Verifying the behavior of agents in BDI4JADE with AspectJ

Francisco J. P. Cunha[1], Marx Leles Viana[1], Márcio R. Rosemberg[1], Carlos J. P. de Lucena[1]

[1]Departamento de Informática – Pontifícia Universidade Católica (PUC-RJ)
Rua Marquês de São Vicente, 255 – 22.451-900 – Rio de Janeiro – RJ – Brasil

{fcunha,mleles,mrosemberg,lucena}@inf.puc-rio.br

Abstract. *The use of multi-agent systems (MAS) for software construction is a promising approach and has been applied in different research areas. So the validation of these systems is crucial to build robust software. However, the methods proposed focused their efforts mainly in how to analyze, to design and to implement a MAS and little attention has been given to how such systems can be tested. Furthermore, some issues related to the controllability and observability makes difficult the verification of behavior. This paper presents an approach to verify the behavior of agents based on the combination and adaptation of ideas already supported by the literature.*

Resumo. *A utilização de sistemas multiagentes (SMA) para construção de software é uma abordagem promissora e tem sido aplicada em diferentes áreas de pesquisa. Assim, a validação destes sistemas é crucial para construção de softwares robustos. No entanto, os métodos propostos concentraram seus esforços para analisar, projetar e implementar SMA, não dando atenção necessária para a forma como tais sistemas podem ser testados. Além disso, questões relacionadas à controlabilidade e observabilidade dificultam a verificação do comportamento. Este artigo apresenta uma abordagem para verificação do comportamento de agentes baseada na combinação e adaptação de ideias já suportadas pela literatura.*

1. Introdução

Com o crescimento da web, sistemas complexos se tornaram uma realidade. Estes são caracterizados por serem distribuídos e composto de entidades autônomas que interagem entre si. Sistemas multiagentes (SMA) são sociedades nas quais entidades autônomas (agentes), heterogêneas e projetadas individualmente, trabalham em função de objetivos que podem ser comuns ou diferentes [López 2003]. Assim, a utilização de agentes para construção de tais sistemas complexos é considerada uma abordagem promissora [Zambonelli et al. 2001]. Com base nestes aspectos, o uso de SMA vem sendo cada vez mais aplicado em diferentes áreas, com diferentes níveis de autonomia, indo desde sistemas quase totalmente controlados com a ajuda de intervenção humana até aqueles quase totalmente automatizados, ou seja, com o mínimo de intervenção humana, o que significa que analisar as escolhas que este tipo de software pode fazer torna-se crucial [Fisher et al. 2013].

No que se refere ao teste de sistemas de agentes de software, algumas questões relacionadas à controlabildade e à observabilidade precisam ser cuidadosamente consideradas, pois: (i) um agente é uma entidade autônoma e, consequentemente, pode ser difícil controlar seu comportamento; (ii) as crenças e objetivos do agente estão embutidos no próprio agente podendo não ser facilmente observados e controlados; (iii) sem a adoção de estratégias eficientes para cobertura dos testes, o teste certamente se torna não escalável, dado o número de possibilidades a serem testadas [Binder 1999] e [Voas e Miller 1995].

Este artigo apresenta uma abordagem que permite a verificação do comportamento de agentes BDI [Rao e Georgeff 1995] desenvolvidos em BDI4JADE [Nunes et al. 2011]. Tal abordagem se baseada na combinação e adaptação de ideias já suportadas pela literatura de testes em agentes, em especial, do JAT *Framework* [Coelho et al. 2007] e no modelo de faltas proposto por Zhang [Zhang 2009].

2. Motivação

Apesar do uso crescente de sistemas multiagentes em cenários críticos, as metodologias propostas até o momento pela Engenharia de Software Orientada a Agentes (AOSE) concentraram seus esforços, principalmente no desenvolvimento de abordagens disciplinadas para analisar, projetar e codificar tais sistemas. No entanto, pouca atenção tem sido empregada na forma como tais sistemas poderiam ser testados [Caire et al. 2004].

Sendo assim, diante da necessidade de verificar e compreender o comportamento complexo executado pelo agente, avaliar sua eficácia e do desafio e implicações referentes à testabilidade dos agentes de software, este trabalho se concentra na tarefa de apresentar uma abordagem que permita apoiar o desenvolvedor de agentes na construção de casos de teste para agentes BDI.

