments (SEEs). Thus, SEEs can benefit from the use of this technology. But we have to consider that integration is a central notion to SEEs and introducing agents in a SEE also introduces another integration dimension: agent integration. Agent integration is closely related to other integration dimensions, such as presentation, data, control and knowledge. This paper presents our approach to deal with agent integration in ODE, an Ontology-based SEE.

1. Introdução

A capacidade de agentes de software para, de forma autônoma, executar (ou apoiar a realização de) algumas tarefas tem estimulado bastante o uso dessa tecnologia na automatização do processo de software. Ambientes de Desenvolvimento de Software (ADSs) são ambientes integrados de suporte automatizado ao processo de software, compostos de diversas ferramentas de apoio ao desenvolvimento e à manutenção de produtos de software ao longo de todo o seu ciclo de vida, incluindo atividades de gerência e controle da qualidade. Assim sendo, tais ambientes são potenciais beneficiários da tecnologia de agentes.

A integração de ferramentas é uma questão fundamental para ADSs. Integração em ADSs envolve diversas dimensões, tais como apresentação, dados, processo, controle e conhecimento [1,2]. A aplicação sistemática da tecnologia de agentes em um ADS introduz uma nova dimensão de integração: a integração de agentes. A dimensão de integração de agentes refere-se ao fato da construção de agentes para atuarem em um ADS dever considerar que os agentes devem apresentar homogeneidade na forma de apresentação, comunicação e atuação. Assim, essa dimensão de integração, além de introduzir novos desafios, está fortemente relacionada a outras dimensões de integração, tais como apresentação, dados, controle e conhecimento.

Este artigo discute como a integração de agentes está sendo tratada no contexto do ambiente ODE (*Ontology-based software Development Environment*) [2]. A seção 2 discute a dimensão de integração de agentes e sua relação com as demais dimensões de integração em ADSs. A seção 3 apresenta ODE brevemente e as primeiras iniciativas de uso de agentes nesse ambiente, bem como os principais problemas encontrados. Na seção 4, é apresentada AgeODE, a infra-estrutura desenvolvida visando a integração de agentes em ODE. Essa infra-estrutura é composta de um *framework* de classes de agentes e de uma ferramenta de apoio à construção de agentes usando o *framework* proposto. Neste artigo, a ênfase recai sobre a ferramenta de apoio, sendo que maiores detalhes sobre o *framework* podem ser encontrados em [3]. Finalmente, as seções 5 e 6 discutem, respectivamente, trabalhos correlatos e as conclusões deste trabalho.

2. Integração de Agentes em Ambientes de Desenvolvimento de Software

A utilização de agentes para a solução de problemas complexos de natureza distribuída e descentralizada tem sido bastante explorada por trabalhos de pesquisa realizados nas últimas décadas. Mais do que simplesmente uma nova tecnologia, a orientação a agentes tem se mostrado uma nova forma de pensar a estratégia de solução de problemas, sobretudo em casos apresentando características de distribuição e descentralização do controle.

Agentes podem ser vistos como sistemas capazes de ações autônomas em algum ambiente, a fim de atingirem seus objetivos de projeto. Um agente tipicamente percebe

seu ambiente através de sensores e tem disponível um repertório de ações que podem ser executadas para modificar o ambiente, o qual pode responder de forma não-determinada à execução de suas ações [4].

Dadas suas características, agentes podem ser muito úteis para tratar a complexidade e a diversidade de problemas enfrentados na construção de Ambientes de Desenvolvimento de Software (ADSs). ADSs são conjuntos de ferramentas CASE integradas que facilitam a realização de atividades de engenharia de software, apoiando todo o ciclo de vida de software [5]. Dentre os diversos requisitos de ADSs, vários deles podem ser mais facilmente satisfeitos através do uso de agentes, tais como [3]: permitir adequação a diferentes perfis de usuários, oferecer apoio ao trabalho cooperativo, apoiar a gerência de conhecimento e oferecer assistência inteligente aos usuários.

Entretanto, há de se considerar que a introdução de agentes em um ADS adiciona novos problemas a um dos aspectos mais desafiantes da área de ADSs: a integração de ferramentas. Integração demanda representação consistente das informações, interfaces padronizadas, significados homogêneos para a comunicação entre desenvolvedores e ferramentas, e uma efetiva abordagem que permita aos ADSs alternar entre várias plataformas [6]. De fato, a integração de ferramentas em ADSs é uma questão que envolve diversas dimensões, dentre elas [1,2]:

- Apresentação: trata de aspectos de padronização das interfaces com o usuário, reduzindo a carga cognitiva no uso de ferramentas individualmente e do ambiente como um todo;
- Dados: diz respeito aos meios como as ferramentas compartilham dados, incluindo serviços de repositório e de compartilhamento de dados;
- Controle: tem por objetivo apoiar a combinação flexível de funções do ambiente e das ferramentas, por meio de compartilhamento de funcionalidades;
- Processo: visa a garantir que as ferramentas interagem efetivamente para apoiar um processo de software definido, estabelecendo uma ligação forte entre as ferramentas e o processo de software;
- Conhecimento: refere-se à gerência do conhecimento capturado durante os projetos de software e à oferta de apoio baseado em conhecimento aos engenheiros de software durante a realização de atividades do processo.

