Em Direção ao Tratamento de Falhas na Plataforma Multi-Agentes JADE *

André Filipe de Moraes Batista, Maria das Graças Bruno Marietto, Robson dos Santos França, Guiou Kobayashi

Universidade Federal do ABC (UFABC)
Centro de Matemática, Computação e Cognição (CMCC)
Santo André - SP - Brasil
{andre.batista,graca.marietto,robson.franca,guiou.kobayashi}@ufabc.edu.br

Abstract To build open and distributed systems, like the Multi-Agent Systems (MAS), it is necessary to deal with a fundamental issue: the treatment of fault tolerance, aiming to guarantee the correct operation of the system, even with the presence or activation of faults. In this work we analyze how the JADE platform detects the occurrence and management of a total fall in its main-container. To deal with this situation, we propose to use the technique of agents' replication, as well as the storage of the MAS´yellow pages in the HSQLDB database. Experiments were realized, and the results of the tests are presented and discussed.

1 Introdução

Uma das principais características da área de Sistemas Multi-Agentes (SMA) é a possibilidade da modelagem e implementação de sistemas computacionais sob a metáfora da inteligência social. Nesta abordagem SMAs partem do pressuposto de uma construção coletiva da solução, onde comportamentos emergem como resultado das interações de seus elementos, seguindo regras locais. Aqui, agentes trabalham de forma autônoma na medida que são capazes de desempenhar determinadas ações de maneira independente. Entretanto, em alguns momentos interações sociais são necessárias objetivando viabilizar a execução de ações coletivas. Para tanto, há a necessidade de desenvolvimento de mecanismos de comunicação, cooperação/colaboração, coordenação, negociação, dentre outros [1].

A autonomia dos agentes permite que tomem suas próprias decisões para o alcance de seus objetivos, como estudado por exemplo em [2]. Sendo assim, agentes podem entrar e sair da sociedade, podem alterar suas regras, papéis, relações de inter-dependência com outros agentes, etc. Esta característica leva a uma nova geração de sistemas e aplicações distribuídas intrinsicamente dinâmica, aberta e complexa. Entretanto, como destacado em [3], como conseqüência deste processo independente de tomada de decisão, há um aumento na imprevisibilidade do sistema. Isto porque, sob o ponto de vista da Engenharia de *Software*, tanto

* Este trabalho possui suporte financeiro da FAPESP, processo n^0 2006/04650-6.

a natureza quanto as conseqüências das relações entre os agentes (em tempo de execução) não podem ser determinadas desde o início pelo projetista.

Pelo o que foi exposto até o momento pode-se perceber que SMAs apresentam características tais como: (a) são sistemas abertos, (b) agentes agem de forma autônoma, tentando alcançar seus objetivos (dialeticamente, mantendo uma relação com seu ambiente), (c) possuem comportamento dinâmico e agregado, (d) a ação de cada agente afeta as ações subsequentes da sociedade, (e) o funcionamento do sistema é socialmente construído e ocorre de maneira emergente, (f) apresentam um elevado grau de imprevisibilidade em seu funcionamento.

Estas características indicam a possibilidade da ocorrência de falhas no funcionamento de SMAs, sendo necessário o desenvolvimento de mecanismos de tolerância que englobem desde a detecção até o tratamento de tais falhas. Entretanto, analisando a literatura da área pode-se observar que há poucos trabalhos que abordam esta questão [4,5,6,7]. Esta observação torna-se mais surpreendente quando verifica-se a grande quantidade de trabalhos em pesquisa e desenvolvimento na área de SMA, e mesmo assim uma questão tão básica em Engenharia de *Software* como tolerância a falhas não está sendo adequadamente tratada, ou divulgada pela comunidade científica. De acordo com [7], uma das principais razões para esta lacuna é que ainda a maioria dos SMAs estão desenvolvidos em pequena escala. A escalabilidade em SMAs refere-se à quantidade de agentes, bem como de *h*osts sobre os quais uma aplicação é distribuída.

