Evolução de um Sistema Multi-Agentes: Implementando Mobilidade em Agentes de Informação

Cláudio Nogueira Sant'Anna Alessandro Fabricio Garcia Carlos José Pereira de Lucena

Laboratório de Engenharia de Software - LES
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio
Rua Marquês de São Vicente, 225 – Ed. Pd. Leonel Franca, 10º Andar
Rio de Janeiro – Brasil

{claudios,afgarcia,lucena}@inf.puc-rio.br

PUC-RioInf.MCC55/03 Novembro, 2003

Abstract: Development of software agents involves multiple concerns (properties), such as autonomy, adaptation, interaction, learning, and mobility. As software engineering for multi-agent systems (MASs) evolves, there is a need for better understanding of the relationships between its concerns and abstractions from object-oriented (OO) software engineering. Hence it is necessary to investigate systematically whether OO techniques and abstractions support explicit modularization of these concerns and consequently the development of software agents that are easier to understand, evolve and reuse. This paper presents a study about the evolution of the design and code of a particular multi-agent system. We used two distinct OO techniques to introduce the mobility property in information agents of this MAS: design patterns and aspects. In this study, we assessed the two techniques in order to verify which approach demands less effort during the realization of this scenario. The assessment was based on a suite of metrics.

Keywords: Separation de *concerns*, multi-agent systems, empirical software engineering, metrics, aspects, design patterns, mobile agents.

Resumo: Desenvolvimento de agentes de software envolve *concerns* (propriedades) múltiplos como autonomia, adaptação, interação, aprendizagem e mobilidade. À medida que a engenharia de software para sistemas multi-agentes (SMAs) evolui, aumenta a necessidade de compreensão das relações entre tais *concerns* e abstrações da engenharia de software orientada a objetos (OO). Deve-se procurar avaliar se técnicas e abstrações OO permitem a modularização destes *concerns* e consequentemente a produção de agentes de software fáceis de entender, evoluir e reutilizar. Neste sentido, esta monografia apresenta um estudo sobre a evolução do projeto e implementação de um SMA particular. A propriedade de mobilidade é introduzida em agentes de informação deste SMA, utilizando duas técnicas OO distintas: padrões de projeto e aspectos. Neste estudo, avaliamos qual das 2 técnicas permitiu a realização deste cenário de evolução com menor esforço. A avaliação foi conduzida segundo um conjunto de métricas.

Palavras Chaves: Separação de *concerns*, sistemas multi-agentes, engenharia de software experimental, métricas, aspectos, padrões de projeto, agentes móveis.

1 Introdução

À medida que a engenharia de software para sistemas multi-agentes evolui, aumenta a necessidade de se entender a relação entre seus *concerns* (propriedades) e abstrações da engenharia de software orientada a objetos (OO). Sistemas multi-agentes (SMAs) de larga escala envolvem *concerns* complexos e não ortogonais, tais como autonomia, adaptação, interação, colaboração, mobilidade, papéis e outros. A análise e o projeto dos *concerns* de SMAs são diretamente apoiados por um número crescente de abstrações associadas a linguagens e metodologias orientadas a agentes [12]. Contudo, as fases de projeto detalhado e implementação ainda dependem, na maioria dos casos, de técnicas de projeto e linguagens de programação orientadas a objetos, tais como Java (figura 1). Essa transição é complexa não apenas devido aos diferentes conjuntos de abstrações dos artefatos gerados, mas também porque muitos *concerns* recorrentes em aplicações de SMAs são essencialmente diferentes dos *concerns* de sistemas OO. *Concerns* de SMAs podem não ser explicitamente separados pelas abstrações OO existentes. Portanto, deve-se procurar avaliar se as abstrações OO permitem a produção de componentes de SMAs com baixo acoplamento, alta coesão, fáceis de entender, evoluir e reusar.

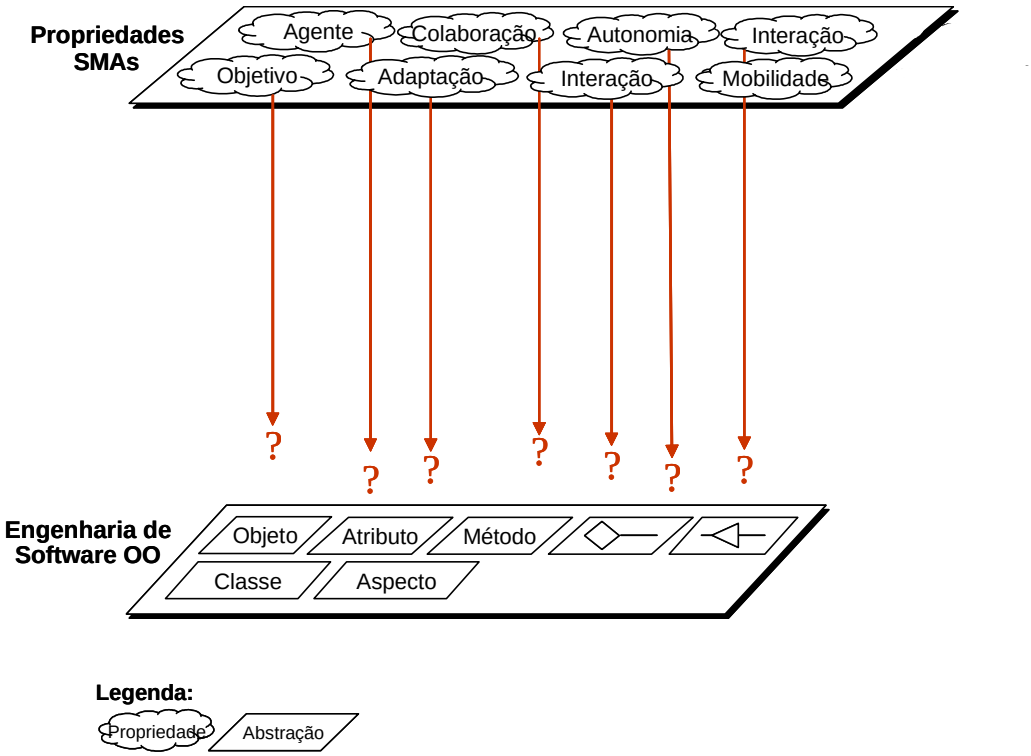

Figura 1. Falta de alinhamento na Engenharia de Software para SMAs.

