# Uma análise comparativa da especificação formal em sistemas multi-agente: os desafios e as exigências uma década depois.

**Carlos Eduardo Pereira Quadros[1], Jader de Freitas Saldanha[1], Narusci Bastos[1], Newton Nyamasege Marube[1], Eder Mateus Gonçalves [1]**

[1] Centro de Ciências Computacionais– Universidade Federal do Rio Grande – FURG, Av. Itália km 8, Bairro Carreiros – 96.203-900 – Rio Grande – RS – Brazil

{cep.quadros,saldanha.jader,naruscibastos,nyamasege,eder.m.goncalves}@gmail.com

***Abstract.*** *Developed from mathematical principles, formal methods are used for computer systems development, giving priority to cohesion. They are an active area of research in multi-agent systems. The present comparative analysis aims to evaluate the adoption of formal specification on a multi-agents system point of view. It revisits a formal specification roadmap written more than a decade ago, highlighting the requirements and challenges proposed at the time, and, comparing these with recent selected research on multi-agent systems from popular academic databases to see how some of the features cited in the roadmap have been addressed. The difference in the nature of the compared works explains the presence or absence of some requirements. It is concluded that formal methods in software design are more evident in critical system development where emergent system properties such as safety, reliability and security are of paramount importance.*

***Resumo.*** *Desenvolvido a partir de princípios matemáticos, métodos formais são usados para o desenvolvimento de sistemas computacionais, dando prioridade à coesão. Eles são uma área de pesquisa ativa em sistemas multi-agente. A presente análise comparativa tem como objetivo avaliar a adoção de especificação formal em ponto de vista de sistemas multi-agente. O trabalho faz visita a um roteiro de especificação formal escrito há mais de uma década atrás, com destaque para as exigências e desafios propostos então, e, comparando estes com pesquisas recentes sobre especificação formal em sistemas multi-agente selecionados e analisados a partir de bancos de dados acadêmicos populares para ver como foram abordadas. A diferença na natureza das obras em analise explica a presença ou ausência de alguns requisitos. Conclui-se que métodos formais em design de software têm sido mais evidentes no desenvolvimento de sistemas críticos, onde as propriedades dos sistemas emergentes, como segurança, confiabilidade e segurança são de suma importância.*

## 1. Introdução

Os agentes, estão sendo utilizados cada vez mais no sentido de contribuir para diminuir as demandas tecnológicas. Pois estes, são mecanismos capazes de tomar decisões autônomas e interagem entre si e com o meio ambiente, Cunha (2014). Cunha afirma

que com o crescimento deste campo de estudo, surgiram necessidades, como a especificação formal desses sistemas.

Especificação formal, conforme Lamsweerde (2000), em termos gerais diz respeito a expressão, em alguma linguagem formal e em algum nível de abstração, de uma coleção de propriedades que um sistema deve satisfazer. Para isso, segundo Gularte et. al (2013), a especificação formal só pode ser considerada formal, se satisfaz os requisitos do formalismo, que para o autor deve ser expressa em uma linguagem composta com três elementos: sintaxe, semântica e teoria de prova.

Além dos requisitos dos formalismos, existem as métricas do formalismo, para avaliar o mesmo. Estas segundo Gularte *et. al* (2013) são: expressividade, construtibilidade, usabilidade e comunicabilidade. Além das métricas destacadas no trabalho de Gularte et. al (2013), Lamsweerde (2000) propôs dezesseis requisitos e/ou desafios que a tecnologia deve cumprir na especificação formal para se tornar um meio essencial para a engenharia ou reengenharia de software de alta qualidade. Os requisitos são: Construtibilidade, suporte a análise comparativa, integração, maior nível de abstração, mecanismos estruturantes com mais recursos, escopo estendido, separação de interesses, técnicas leves, especificação multiparadigma, análise polivalente, especificação multiformato, raciocínio apesar de erros, feedback construtivo a partir de ferramentas, suporte a evolução, suporte a reuso e mensurabilidade de progresso.

O presente trabalho tem como objetivo analisar os requisitos e desafios da especificação formal propostos por Lamsweerde (2000) para o futuro, buscando detectar os mesmos em estudos atuais.