As seguintes questões surgem como motivação para este trabalho: "*Como podemos apoiar o desenvolvimento de sistemas multiagentes através da construção e manutenção de casos de testes para agentes BDI?*" e "*Como podemos controlar e observar as ações executadas durante o ciclo de raciocínio de agentes BDI?*".

3. Trabalhos Relacionados

Embora existam trabalhos que abordam e apresentam estratégias para o teste de agentes BDI nenhum deles se preocupa em fornecer mecanismos que auxilie na identificação de falhas de implementação e na observação do estado interno dos elementos do agente (*beliefs*, *goals*, *plans*, *events*, etc.) durante o processo de desenvolvimento do agente.

[Winikoff e Cranefield 2010] apresentam uma análise da flexibilidade e das características adaptativas dos agentes BDI observando o espaço comportamental do agente, ou seja, o número de caminhos possíveis para alcançar um objetivo. O trabalho buscou entender a viabilidade de assegurar a eficácia dos sistemas multiagentes através dos testes. Para isso, relacionaram a viabilidade do teste de um sistema multiagente à proporção de caminhos que podem ser percorridos do espaço comportamental, considerando ainda que, executar um teste consiste em observar um caminho de execução e determinar se este está correto ou não. Em última análise, os autores concluem que testar o espaço comportamental de um sistema multiagente como um

todo é inviável dado o seu comportamento assintótico. A preocupação apresentada neste trabalho não está em verificar se um determinado caminho está correto, nem em fornecer mecanismos para identificação de falhas, mas em determinar se é possível garantir a eficácia do sistema por meio de testes. As conclusões de Winikoff e Cranefield sobre a viabilidade do teste de sistemas multiagentes corroboraram para a decisão de propor uma abordagem que considere o teste unitário de agentes.

No trabalho [Coelho et al. 2007] apresenta o JAT (Jade Agent Testing Framework), um *framework*, capaz de criar testes para sistemas multiagentes escritos em JADE baseado na utilização de "agentes mock" [Coelho et al. 2006], ou seja, na implementação "falsa" de um agente real com o propósito exclusivo de testar a comunicação entre os agentes. Através do monitoramento do estado interno dos agentes é possível controlar e observar a interação entre os agentes mock e o agente em teste. Apesar da boa contribuição no que tange ao teste de agentes de software, este trabalho se limita a testar agentes de comportamentos reativos onde são verificadas, basicamente, falhas no protocolo de comunicação entre os agentes.

Em [Zhang et al. 2009] é apresentado um *framework* para geração automática de casos de testes para sistemas multiagentes. Este trabalho considera a construção de sistemas multiagentes baseados em modelos [Apfelbaum e Doyle 1997] [El-Far e Whittaker 2001]. É apresentado, ainda, um modelo de faltas para identificar possíveis falhas e as condições em que as mesmas podem ocorrer. Este modelo de faltas foi utilizado pela abordagem descrita neste artigo.

4. Uma abordagem para testes de agentes BDI4JADE

Esta seção apresenta uma visão geral da abordagem proposta, apoiada nos trabalhos de [Coelho et al. 2007] e [Zhang et al. 2009].

4.1. Visão geral da abordagem

Cada agente possui sua própria "*thread*" de execução e seu comportamento é determinado por um conjunto de inferências feitas mediante suas crenças, objetivos e planos [Garcia et al. 2004]. Com o objetivo de testar efetivamente o agente – considerando que os testes são baseados em alguma forma de comparação do resultado esperado com o resultado produzido – é necessário saber como cada um dos elementos do agente se comportou durante sua execução. Portanto, a informação sobre o estado das crenças, os planos que foram selecionados e abandonados, os objetivos, as mensagens trocadas e os eventos disparados são cruciais para verificar se o agente executou conforme esperado.

A ideia de criar estruturas capazes de manter as informações sobre as transições dos estados do agente [Coelho et al. 2007], motivou, de forma análoga, a criação de um conjunto de estruturas capazes de armazenar as informações dos elementos dos agentes, ocorridas durante a execução. Sendo assim, é possível consultar posteriormente as estruturas preenchidas, e verificar se se estas se comportaram como esperado e identificar a ocorrência de possíveis falhas de acordo com [Zhang et al. 2009].