Estudos experimentais [1] são a forma mais eficaz para a obtenção de evidências empíricas que podem melhorar o conhecimento sobre fenômenos da engenharia de software. No entanto, até agora, poucos estudos sistemáticos foram realizados para avaliar se os *concerns* de SMAs podem ser naturalmente apoiados por abstrações já bem conhecidas. Além disso, poucos estudos empíricos encontrados na literatura investigaram a relação entre as abstrações orientadas a agentes e as abstrações OO. Nesse contexto, esta monografia

apresenta a continuação do estudo empírico apresentado em [7], que usa duas técnicas orientadas a objetos para o desenvolvimento de um SMA. O objetivo geral desse estudo é avaliar a facilidade de evolução e reuso de um SMA desenvolvido com cada uma das técnicas investigadas. Em [7], duas versões de um mesmo SMA foram projetadas e implementadas. Uma versão usou a técnica baseada em padrões e a outra usou a técnica baseada em aspectos. Em seguida, cenários envolvendo alterações relativas aos *concerns* de agência foram simulados em ambas versões, e a facilidade de evolução e reuso foi avaliada a partir do uso de um conjunto de métricas.

Essa monografia continua o trabalho apresentado em [7] descrevendo mais um cenário de evolução aplicado nas duas versões do sistema. Este novo cenário realiza a inclusão da propriedade de mobilidade em um dos tipos de agentes do SMA. A avaliação é conduzida segundo um framework desenvolvimento previamente [7, 17]. Este framework possui um conjunto de métricas que permite capturar o esforço de evolução e reutilização das modificações realizadas.

O restante do texto está organizado da seguinte maneira. A Seção 2 conceitua a propriedade de mobilidade em SMAs. A Seção 3 descreve o estudo empírico. Alguns detalhes do projeto são apresentados na Seção 4. A Seção 5 apresenta os resultados da comparação entre a abordagem orientada a aspectos e a abordagem orientada a padrões. A Seção 6 acrescenta as conclusões.

2 Mobilidade

Esta seção descreve sucintamente a propriedade de mobilidade e sua importância em sistemas multi-agentes. Além disso, justifica sua inclusão nos agentes do sistema desenvolvido no nosso estudo empírico.

Um agente móvel é um programa que, durante sua execução, pode migrar de um computador para outro em uma rede heterogênea [11]. Em cada computador, o agente interage com outros agentes e recursos para realizar sua tarefa. Lange [14] afirma que o uso de agentes móveis para o desenvolvimento de sistemas distribuídos traz uma série de benefícios, tais como redução do tráfego na rede, redução da latência da rede, encapsulamento de protocolos, e a possibilidade de execução assíncrona e autônoma.

Atualmente, muitas plataformas e frameworks para o desenvolvimento de sistemas multi-agentes dão suporte a mobilidade. Alguns deles são específicos apenas para o desenvolvimento de agentes móvies, tais com Aglets [14], D'Agents [3], Odyssey [16]. Outros, como Jade [13], são plataformas mais abrangentes que dão suporte também a outras propriedades, por exemplo, interação. Além disso, com o objetivo de promover a interoperabilidade entre agentes de diferentes plataformas e sistemas, alguns esforços para padronizar aspectos importantes da tecnologia de agentes móveis foram realizados. MASIF [15], por exemplo, é um padrão adotado pela OMG que uniformiza as seguintes áreas: gerenciamento de agentes, transferência de agentes, serviço de nomes de agentes e serviços de localização de agentes.

De acordo com [14], comércio eletrônico, assistência pessoal, trabalho em grupo apoiado por computador e disseminação de informação são domínios de aplicação que podem se beneficiar com o paradigma de agentes móveis. Segundo [11], o processamento distribuído

de informações é uma das aplicações mais comuns de agentes móveis. O agente móvel pode deixar seu sistema de origem, se mover para o local da rede onde existe a informação desejada e realizar localmente a procura pela informação. Pode-se ver, então, que mobilidade é uma propriedade recorrente em SMAs, o que justifica sua implementação como um cenário de evolução de um sistema multi-agentes.

3 O Estudo Empírico

Como já foi dito antes, esta monografia apresenta a continuação de um estudo empírico descrito em [7]. O estudo foi dividido em duas fases: a fase de construção e a fase de evolução e reuso. Na fase de construção, os participantes desenvolveram duas versões de um SMA usando diferentes técnicas de orientação a objetos. Uma versão orientada a aspectos (AO) e outra versão orientada a padrões (OP). Como essas duas técnicas não levam em consideração os *concerns* de SMAs, dois métodos [9, 10], especialmente adaptados ao desenvolvimento de SMAs, foram associados e usados na aplicação de cada técnica. Todas as pessoas participaram do desenvolvimento das duas versões. O conjunto de modelos orientados a agentes, projeto OO e implementação foram baseados nas mesmas especificações de requisitos.