A seção 2 trata de trabalhos relacionados, logo são abordados os materiais e métodos utilizados para o desenvolvimento deste trabalho, e a seção 4 busca discutir e analisar os requisitos nos trabalhos utilizados para análise e a seção 5 traz a conclusão.

## 2. Trabalhos Relacionados

O trabalho com especificação formal para sistemas multi-agente requer um *background* mínimo para desenvolver qualquer tipo de análise. Os artigos citados abaixo, investigados durante o decorrer da disciplina de Especificação formal para sistemas multi-agente forneceram uma base de discussão suficiente para propor a escrita da análise sobre requisitos que deveriam ser contemplados no futuro segundo Lamsweerde (2000).

Três áreas básicas são necessárias para entendimento de especificação formal de sistemas multiagente, teoria dos agentes, arquitetura e linguagens. O trabalho de Wooldridge (1994) contempla essas três áreas em uma pesquisa.

Há na literatura uma quantidade significativa de trabalhos que utilizam redes de Petri para modelar suas metodologias em especificações formais para sistemas multiagente, como exemplo em Gularte (2013) e Chang (2011).

A modelagem utilizando Agent-Oriented Software Engineering (AOSE) tem ganhado certa atenção nos últimos anos. Segundo Vicari (2012) a área mescla conceitos tanto da Inteligência Artificial como os da Engenharia de Software. O trabalho de Fuentes (1999) define dois pontos principais: a) as estruturas conceituais atingiram um nível de maturidade que torna razoável dedicar esforço para buscar um consenso em linguagens de modelagem, incluindo suporte de ferramenta. b) a influência de engenharia orientada a modelo enfatiza o valor potencial de ter modelos no cerne dos processos de desenvolvimento.

O trabalho de Horling (2004) apresenta um levantamento dos principais paradigmas organizacionais utilizados em sistemas multiagente, entre eles: hierarquias,

oligarquias, coalizões, equipes, congregações, sociedades, federações, mercados e organizações matriciais.

## 3. Materiais e Métodos

Para o desenvolvimento deste trabalho foram selecionadas as palavras chaves dos artigos estudados em durante a disciplina, posteriormente foi feito o ranking destas palavras selecionando as que tiveram maior incidência, que foram: especification language; semantics; formal specification; knowledge-based systems; agent-oriented modeling.

A partir da seleção das palavras chave, através dos motores de busca IEEE e SCHOLAR, foi feita a busca de artigos para compor a análise deste trabalho. Foram extraídos e analisados trabalhos no período de 2001 até 2015.

No primeiro quinquênio foram encontrados os seguintes trabalhos: Wood (2001), Aguirre (2001), Esteva (2001), Dumas (2001), Armoni (2002), Karsai (2003), Gruninger (2003), Lötzsch (2003), e Pustejovsky (2005).

No segundo quinquênio encontramos os trabalhos de: Che (2006), Monostori (2006), Lin (2006), Medina (2007), Bosse (2007), Wang (2007), Cabral (2008), Pagliarecci (2008), Zhao (2010) e Wang (2010).

Por fim, no último quinquênio, foram encontrados os seguintes trabalhos: Urban (2011), Ramzan (2011), Saburraj (2013), Yuan (2013), Lin (2014), Hussain (2014), Tao (2015), Van (2015), Ormandjieva (2015),

Com base no que foi encontrado foram escolhidos artigos mais atuais no período compreendido entre 2013 até 2015. A investigação baseia-se na presença ou não dos itens descritos e previstos por Lamsweerde (2000), que seguem: Construtibilidade, Suporte a análise comparativa, Integração, Maior nível de abstração, Mecanismos estruturantes com mais recursos, Escopo estendido, Separação de interesses, Técnicas leves, Especificação Multiparadigma, Análise polivalente, Especificação multiformato, Raciocínio apesar de erros, Feedback construtivo a partir de ferramentas, Suporte a evolução, Suporte a reúso, Mensurabilidade de progresso.

## 4. Análise e discussão

Os requisitos e desafios para a especificação formal propostos por Lamsweerde (2000) são detalhados abaixo seguido por tabela 1, que mostrar um resumo dos itens analisados por artigo.

