Uma Revisão dos Métodos para Geração e Execução de Testes em Sistemas Multiagentes

Ricardo A. Machado[1], **Eder M. Gonçalves**[1]

[1]Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Rio Grande – RS – Brazil

ricardoarend@gmail.com, edergoncalves@furg.br

Abstract. *As you know every software system needs to be properly tested before entering the market so there is a guarantee of its operation and a safety to the user. However, the test of multi-agent systems (MAS) is a challenging task due to the autonomous, proactive and non-deterministic behavior of the agents, which makes it very difficult to predict all the test possibilities necessary for their complete validation.*
This article presents a review and classification of published papers focusing on the MAS tests. The motivations that led to the choice of the content studied here were based on the search for answers to questions such as: what difficulties exist to test an MAS, what levels of test can be performed and what techniques are most used to generate test cases. The objectives are, in addition to answering such questions, to analyze and describe the main approaches for generating and executing tests in these systems.

Resumo. *Como se sabe todo sistema de software precisa ser devidamente testado antes de entrar no mercado para que haja uma garantia de seu funcionamento e uma segurança ao usuário. Porém o teste de sistemas multiagentes (SMA) é uma tarefa desafiadora devido ao comportamento autônomo, proativo e não-determinístico dos agentes, o que faz com que seja muito difícil prever todas as possibilidades de teste necessárias para sua completa validação.*
Esse artigo apresenta uma revisão e classificação de trabalhos publicados que tenham como foco os testes em SMA. As motivações que levaram a escolha do conteúdo aqui pesquisado foram baseadas na busca de respostas para questões como: quais as dificuldades existentes para testar um SMA, que níveis de teste podem ser realizados e quais técnicas são mais utilizadas para gerar casos de teste. Os objetivos são, além de responder tais questões, analisar e descrever as principais abordagens existentes para geração e execução de testes nesses sistemas.

1. Introdução

Agentes são entidades autônomas, que gerenciam por si mesmos seu próprio estado e comportamento. Os agentes podem se comunicar com outros agentes ou usuários além de interagir com o ambiente, por meio de protocolos de comunicação e interação. Um sistema multiagente (SMA) é uma sociedade de agentes autônomos, que evolui em cooperação em seu ambiente para atingir coletivamente um objetivo global [Barnier et al. 2017].

Uma das grandes dificuldades ao trabalhar com SMA está relacionado com a realização de testes. O teste é uma atividade de desenvolvimento de software, dedicada a avaliar a qualidade do produto e melhorá-lo, identificando defeitos e problemas [Kerraoui et al. 2016]. No que diz respeito aos sistemas multiagentes, o teste é uma tarefa desafiadora, e requer novas técnicas que lidem com sua natureza específica. Segundo [Houhamdi 2011] as principais razões para esse grau de dificuldade são:

- maior complexidade, pois existem vários processos distribuídos que são executados de forma autônoma e simultânea;
- quantidade de dados, pois os sistemas podem ser compostos por milhares de agentes, cada um com seus próprios dados;
- efeito de irreprodutibilidade, já que não há garantia que duas execuções do mesmo sistema com as mesmas entradas levem ao mesmo estado, dificultando a procura de erros;
- eles são não-determinísticos, uma vez que não é possível determinar antecipadamente todas as interações de um agente durante sua execução.

Com o objetivo de responder algumas perguntas de pesquisa e gerar uma fundamentação para futuros trabalhos foi feita uma revisão de literatura tendo como base o teste de SMA. Para tal foi adotada uma metodologia que é descrita na seção 2 com os passos para a geração da busca e seleção dos artigos. Na seção 3 os artigos selecionados são classificados e descritos. Por fim, a seção 4 traz uma conclusão do trabalho realizado a partir das respostas encontradas para as perguntas de pesquisa.

2. Metodologia da Revisão

2.1. Perguntas de Pesquisa

O principal objetivo desta revisão sistemática da literatura é reconhecer e categorizar a literatura existente sobre abordagens de geração de casos de teste em SMA. Portanto, as questões de pesquisa abordadas nesta revisão são:

1. Quais as dificuldades para testar um sistema multiagente?
2. Que níveis de teste podem ser realizados num sistema multiagente?
3. Quais são as principais técnicas para geração de casos de teste em sistemas multiagentes?