A fase de evolução e reuso envolveu os mesmos participantes. O objetivo dessa fase foi comparar a facilidade de evolução e reusabilidade dos componentes da solução orientada a padrões e da solução orientada a aspectos. Para isso, um conjunto de cenários relevantes de evolução e reuso foram executados no projeto e código de ambas soluções. Esses cenários estão descritos em [7]. A monografia descreve a execução de mais um cenário que está detalhado na Seção 3.2.

3.1 O Sistema Multi-Agentes

O projeto do sistema multi-agentes (denominado Portalware [9]) foi realizado pelo grupo TecComm/SoC+Agents da PUC-Rio. Portalware é um ambiente que apóia o desenvolvimento e gerenciamento de portais de conteúdo da Internet. Para se manter útil, o Portalware deve oferecer facilidade de extensão e modificação de suas funcionalidades, portanto seu design e implementação devem ser flexíveis para acomodar novas mudanças.

Os *concerns* manipulados nesse projeto são *concerns* encontrados em SMAs reativos do mundo real, típicos de aplicações existentes. Este SMA compreende vários *concerns* de agência, incluindo tipos de agentes, papéis, colaboração, interação, adaptação, autonomia e outros. O ambiente inclui alguns *tipos de agentes* para controlar portais e para cordenar e automatizar atividades repetitivas e que consomem muito tempo dos grupos de desenvolvimento de portais. Portalware tem quatro tipos de agentes: (i) *agentes mediadores,* (ii) *agentes de interface*, (iii) *agentes de usuário* e (iv) *agentes de informação*. Os tipos de agentes implementam os *concerns* da parte fundamental do agente e outros *concerns*. Nesse sistema, a parte fundamental é formada por três propriedades de agência: interação, autonomia e adaptação. Cada instância de agente tem diferentes propriedades e desempenham papéis distintos. A figura 2 mostra os tipos de agentes existentes no Portalware e as propriedades e papéis associados a eles.

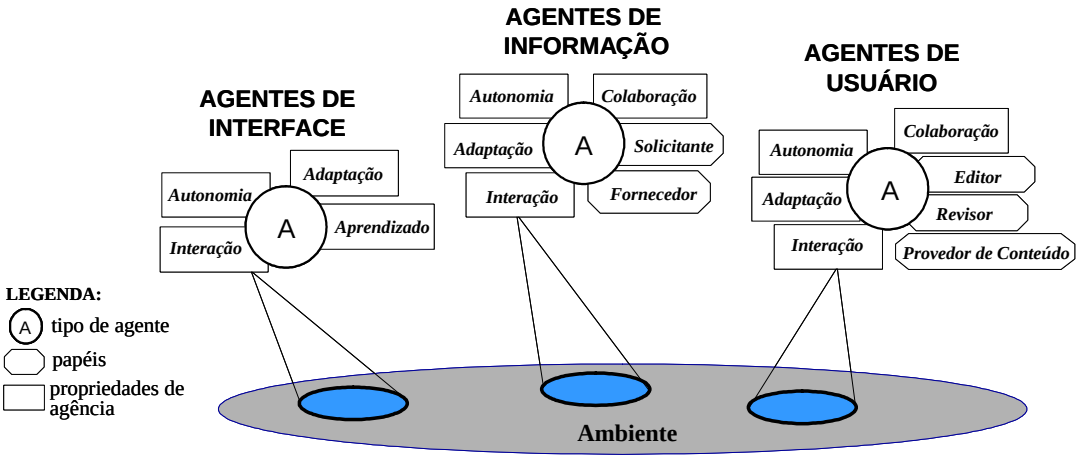

Figura 2. Agentes do Portalware

Agentes mediadores fazem a mediação das conversações entre todos os agentes do sistema, provendo alguns serviços, tais como serviços de nome. Os agentes de interface monitoram a interface gráfica para melhorar a interação com os usuários. Eles aprendem a partir das preferências ou de instruções explícitas do usuário. Agentes de usuário representam os usuários do Portalware e existem para reduzir a necessidade de conversa direta entre eles. Uma vez que, *editores*, *revisores* e *provedores de conteúdo* precisam se comunicar entre si para manter o portal, os agentes de usuários incorporam esses papéis e capacidades para automatizar e apoiar a colaboração em diferentes contextos. O agente, ao desempenhar o papel de editor, tem a responsabilidade de contactar os revisores e provedores de conteúdo e negociar com eles para usar seus serviços. Os usuários do Portalware freqüentemente necessitam pesquisar informações que estão armazenadas em bancos de dados. Cada agente de informação controla um banco de dados e contém planos para a pesquisa de informação. Um plano alternativo de colaboração é usado quando um agente de informação não é capaz de encontrar a informação desejada em seu banco de dados. Durante a execução desse plano, o agente de informação desempenha o papel de *solicitante* para chamar os outros agentes de informação e requisitar a informação desejada. De forma similar, os agentes de informação chamados desempenham o papel de *fornecedor* para poderem receber a requisição e enviar o resultado da pesquisa.

3.2 Incluindo Mobilidade nos Agentes de Informação

O novo cenário de evolução se caracteriza pela inclusão da propriedade de mobilidade nos agentes de informação. Como foi dito anteriormente, cada agente de informação controla um banco de dados. Ao ser requisitado, o agente de informação pesquisa pela informação desejada no seu banco de dados. Caso não a encontre, ele verifica quais são os outros agentes de informação que existem no mesmo ambiente e inicia uma colaboração com um deles, solicitando a informação. O segundo agente de informação procura a informação no seu banco de dados e responde ao agente solicitante. Se a informação também não estiver disponível no banco de dados do segundo agente, ele informa isso ao agente solicitante e a colaboração é finalizada. Atualmente o agente de informação não faz nada em seguida para buscar a informação em outras fontes.