### a) Construtividade

Em seu artigo, Lamsweerde (2000) afirma que as especificações são construídas de forma incremental desde as de nível superior de uma forma que garante alta qualidade de construção. Só então se poderia realmente falar de um método, normalmente feita de um conjunto de estratégias de construção de modelos, regras de seleção de estilo, regras de especificação de derivação, diretrizes e heurística. Um exemplo disto é visto em Van Nguyen et al (2015), que explorou a linguagem de especificação TESL e apesar dos artefatos semânticas limitados, eles foram capazes de permitir a especificação de sistemas cronometrados cujos componentes obedecem a padrões de sincronização complexos, adotando técnicas construtivas no processo. Em Tao et al (2015), os autores desenvolvem um quadro que permite semântica operacional Kripke que pode ajudar a converter quadro da política de obrigação do modelo de entrada ao modelo verificador MCMAS (Multi-Agent System Model Checker), ao mesmo tempo, as propriedades do

framework dependente de políticas de conflitos são representados com CTL (Computational Tree Logic), e as violações de propriedades são detectada usando MCMAS que pode fornecer o contra-exemplo e relacioná-las com os erros na política de interação. Hussain et al (2014) descrevem uma nova lógica espaço-temporal probabilística que combina a noção de tempo linear com espaço-tempo Minkowski em uma estrutura probabilística. Eles provam que, apesar de validação e verificação serem vitais, especialmente em epidemiologia, a construtividade atingido torna mais fácil desenvolver software apropriado.

**b) Suporte a análise comparativa**

De acordo com Lamsweerde (2000), o suporte para análise comparativa refere-se a ausência de parâmetros comparativos. O mesmo problema acontece nos programas, porém, esse caso é resolvido na execução do programa se este faz ou não o que deveria.

O trabalho de Brazier et al. (1997) descreve um framework chamado DESIRE, para especificar um sistema multiagente operacional. O suporte da ferramenta implementado neste esforço de pesquisa permite a execução da especificação formal de sistemas de agente. A análise da correção sintática e semântica das especificações formais do agente foram atingidos por conta do apoio fornecido pela ferramenta implementada em Java. A sobrecarga de validar manualmente a correção da especificação foi eliminada neste esforço de investigação através do apoio ferramenta desenvolvida. O trabalho sugere que a validação do modelo deve ser tratada como uma parte fundamental do processo de desenvolvimento e uso. Por conseguinte, deve ser realizada periodicamente a verificação durante estas fases. Tal validação contínua dos modelos epidemiológicos contra observações em tempo real ajuda a manter a confiança do usuário na análise realizada utilizando esses modelos. A pesquisa de Tao et. al (2015) utiliza MCMAS (model checker) para verificar se o modelo OPL (Obligation Policy Language) atende a propriedade representado pela fórmula CTL (computational tree logic), e apresenta um estudo de caso real para demonstrar a eficácia e aplicabilidade da nossa abordagem.

**c) Integração**

Segundo Lamsweerde (2000) a integração refere-se a tecnologia do amanhã, da qual deve ter a atenção vertical e horizontal de especificações durante o ciclo de software - de objetivos de alto nível ao design funcional, a componentes arquiteturais e de formulação informal para especificações formais aos produtos relacionados. Em estudos de Subburaj et. al (2013) a integração é explorada da seguinte maneira: As extensões feitas na linguagem seguem um desenvolvimento modular top-down permitindo o desenvolvimento de decomposição e incrementação de grandes sistemas de agentes. Além disso, seis conceitos foram adicionados a linguagem Descartes para especificar e validar sistemas de software de agentes. Dos quais são: construtor de agente, objetivo de agente, atributos de agentes, papéis de agentes, planos de agentes e protocolo de comunicação. Observa-se uma atenção dos autores da utilização da linguagem no ciclo de desenvolvimento de um sistema, contemplando o tópico apresentado. Em Hussain et. al (2014) os autores sugerem que a validação de modelos deve ser tratada como uma parte fundamental do modelo de desenvolvimento e processo de uso, além disso, dizem que deve ser efetuada periodicamente durante essas fases. Sendo assim, a linguagem desenvolvida permite a verificação e validação de modelos epidemiológicos propostos.

Em Tao et. al (2015) não está claro como o método formal de verificação criado pode ser integrado durante o ciclo de desenvolvimento de software.