A integração de agentes em um ADS é mais uma dimensão a ser tratada e inclui o estabelecimento de formas uniformes de atuação, apresentação e comunicação entre agentes. Além disso, pode-se notar que a dimensão de integração de agentes está fortemente relacionada com as dimensões de integração anteriormente citadas. No que se refere à dimensão de apresentação, agentes que atuam junto aos usuários têm de ter uma forma padrão de apresentação de informações. No que concerne à dimensão de dados, agentes têm de ter uma maneira uniforme de acessar os objetos do ADS, já que esses tipicamente compõem suas bases de conhecimento. Em relação à dimensão de processo, é necessário saber que agentes atuam em que atividades do processo de software, para que eles possam ser oportunamente iniciados. Finalmente, no que diz respeito à integração de conhecimento, é importante estabelecer formas também uniformes para que os agentes possam oferecer apoio baseado em conhecimento, o que tem um impacto grande na arquitetura dos agentes.

Deve-se realçar, ainda, que além das questões apontadas anteriormente, a integração de agentes em um ADS requer, também, uma forma padrão de comunicação entre agentes. Agentes tipicamente atuam na forma de sociedades e, portanto, é fundamental que eles compartilhem um mecanismo padrão de comunicação.

Visando a tratar a dimensão de integração de agentes no ambiente ODE, foi desenvolvida AgeODE, uma infra-estrutura de apoio à construção e à integração de agentes atuando nesse ambiente. AgeODE é composta por um *framework* de classes de agentes, uma ferramenta de apoio à construção de agentes usando o *framework* proposto e uma estrutura padrão de apresentação de avisos dos agentes para os usuários de ODE. A seguir, o ambiente ODE é brevemente apresentado, bem como são discutidas as primeiras iniciativas de se introduzir agentes em ODE e os problemas enfrentados que, em última instância, motivaram o desenvolvimento de AgeODE.

3. ODE: Um Ambiente de Desenvolvimento de Software Baseado em Ontologias

ODE (*Ontology-based software Development Environment*) [2] é um ADS centrado em processo, que tem sua fundamentação baseada em ontologias. Em ODE, parte-se do pressuposto que, se as ferramentas do ADS são construídas baseadas em ontologias, a integração pode ser facilitada, pois os conceitos envolvidos são bem definidos pelas ontologias [2].

ODE está em desenvolvimento no Laboratório de Engenharia de Software da Universidade Federal do Espírito Santo (LabES/UFES) e sua versão 1.0 está implantada em uma *software house*. ODE é implementado em uma plataforma livre, que inclui Java, PostgreSQL e Linux. Dentre as várias ferramentas de ODE, podem ser citadas as ferramentas de apoio à definição de processos de software [7], de apoio à gerência de riscos [8], de apoio à documentação [9], de apoio à gerência de recursos humanos, de apoio à realização de estimativas e a de modelagem segundo a UML, dentre outras.

As primeiras iniciativas de se utilizar agentes em ODE foram feitas no sentido de apoiar a customização do ambiente para usuários específicos e a realização de sub-atividades do planejamento, tal como a alocação de recursos. Além desses, uma infra-estrutura preliminar de agentes para apoiar a disseminação de conhecimento em ODE foi desenvolvida e experimentada na disseminação de conhecimento relevante para a atividade de planejamento da qualidade [10]. No desenvolvimento desses agentes, alguns problemas foram detectados, a saber: (i) agentes não possuíam autonomia, uma vez que eram tratados como objetos e, portanto, não possuíam linha de controle própria nem eram tratados como processos separados; (ii) cada agente era construído de uma maneira diferente por cada um dos desenvolvedores e, deste modo, não havia padronização nem uniformidade na construção dos agentes, o que provocava problemas de integração; (iii) finalmente, cada desenvolvedor partia do zero na árdua tarefa de se construir agentes, não havendo, deste modo, nenhuma forma de reuso [3].

Visando a tratar esses problemas, foi desenvolvida AgeODE, a infra-estrutura apresentada na próxima seção. Ainda que existam várias infra-estruturas de apoio à construção de agentes, de modo geral, elas não se destinam a apoiar especificamente a construção de agentes para atuarem em um ADS e, conseqüentemente, não tratam aspectos relacionados à dimensão de integração de agentes. Assim, a aplicabilidade da tecnologia de agentes para o desenvolvimento de funcionalidades de um ADS, a inexistência de uma infra-estrutura voltada especificamente para este fim e a grande preocupação com a questão da integração em ODE foram os principais fatores de motivação para o desenvolvimento de AgeODE [3].

Desenvolvida AgeODE, os agentes inicialmente desenvolvidos foram reconstruídos, bem como foi definido um sistema multiagente para apoiar a disseminação de conhecimento na infra-estrutura de gerência de conhecimento de ODE [11].