2.2. Protocolo da Busca

Uma vez com as perguntas definidas, a próxima etapa foi gerar uma *string* para iniciar as buscas. O processo para realizar as buscas da pesquisa teve os seguintes passos:

1. Definir os termos primários da busca que são: "*agent system*", "*multi-agent system*", "*Multiagent System*".
2. Identificar termos específicos que tenham relação direta com o objetivo do trabalho sendo: "*test case*".
3. Identificar os termos usados para descrever os resultados da pesquisa: "*correction*", "*verification*", "*validation*", "*error*", "*faults*", "*failure*".
4. Utilizar os Booleanos *OR* e *AND* para concatenar as palavras acima listadas de forma que gerasse uma busca satisfatória.

Os termos de pesquisa resultantes são descritos através de uma *string* que ficou da seguinte maneira: *("agent system"OR "multi-agent system"OR "multiagent system") AND ("test case") AND (correction OR verification OR validation OR error OR faults OR failure).*

Com a *string* definida foram realizadas as buscas nos repositórios do Google Scholar e do Springer sendo que ela foi utilizada exatamente da maneira acima descrita em ambos. O motivo da escolha do Scholar foi a grande diversidade de outros repositórios indexada nele. No caso do Springer foi encontrados diferentes artigos que não haviam sido anteriormente retornados pela busca do Scholar. O período analisado foi entre o ano de 2007 até 2018 com o intuito de fechar uma década de abrangência da pesquisa.

2.3. Seleção dos Artigos

Na busca foram encontrados 2720 resultados sendo 2040 no Scholar e 680 no Springer. Para a seleção dos artigos mais relevantes foram adotados critérios de inclusão e exclusão sendo eles listados a seguir:

Inclusão:

1. Artigos que satisfazem as palavras chaves de busca e tratam das questões de pesquisa.
2. Artigos que tratam de testes em sistemas multiagentes ou agentes.
3. Artigos que apresentam uma técnica ou ferramenta para geração de casos de teste em sistemas multiagentes.

Exclusão:

1. Artigos que apresentam sistemas multiagentes mas não possuem nenhum objetivo ou método com finalidade de testar esses sistemas.
2. Artigos que tratam de teste de softwares mas não com foco em sistemas multiagentes ou em agentes.

Os critérios acima foram utilizados como filtros através da leitura do título e do abstract de cada artigo. No final dessa primeira triagem restaram um total de 33 artigos. Em seguida foram descartados artigos de teses assim como artigos duplicados em ambos repositórios ou mesmo sequências anteriores de um mesmo trabalho. Por fim restaram 17 artigos sendo 12 do Scholar e 5 do Springer que serão apresentados detalhadamente no capítulo seguinte.

3. Classificação dos Resultados

3.1. Níveis de Classificação

Como existem diferentes níveis de testes que podem ser aplicados num sistema de agentes se torna necessário classificar os resultados aqui obtidos de maneira organizada com base nesses níveis. Uma classificação anterior foi descrita por [Nguyen 2009] que separa os testes em: unidade, agente, integração, sistema e aceitação. A descrição individual desses níveis pode ser vista a seguir:

- *Unidade*: Testar unidades de código e módulos que compõem os agentes como metas, planos, crenças, sensores, mecanismo de raciocínio e assim por diante.

- *Agente*: Testar a integração dos diferentes módulos dentro de um agente; testar os recursos dos agentes para preencher seus objetivos e detectar o ambiente.
- *Integração*: Testar a interação de agentes, protocolos de comunicação e semântica, interação de agentes com o ambiente, integração de agentes com recursos compartilhados, cumprimento de normas; observar propriedades emergentes; certificar que um grupo de agentes e os recursos do ambiente funcionem corretamente juntos.
- *Sistema*: Testar o SMA como um sistema em execução no ambiente operacional de destino; teste para propriedades de qualidade que o sistema pretendido deve atingir, como adaptação, abertura, tolerância a falhas, desempenho.
- *Aceitação*: Testar o SMA no ambiente de execução do cliente e verificar se ele atende aos objetivos das partes interessadas.

Sendo a classificação acima descrita bem detalhada foi entendido que ela é válida para essa revisão. Portanto os artigos foram organizados com base nessa classificação, não havendo necessidade de criar algo novo para esse trabalho.

3.2. Classificação dos Artigos

A Tabela 1 apresenta uma classificação das publicações de acordo com os níveis descritos na seção anterior. Nela podemos ter uma visão geral dos trabalhos na área de teste de SMA que foram publicados na última década. Como podemos visualizar alguns trabalhos abrangem em sua abordagem diferentes níveis de teste.

