Anais do XIII Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCações

— WESAAC 2019 —

Organizado por

Jerusa Marchi

Gleifer Vaz Alves

Gustavo Guiménez Lugo

Florianópolis, 02 – 04 de Maio, 2019.

Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCações — XIII WESAAC / Marchi, J.; Alves, G. V.; Lugo, G. G. (Org). ANAIS.— — Florianópolis, 2019.

270p. :il.

ISSN 2177-2096

1. Agentes Inteligentes. 2. Sistemas de Agentes de Software. 3. Ambientes para Agentes. 4. Aplicações de Agentes. I. Marchi, J. II. Alves, G. V. III. Lugo, G. G.

CDD

PREFÁCIO

Este documento contém os trabalhos apresentados na Décima Terceira Edição do WESAAC (Workshop Escola de Sistemas de Agentes, seus Ambientes e apliCações), realizado na cidade de Florianópolis/SC, com o apoio da Universidade Federal de Santa Catarina (UFSC), Universidade Tecnológica Federal do Paraná (UTFPR) e Universidade do Sul do Estado de Santa Catarina (UNISUL), entre os dias 02 e 04 de Maio de 2018.

Com o objetivo de integrar pesquisadores e estudantes de todos os níveis na área de Agentes e Sistemas de Agentes e divulgar as atividades dos diversos grupos de pesquisa do Brasil, bem como promover o intercâmbio de conhecimentos e experiências, a 13ª edição do WESAAC contou com Oficinas e Palestras proferidas por professores e pesquisadores com excelência em suas áreas de conhecimento e com apresentações de Trabalhos Completos e Resumos Estendidos.

Gostaríamos de agradecer aos pesquisadores convidados, Cristina Baroglio, Maiquel de Brito e Ingrid Nunes que abrilhantaram o evento com suas palestras. Também agradecemos à Diana Adamatti e Maicon Zatelli pelas oficinas ministradas.

Neste ano recebemos trabalhos de diversas regiões do país. Agradecemos a todos os pesquisadores que submeteram seus artigos e que proporcionaram a troca de conhecimentos e a integração entre pesquisadores e grupos de pequisa. Gostaríamos de agradecer aos membros do comitê de programa, aos revisores adicionais pelo criterioso trabalho desenvolvido. Nossos sinceros agradecimentos à comissão de realização local, Professores Jomi Fred Hubner, Saulo Popov Zambiasi, Ricardo Azambuja Silveira e Maicon Zatelli por todo apoio dado para a realização do evento. Para esta edição, tivemos o suporte financeiro da Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC) (www.fapesc.sc.gov.br), que foi fundamental para a viabilidade do evento. Nosso agradecimento também ao Programa de Pós-Graduação em Engenharia de Controle e Automação e as instituições organizadoras, Universidade Federal de Santa Catarina, Universidade Tecnológica Federal do Paraná e Universidade do Sul do Estado de Santa Catarina (UNISUL) por todo apoio para a realização do evento. Também agradecemos à Fundação Stemmer para Pesquisa, Desenvolvimento e Inovação (FEESC) pelo apoio.

Florianópolis, 2019.
Jerusa Marchi
Gleifer Vaz Alves
Gustavo Guiménez Lugo

Organização e Patrocínio

Organização

Organização Geral

Jerusa Marchi (UFSC)

Coordenação do Comitê de Programa

Gleifer Vaz Alves (UTFPR)

Gustavo Guiménez Lugo (UTFPR)

Organização Local

Jomi Fred Hubner (UFSC)

Maicon Rafael Zatelli (UFSC)

Ricardo Azambuja Silveira (UFSC)

Saulo Popov Zambiasi (UNISUL)

Comitê Consultivo

Anarosa Brandão (USP)
Antônio Carlos Rocha Costa (UFRGS)
Diana Francisca Adamatti (FURG)
Gustavo Alberto Giménez Lugo (UTFPR)
Jaime Simão Sichman (USP)
Jomi Fred Hübner (UFSC)
Mariela Inés Cortés (UECE)
Jerusa Marchi (UFSC)