4.2. Monitorando as informações do ciclo de raciocínio com ASPECTJ

Para obter esse resultado, deve-se adicionar código (nos agentes e plataformas envolvidas, neste caso, o BDI4JADE) nos pontos onde ocorrem alterações nas crenças,

planos, troca de mensagens e eventos assim como nos locais onde são tomadas decisões importantes do mecanismo deliberativo.

Fazendo assim, no entanto, o código adicionado estaria espalhado em muitos pontos e por muitos módulos da plataforma. Logo, monitorar o ciclo de raciocínio do agente é, naturalmente, um interesse transversal. Nestes casos, uma solução amplamente adotada é definir um aspecto para apontar diretamente os locais de execução no agente e na plataforma que registram as transições no estado dos componentes [Briand et al. 2005]. O monitoramento do ciclo de raciocínio do agente e da plataforma BDI4JADE é realizado através de um aspecto, implementado na linguagem ASPECTJ, conforme pode ser visto na Figura 1.

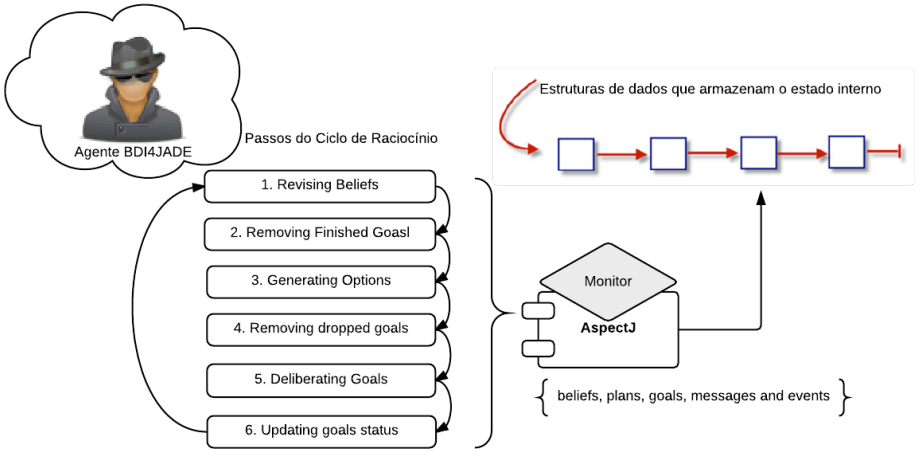

Figura 1. Armazenando as informações dos agentes durante a execução.

Cada passo do ciclo do raciocínio é responsável por uma etapa do processo deliberativo. Sendo assim, durante sua execução são criadas estruturas associadas a cada etapa do ciclo de raciocínio, capazes de armazenar as informações e transições ocorridas na execução da etapa. Por exemplo, durante a etapa de revisão das crenças, executada no início do ciclo de raciocínio do agente são criadas estruturas capazes de armazenar as crenças iniciadas com a criação do agente, as crenças que foram removidas durante a execução do agente, as crenças que tiveram seu valor alterado e assim por diante. Após a execução do agente as estruturas estão preenchidas e podem ser consultadas. Dessa forma, o testador pode consultar o conteúdo dessas estruturas e verificar se um comportamento ocorreu conforme esperado (se o valor de uma crença foi alterado, por exemplo).

O conteúdo das estruturas pode ser consultado através de métodos assertivos facilitadores no estilo JUNIT. Um exemplo de um método assertivo que permite ao desenvolvedor verificar a alteração no valor de uma crença é: *assertWasUpdatedBelief*. Assim, através de um conjunto de métodos assertivos capazes de consultar as informações e transições dos elementos do agente ocorridas durante o ciclo de raciocínio é possível, por meio de casos de testes definidos pelo desenvolvedor do agente, verificar o comportamento do agente e identificar possíveis condições de falhas.

5. Conclusão

Este artigo propôs uma abordagem para verificação do comportamento de agentes BDI desenvolvidos em BDI4JADE capaz de apoiar o desenvolvimento de agentes de software através da construção e manutenção de casos de testes. Tal abordagem apoiou-se nas ideias suportadas por [Coelho et al. 2007] e no modelo de faltas proposto por [Zhang 2009].