Tabela 1. Classificação dos Artigos

Unidade	[Coelho et al. 2007], [Poutakidis et al. 2009], [Zhang et al. 2009], [Salamon 2009], [Winikoff 2017].
Agente	[Kissoum and Sahnoun 2007], [Nguyen et al. 2007], [Gomez-Sanz et al. 2008], [Nguyen et al. 2008], [Wang and Zhu 2012], [Eassa et al. 2014], [Kerraoui et al. 2016], [do Nascimento et al. 2017], [Winikoff 2017], [Barnier et al. 2017].
Integração	[Coelho et al. 2007], [Salamon 2009], [Poutakidis et al. 2009],[Gomez-Sanz et al. 2008], [Miller et al. 2010], [Eassa et al. 2014], [Ur Rehman and Nadeem 2015], [Kerraoui et al. 2016], [do Nascimento et al. 2017], [Barnier et al. 2017].
Sistema	[Coelho et al. 2007],[Salamon 2009], [Houhamdi and Athamena 2011],[Kerraoui et al. 2016], [Winikoff 2017].
Aceitação	[Barnier et al. 2017]

A Figura 1 mostra um gráfico com a quantidade de publicações para cada nível de teste. O objetivo desse gráfico é facilitar a visualização da informação gerada por essa revisão. Os testes de unidade e de sistema foram utilizados em cinco artigos cada um,

os testes de agente e integração são dez em cada, já o teste de aceitação tem apenas um artigo. Nota-se que existe maior foco nos testes de agente e de integração. Como já foi comentado um mesmo artigo pode fazer uso de vários níveis o que fez com que o total de artigos selecionados não coincida com o total listado no gráfico.