4. AgeODE: Uma Infra-estrutura para Apoiar a Integração de Agentes em ODE

A construção de sistemas multiagentes não é uma tarefa simples. Aspectos como autonomia, capacidade de percepção, comunicação e ação são, muitas vezes, difíceis de serem tratados no desenvolvimento de um sistema dessa natureza. Essa constatação, citada corriqueiramente na literatura, pôde ser observada nas primeiras iniciativas de se utilizar agentes em ODE.

Seguindo uma tendência da área de Engenharia de Software Orientada a Agentes, decidiu-se desenvolver uma infra-estrutura para apoiar a construção de agentes, no caso, agentes atuando em um ADS, mais especificamente, atuando em ODE. Observando algumas das diversas infra-estruturas de apoio à construção de agentes existentes, tais como JATLite [12] e ZEUS [13], foi possível constatar a importância de se oferecer aos desenvolvedores um *framework* de classes de agentes, associado a uma ferramenta de apoio à construção de agentes usando o *framework* proposto. Assim, AgeODE contém esses dois elementos principais. Além disso, AgeODE possui, também, uma estrutura padrão de apresentação de avisos dos agentes para os usuários de ODE. Essa estrutura visa a tratar a forte relação entre a dimensão de integração de agentes e a dimensão de integração de apresentação.

Vale ressaltar, ainda, que AgeODE precisa tratar as questões envolvidas na integração de agentes e, portanto, o *framework* proposto e a ferramenta de apoio à construção de agentes associada, em conjunto, têm de considerar a arquitetura dos agentes, a comunicação entre eles, o acesso aos objetos do ambiente e a associação dos agentes ao processo de software.

4.1. O *Framework* de Classes de Agentes de AgeODE.

Dependendo dos objetivos para os quais são projetados, agentes podem apresentar características bastante distintas. Esse fato conduz, naturalmente, ao estabelecimento de classificações de agentes, na qual cada tipo de agente tipicamente inclui diferentes interesses [3]. Existem várias classificações de agentes, tais como as sugeridas por [14], [15] e [16].

Visando à construção de classes específicas de agentes no *framework* de AgeODE, buscou-se estabelecer uma classificação que contemplasse um conjunto básico de tipos de agentes capaz de abranger as formas de atuação mais comuns no contexto de Ambientes de Desenvolvimento de Software, tendo se chegado ao seguinte conjunto [3]:

- **Agente de Interface:** Visa a fornecer ao usuário do ADS uma interface mais amigável, com características pró-ativas, detectando as ações do usuário quando usando a interface do ambiente.
- **Agente de Usuário:** Usa o conhecimento que possui sobre determinado usuário para permitir ou não a execução de atividades por ele. Deve ser capaz de estabelecer o perfil do usuário, buscando informações pertinentes ao usuário no contexto do ADS ou de um projeto específico. Um agente de usuário pode interagir com o usuário que ele representa, auxiliando-o na realização de atividades e na tomada de decisões.
- **Agente de Informação:** É responsável pela realização de alguma funcionalidade do sistema.

Sua tarefa principal é buscar informações e realizar tarefas dentro do domínio de problemas do ADS.

- **Agente Coordenador:** Visa a coordenar as tarefas a serem executadas em um dado momento pelos agentes do ADS. Para tal, dentre outros, deve ser capaz de: distribuir tarefas aos agentes, consolidar resultados de tarefas e/ou conjuntos de informações recuperadas de um ou mais agentes dispersos na sociedade e conhecer os agentes (e suas capacidades / habilidades específicas) que estão sob seu domínio de coordenação.

Como é possível notar pela descrição dos tipos de agentes acima, para cumprir suas responsabilidades, agentes têm de ter acesso aos objetos existentes em ODE, que podem ser vistos como parte de suas bases de conhecimento. Para tratar essa questão, um mecanismo padrão foi definido em AgeODE, pelo qual agentes invocam métodos dos objetos diretamente, abrindo um canal de comunicação entre um agente e ODE, usando *sockets* [3].

Agentes necessitam, ainda, cooperar e negociar com outros agentes. Para permitir a integração, é necessário adotar uma linguagem padrão de comunicação entre agentes. Em AgeODE, adotou-se KQML [17], dado que AgeODE foi construída usando JAT*Lite* [12] que já adota essa linguagem [3].

Uma vez que construir uma infra-estrutura com tais características é uma tarefa complexa, visando ao reuso, AgeODE foi definida como uma camada sobre a infra-estrutura JAT*Lite* [12], utilizando algumas de suas classes, principalmente para tratar a comunicação entre agentes e a autonomia dos mesmos [3]. A Figura 1 mostra um modelo parcial do *framework* de AgeODE.