Objetivando auxiliar no desenvolvimento de metodologias e técnicas de tolerância a falhas em SMAs, neste trabalho é apresentado um estudo de caso sobre a plataforma de agentes JADE [8]. Mais especificamente, este estudo foca as funcionalidades da plataforma para o tratamento de uma falha total em seu gerenciador de agentes, denominado *main-container*. Caso esta falha não seja detectada e tratada de forma eficiente, o SMA como um todo pode parar de funcionar. Ou então, caso haja possibilidade de restauração, a configuração do sistema no momento da queda pode não ser recuperada com a qualidade necessária para garantir a confiabilidade e robustez do sistema.

Duas técnicas da plataforma JADE para tratar a queda de um *main-container* serão analisadas. A primeira delas consiste em criar uma réplica deste *container* principal para que, em caso de problemas em seu funcionamento, a réplica assuma o controle dos agentes do SMA. A segunda técnica refere-se ao armazenamento do arquivo que contém a lista de serviços disponibilizados pelos agentes (páginas amarelas) no gerenciador de banco de dados HSQLDB [9]. A persistência em um banco de dados é importante pois, por *default* estas informações estão armazenadas na memória do *host* principal, e com a queda do *main-container* não ficam mais disponíveis.

O restante do artigo está organizado da seguinte maneira. Na Seção 2 a estrutura de SMAs é apresentada, analisando situações em que estes podem falhar. Adicionalmente, na Subseção 2.1 alguns estudos que também tratam da replicação de agentes são discutidos. Na Seção 3 tem-se o detalhamento da arquitetura da plataforma de agentes JADE. A Seção 4 apresenta as técnicas de

tolerância a falhas desta plataforma. A Seção 5 relata os experimentos executados para validação destas técnicas. A Seção 6 apresenta os resultados obtidos com a execução dos experimentos e, por último a Seção 7 apresenta as considerações finais deste trabalho.

2 Tolerância a Falhas em Sistemas Multiagentes

Quando o funcionamento de um sistema exige alta confiabilidade e robustez, técnicas relacionadas à prevenção e à remoção de falhas podem não ser suficientes. Neste caso é indicado o uso de técnicas de tolerância a falhas, que visam garantir o funcionamento correto do sistema mesmo com a presença ou ativação de falhas [10].

Sistemas Multiagentes são sistemas constituídos de múltiplos agentes que interagem de forma a realizar um determinado conjunto de tarefas. A metáfora da inteligência utilizada por estes sistemas é a de comunidade inteligente, em que o comportamento social é a base para o funcionamento do sistema. A partir disto, é possível distribuir os agentes em áreas de especialização e, com a interação destes, um problema complexo pode ser resolvido de uma forma mais rápida e dinâmica [11].

Entretanto, como SMAs são sistemas distribuídos, estão expostos a altas taxas de falhas em tempo de execução. Por isto, mecanismos de tolerância a falhas são imprescindíveis em tais ambientes, especialmente naqueles onde questões de segurança são fundamentais como por exemplo em instituições financeiras, comércio eletrônico, controle de tráfego aéreo, sistemas de telecomunicação, dentre outros.

De forma geral, pesquisas em SMA na área de tolerância a falhas objetivam garantir a continuidade da computação de cada agente e, como conseqüência, da agência como um todo. De forma específica, estudos são realizados relacionados a algumas características destes sistemas que podem levar a falhas em seu funcionamento. Dentre elas três (03) podem ser destacadas [3,4,11]. A primeira característica considerada é a autonomia dos agentes que atuam em um SMA, significando que executam a maior parte de suas ações sem interferência de outros agentes (humanos ou computacionais). Tal autonomia permite ao agente agir sem autorização ou conhecimento por parte dos demais elementos do ambiente, podendo assim alterar a configuração ao seu redor. Por exemplo, um agente pode quebrar uma ordem normativa, não respeitando o consenso em relação às expectativas de como cada membro deve se comportar na agência. Isto leva a um alto grau de imprevisibilidade no funcionamento do sistema, devido à impossibilidade de um agente predizer o comportamento dos outros agentes da sociedade.