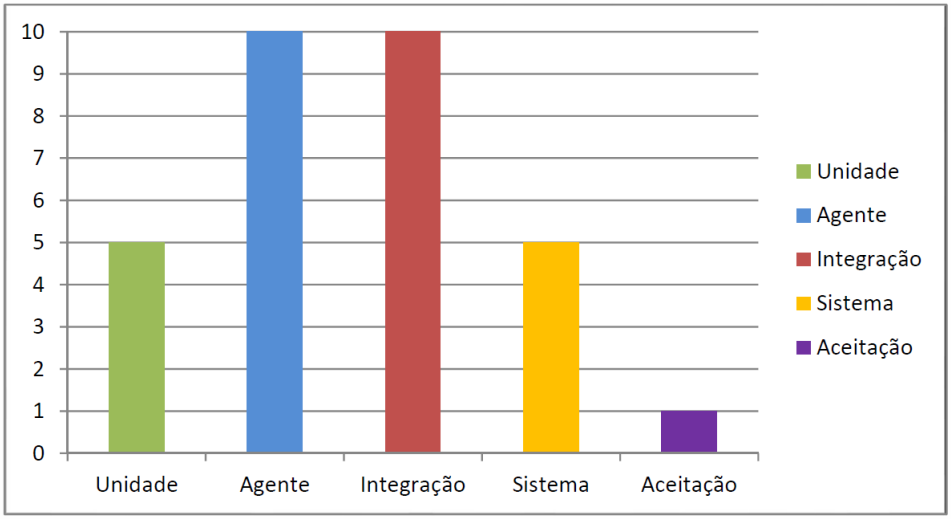

Figura 1. Níveis de Teste

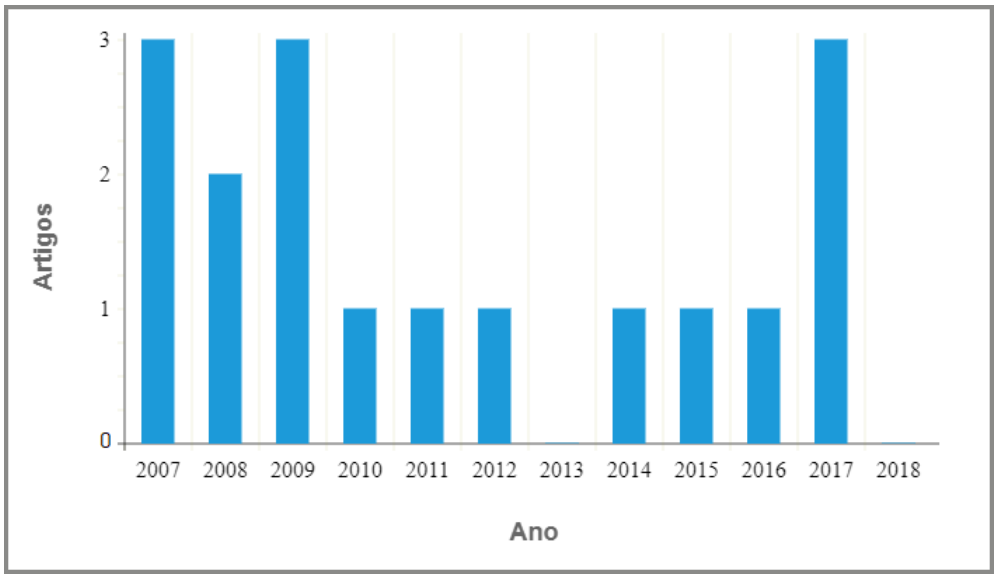

Figura 2. Linha do tempo

Na Figura 2 podemos visualizar a quantidade de artigos selecionados por ano de publicação, com o objetivo de avaliar a quantidade de contribuições geradas ao longo do tempo. Como é possível notar os três primeiros anos pesquisados tiveram maior número de resultados encontrados com oito artigos. Já entre 2010 a 2016 os valores se mantiveram baixos, tendo sido selecionados apenas sete artigos nesse período. No ano de 2017 houve um aumento para três novamente e em 2018, apesar de não ter encontrado nenhum artigo, como a pesquisa foi realizada em outubro ainda é possível que novas publicações sejam feitas.

3.3. Resumo dos Artigos

A seguir serão apresentados os artigos selecionados nessa revisão com uma breve descrição dos mesmos.

[Kissoum and Sahnoun 2007] descreve uma abordagem formal para especificar, desenvolver e testar SMA. Esta abordagem baseia-se na utilização da linguagem algébrica de especificação formal Maude [Clavel et al. 2002], que facilita as descrições e representações das relações entre os elementos de forma transparente no que diz respeito ao comportamento interno de cada classe de agentes. Também foi utilizado um conjunto de critérios de cobertura que podem orientar a geração de sequências de teste e calcular a eficiência da abordagem de teste. Vale ressaltar que os testes propostos no artigo estão restritos à interação do agente e seu comportamento interno.

[Coelho et al. 2007] apresenta um *framework* para teste de agentes desenvolvido na plataforma JADE. A ferramenta realiza os testes utilizando agentes falsos "*mock agents*", para testar outro agente através do monitoramento do comportamento dele e além disso é gerado um cenário de testes para definir uma série de condições onde o agente em teste será exposto. A ferramenta é capaz de realizar testes de unidade, integração e sistema.

[Nguyen et al. 2007] desenvolve uma ferramenta para gerar casos de teste de forma automática chamada eCAT em que são apresentadas duas técnicas. A primeira é randômica, onde um agente autônomo de teste é capaz de gerar casos de testes de forma aleatória utilizando troca de mensagens com o agente em teste enquanto agentes de monitoramento analisam as reações. A segunda chamada de mutação evolutiva é uma combinação do teste de mutação, onde os operadores de mutação são aplicados ao programa original para introduzir artificialmente defeitos conhecidos, com o teste evolucionário onde suítes de teste são desenvolvidas aplicando operadores de mutação nos casos de teste.

[Nguyen et al. 2008] define uma abordagem para geração de casos de teste baseado em ontologia para servir como guia na geração de testes utilizando o *framework* desenvolvido anteriormente pelo mesmo autor. Um conjunto de regras de geração foi definido e, usando-as, uma estrutura de teste automatizada pode testar extensivamente um determinado agente por meio de um grande e diverso número de casos de teste.

[Gomez-Sanz et al. 2008] apresenta avanços em testes e depuração feitos pela metodologia INGENIAS [Pavón et al. 2005]. Baseado numa abordagem direcionada por modelos ele possibilita especificar os casos de teste durante a modelagem do sistema. O modelo é então transformado num código onde os testes podem ser executados e o desenvolvedor coletar informações relativas tanto às interações entre os agentes como ao estado mental deles em tempo de execução. Esse trabalho foca no teste do agente e das interações entre eles.