Resolveu-se então evoluir o agente de informação no sentido de torná-lo um agente móvel que possa deslocar-se, através da rede, em busca da informação em outros ambientes, mais especificamente, em outras instâncias do Portalware sendo executadas em computadores espalhados pela rede. A implementação desse cenário permite que o agente de informação realize os seguintes passos quando não obtém sucesso ao colaborar com outro agente do seu próprio ambiente:

1) O agente avisa a todos os outros agentes de seu ambiente que irá se mover. Com isso, os outros agentes o excluem da lista de agentes conhecidos;
2) O agente apaga seu registro no ambiente atual;
3) O agente move-se para outro ambiente;
4) O agente registra-se no novo ambiente. A partir disso, ele passa a conhecer os agentes que habitam o novo ambiente;
5) O agente avisa a todos os agentes que ele acabou de chegar no ambiente. Com isso, cada agente o inclui na lista de agentes conhecidos;
6) O agente verifica os agentes de informação existentes e colabora com um deles, solicitando a informação desejada;
7) De posse do resultado da colaboração, o agente retorna ao seu ambiente original.

A implementação desse cenário foi realizada tanto na versão orientada a padrões quanto na versão orientada a aspectos e está descrita na próxima seção.

4 Projeto do Agente de Informação Móvel

Esta seção apresenta alguns detalhes do projeto e implementação do agente de informação, tanto da solução orientada a padrões quanto da solução orientada a aspectos, dando ênfase às modificações necessárias para a inclusão da propriedade de mobilidade.

4.1 O Framework JADE

No intuito de facilitar a implementação de mobilidade nos agentes de informação, resolveu-se fazer a integração do Portalware com um framework para desenvolvimento de sistemas multi-agentes que provê suporte a essa propriedade. O framework escolhido foi o JADE [13]. JADE é um framework pare o desenvolvimento de sistemas multi-agentes que segue os padrões estabelecidos pela FIPA [5]. Sua escolha deveu-se ao fato dele ser um framework já bastante utilizado e dar suporte a outras características de agentes, além de mobilidade. Isso possibilita que futuramente outras evoluções sejam realizadas no Portalware, por exemplo, uso da linguagem de comunicação FIPA ACL [4], com o uso do mesmo framework.

JADE é totalmente implementado em Java e fornece uma biblioteca de classes para a implementação de agentes. Além disso, JADE provê uma plataforma distribuída sobre a qual os agentes serão executados. Por ser uma plataforma distribuída, pode-se ter agentes e SMAs em vários computadores de uma rede, e agentes migrando de um computador para o outro. JADE fornece, também, uma interface gráfica por onde os agentes podem ser gerenciados. A figura 3 mostra essa interface, onde aparecem quatro agentes. Os agentes *Agent_1* e *Agent_2* estão sendo executados em uma instância do Portalware e os agentes *Agent_3* e *Agent_4* em outra instância distinta. A movimentação dos agentes pode ser monitorada por essa interface. As seções 4.2 e 4.3 mostram como JADE foi integrado ao Portalware nas duas soluções desenvolvidas.

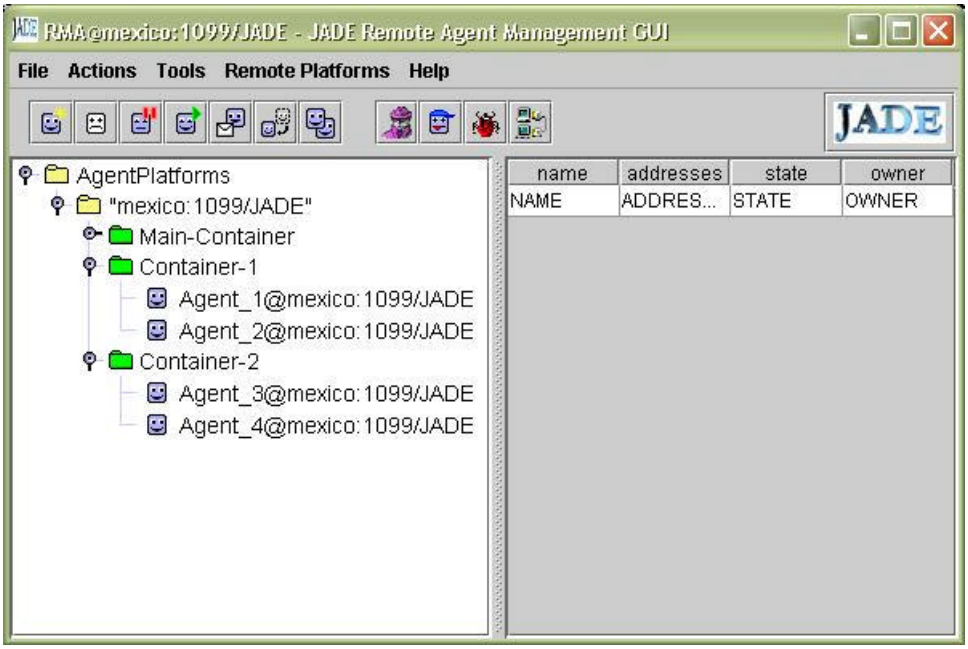

Figura 3. A interface gráfica do framework JADE.