Outra característica é o dinamismo do sistema, dinamismo este vinculado ao princípio de que SMAs apresentam comportamentos emergentes que resultam de ações locais de seus agentes e suas interações. Vale ressaltar que tais comportamentos emergentes não são explicitamente programados nos agentes, pois são comportamentos socialmente construídos. Por exemplo, a multitude das possíveis

interações entre os agentes leva a um alto grau de imprevisibilidade no processo de tomada de decisão. Com esta imprevisibilidade as alterações no ambiente são, então, dinâmicas e emergentes, aumentando a complexidade no planejamento de possíveis situações que podem ocorrer no sistema. Tal dinamismo emergente dificulta, por exemplo, a capacidade de um SMA em recuperar seu estado caso haja necessidade de restauração.

Por último, agentes em um SMA podem agir de forma cooperativa, colaborativa ou competitiva. Sendo assim, um processo de coordenação deve gerenciar o comportamento dos agentes objetivando evitar conflitos, esforços redundantes, intertravamento (*deadlock*), etc. Quanto maior a incidência de situações deste tipo menos coordenado é o sistema e, por isto, com mais chances de propagações imprevisíveis de falhas [4,11].

De acordo com [4] os tipos de falhas que podem ocorrer em SMAs, a partir destas características são: (i) estados e comportamentos não previstos pelos desenvolvedores, fazendo com que o sistema não consiga manipular tais resultados, (ii) falhas no gerenciador de agentes, podendo ser uma queda total ou a falta de alguns recursos da plataforma, (iii) falhas de comunicação, (iv) intertravamento no funcionamento dos agentes, (v) falhas de interação provocadas por agentes maliciosos.

Há técnicas específicas para o tratamento de cada uma destas falhas, objetivando garantir a robustez e confiabilidade de SMAs [4]. Na Seção 4 a característica relacionada ao dinamismo de um SMA será analisada, considerando como recuperar o estado de um sistema quando de sua interrupção. Na Seção 2.1 apresentamos alguns trabalhos que também abordam como recuperar o estado do sistema, usando a técnica de replicação.

2.1 Replicação de Componentes: Trabalhos Relacionados

Parte das técnicas de tolerância a falhas se baseia na replicação de componentes de *hardware* e/ou *software*. Neste contexto, caso algum dos componentes venha a falhar, uma réplica do mesmo será ativada para continuar a prestar o serviço que antes era fornecido pelo componente que falhou. De forma específica, para SMAs o ato de replicar agentes consiste em criar uma ou mais réplicas (*backups*) de um ou mais agentes. Como destacado em [12], a replicação de componentes é uma das técnicas mais eficientes para o tratamento de falhas. Dentre as vantagens do uso da replicação tem-se [12,13]: (i) é menos intrusiva durante o tempo de execução, (ii) é mais adequada para tratar escalabilidade, (iii) é relativamente genérica e transparente para o domínio da aplicação.

Os trabalhos de [7,13] tratam a questão da tolerância a falhas em SMAs, propondo uma nova abordagem para avaliar dinamicamente a criticalidade de agentes. Esta abordagem está baseada nos conceitos de papéis e atividade. A criticalidade do agente é, posteriormente, usada para replicar agentes com o objetivo de maximizar a segurança do sistema. Para validar esta proposta foi desenvolvido um *f*ramework de replicação tolerante a falhas, denominado DarX. Em tal ambiente SMAs foram criados para decidirem, de forma automática e

dinâmica, quais entidades devem ser replicadas e como parametrizar estas replicações.

Este trabalho é uma continuidade da pesquisa desenvolvida em [14]. Aqui apresentamos o problema com um foco mais teórico, e em [14] a descrição da experiência é destacada.