[Salamon 2009] apresenta uma abordagem para teste de SMA com três camadas. Sendo a primeira camada responsável pela condução de teste de unidade em agentes. A segunda camada é referente aos testes das interações entre esses agentes onde erros como um *deadlock* podem ser detectados. Por fim a terceira camada constitui o teste do SMA como um todo onde gargalos no sistema ou outros problemas estruturais podem ser corrigidos.

[Poutakidis et al. 2009] apresenta um *framework* para teste e depuração de SMA baseado no uso de artefatos de design. Mais precisamente foi demonstrado dois conceitos, um gerando artefatos de design para gerar casos de teste em testes de unidade e o outro usando esses artefatos para auxiliar na depuração do código.

[Zhang et al. 2009] descreve uma abordagem para teste unitário baseado em planos de sistemas de agentes, com foco na geração automática de casos de teste. O *framework* se concentra em determinar a ordem na qual as unidades serão testadas, testar planos de agentes (unidades), incluir no código fonte do programa a ser testado mecanismos que possam gerar informações adicionais para o sistema de teste e por fim executar os casos de teste para coletar e analisar os resultados.

[Miller et al. 2010] define critérios de cobertura para testes de interações em SMA. Esses critérios de cobertura de testes tem como finalidade medir a qualidade dos casos de teste que serão executados. Dois tipos de critérios são apresentados sendo um baseado apenas na especificação de protocolos de mensagem e o outro que leva em consideração também os planos dos agentes para a troca dessas mensagens. O artigo provê uma base para futuras especificações de geração de casos de teste projetados para fornecer uma boa cobertura.

[Houhamdi and Athamena 2011] apresenta uma abordagem para testes estruturais em SMA especificando um processo de teste que complementa a metodologia orientada a metas chamada Tropos [Giunchiglia et al. 2002]. Nele é fornecida uma orientação sistemática para gerar conjuntos de testes a partir de artefatos de modelagem produzidos junto com o processo de desenvolvimento. Esses conjuntos de testes podem ser usados para refinar a análise de metas dos agentes e detectar problemas no início do processo de desenvolvimento.

[Wang and Zhu 2012] propôs um *framework* de automação de testes chamado CATest utilizando a linguagem de especificação formal baseada em agentes SLABS [Zhu 2001]. Durante a execução do programa de teste o comportamento dos agentes é monitorado e gravado. Então esses comportamentos gravados são checados através de um verificador de correção e um verificador de adequação para garantir que estejam funcionando de acordo com a especificação. Uma limitação é a inexistência de um gerador de casos de teste, eles precisam ser definidos manualmente pelo usuário, e a outra é que o sistema testa apenas o comportamento do agente individualmente.

[Eassa et al. 2014] desenvolveu ferramenta de teste dinâmico que usa uma linguagem de asserção, declarativa, de lógica temporal para detectar erros em tempo de execução dos agentes. A linguagem foi uma proposta do próprio autor e através das declarações inseridas a ferramenta pode identificar erros dinâmicos durante a execução do programa de teste. Essa ferramenta gera agentes de teste para monitorar, controlar e gerar os testes que podem ser tanto a nível de agente quanto de integração.

[Ur Rehman and Nadeem 2015] apresenta uma abordagem para teste em SMA baseado em design de artefatos da ferramenta Prometheus [Padgham et al. 2008]. A ideia é gerar através de um diagrama de protocolos um gráfico de protocolos. Os dados deste gráfico juntamente com critérios de cobertura pré definidos servem de entrada para gerar caminhos de teste que cubram as interações entre os agentes. Esses caminhos de teste podem ser usados num trabalho futuro para uma geração automática de casos de teste.

[Kerraoui et al. 2016] propôs uma abordagem de teste em SMA baseada em Modelos. Primeiramente é gerado e validado um modelo que represente o comportamento do sistema através de uma rede de petri. Em seguida os casos de teste são gerados automaticamente tendo como entrada o modelo comportamental do sistema. Por fim uma versão instrumentada do modelo é gerada para a execução dos casos de teste e geração dos resultados. O modelo abrange os níveis de teste de agente, integração e sistema e uma das limitações é que o sistema consegue realizar apenas testes funcionais.