**d) Maior nível de abstração**

Para Lamsweerde (2000) técnicas de especificação devem ir de um projeto funcional a engenharia de requisitos, em que os impactos dos erros são ainda mais cruciais. Tendo a necessidade, assim, de idiomas, métodos e ferramentas que suportem ontologias mais ricas orientadas para o problema. Tao et. al (2015) utiliza o conceito de compromisso social para definir a obrigação da semântica formal na política de interações de SMA. As obrigações são ações que os agentes são obrigados a tomar ou algum estado de coisas que devem ser mantidas, modelagem formal, obrigações de requisitos de alto nível, protocolo de comunicação para restringir a interação do agente pode melhorar a exatidão do projeto do sistema. Hussain et. al (2014) utilizou a verificação formal para analisar modelos epidemiológicos, as técnicas para podem resolver tanto a calibração de modelos raros como a descoberta de problemas no comportamento que envolvem o controle estatístico do modelo e o uso de solucionadores de restrição poderosos. Pode ser traduzido qualquer conjunto de especificação de comportamento escrito em EpiSpec em uma forma que pode ser verificada durante a simulação do modelo usando algoritmos de monitoramento. Estes métodos baseados em técnicas formais, combinados com o trabalho em algoritmos para lidar com dados eXtremeScale pode ser utilizado para criar ferramentas poderosas para solucionar problemas. Entende-se que as questões colocadas nos trabalhos de Tao et. al (2015), e Hussain et. al (2014) apresentam ferramentas que suportam ontologias mais ricas orientadas para resolver problemas.

**e) Mecanismos estruturantes com mais recursos**

Na época da publicação de Lamsweerde (2000), a maioria das construções disponíveis para modularização de grandes especificações teriam sido tirado de outras formas de programação. Segundo o autor, as construções orientadas a programas devem estar disponíveis também. Em seu trabalho, Hussain et al. (2014) fazem uso ás novas construções para apresentar uma linguagem de especificação formal EpicSpec, para modelos baseados em agentes parametrizados contra epidemiologias reais. Os autores usam técnicas como a verificação de modelo estatístico para fazer verificação formal.

**f) Escopo estendido**

Escopo estendido, segundo autor do trabalho base, define que as especificações devem ter mais categorias de propriedades não-funcionais que são desencadeadas durante a engenharia de requisitos.

O trabalho de Brazier et al. (1997) descreve um framework chamado DESIRE. No trabalho, seis conceitos foram adicionados à linguagem de especificação de Descartes para especificar e validar sistemas de software do agente. Os conceitos adicionados são: (1) Agente de construir; (2) objetivo agente; (3) atribui agente; (4) funções de agentes; (5) planeja agente; e (6) o protocolo de comunicação.

**g) Separação de interesses**

Como discutido em Lamsweerde (2000), linguagens de especificações formais devem estritamente focar na separação entre propriedades descritivas e prescritivas, das quais devem ser exploradas por ferramentas de análise de conformidade. Nos trabalhos analisados não ficou clara a abordagem destes conceitos como sugerido por Lamsweerde (2000).

**h) Técnicas leves**

O uso de especificações formais, de acordo com Lamsweerde(2000) não deve exigir conhecimento avançado em sistemas formais. As complexidades matemáticas devem ser escondidas; ferramentas de análise devem poder ser utilizadas em compiladores. O trabalho de Subburaj et. al (2013) descreve um sistema orientado a agente em conjunto com a linguagem de especificação Descartes voltada para sistemas de agentes definidos pelos autores. Para os autores a integração de sistemas complexos que especificam formalmente e utilizam a tecnologia orientada a agente com a construção de linguagem utilizando o Descartes terá um impacto sobre as formas atuais de especificação de sistemas de software. No que diz respeito aos requisitos propostos por Lamsweerde (2000) o estudo de Subburaj et. Al (2013) a linguagem de especificação Descartes permite que a especificação seja facilmente compreendida e construída, mesmo quando os sistemas especificados sejam de agentes complexos. O idioma original para a especificação faz uso de semântica formal, porém simples e poderosa. A definição da linguagem Descartes consiste em uma definição sintática formal e uma notação semântica formal em que ambas não são complexas, o agente age de forma autônoma para realizar seus objetivos baseado no conjunto de regras definidas na base de conhecimento/crença.