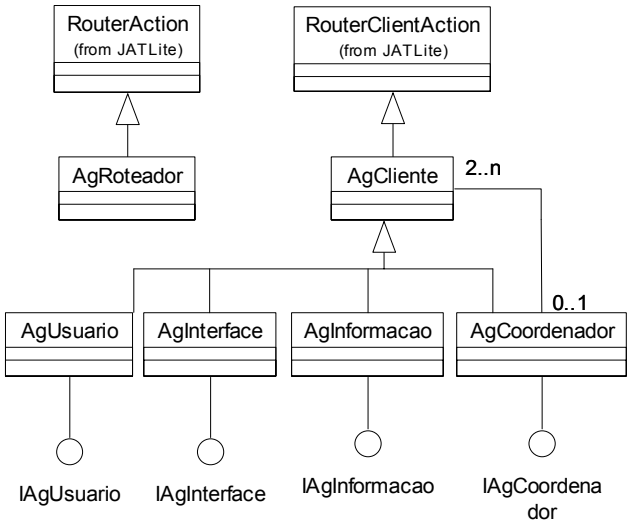

Figura 1. O *Framework* de Classes de AgeODE

Agentes de ODE podem assumir os perfis de mais de um tipo de agente cliente. Assim, para cada classe de agente cliente, há também uma interface associada. Vale lembrar que, como o *framework* de AgeODE é implementado em Java, uma linguagem que não suporta herança múltipla, os agentes só podem herdar de uma classe de agente, aquela que representa seu tipo principal, e devem implementar as interfaces de seus demais tipos. Desta forma, se um agente possui características tanto de um Agente de Interface quanto de um Agente de Usuário, então, ele pode ser implementado, por exemplo, herdando de *AgInterface* e implementando *IAgUsuario.* Maiores detalhes sobre o *framework* de AgeODE podem ser encontrados em [3].

4.2. Integração de Apresentação e AgeODE.

Tendo em vista a importância da integração de apresentação, discutida na seção 2, os agentes de ODE têm de ter uma forma padrão para enviar avisos e sugestões para os usuários do ambiente. Assim, AgeODE oferece uma interface padrão para o envio de mensagens de agentes para os usuários de ODE, mostrada na Figura 2. Esse mecanismo permite que os agentes interajam com os usuários do ambiente sem ter de conhecer ou interferir diretamente nas interfaces gráficas do mesmo.

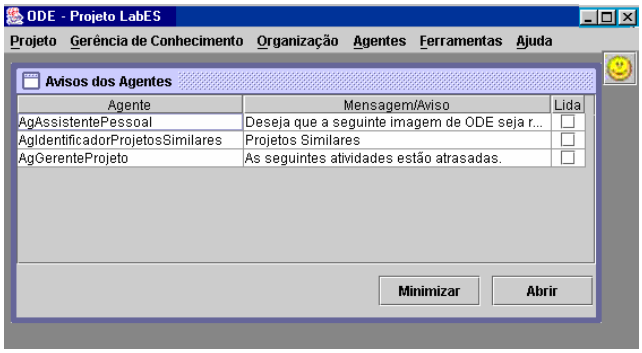

Figura 2. Janela de Mensagens de Agentes

Sempre que ODE é executado com o apoio de agentes, essa janela é aberta assim que o usuário se autentica no ambiente. A partir daí, qualquer agente que quiser enviar uma mensagem para o usuário, basta publicá-la nessa interface, executando o método `pubIncluirAvisoAgente()`, cujos parâmetros são o nome do agente, o aviso e o conteúdo. Quando o usuário de ODE recebe um aviso de um agente e o aceita, ou seja, clica no botão OK da janela de aviso (vide Figura 3), uma mensagem é enviada ao agente, informando que o usuário aceitou seu aviso / sugestão.

Vale destacar que, para que os usuários de ODE fiquem cientes das mensagens enviadas pelos agentes, há no canto superior direito da janela principal de ODE, mostrada ao fundo da Figura 2, um ícone imitando um rosto. Sempre que o usuário possuir mensagens enviadas

por agentes ainda não lidas, esse rosto fica amarelo e sorrindo, indicando que há avisos ou sugestões dadas pelos agentes para apoiar a realização de tarefas.

Um exemplo da publicação de mensagens na janela de avisos dos agentes é o caso da recuperação da imagem de ODE. Quando o usuário encerra uma sessão de ODE deixando algum(ns) artefato(s) ou ferramenta(s) aberta(s), bem como, o projeto em que ele estava trabalhando, a imagem (configuração) do ambiente para esse usuário é guardada. Constituem a imagem do usuário no ambiente: a data do último acesso, o projeto em que ele trabalhou, bem como a(s) atividade(s) que ele estava desenvolvendo ao fechar o ambiente e o(s) artefato(s) e a(s) ferramenta(s) que estavam em uso. Quando o mesmo usuário acessar novamente o ambiente, um agente, denominado Assistente Pessoal (*AgAssistentePessoal*), recupera a imagem do ambiente para o usuário e publica uma mensagem na janela de mensagens (a primeira mensagem da lista mostrada na Figura 2). Caso o usuário selecione essa mensagem, um resumo da imagem lhe é apresentado na janela de avisos (Figura 3) e lhe é perguntado se deseja que essa imagem seja restaurada. Se ele quiser, a configuração é restabelecida, não necessitando esforço para se voltar ao contexto no qual havia parado o trabalho.