Como trabalhos futuros, vemos a necessidade de implementar uma ferramenta adotando a abordagem proposta e também executar um conjunto de cenários de uso como prova de conceito da abordagem. Em seguida, um experimento controlado pode ser realizado para obter junto a comunidade de desenvolvedores de agentes um *feedback* do uso da ferramenta. Ainda como trabalho futuro, uma análise desta abordagem pode ser aplicada em outras plataformas de desenvolvimento de agentes tais como: JASON [Bordini et al. 2007], JADEX [Braubach et al. 2003]e JACK [Howden et al. 2001], por exemplo. Procurando entender suas implicações e caso seja necessário, adaptações podem ser realizadas.

References

Apfelbaum, L. and Doyle, J. (1997) "Model Based Testing. International Software Quality Week Conference". CA – USA.

Binder, R (1999) "Testing Object-Oriented Systems: Models, Patterns, and Tools", Addison-Wesley, 1999.

Bordini, R. H., Hübner, J. F. and Wooldridge, M. (2007) "Programming Multi-Agent Systems in AgentSpeak using Jason", In Wiley Series in Agent Technology.

Braubach, L., Lamersdorf, W., and Pokahr, A. (2003) "Jadex: Implementing a BDI-infrastructure for JADE agents".

Briand, L., Labiche, Y. and Leduc, J. (2005) "Tracing Distributed Systems Executions Using AspectJ", In: Proceedings of ICSM.

Caire, G., Cossentino, M., Negri, A., Poggi, A. and Turci, P. (2004) "Multi-agent systems implementation and testing", In Proceedings of 4th International Symposium - From Agent Theory to Agent Implementation.

Coelho, R, Cirilo, E., Kulesza, U., Staa, A., Rashid A. and Lucena, C. (2007) "JAT: A Test Automation Framework for MultiAgent Systems", In International Conference on Software Maintenance. ICSM.

Coelho, R., Kulesza, U., Staa, A. and Lucena, C. (2006) "Unit Testing in Multi-agent Systems using Mock Agents and Aspects", In International Workshop on Software Engineering for Large-Scale Multi-Agent Systems. ICSE.

El-Far, I. and Whittaker, J. (2001) "Model-Based Software Testing", In Encyclopedia of Software Engineering, pages 825-837. Wiley, Chichester.

Fisher, M., Dennis, L. and Webster, M. (2013) "Verifying Autonomous Systems", In Communications of the ACM, Vol. 56 No. 9, Pages 84-93.

Garcia, A., Lucena, C. and Cowan, D. (2004) "Agents in Object-Oriented Software Engineering. Software Practice & Experience", Elsevier, 34(5), pages 489-521.

Howden, N., Rönnquist, R., Hodgson, A., & Lucas, A. (2001, May). “JACK intelligent agents-summary of an agent infrastructure”. In 5th International conference on autonomous agents.

López, F. L. (2003) “Social Power and Norms”.

Nunes, I., Lucena, C. and Luck, M. (2011) “BDI4JADE: a BDI layer on top of JADE”, In: International Workshop on Programming Multi-Agent Systems - ProMAS.

Pokahr, A., Braubach L. and Lamersdorf W. (2005) “Jadex: A BDI Reasoning Engine”, Multi-Agent Programming - Springer US.

Rao, A. and Georgeff, M. (1995) “BDI-agents: from theory to practice”, In: Proceedings of the First International Conference on Multiagent Systems.

Voas, J. and Miller, K. (1995) “Software Testability: The New Verication”, In: IEEE Software, 1995.

Winikoff, M. (2005) “Jack™ Intelligent Agents: An Industrial Strength Platform”, In: Multi-Agent Programming - Springer US.

Winikoff, M. and Cranefield, S. (2010) “On the testability of BDI agents”, In: European Workshop on Multi-Agent Systems.

Zambonelli, F., Jennings, N., Omicini, A. and Wooldridge, M. (2001) “Agent-oriented software engineering for internet applications”, Coordination of Internet Agents, p. 326–346. Springer Verlag.

Zhang, Z., Thangarajah, J. and Padgham, L. (2009) “Model based testing for agent systems”, In: International Conference on Autonomous Agents and Multiagent Systems.