Já no trabalho de Tao et. al (2015) a linguagem de obrigação pode especificar um grande número de exigências práticas da vida real e é simples ao ponto de permitir que não especialistas possam compreendê-los e usá-los, isso devido ao fato do modelo estar ligado ao OPL, que é um modelo baseado no estado da obrigação, que esclarece a semântica da obrigação, identificando os diferentes estados e transições. Porém, tanto o trabalho de Subburaj et. Al (2013) como o de Tao et. al (2015) apresentam linguagem que não exigem que o usuário seja especialista para compreender a especificação, por esse motivo os trabalhos apresentados pelos autores atendem a esse requisito.

**i) Especificação Multiparadigma**

Uma vez que nenhum único paradigma vai servir todos os efeitos, devido a preconceitos semânticos, um framework multi-paradigma permite que frameworks diferentes integrem várias linguagens formais, semi-formais e linguagem natural, juntamente com técnicas e ferramentas de análise [Lamsweerde, 2000]. Em comparação com os requisitos sugeridos pelo autor supracitado, sobre os futuros desafios na especificação formal, o progresso tem sido feito para desenvolver frameworks multiparadigma nos esforços de especificação. Van et Al (2015). Relacionar multiparadigms para problemas heterogêneos quando combinado com problemas de modelagem industriais, mostrando que é um requisito fundamental na concepção de sistemas multi-agente. As outras obras nesta análise comparativa pouco fizeram para mostrar especificação multiparadigma em seus trabalhos, sugerindo que ainda é uma questão a ser abordada.

**j) Análise polivalente**

Um quadro de suporte multiparadigma deve apoiar diferentes níveis de análise, da mais simples até a mais complexa. Em sistema multiagente, as obrigações são ações que os agentes são obrigados a tomar ou alguns estados de coisas que devem ser mantidos. Neste sentido, o artigo de Tao et. al (2015), apresenta um quadro formal para a modelagem da política de obrigação. O quadro é formalizado utilizando conceitos de compromissos sociais. A linguagem de política de obrigação pode especificar um grande número de requisitos e práticas simples da vida real para permitir que não-especialistas possam compreendê-lo e usá-lo. O modelo OPL (Obligation Policy

Language) está associado a um modelo baseado no estado de obrigação que esclarece a semântica da obrigação, identificando os diferentes estados, obrigação e transições de estado. O modelo OPL usa o conceito de contexto para representar as condições de interação dos agentes. Além disso, define a semântica operacional do modelo OPL que pode ajudar a converter o quadro da política de obrigação para o modelo de entrada do model checker MCMAS (model checker for multi-agent systems).

**k) Especificação multiformato**

Lamsweerde (2000) define a comunicabilidade do fragmento de especificação como melhor forma que seja mantida sobre várias sintaxes concretas como: tabulares, diagramas e textuais. Dessa forma atingiria diferentes produtores/consumidores. (Subburaj, Urban, 2013) A linguagem Descartes consiste em uma definição de sintaxe formal e uma notação semantica formal das quais não são complexas, também suporta a especificação de sistemas multiagentes pela interação entre os agentes. Não possui variadas sintaxes, como sugerida por Lamsweerde (2000). A linguagem de especificação EpiSpec é definida em uma estrutura probabilistica spatio-temporal, somente (Hussain et. al 2014). Os autores não apresentam outros tipos de representação. Tao et. al (2015) define como uma linguagem de sistema de transição de estados, não apresentando outras formas de reprentação.

**l) Raciocínio apesar de erros**

Grande número de técnicas de especificação formal exige que a especificação esteja completa antes mesmo da análise iniciar. Para Lamsweerde (2000) deve ser possível iniciar a análise sobre os projetos de especificação, mais cedo e de forma incremental. No trabalho apresentado por Subburaj et. al (2013) as regras de contexto são definidas na base de conhecimento/crença, fazendo o uso apenas de primitivos lógicos no Descartes. As primitivas lógicas recém adicionadas não só permitem que o usuário determine o valor verdadeiro ou falso para os argumentos, mas também tomar decisões lógicas baseadas em crenças dos agentes. Essas primitivas podem ser usados para definir as regras de contexto na base de dados de conhecimento/crença , sendo satisfeito antes que a execução da especificação comece, a fim de ser como uma prova de correção.