3 Plataforma de Agentes JADE

A plataforma *Java Agent Development Framework* (JADE) é um ambiente para o desenvolvimento de aplicações baseadas em agentes, que segue as especificações do padrão FIPA [15] e permite a interoperabilidade entre SMAs. JADE está implementada na linguagem Java, tendo sido desenvolvida na Universidade de Parma, Itália [8].

Dentre as especificações FIPA inseridas na plataforma JADE destacam-se neste trabalho as páginas brancas e páginas amarelas. As páginas brancas relacionam o nome de cada agente com seu endereço, permitindo que um agente seja encontrado na agência. As páginas amarelas listam os serviços oferecidos pelos agentes da sociedade. Para implementar estas funcionalidades a plataforma conta com agentes que atuam como páginas brancas e páginas amarelas. Estes agentes são o *Agent Management System* (AMS) e *Directory Facilitator* (DF), respectivamente.

A arquitetura da plataforma JADE permite dividir sua execução em diferentes máquinas (*hosts*), sendo que em cada *host* deve haver uma Máquina Virtual Java instalada.Um *container* no ambiente JADE é um local onde agentes residem e entre os quais podem se mover. Cada *container* oferece suporte para a execução de seus agentes. A plataforma JADE conta com um *container* principal denominado *main-container*, onde encontram-se os agentes AMS e DF. Todos os demais *containers* se registram neste *container* principal.

4 Tolerância a Falhas na Plataforma JADE

Neste trabalho será desenvolvido um estudo de caso na plataforma JADE focando como esta plataforma detecta e gerencia a ocorrência da falha do tipo (ii), mais especificamente a queda total de seu gerenciador de agentes (*main-container*). Para tratar esta falha uma possível abordagem é a técnica de replicação de agentes. No contexto de um SMA, o principal critério para se definir uma replicação é o grau de criticalidade de um agente. Para [4] "*criticalidade é o quão importante um agente é para o sistema, isto é, qual o potencial impacto que um agente defeituoso pode causar*".

Analisando a plataforma JADE neste contexto de criticalidade, o funcionamento do agente *main-container* tem alto impacto na confiabilidade de um SMA. Isto porque há operações executadas apenas por ele, como por exemplo o gerenciamento dos demais *containers*, hospedagem dos agentes responsáveis pelas páginas brancas e páginas amarelas (AMS e DF, respectivamente), conexão inter-plataformas, mobilidade de agentes, dentre outros. Esta centralização revela uma

vulnerabilidade da plataforma pois, caso ocorra uma falha no *main-container* interrompendo seu funcionamento, as atividades dos agentes da sociedade serão interrompidas. Para tratar este problema a plataforma JADE oferece duas funcionalidades [8]. A primeira é a técnica de replicação de agentes, implementada com o serviço *Main Container Replication Service* (MCRS). A segunda consiste no armazenamento das páginas amarelas do SMA no banco de dados HSQLDB. Nas próximas subseções estas funcionalidades são analisadas.

4.1 *Main Container Replication Service*

A Figura 1 ilustra o esquema do *Main Container Replication Service* (MCRS) da plataforma JADE, mostrando o *main-container* e os demais *containers* da plataforma (C1, C2, C3) conectados a este. Também apresenta a réplica do *main-container*, que é um outro gerenciador de agentes executado concorrentemente. Esta réplica envia mensagens de tempos em tempos para o *main-container*, monitorando seu funcionamento. Quando a réplica não recebe resposta a uma mensagem enviada, uma exceção é gerada e os demais *containers* (C1, C2 e C3) conectam-se à réplica, que assume o papel de *container* principal da plataforma.

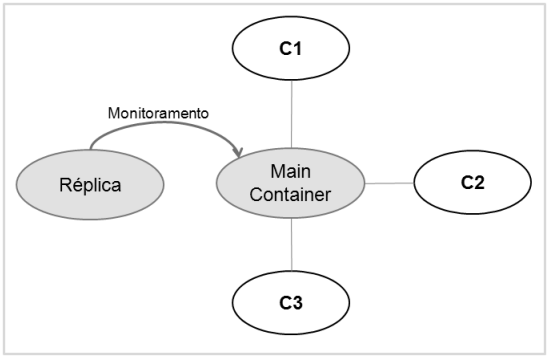

Figura 1. Serviço de Replicação de um *Main-Container* da Plataforma JADE.