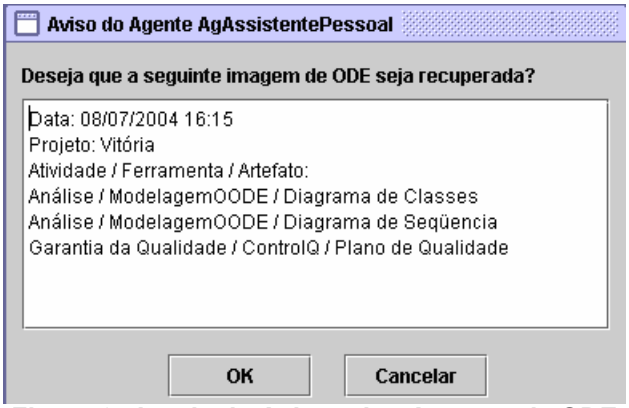

Figura 3. Janela de Avisos dos Agentes de ODE

4.3. A Ferramenta de Apoio à Construção e Integração de Agentes de AgeODE

Mesmo com um *framework* simples como o definido por AgeODE, pode ser difícil construir agentes e integrá-los ao ambiente no qual atuarão. Assim, AgeODE oferece, ainda, uma ferramenta de apoio à construção e integração de agentes em ODE, denominada CadAgeODE.

CadAgeODE apóia o processo de construção de agentes usando o *framework* de AgeODE, guiando o desenvolvedor na realização dos passos necessários para se construir agentes, gerando parcialmente o código dos agentes e registrando-os no Cadastro de Agentes de ODE.

Vale destacar que é muito importante que todos os agentes de ODE sejam cadastrados no ambiente, pois alguns necessitam conhecer outros para o bom funcionamento do ambiente. Seja o exemplo do *AgAssistentePessoal*, informalmente introduzido anteriormente. Esse agente é responsável por acompanhar um usuário de ODE desde o momento em que ele acessa o ambiente até o momento em que ele deixa de usá-lo. *AgAssistentePessoal* deve conhecer todos os agentes que atuam em ferramentas de ODE, para poder iniciá-los quando a respectiva ferramenta for iniciada pelo usuário. Caso um agente que atue em uma ferramenta não esteja cadastrado no cadastro de agentes de ODE, o *AgAssistentePessoal* não saberá de sua existência e, conseqüentemente, não poderá iniciá-lo. Assim, o cadastro de agentes de ODE serve, principalmente, de base de conhecimento sobre os agentes atuando em ODE, de modo que os agentes se conheçam e para que futuros desenvolvedores possam saber quais agentes existem.

Uma vez que todos os agentes atuando em ODE têm de ser cadastrados, optou-se por incorporar essa funcionalidade a CadAgeODE, fazendo com que essa ferramenta disponibilize funcionalidades de apoio ao cadastro e à construção de agentes para atuarem em ODE.

Durante o cadastro de um agente em CadAgeODE, devem ser informados seu(s) tipo(s) (*TipoAgente*), definindo qual o tipo principal, seus atributos (*AtributoAgente*), atividades (*AtividadeAgente*) e protocolos (*ProtocoloAgente*), bem como em qual ferramenta de software de ODE o agente atuará. Caso um agente seja do tipo coordenador, deve-se informar, também, quais agentes ele coordena. A Figura 4 mostra o modelo de classes correspondente ao cadastro de agentes em ODE.

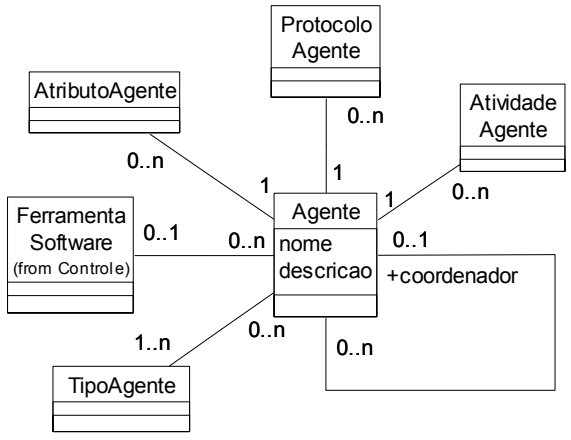

Figura 4 – Modelo de Classes de CadAgeODE

A Figura 5 mostra a janela de CadAgeODE com a aba "Dados Gerais" aberta, quando devem ser informados os dados gerais do agente. Na aba "Tipos de Agente", pode-se selecionar qual(is) o(s) tipo(s) do agente. Caso um

agente seja do tipo coordenador, devem ser informados, também, quais agentes ele coordenará, conforme citado anteriormente. Novos atributos do agente podem ser cadastrados na aba "Atributos". Basta fornecer o nome e o tipo do atributo. As atividades dos agentes podem ser cadastradas na aba "Atividades". Analogamente, os protocolos dos agentes podem ser informados na aba "Protocolos". Neste caso, são informadas apenas as mensagens que serão enviadas de um agente para outro. O tratamento dessas mensagens deve ser implementado pelo desenvolvedor após o código do agente ter sido parcialmente gerado por CadAgeODE. O desenvolvedor deve informar um nome e uma descrição para o protocolo, quem é o agente que receberá a mensagem, a performativa e o conteúdo da mensagem, caso o desenvolvedor já saiba de antemão.