**m) Feedback construtivo a partir de ferramentas**

Em seu artigo, Lamsweerde (2000) sugere que em vez de apenas apontar problemas; futuras ferramentas devem ajudar a resolvê-los. Este desafio parece não ter sido atendido década depois. As pesquisas analisadas no presente estudo não apresentaram quaisquer formas de resolver problemas que surgem durante o processo de especificação-verificação. Isso mostra que mesmo com o design de sistemas multiagentes amadurecendo nos últimos anos, ferramentas de feedback estão ainda a acompanhar esta tendência.

**n) Suporte a evolução**

Duas questões poderiam responder se existe ou não o item nos demais trabalhos. Questão 1: Há aprendizado? Questão 2: Tem arquitetura central estável para manter o resto do sistema? Diante disso, o trabalho de Brazier et al. (1997) apresenta um framework chamado DESIRE, onde, cada agente em um sistema de agentes interage com a base de conhecimento para ler conjunto inicial do agente de crenças e tomar medidas em conformidade. Diferente de um framework, Tao et. al (2015) apresenta um

verificador com funções de evolução. MCMAS é um verificador de modelos para sistemas multiagente com ISPL (Interpreted Systems Programming Language), que nos permite modelar SMA. No modelo ISPL, o SMA é distinto em agente ambiente e agentes padrão. Agente ambiente é utilizado para descrever condições de contorno, infra-estruturas e as variáveis de observação compartilhadas por agentes padrão. Os agentes são modelados como autômato não determinístico na forma de um conjunto de estados locais, um conjunto de ações, funções de protocolo e funções de evolução

**o) Suporte ao reuso**

Lamsweerde (2000) apresenta o suporte ao reuso de especificações formais como forma a de fato reaproveitar solução a diferentes domínios de sistemas similares, assim semelhante ao reuso de código. Subburaj et. al (2013) não abordado no trabalho. Hussain et. al (2014) não abordado. Tao et. al (2015) não abordado.

**p) Mensurabilidade de progresso**

O benefício de usar especificações formais em engenharia de software deve ser possível devido a métricas semelhantes utilizadas para a medição do aumento de produtividade de software. Subburaj et. al (2013) a especificação Descartes a cerca de sistemas de agentes foi desenvolvido para preservar a facilidade de compreensão e construtibilidade das características da linguagem de especificação descartes. Este esforço de investigação permitiu a validação da declaração: análise inicial de especificação de correção pode reduzir o custo do desenvolvimento de software através da detecção de erros durante a fase inicial de desenvolvimento do software. A identificação desta propriedade do agente durante o desenvolvimento precoce do sistema de agente reduz o custo de desenvolvimento de software envolvido no desenvolvimento do agente, resultando no desenvolvimento de um sistema de especificação formal para agente completo e fiável. Hussain et. al (2014) sugere o uso de EpiSpec para escrever propriedades epidemiológicas que os estudiosos desejam investigar. Com a disponibilidade de arquitetura de computação de alto desempenho, um modelo de rápida verificação de algoritmos e ferramentas que permitem sua implementação, os modelos epidemiológicos podem ser analisados em tempo real para responder a emergências como encontrar estratégias de contenção de pandemias.

**Tabela 1. Incidência (▓) ou não (▒), das características em avaliação dos artigos selecionados para o estudo**

| | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Lamsweerde (2000)* | ▓ | ▓ | ▓ | ▓ | ▓ | ▓ | ▓ | ▓ | ▓ | ▓ | ▓ | ▓ | ▓ | ▓ | ▓ | ▓ |
| *Subburaj et. al (2013)* | ▓ | ▓ | ▓ | ▒ | ▒ | ▓ | ▒ | ▓ | ▓ | ▒ | ▒ | ▓ | ▒ | ▓ | ▒ | ▓ |
| Hussain et. al (2014) | ▓ | ▒ | ▓ | ▓ | ▓ | ▒ | ▒ | ▓ | ▓ | ▒ | ▒ | ▒ | ▓ | ▒ | ▒ | ▓ |
| *Tao et. al (2015)* | ▓ | ▓ | ▒ | ▓ | ▒ | ▒ | ▒ | ▓ | ▒ | ▓ | ▒ | ▒ | ▒ | ▓ | ▒ | ▒ |

## 5. Conclusão

Métodos formais são usados no desenvolvimento de sistemas e são uma área ativa de pesquisa em sistemas multi-agente, este artigo apresentou uma análise comparativa que avaliou a adoção de especificações formais em sistemas multiagente, contrapondo um *roadmap* escrito há uma década, analisando o que de fato foi adotado em trabalhos atuais da academia. Com esta analise podemos concluir que ainda há incipiência por parte de algumas pesquisas que elucidam sistemas multiagente no tange sua globalidade no desenvolvimento de software, este trabalho serve como pontapé inicial para pesquisas que tratam destes dois universos.