[do Nascimento et al. 2017] modelou e desenvolveu uma arquitetura baseada em publicação de assinaturas para facilitar a implementação de sistemas que testem os SMA nos níveis de agente e grupo. O autor utilizou uma plataforma chamada RabbitMQ [Richardson et al. 2012] para entregar os *logs* "publicações", dos agentes para serem utilizados por aplicações de teste "assinantes". Usando máquinas de estado os aplicativos de teste puderam validar esses casos de teste comparando os *logs* consumidos do editor MAS com os *logs* listados para validação. No final o desenvolvedor pode usar a interface para identificar a falha e reduzir o tempo de diagnóstico. Uma limitação citada é a falta de recursos para lidar com as características não-determinísticas dos sistemas multiagentes.

[Winikoff 2017] com foco em programas de agentes BDI o autor analisa sua testabilidade em relação ao critério de adequação de testes de todas as arestas "*all edges*". Para isso eles fazem uma comparação de quantidades de testes necessário entre os critérios de todas as arestas e todos os caminhos "*all paths*"e mostram que o número de testes necessários para cobrir todas as arestas é bem menor do que todos os caminhos. Esse artigo também faz uma comparação para mostrar o quanto programas BDI são mais difíceis de testar do que programas processuais de tamanho equivalente. Portanto ele realiza uma avaliação de certos critérios para que possam ser analisados e utilizados em futuras aplicações. Segundo o autor as comparações que foram feitas podem ser representadas a nível de agente, partes de um agente (unidade) ou o sistema todo.

[Barnier et al. 2017] apresenta uma nova estratégia para testes em sistemas multiagentes embarcados. Sua metodologia foca em três itens principais que são: teste de agente, teste de recursos coletivos e teste de aceitação. Em cada item o autor define quais as metas que devem ser alcançadas para que o teste seja bem sucedido. Como por exemplo o teste de agente tem como metas os testes de software, hardware e integração do agente assim como a análise do tempo de resposta do mesmo.

4. Conclusão

Esse artigo apresentou como revisão uma série de abordagens que foram desenvolvidas nos últimos anos que tivessem como foco principal desenvolver uma solução, seja ela um modelo, ferramenta ou método, para testar um agente ou um SMA completo. Os resultados da busca foram classificados em diferentes níveis utilizando uma classificação existente que se mostrou válida e atualizada. Com esses resultados foi possível analisar as evoluções feitas na área de testes durante a última década.

As perguntas de pesquisa que foram apresentadas no capítulo 2 serão respondidas a seguir com base no conteúdo dos artigos selecionados:

1. **Quais as dificuldades para testar um SMA?** Como visto em [Houhamdi 2011] devido ao seu comportamento não-determinístico os agentes possuem uma complexidade maior do que um software tradicional. Isso significa que suas ações

são pouco previsíveis e para gerar e executar todos os testes necessários é preciso prever todos os caminhos de teste possíveis. Além disso segundo [Barnier et al. 2017] testar um SMA implica não só em monitorar o comportamento individual do agente, mas também a interação entre os agentes e o sistema global precisam ser testados. Por isso testar esses sistemas requer novas técnicas de teste que lidem com sua natureza específica.

2. **Que níveis de teste podem ser realizados em um SMA?** Existem diferentes níveis em que um teste de software pode ser empregado. No caso dos SMA a maioria dos autores classificam esses níveis em 5. Como foi descrito no item 3.1, de acordo com [Nguyen 2009] esses níveis são o teste de unidade, agente, integração, sistema e aceitação. Essa classificação pôde ser confirmada e utilizada nos artigos aqui selecionados. Cada um desses níveis representa uma etapa no desenvolvimento do sistema, e para que o sistema possa ser considerado confiável, os testes em todos esses níveis precisam ser realizados
3. **Quais são as principais técnicas para geração de casos de teste em SMA?** Praticamente todos os artigos utilizam como técnica a geração automática de casos de teste através de *frameworks*, isso se deve pelo fato que como os agentes tem um comportamento imprevisível o número de casos de teste possíveis num SMA é muito grande para que seja feito manualmente o que consumiria muito tempo.

Apesar de ser essencial realizar testes em um SMA para garantir seu perfeito funcionamento, ainda existem muitas dificuldades que precisam ser superadas. Além disso o número de artigos publicados nos últimos tempos tem se mantido baixo o que faz com que essa seja uma área de conhecimento que ainda pode ser bastante explorada.

Referências

Barnier, C., Mercier, A., Jamont, J.-P., et al. (2017). Toward an embedded multi-agent system methodology and positioning on testing. In *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 239–244. IEEE.