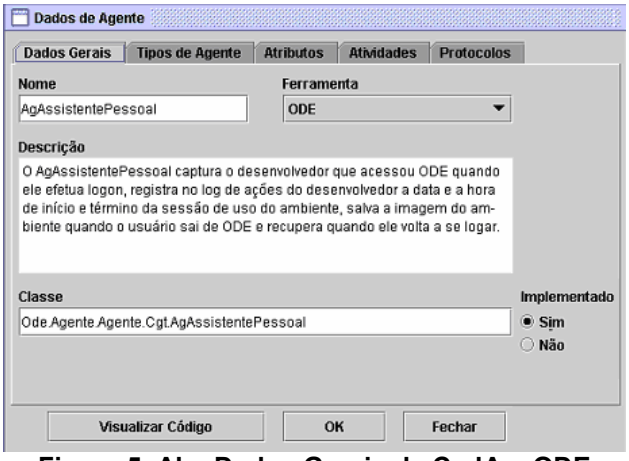

Figura 5. Aba Dados Gerais de CadAgeODE

Para gerar o código do agente, basta clicar no botão "Visualizar Código" (vide botão na Figura 5). O código é, então, gerado e apresentado, como mostra a Figura 6. Vale destacar que esse código contém: (i) as principais importações de bibliotecas; (ii) os atributos que foram cadastrados; (iii) o método construtor do agente; (iv) a assinatura dos métodos para tratar o retorno de mensagens vindas de ODE (`tratarRetorno()`) e para interpretar as mensagens recebidas (`interpretarMensagem()`), que devem ser implementados para cada agente; (v) a assinatura das atividades cadastradas, que devem ser implementadas conforme o propósito das mesmas; (vi) a assinatura dos métodos abstratos das interfaces dos tipos de agentes que o agente implementa, caso o mesmo seja de mais de um tipo; (vii) os protocolos cadastrados, que são implementados de modo a enviar uma mensagem para um agente, sendo que esta mensagem é formada reunindo os dados que foram informados no cadastro de protocolos; e (viii) o método main(), que, enquanto o agente existir, fica em *loop* lendo as mensagens que chegam para o agente.

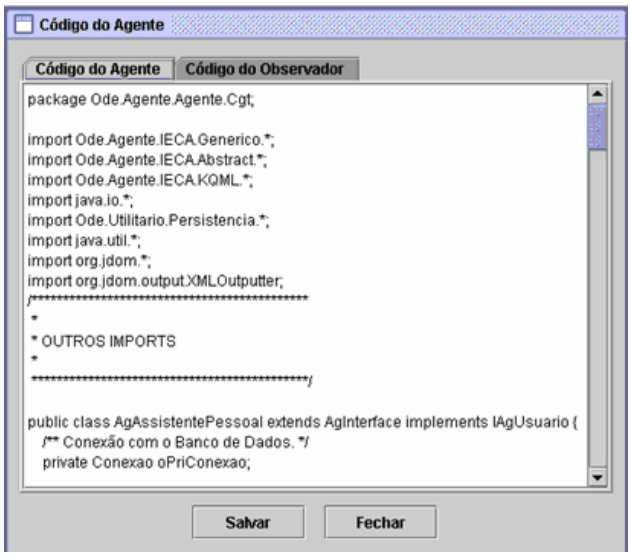

Figura 6. Código gerado por CadAgeODE

4.4. Usando AgeODE

Conforme citado anteriormente, AgeODE foi utilizada para re-implementar os agentes desenvolvidos para ODE, bem como para desenvolver um sistema multiagente para a disseminação de conhecimento em ODE [11].

ODE possui uma infra-estrutura de gerência de conhecimento que suporta captura, recuperação, disseminação, uso e manutenção de conhecimento [10]. Para tratar a disseminação, agentes são usados. Toda ferramenta que necessitar implementar facilidades de disseminação de conhecimento tem de ter um agente ou um sistema multiagentes associado a ela, de acordo com a complexidade da tarefa por ela suportada [11].

No caso de se ter um sistema multiagentes, necessita-se de um agente coordenador como agente principal da ferramenta. Esse coordenador é responsável por coordenar os agentes internos da ferramenta e também por interagir com o agente *AgAssistentePessoal*. Em qualquer um dos casos, *AgAssistentePessoal* tem que conhecer o agente principal da ferramenta. Esta abordagem foi seguida para reconstruir os serviços de disseminação de conhecimento de duas ferramentas de ODE: alocação de recursos humanos e gerência de riscos. A primeira possui um único agente a atuar na ferramenta, uma vez que ela é relativamente simples. A segunda, denominada GeRis [8], possui um sistema multiagente embutido nela, composto de quatro agentes, uma vez que ela apóia atividades mais complexas. Devido a limitações de espaço, neste artigo discutimos apenas o sistema multiagente da ferramenta GeRis. Maiores detalhes podem ser encontrados em [11].

A Figura 7 mostra os agentes de GeRis e como eles se encaixam na estrutura de AgeODE.