Para trabalhos futuros os autores ampliarão as buscas para incluir mais motores de buscar além de incluir trabalhos considerados de mais impacto na área de especificação formal em sistemas multiagentes além de incluir trabalhos em periódicos.

## Referencias

Aguirre, J. L., Brena, R., & Cantu, F. J. (2001). Multiagent-based knowledge networks. Expert Systems with applications, 20(1), 65-75.

Armoni, R., Fix, L., Flaisher, A., Gerth, R., Ginsburg, B., Kanza, T., ... & Vardi, M. (2002). The ForSpec temporal logic: A new temporal property-specification language. In Tools and Algorithms for the Construction and Analysis of Systems (pp. 296-311). Springer Berlin Heidelberg.

Bosse, T., Hoogendoorn, M., Serban, R., & Treur, J. (2007). A Specification Language for Coordination in Agent Systems. In Intelligent Agent Technology, 2007. IAT'07. IEEE/WIC/ACM International Conference on (pp. 252-256). IEEE.

Brazier, F. M. T., Dunin-Keplicz, B. M., Jennings, N. R., & Treur, J. (1997). Desire: Modelling multi-agent systems in a compositional formal framework.*Int. Journal of Cooperative Information Systems*, *6*(1), 67-94.

Cabral, G., & Sampaio, A. (2008). Formal specification generation from requirement documents. Electronic Notes in Theoretical Computer Science,195, 171-188.

Chang, L., He, X., & Shatz, S. M. (2012). A methodology for modeling multi-agent systems using nested Petri nets. International Journal of Software Engineering and Knowledge Engineering, 22(07), 891-925.

Che, H. Y., Sun, J. G., & Yu, H. B. (2006). A Description Logic Method of Formalizing the Specification of Multi-Agent System. In Machine Learning and Cybernetics, 2006 International Conference on (pp. 61-65). IEEE.

Dumas, M., & Ter Hofstede, A. H. (2001). UML activity diagrams as a workflow specification language. In ≪ UML≫ 2001—The Unified Modeling Language. Modeling Languages, Concepts, and Tools (pp. 76-90). Springer Berlin Heidelberg.

Esteva, M., Rodriguez-Aguilar, J. A., Sierra, C., Garcia, P., & Arcos, J. L. (2001). On the formal specification of electronic institutions. In Agent mediated electronic commerce (pp. 126-147). Springer Berlin Heidelberg.

Fuentes, R, F., Henderson, B. H., Argente, E. S., Beydoun, G., and Low, G. (1999) "Agent-oriented software engineering". In Modelling with Agents, pages 151–162. Springer.

Gruninger, M., & Menzel, C. (2003). The process specification language (PSL) theory and applications. AI magazine, 24(3), 63.

Gularte, A., Gonçalves, E., Jung M., da Rosa, A.(2013)" Two different perspectives to specify and implement multiagent systems". WESAAC.

Horling, B., & Lesser, V. (2004). A survey of multi-agent organizational paradigms. The Knowledge Engineering Review, 19(04), 281-316.

Hussain, F., Ramanathan, A., Pullum, L. L., & Jha, S. K. (2014). EpiSpec: A formal specification language for parameterized agent-based models against epidemiological ground truth. In Computational Advances in Bio and Medical Sciences (ICCABS), 2014 IEEE 4th International Conference on (pp. 1-6). IEEE.

Karsai, G., Agrawal, A., Shi, F., & Sprinkle, J. (2003). On the use of graph transformation in the formal specification of model interpreters. J. UCS,9(11), 1296-1321.

Lamsweerde, A. V. (2000). Formal specification: a roadmap. InProceedings of the Conference on the Future of Software Engineering (pp. 147-159). ACM.