4.2 Projeto Orientado a Padrões

O framework JADE disponibiliza uma classe chamada *Agent* que deve ser estendida para que o agente a ser desenvolvido herde as funcionalidades de agência já implementadas por ele. No caso do Portalware, optou-se por não fazer com que a classe do agente básico (*PAgent*) especializasse a classe *Agent* de JADE, pois dessa forma a propriedade de mobilidade, provida pelo framework, ficaria misturada dentro da classe *PAgent*. Resolveu-se, então, criar a classe *JadeAgent*, que especializa o agente de JADE. A classe *Mobility* implementa a propriedade de mobilidade no sentido que faz a integração da classe *JadeAgent* com o agente de informação do Portalware.

A figura 4 mostra o diagrama de classes que descreve a estruturação das propriedades do agente de informação. Pode-se observar que a propriedade de mobilidade foi estruturada de forma semelhante às outras propriedades (interação, autonomia, adaptação) especializando a classe *Property*, que caracteriza o padrão *Mediator* [6].

A figura 5 descreve como ficou, após a inclusão de mobilidade, a seqüência de ações que o agente executa para buscar uma informação. Primeiro ele procura a informação no seu banco de dados (método *search()*). Se não encontrar a informação, ativa o papel de solicitante, implementado pela classe *Caller*, e passa a colaborar com outro agente (métodos *sendAndLockInformationAsk()* e *receiveAndLockInformationAsk()*). Se mesmo assim, não obtiver sucesso, chama o método *move()* da classe *Mobility* e se move para outra instância remota do Portalware.

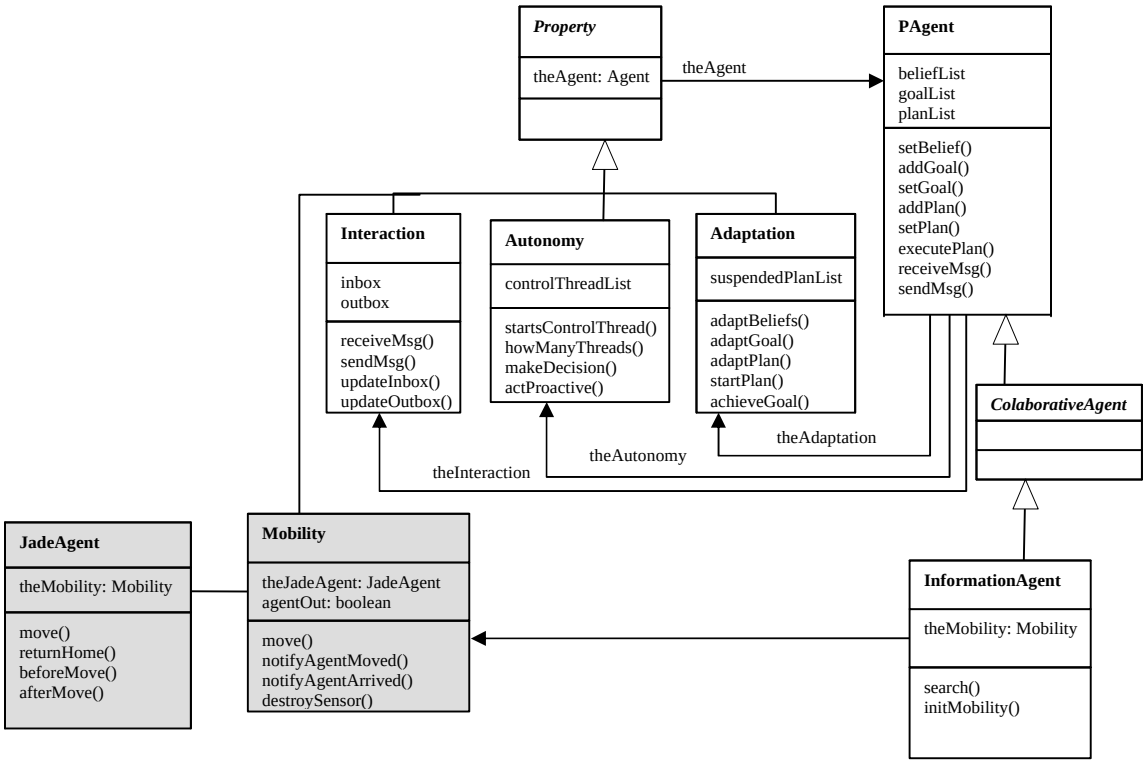

Figure 4. Propriedades do agente de informação da solução OP (Diagrama de Classes)