4.2 Persistência das Páginas Amarelas

Após uma falha total do *main-container* sua réplica pode não ter acesso à lista de serviços disponibilizados pelos agentes, pois por *default* o *main-container* armazena em memória as páginas amarelas do SMA. Para tratar esta vulnerabilidade a plataforma JADE permite a persistência do arquivo DF de forma compartilhada, utilizando o banco de dados HSQLDB executado na forma de servidor. Quando o *main-container* e a réplica são executados, ambos conectam-se ao servidor HSQLDB. Ocorrendo uma falha no *main-container*, a réplica assume a liderança do SMA e consulta o servidor para obter as informações das páginas amarelas.

5 Plataforma JADE e o Tratamento da Criticalidade do *Main-Container*

Neste trabalho experimentos foram modelados e executados objetivando analisar o comportamento da plataforma JADE com relação ao tratamento de uma falha geral no *main-container*. A configuração de tais experimentos está apresentada nesta seção, e os resultados obtidos estão descritos na Seção 6.

5.1 Objetivo dos Experimentos

Os experimentos realizados visam testar a tolerância a falhas da plataforma JADE considerando que (a) o *main-container* possui uma réplica sincronizada, (b) o gerenciador de banco de dados HSQLDB é usado para garantir a persistência e o compartilhamento das páginas amarelas do SMA.

5.2 Configuração dos Cenários

Os testes foram estruturados para medir o tempo de atuação do agente réplica desde o momento da percepção de uma falha no *main-container*, até o momento em que a réplica assume a liderança do SMA. Esta liderança se efetiva quando todos os *containers*, antes conectados ao *container* principal, se registram no agente réplica.

Em cada experimento uma falha geral no *main-container* é intencionalmente ocasionada para que o processo de reestruturação da plataforma seja iniciado. Os valores de tempo utilizados nos experimentos foram disponibilizados pela plataforma JADE, como por exemplo o instante em que os *containers* se registram na réplica bem como o instante que ocorre a falha no *main-container*. Isto é possível porque a plataforma, na ocorrência de qualquer evento, gera uma mensagem com informações a respeito deste evento, inclusive o instante em que o mesmo ocorreu.

Foram definidos quatro (04) cenários, cada um composto por um (01) agente *main-container*, dez (10) *containers* a ele conectados e um (01) agente réplica. Tanto a réplica quando o *main-container* estão conectados ao servidor de dados. A Figura 2 ilustra esta configuração. Os cenários foram configurados alterando-se a quantidade de agentes em cada *container*. A Tabela 1 apresenta a distribuição proposta. Por exemplo, no cenário 2 há dez *containers* com 100 agentes em cada um deles. Portanto, ao total tem-se 1.000 agentes conectados ao *main-container*.

Os agentes dos *containers* são implementados a partir do modelo `PingAgent`, disponibilizado pela plataforma JADE. Este agente tem a funcionalidade de se registrar nas páginas amarelas do *container* principal. Para a instanciação dos `PingAgent` foi desenvolvido o agente `Criador`. Após a instanciação dos agentes o `Criador` se destrói, deixando a quantidade exata de agentes por *container*. Cada cenário foi repetido três vezes. Assim, por três vezes os agentes foram instanciados, simulou-se uma falha no *main-container* e mediu-se o tempo que o agente réplica precisou para obter a liderança do SMA. Ao final, para cada cenário foi feita uma média aritmética dos tempos obtidos.