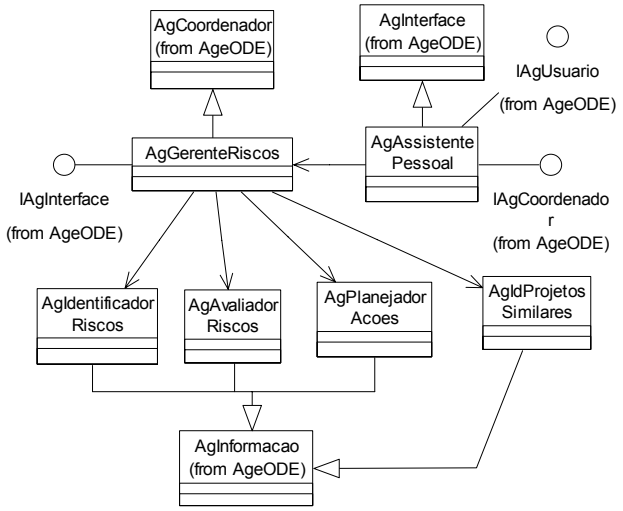

Figura 7. Usando AgeODE

AgGerenteRiscos é o agente responsável por GeRis, sendo, portanto, o coordenador do sistema multiagente dessa ferramenta. Deste modo, *AgGerenteRiscos* herda de *AgCoordenador* e é o único desta ferramenta que é conhecido pelo *AgAssistentePessoal.*

Quando GeRis é iniciada, *AgAssistentePessoal* inicia *AgGerenteRiscos*, que por sua vez, é responsável por iniciar os outros agentes da gerência de riscos que ele coordena. Uma vez que *AgGerenteRiscos* tem que monitorar a interface com o usuário de GeRis, ele implementa, também, a interface *IAgInterface*.

Os demais agentes atuando em GeRis são: *AgIdentificadorRiscos*, que atua durante a identificação de riscos de um projeto, sugerindo riscos a serem considerados com base em projetos similares já concluídos; *AgAvaliadorRiscos*, que atua na avaliação da probabilidade e do impacto de um risco, sugerindo, ainda, que riscos devem ser gerenciados; e *AgPlanejadorAcoes*, que sugere ações de mitigação e contingência a serem tomadas para tratar riscos, tomando por base projetos similares já concluídos. Esses três agentes são agentes de informação.

Como todos esses agentes precisam de informações de projetos similares, *AgGerenteRiscos* se encarrega de obtê-los interagindo com o agente identificador de projetos similares (*AgIdProjetosSimilares*).

Todos esses agentes foram construídos e cadastrados usando CadAgeODE.

5. Trabalhos Correlatos

Existem diversas infra-estruturas de apoio à construção de agentes descritas na literatura, dentre elas JATLite [12] e ZEUS [13]. Entretanto, nenhuma delas se destina a ser uma infra-estrutura específica para construção de agentes em um ADS, considerando, portanto, aspectos de integração nesses ambientes.

Alguns ADSs também foram estudados e, em particular, os Ambientes PROSOFT [18,19] e Odyssey [20] fazem uso da tecnologia de agentes. Em PROSOFT, foram desenvolvidos um modelo de simulação de processos *AgentProcess* (Simulação de Processo de Software baseado em Agentes Cooperativos) [18] e um modelo de um ambiente gerenciador de processos [19]. Além disso, agentes assistentes pessoais são utilizados. Dentre do contexto do Ambiente Odyssey, foi desenvolvida CHARON, uma máquina de processos extensível baseada em agentes inteligentes [20]. Entretanto, em ambos os casos não há uma infra-estrutura definida com o propósito de apoiar a construção e integração de agentes no ADS.

6. Conclusões

A tecnologia de agentes é bastante aplicável na construção de Ambientes de Desenvolvimento de Software (ADSs) e, portanto, é importante ter uma infra-estrutura que apóie a construção e integração de agentes para atuarem nesse contexto. Entretanto, as infra-estruturas existentes são de caráter geral, não levando em conta características de agentes atuando em um ADS. Assim, foi desenvolvida AgeODE, uma infra-estrutura desse tipo, integrada ao ambiente ODE.

Com base nas primeiras experiências de uso de AgeODE, foi possível levantar alguns pontos fortes e fracos da infra-estrutura. Dentre os pontos fortes relacionados ao objetivo deste trabalho, destacam-se os seguintes:

- AgeODE favorece a padronização e o reúso na construção de agentes, facilitando, portanto, a integração. O *framework* de apoio à construção de agentes é fundamental neste contexto;
- CadAgeODE permite a automatização parcial da construção dos agentes, sendo possível gerar parcialmente seu código, o que agiliza e facilita o trabalho do desenvolvedor, além de favorecer a padronização de código.

Dentre os pontos fracos, destacam-se:

- Pela forma como foi concebida AgeODE, valorizando características inerentes a agentes, tal como linha de controle própria, o ambiente ODE quando rodando com agentes passa a apresentar problemas de desempenho. Isso decorre de diversos fatores, tal como a necessidade de haver várias conexões com o banco de dados (uma para cada agente);
- O desenvolvedor de um agente que necessite acessar o método de um objeto de ODE deve saber e codificar corretamente o caminho completo da

classe do objeto, bem como o nome do método a ser acessado. AgeODE apenas gera a interface dos métodos das classes de agentes e não apóia a sua implementação.