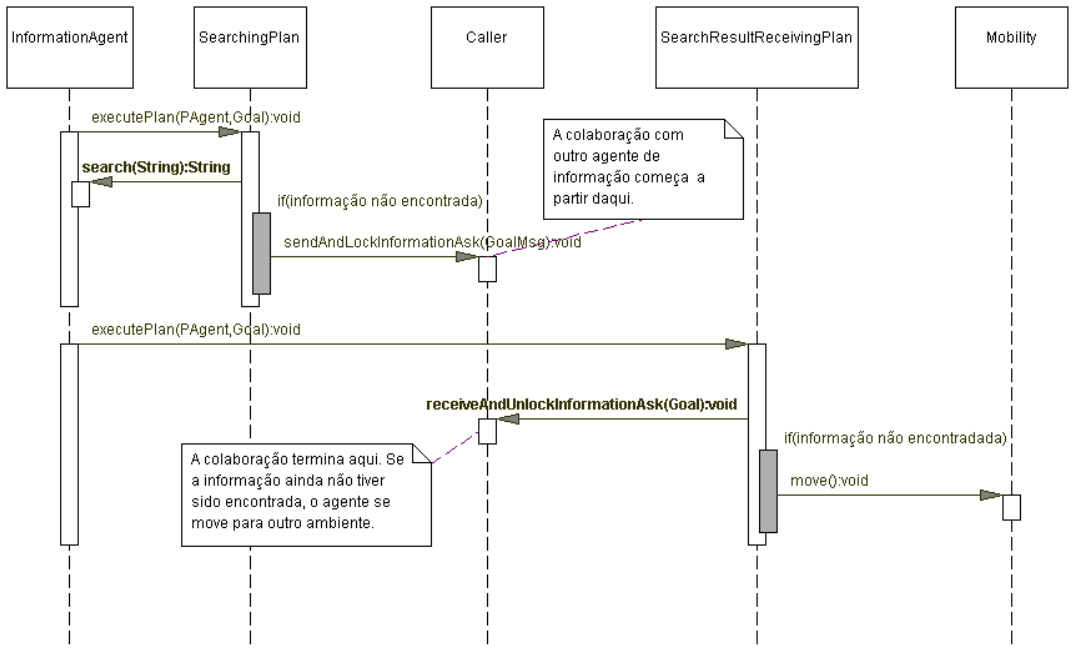

Figure 5. Seqüência de ações do agente em busca de informação.

A figura 6 mostra a seqüência de operações que são executadas para que o agente de informação possa se mover. O método *move()* da classe *JadeAgent* é responsável por manipular todos os mecanismos fornecidos por JADE para que haja a mobilidade física do agente. Além disso, a classe *JadeAgent* reescreve os métodos *beforeMove()* e *afteMove()* da classe *Agent* de JADE. Esses métodos são executados imediatamente antes e depois do agente se mover, respectivamente. Antes de sair do seu ambiente, o agente precisa avisar aos outros agentes que está saindo (método *notifyAgentMoved()*), para que eles possam excluí-lo da suas listas de agentes conhecidos. Além disso, ele destrói o seu sensor ligado ao ambiente (método *destroySensor()*). Após mover-se, o agente de informação: (i) limpa sua lista de agentes conhecidos (método *clearBeliefAgents()*), (ii) cria um novo sensor ligado, agora, ao ambiente para o qual ele acaba de se mover (método *createSensor()*), o que faz com que ele passe a conhecer os agentes desse ambiente, (iii) notifica aos agentes que ele acabou de chegar (método *notifyAgentArrived()*), (iv) colabora com outro agente em busca da informação desejada (método *colaborate()*), e (v) retorna ao seu ambiente de origem (método *returnHome()*).

Estes são os pontos principais do projeto. Muitos outros detalhes tiveram que ser implementados para viabilizar a mobilidade do agente de informação, mas estão fora do escopo deste trabalho.

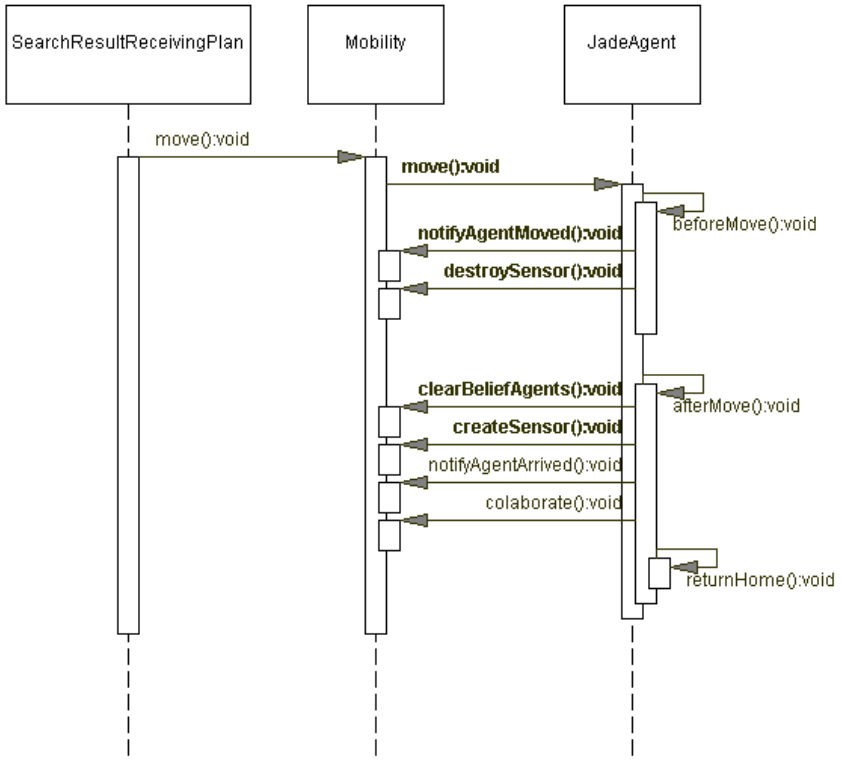

Figure 6. Seqüência de operações executadas enquanto a agente se move.

4.3 Projeto Orientado a Aspectos

A solução orientada a aspectos é bem parecida com a solução orientada a padrões. Da mesma maneira, utilizou-se a classe *JadeAgent* para estender a classe *Agent* do framework JADE e encapsular todos os mecanismos responsáveis pela mobilidade física do agente. A diferença dá-se pelo fato de que a integração entre a classe *JadeAgent* e o agente de informação é implementada por um aspecto, chamado de *Mobility*. Ou seja, o aspecto *Mobility* representa a propriedade de mobilidade do agente e, assim como no projeto OP, é implementada da mesma forma que as outras propriedades de agência. A figura 7 mostra o diagrama que descreve a estrutura das propriedades do agente de informação da solução OA. Nesse diagrama as caixas com um losango superior representam os aspectos, e os relacionamentos com o estereótipo <<crosscuts>> indicam que o aspecto intercepta a classe ao qual ele está ligado. Pode-se observar que aqui, diferentemente da solução OP, o agente de informação na tem uma referência para a propriedade de mobilidade.