Lin, H. I., & Cheng, C. H. (2014). Behavior-Based Manipulator Programming Based on Extensible Agent Behavior Specification Language. In Control, Automation and Systems (ICCAS), 2014 14th International Conference on (pp. 808-813). IEEE.

Lin, H., & Yang, C. (2006). Specification of Multi-Agent Systems in the Gamma Language.

Lötzsch, M., Bach, J., Burkhard, H. D., & Jüngel, M. (2003). Designing agent behavior with the extensible agent behavior specification language XABSL. InRoboCup 2003: Robot Soccer World Cup VII (pp. 114-124). Springer Berlin Heidelberg.

Medina, M. A., & Urban, J. E. (2007). An Approach to Deriving Reactive Agent Designs from Extensions to the Descartes Specification Language. In Autonomous Decentralized Systems, 2007. ISADS'07. Eighth International Symposium on (pp. 363-367). IEEE.

Monostori, L., Váncza, J., & Kumara, S. R. (2006). Agent-based systems for manufacturing. CIRP Annals-Manufacturing Technology, 55(2), 697-720.

Ormandjieva, O., Bentahar, J., Huang, J., & Kuang, H. (2015). Modelling Multi-agent Systems with Category Theory. Procedia Computer Science, 52, 538-545.

Pagliarecci, F., Spalazzi, L., Stehr, M. O., & Talcott, C. L. (2008). Formal specification of agent-object oriented programs. In Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on(pp. 127-134). IEEE.

Pustejovsky, J.,Ingria, B.,Sauri, R.,Castano, J.,Littman, J., Gaizauskas, R., Setzer, A.,Katz, G. and Miani, I. (2005) The Specification Language TimeML. In I. Mani, J. Pustejovsky, and R. Gaizauskas, (eds.), The Language of Time: A Reader. Oxford University Press.

Ramzan, M., Ali, A., Akram, S., & Qayyum, Z. U. (2011). Formal specification of multi-agent environment using VDM-SL. In Computer Sciences and Convergence Information Technology (ICCIT), 2011 6th International Conference on (pp. 150-154). IEEE.

Subburaj, V. H., & Urban, J. E. (2013). A formal specification language for modeling agent systems. In Informatics and Applications (ICIA), 2013 Second International Conference on (pp. 300-305). IEEE.

Tao, Z., Hong, X., & Shao-bin, H. (2015). A Formal Verification Method of Obligation Policy in Multi-agent System.

Urban, J. E., Subburaj, V. H., & Ramamoorthy, L. (2011). Extending the descartes specification language towards process modeling. InComputer Science and Information Systems (FedCSIS), 2011 Federated Conference on (pp. 337-340). IEEE.

Van, H. N., Balabonski, T., Boulanger, F., Taha, S., Valiron, B., Wolff, B., & Ye, L. (2015). Towards a formal semantics of the TESL specification language⋆. In 3rd

International Workshop on the Globalization Of Modeling Languages (GEMOC 2015) (No. 1511, pp. 14-19).

Vicari, R. M. (2012). Um Metamodelo UML para a Modelagem de Requisitos em Projetos de Sistemas MultiAgentes (Doctoral dissertation, UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL).

Wang, L., & Hongshuai, Z. (2010). Ontology for communication in distributed multi-agent system. In Distributed Computing and Applications to Business Engineering and Science (DCABES), 2010 Ninth International Symposium on (pp. 588-592). IEEE.

Wang, Y., & Singh, M. P. (2007). Formal Trust Model for Multiagent Systems. In IJCAI (Vol. 7, pp. 1551-1556).

Wood, M. F., & DeLoach, S. A. (2001). An overview of the multiagent systems engineering methodology. In Agent-Oriented Software Engineering (pp. 207-221). Springer Berlin Heidelberg.

Wooldridge, M., & Jennings, N. R. (1994). Agent theories, architectures, and languages: a survey. In Intelligent agents (pp. 1-39). Springer Berlin Heidelberg.

Yuan, L., & Fan, P. (2013). Formal Modeling and Verification of Multi-agent System Architecture. AASRI Procedia, 5, 126-132.

Zhao, Z., & Xu, Y. (2010). DPMAS: A Design Method for Multi-agent System Using Agent UML. In Information and Computing (ICIC), 2010 Third International Conference on (Vol. 4, pp. 137-140). IEEE.