Como perspectiva futura, espera-se tratar os pontos fracos detectados, além de explorar novos tipos de agentes e ampliar a infra-estrutura proposta, permitindo tratar agentes inteligentes e agentes móveis. Para tal, facilidades de inferência têm de ser incorporadas a AgeODE.

7. Agradecimentos

Este trabalho foi realizado com o apoio do CNPq, uma entidade do Governo Federal Brasileiro dedicada ao desenvolvimento científico e tecnológico, e da FAPES, Fundação de Apoio à Ciência e Tecnologia do Estado do Espírito Santo.

8. Referências

[1] I. Thomas, B.A. Nejmeh, "Definitions of Tool Integration for Environments", IEEE Software, 29-35, March 1992.

[2] R.A. Falbo, A.C.C. Natali, P.G. Mian, G. Bertollo, F.B. Ruy, "ODE: Ontology-based software Development Environment", IX Argentine Congress on Computer Science, p. 1124-1135, La Plata, Argentina, 2003.

[3] J. Pezzin, R.A. Falbo, "AgeODE: Uma Infra-estrutura para Apoiar a Construção de Agentes para Atuarem no Ambiente de Desenvolvimento de Software ODE", IV Jornadas Iberoamericanas de Ingeniería de Software y Ingeniería del Conocimiento – JIISIC'04, Madri, Espanha, 2004.

[4] M. Wooldridge, "Intelligent Agents". In: *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*, London, The MIT Press, p. 3-51, 1999.

[5] W. Harrison, H. Ossher, P. Tarr, "Software Engineering Tools and Environments: A Roadmap", in Proc. of the Future of Software Engineering, ICSE'2000, pp. 263-277, Ireland, 2000.

[6] R.S. Pressam, *Software Engineering: A Practiotioner's Approach*, Fifth Edition, McGraw Hill, 2001.

[7] F. Ruy, G. Bertollo, R.A. Falbo, "Knowledge-based Support to Process Integration in ODE". Clei Electronic Journal, Volume 7, Number 1, June 2004.

[8] R.A. Falbo, F.B. Ruy, G. Bertollo, D.F. Togneri, "Learning How to Manage Risks Using Organizational Knowledge", Advances in Learning Software Organizations (Proceedings of the 6th International Workshop on Learning Software Organizations - LSO'2004), Melnik G. and Holz, H. (Eds.): LNCS 3096, pp. 7-18, 2004.

[9] V.B. Nunes, A.O. Soares, R.A. Falbo, "Apoio à Documentação em um Ambiente de Desenvolvimento de Software, VII Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software, IDEAS'2004, pp 50-55, Arequipa, Peru, Maio 2004.

[10] R.A. Falbo, D.O. Arantes, A.C.C. Natali, "Integrating Knowledge Management and Groupware in a Software Development Environment", 5th International Conference on Practical Aspects of Knowledge Management, Vienna, Austria, December 2004.

[11] R. A. Falbo, J. Pezzin, M. M. Schwambach, "A Multi-Agent System for Knowledge Delivery in a Software Engineering Environment", SEKE'2005, Taipei, Taiwan, 2005.

[12] H. Jeon, C. Petrie, M.R. Cutkosky, "JATLite: A Java Agent Infrastructure with Message Routing". IEEE Internet Computing, March /April 2000.

[13] J. Collis, D. Ndumu, "The ZEUS Agent Building Toolkit – ZEUS Technical Manual", September, 1999.

[14] Garcia, A., Silva, V., Chavez, C., Lucena, C., "Engineering Multi-Agent Systems with Aspects and Patterns". Journal of the Brazilian Computer Society, number 1, Vol. 8, July 2002.

[15] Brugali, D., Sycara, K.: Towards Agent Oriented Application Frameworks. ACM Computing Surveys, Volume 32, Issue 1, March 2000.

[16] Juchem, M., Bastos, R. M., "Projetando Sistemas Multiagentes em Organizações Empresariais". XVI Simpósio Brasileiro de Engenharia de Software, Gramado, Brasil, 2002.

[17] T. Finin, Y. Labrou, J. Mayfield, "KQML as an agent communication language", Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM94), ACM Press, December 1994.

[18] F. A. D. Silva, R. Q. Reis, C. A. L. Reis, D. J. Nunes, "Um Modelo de Simulação de Processos de Software Baseado em Agentes Cooperativos", XIII Simpósio Brasileiro de Engenharia de Software, 1999.

[19] C. A. G. de Lima, D. J. Nunes, R. Q. Reis, "Gerenciamento de Processo de Desenvolvimento Cooperativo de Software no Ambiente PROSOFT", XII Simpósio Brasileiro de Engenharia de Software, 1998.

[20] L. G. P. Murta, "CHARON: Uma Máquina de Processos Extensível Baseada em Agentes Inteligentes", Tese de Mestrado COPPE – UFRJ – Rio de Janeiro, RJ – Brasil - Março de 2002.