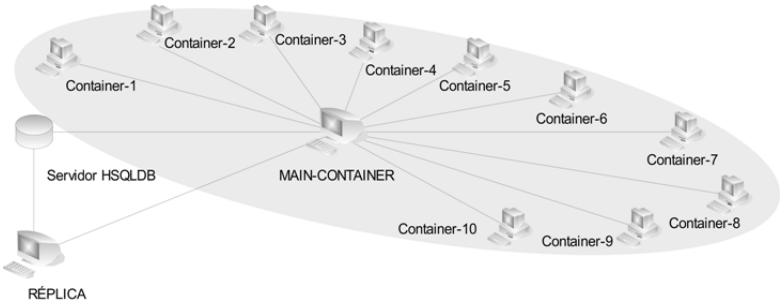

Figura 2. Configuração dos Cenários.

Tabela 1. Número de Agentes por *Container* em cada Cenário.

Cenário	Agentes instanciados em cada *container*	Total de Agentes Conectados ao *Main-Container*
1	10	100
2	100	1.000
3	1.000	10.000
4	5.000	50.000

5.3 Configuração de *Hardware* e *Software*

Os experimentos foram executados em micro-computadores com processador Pentium 3.4 GHz e 1 GB de RAM, BIOS versão 2.53. O sistema operacional instalado foi o Microsoft Windows Professional, versão Service Pack 2. As versões dos *softwares* usados são as seguintes: (a) plataforma Java Sun JDK 1.6, (b) plataforma JADE 3.5, (c) gerenciador de banco de dados HSQLDB 1.8.0.

6 Análise dos Resultados

Após a execução dos experimentos obteve-se o tempo médio do tratamento da falha simulada em cada cenário. Este tempo foi medido iniciando no instante em que o agente réplica detectou uma falha no *main-container*, até o instante em que os demais *containers* se conectaram à réplica. Os resultados obtidos estão apresentados no gráfico da Figura 3.

O tempo de reestruturação da plataforma com 100, 1.000, 10.000 e 50.000 agentes foi de 167, 177, 203,8 e 243,4 segundos, respectivamente. Tais resultados indicam uma escalabilidade linear no tratamento à falha considerada, pois o crescimento do tempo de reestruturação da plataforma mostrou-se linear em função do aumento do número de agentes. O tempo de reestruturação de tal SMA, em caso de falha de seu gerenciador de agentes, é crucial para garantir a confiabilidade e robustez deste sistema.

O armazenamento do arquivo das páginas amarelas do SMA no gerenciador de banco de dados HSQLDB, rodando como servidor, ocorreu de forma estável

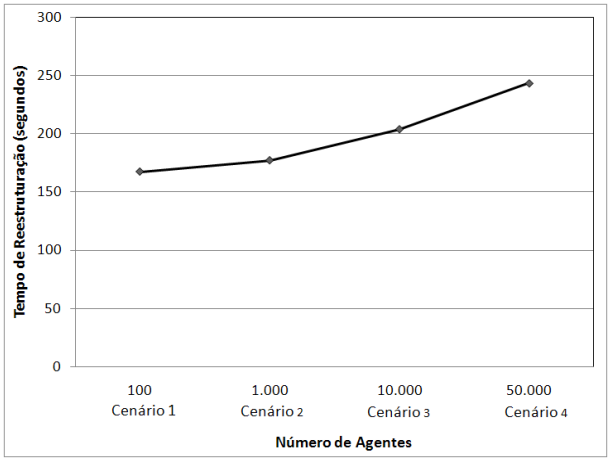

Figura 3. Número de Agentes *versus* Tempo de Reestruturação.

e não apresentou falhas em sua execução. Esta técnica garante a persistência das informações, bem como a confiabilidade da plataforma JADE. Se estas informações não estivessem armazenadas em uma base de dados, com a queda do *main-container* elas seriam perdidas e o SMA teria sua estabilidade afetada, uma vez que as informações sobre os agentes e seus respectivos serviços não estariam mais disponíveis, dificultando a interação entre os agentes.

6.1 Instanciação de Agentes na Plataforma JADE: O Céu não é o Limite

O número de agentes que podem ser executados em um *container* da plataforma JADE é limitado pelo fato que cada agente é encapsulado em uma *thread*.