A seqüência de operações para que o agente se mova é semelhante à mostrada na Seção 4.2. A principal diferença é que, na solução OP, o método *move()* é explicitamente chamado por um plano do agente de informação. Enquanto que, na solução OA, o aspecto *Mobilty* transfere o controle da execução para si e faz com que o agente se mova quando for necessário, sem o conhecimento das classes que implementam o agente de informação.

4.4 Dificuldades

A maioria das dificuldades encontradas durante o projeto das duas soluções esteve ligada ao uso do framework JADE. A principal delas ocorreu porque apenas os mecanismos de JADE relativos à implementação de mobilidade foram utilizados. Preferiu-se continuar usando os mecanismos de troca de mensagem e serviços de nomes já implementados no Portalware. Com isso, todo o tratamento automático que JADE dá em relação a esses mecanismos, quando os agentes se movem, não foi aproveitado. Por isso, foi preciso implementar todo esse tratamento no Portalware para viabilizar a utilização de agentes móveis.

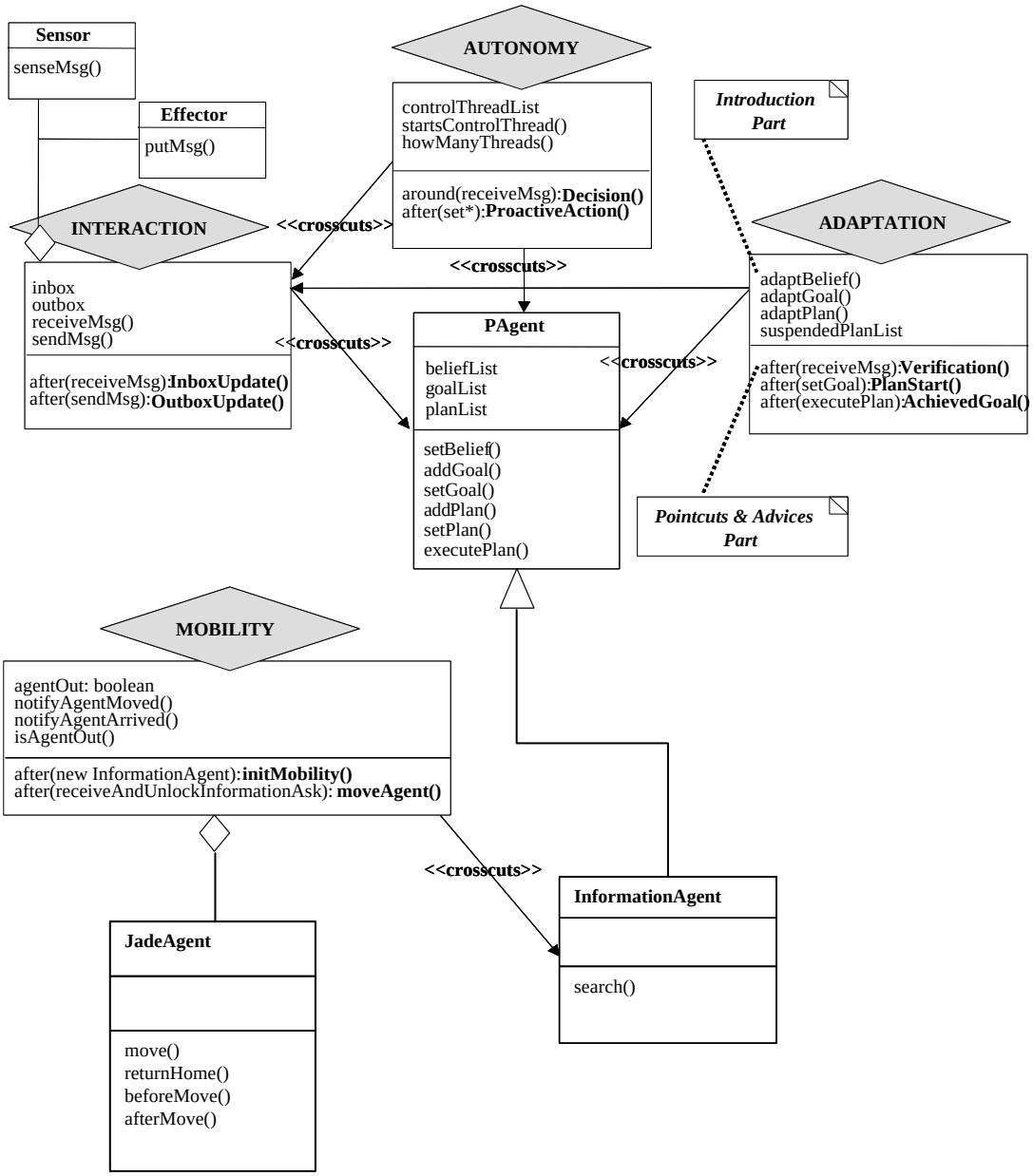

Figure 7. Propriedades do agente de informação da solução OA.

5 Resultados

Esta seção mostra os resultados da aplicação de algumas métricas para tentar mensurar o esforço despendido ao realizar as alterações nas duas versões do sistema, e poder assim fornecer indicações de qual solução propiciou maior facilidade de evolução no sentido de incluir a propriedade de mobilidade nos agente de informação do Portalware.