Clavel, M., Durán, F., Eker, S., Lincoln, P., Martı-Oliet, N., Meseguer, J., and Quesada, J. F. (2002). Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 285(2):187–243.

Coelho, R., Cirilo, E., Kulesza, U., von Staa, A., Rashid, A., and Lucena, C. (2007). Jat: A test automation framework for multi-agent systems. In *Software Maintenance, 2007. ICSM 2007. IEEE International Conference on*, pages 425–434. IEEE.

do Nascimento, N. M., Viana, C. J. M., von Staa, A., and Lucena, C. (2017). A publish-subscribe based architecture for testing multiagent systems. In *SEKE*, pages 521–526.

Eassa, F. E., Osterweil, L. J., Fadel, M. A., Sandokji, S., and Ezz, A. (2014). Dttas: A dynamic testing tool for agent-based systems. *Pensee Journal*, 76(5).

Giunchiglia, F., Mylopoulos, J., and Perini, A. (2002). The tropos software development methodology: processes, models and diagrams. In *International Workshop on Agent-Oriented Software Engineering*, pages 162–173. Springer.

Gomez-Sanz, J. J., Botía, J., Serrano, E., and Pavón, J. (2008). Testing and debugging of mas interactions with ingenias. In *International Workshop on Agent-Oriented Software Engineering*, pages 199–212. Springer.

Houhamdi, Z. (2011). Multi-agent system testing: A survey. *International Journal of Advanced Computer*.

Houhamdi, Z. and Athamena, B. (2011). Structured system test suite generation process for multi-agent system. *International Journal on Computer Science and Engineering*, 3(4):1681–1688.

Kerraoui, S., Kissoum, Y., Redjimi, M., and Saker, M. (2016). Matt: Multi agents testing tool based nets within nets. *Journal of Information and Organizational Sciences*, 40(2):165–184.

Kissoum, Y. and Sahnoun, Z. (2007). Test cases generation for multi-agent systems using formal specification. *Computer Systems and Applications*, pages 76–83.

Miller, T., Padgham, L., and Thangarajah, J. (2010). Test coverage criteria for agent interaction testing. In *International Workshop on Agent-Oriented Software Engineering*, pages 91–105. Springer.

Nguyen, C. D., Perini, A., and Tonella, P. (2008). Ontology-based test generation for multiagent systems. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1315–1320. International Foundation for Autonomous Agents and Multiagent Systems.

Nguyen, C. D., Perini, A., Tonella, P., and Kessler, F. B. (2007). Automated continuous testing of multi-agent systems. In *The fifth European workshop on Multi-agent systems*. Citeseer.

Nguyen, D. C. (2009). *Testing techniques for software agents*. PhD thesis, University of Trento.

Padgham, L., Thangarajah, J., and Winikoff, M. (2008). Prometheus design tool. In *AAAI 2008*. AAAI Press.

Pavón, J., Gómez-Sanz, J. J., and Fuentes, R. (2005). The ingenias methodology and tools. In *Agent-oriented methodologies*, pages 236–276. IGI Global.

Poutakidis, D., Winikoff, M., Padgham, L., and Zhang, Z. (2009). Debugging and testing of multi-agent systems using design artefacts. In *Multi-Agent Programming:*, pages 215–258. Springer.

Richardson, A. et al. (2012). Introduction to rabbitmq. *Google UK, available at http://www. rabbitmq. com/resources/google-tech-talk-final/alexis-google-rabbitmq-talk. pdf, retrieved on Mar*, 30:33.

Salamon, T. (2009). A three-layer approach to testing of multi-agent systems. In *Information Systems Development*, pages 393–401. Springer.

Ur Rehman, S. and Nadeem, A. (2015). An approach to model based testing of multiagent systems. *The Scientific World Journal*, 2015.

Wang, S. and Zhu, H. (2012). Catest: a test automation framework for multi-agent systems. In *Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual*, pages 148–157. IEEE.

Winikoff, M. (2017). Bdi agent testability revisited. *Autonomous Agents and Multi-Agent Systems*, 31(5):1094–1132.

Zhang, Z., Thangarajah, J., and Padgham, L. (2009). Automated testing for intelligent agent systems. In *International Workshop on Agent-Oriented Software Engineering*, pages 66–79. Springer.

Zhu, H. (2001). Slabs: A formal specification language for agent-based systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(05):529–558.