O *main-container*, por sua vez, tem uma capacidade maior de agentes que podem ser gerenciados. Isto porque as *threads* dos agentes não são executadas no *main-container*, mas sim em seus *containers* de origem. A plataforma JADE, por *default*, aloca no *main-container* itens tais como (a) os agentes que atuam como páginas brancas e amarelas da plataforma, (b) as ferramentas de análise e de apoio administrativo da plataforma (tais como um agente que intercepta mensagens, uma interface GUI do repositório do DF, dentre outras). Sendo assim, o *main-container* também apresenta um limite definido pela quantidade de memória em seu *host*.

7 Conclusões

A possibilidade de falhas parciais é uma das características fundamentais de aplicações distribuídas e abertas, como é o caso de SMAs. Objetivando deixar tais sistemas mais confiáveis e robustos, neste trabalho foi apresentado um estudo

de caso sobre como a plataforma JADE trata a falha em um de seus componentes que apresenta um alto grau de criticalidade: o *main-container*. As técnicas nativas da plataforma para a réplica do *main-container* e a persistência de dados no servidor HSQLDB mostraram-se eficientes, gerenciando a mudança de liderança do SMA sem perda de consistência dos estados dos agentes. Os testes indicaram que JADE apresenta uma escalabilidade linear no tratamento a esta falha, pois em todos os experimentos o aumento no número de agentes resultou em um aumento linear no tempo de reestruturação do sistema.

Referências

1. Weiss, Gerhard. (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT Press.
2. C., C. and R., F. (2003). *Agent Autonomy*, chapter From Automaticity to Autonomy: the Frontier of Artificial Agents, pages 103–136. Kluwer Academic Publishers.
3. Potiron, K., Taillibert, P., and Fallah Seghrouchni, A. (2008). A step towards fault tolerance for multi-agent systems. pages 156–172.
4. Gatti, M. A. C. (2006). Fidedignidade de sistemas multiagentes abertos governados por leis. Dissertação de mestrado, Pontifícia Universidade Católica do Rio de Janeiro.
5. Gatti, M. A., Carvalho, G., Paes, R., von Staa, A., Lucena, C., and Brio, J.-P. O rationale da fidedignidade em sistemas multiagentes abertos governados por leis.
6. Guessoum, Z., Faci, N., and Briot, J.-P. (2005). Adaptive replication of large-scale multi-agent systems: towards a fault-tolerant multi-agent platform. *SIGSOFT Softw. Eng. Notes*, 30(4):1–6.
7. Guessoum, Z., Briot, J.-P., Sens, P., and Marin, O. Toward fault-tolerant multi-agent systems.
8. Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. John Wiley & Sons.
9. HSQLDB 2008. *HSQLDB HomePage*. Disponível em http://www.hsqldb.org.
10. Avizienis, A. (1967). Design of fault-tolerant computers. *Fall Joint Computer Conference*, v.31:p.733–743.
11. Marietto, M. G. B. (2000). *Definição Dinâmica de Estratégias Instrucionais em Sistemas de Tutoria Inteligente: Uma Abordagem Multiagentes na WWW*. PhD thesis, Instituto Tecnológico de Aeronáutica.
12. Guerraoui, R. and Schiper, A. (1997). Software-based replication for fault tolerance. *Computer*, 30(4):68–74.
13. De Luna Almeida, A., Briot, J.-P., Aknine, S., Guessoum, Z., and Marin, O. (2007). Towards autonomic fault-tolerant multi-agent systems. In *2nd Latin American Autonomic Computing Symposium (LAACS'07)*, Petropolis, RJ, Brasil.
14. Batista, A. F. M., Marietto, M. G. B., França, R. S., Kobayashi, G. (2009). Multi-agent systems and fault tolerance in the jade platform. In *International Conference on Enterprise Information Systems and Web Technologies (EISWT-09)*.
15. FIPA 2008. *Foundation for Intelligent Physical Agents*. Disponível em http://www.fipa.org.