As métricas usadas aqui foram as mesmas utilizadas para avaliar os cenários do estudo realizado anteriormente [7]. São elas: (1) número de componentes (classes/aspectos) adicionados, (2) número de componentes alterados, (3) número de relacionamentos adicionados, (4) número de relacionamentos alterados, (5) número de operações (métodos/advices) adicionadas, (6) número de operações alteradas, (7) número de linhas de código (LOC) adicionadas e (8) número de LOC alteradas. A tabela 1 resume os resultados obtidos a partir da evolução das duas versões do SMA. Percebe-se que os resultados foram semelhantes em ambas soluções. As maiores diferenças, a favor da solução OA, ocorreram no número de linhas de código e operações adicionadas. Isso se deve principalmente ao fato de que o agente de informação na solução OP tem uma referência para a propriedade de mobilidade e precisa iniciá-la e acioná-la explicitamente por meio de chamada de métodos.

Métrica	Solução OP	Solução OA
Componentes incluídos	3	3
Componentes alterados	11	11
Operações adicionadas	30	25
Operações alteradas	9	8
Relacionamentos adicionados	12	12
Relacionamentos alterados	0	0
LOC adicionadas	191	181
LOC alteradas	20	22

Tabela 1. Resultados do cenário de evolução.

6 Conclusões

A separação dos *concerns* de agência é essencial para que os engenheiros de SMAs possam decidir melhor como estender e modificar os *concerns* a medida que o sistema evolui [8]. O desenvolvimento de SMAs complexos enfrenta a transição de modelos orientados a agentes, construídos de acordo com metodologias orientadas a agentes e suas abstrações, para projetos e implementações orientadas a objetos. Dentre os problemas inerentes a essa transição, nenhum é mais sério que a dificuldade de tratar as diferenças conceituais entre agentes e objetos. Por isso, existe a necessidade de se entender melhor os relacionamentos entre o paradigma orientado a objetos e o paradigma orientado a agentes.

Em um trabalho anterior [7], foi realizado um estudo empírico para comparar a manutenibilidade e reusabilidade de duas versões de uma mesmo SMA, desenvolvidas com técnicas de orientação a objetos diferentes. Naquele estudo, vários cenários de evolução foram realizados e avaliados. Esta monografia trata da continuação desse trabalho com a realização de mais um cenário de evolução. Neste cenário, introduziu-se a propriedade de mobilidade em um tipo de agente do sistema. Para isso utilizou-se JADE, um framework para desenvolvimento de sistemas multi-agentes. A monografia apresenta alguns detalhes do projeto, tanto da solução orientada a padrões como da solução orientada a aspectos. Os

resultados da aplicação de algumas métricas mostraram que o projeto e implementação das duas soluções demandaram um nível de esforço similar, com uma diferença sutil em favor da solução orientada a aspectos.

Referências Bibliográficas

[1] Basili, V., Selby R., Hutchins D. “Experimentation in Software Engineering”. IEEE Transactions on Software Engineering, SE-12, 1986 p. 733-743.

[2] Bellifemine, F., Caire, G., Trucco, G., Rimassa, G. “JADE Programmer’s Guide. URL: http://sharon.cselt.it/projects/jade/

[3] Dartmouth College. “D’Agents”. URL: http://agent.cs.dartmouth.edu/.

[4] FIPA ACL Message Structure Specification. URL: http://www.fipa.org/.

[5] FIPA website. URL: http://www.fipa.org/

[6] Gamma, E. et al. “Design Patterns: Elements of Reusable Object-Oriented Software”. Addison-Wesley, Reading, 1995.

[7] Garcia, A. et al. “Agents and Objects: An Empirical Study on the Design and Implementation of Multi-Agent Systems”. Proc. of the SELMAS’03 Workshop at ICSE’03, Portland, USA, May 2003, pp. 11-22.

[8] Garcia, A., Lucena, C. Software Engineering for Large-Scale Multi-Agent Systems – SELMAS 2002. (Post-Workshop Report) ACM Software Engineering Notes, August 2002.

[9] Garcia, A., Lucena, C., Cowan, D. “Agents in Object-Oriented Software Engineering”. Software: Practice and Experience, Elsevier, 2003. (Accepted to Appear)

[10] Garcia, A., Silva, V., Chavez, C., Lucena, C. “Engineering Multi-Agent Systems with Aspects and Patterns”. Journal of the Brazilian Computer Society, November, 2002.

[11] Gray, R. et al. “Mobile Agents: Motivations and state-of-art systems”, Relatório Técnico disponível na URL ftp://ftp.cs.dartmouth.edu/TR/TR2000-365.ps.Z

[12] Iglesias, C. et al. “A Survey of Agent-Oriented Methodologies”, Proceedings of the ATAL-98, Paris, France, July 1998, pp. 317-330.

[13] JADE website. URL: http://sharon.cselt.it/projects/jade/

[14] Lange, D., Oshima, M. “Programming and Deploying Java Mobile Agents with Aglets”. Addison Wesley, 1998.

[15] Milojicic, D. et al. “MASIF: The OMG Mobile Agent System Interoperability Facility”. In Dejan Milojicic, Frederik Douglis, and Richard Wheeler, editors, Readings in Agents, chapter 14, pages 628-642. ACM Press, 1999.

[16] Projeto Odyssey. URL: www.generalmagic.com

[17] Sant’Anna, C., Garcia, A., Chavez, A., Lucena, C., Staa, A. “On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework”. Proceedings of the XVII Brazilian Symposium on Software Engineering, Manaus, Brazil, October 2003, pp. 19-34.

[18] White J. “Mobile Agents”. In Software Agents, J. Bradshaw, Ed. MIT Press, 1997, pp. 437-472.