Comitê de Programa

Adamatti, Diana	FURG (Brasil)
Alencar, Fernanda	UFPE (Brasil)
Alves Franco Brandão, Anarosa	USP (Brasil)
Azambuja Silveira, Ricardo	UFSC (Brasil)
Barbosa, Raquel	FURG (Brasil)
Baségio, Túlio Lima	PUCRS (Brasil)
Billa, Cleo	FURG (Brasil)
Boissier, Olivier	EMSE (França)
Borges, André Pinz	UTFPR (Brasil)
Brito, Maiquel de	IFRS (Brasil)
Campos, Gustavo Augusto	UECE (Brasil)

Casare, Sara	LIP6 (França)
Castro, Paulo André L.	ITA (Brasil)
Choren, Ricardo	IME/RJ (Brasil)
Coelho, Helder	FCUL (Portugal)
Cortés, Mariela Inés	UECE (Brasil)
Dimuro, Graçaliz	FURG (Brasil)
Freitas, Artur Silva Da Cunha	PUCRS(Brasil)
Frozza, Rejane	UNISC (Brasil)
Giménez Lugo, Gustavo	UTFPR (Brasil)
Gonçalves, Eder Mateus Nunes	FURG (Brasil)
Grunitzki, Ricardo	UFRGS (Brasil)
Hubner, Jomi Fred	UFSC (Brasil)
Leitão, Paulo	Instituto Politécnico de Bragança (Portugal)
Leite, João	Nova LINCS (Portugal)
Lemke, Ana Paula	IFRS (Brasil)
Marchi, Jerusa	UFSC (Brasil)
Moreira, Álvaro	UFRGS (Brasil)
Nardin, Luis Gustavo	University of Idaho (USA)
Ramos, Gabriel De O.	Vrije Universiteit Brussel (Bélgica)
Rocha Costa, Antônio Carlos	UFRGS (Brasil)
Sanchotene de Aguiar, Marilton	UFPel (Brasil)
Sichman, Jaime Simão	USP (Brasil)
Silva, João Luis	UCS (Brasil)
Simari, Guillermo R.	Universidad del Sur in Bahia Blanca (Argentina)
Tacla, Cesar A.	UTFPR (Brasil)
Vaz Alves, Gleifer	UTFPR (Brasil)
Webber, Carine	UCS (Brasil)
Werneck, Vera	UFRJ (Brasil)
Xavier, Clarissa	UFRGS (Brasil)
Zatelli, Maicon	UFSC (Brasil)

SUMÁRIO

ARTIGOS COMPLETOS

RESUMOS ESTENDIDOS

ARTIGOS COMPLETOS

Análise Comparativa entre Ferramentas de Simulação Multiagente

Ricardo A. Machado[1], **Gíulia B. Silveira** [1], **Marla P. Melo** [1], **Diana F. Adamatti**[1], **Cleo Z. Billa** [1]

[1]Programa de Pós-Graduação em Computação
Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Rio Grande – RS – Brasil

{ricardoarend,contateagiulia,marlamelo.sinfo}@gmail.com

{dianaadamatti,cleobilla}@furg.br

Abstract. *The use of tools to assist in the process of modeling and developing simulations is very important, since it streamlines the whole process as well as tends to minimize errors. In this way, their study and analysis are for choosing the tool that best suits a simulation.*
This paper presents a comparative study between three multiagent simulation tools, JADE, MESA and CORMAS, applied to the same problem. The objective is to present a description of each tool, develop the same application in each tool and run simulations. The entire development process until execution is used for comparative purposes in different parameters.

Resumo. *O uso de ferramentas para auxiliar no processo de modelagem e desenvolvimento de simulações é muito importante, pois agiliza todo processo, bem como tende a minimizar erros. Desta forma, seu estudo e análise são vitais para a escolha da ferramenta que melhor atende aos requisitos desejados para a simulação a ser realizada.*
Este artigo apresenta um estudo comparativo entre três ferramentas de simulação multiagente, JADE, MESA e CORMAS, aplicadas a um mesmo problema. O objetivo é apresentar uma descrição de cada ferramenta, realizar o desenvolvimento de uma mesma aplicação em cada ferramenta e executar simulações. Todo o processo de desenvolvimento até a execução é utilizado para fins comparativos em diferentes parâmetros.

1. Introdução

A Simulação Multiagente(*Multi-Agent-Based Simulation - MABS*) surge da união das tecnologias de Simulação e Sistemas Multiagente (SMA) [Adamatti et al. 2007]. A simulação é uma excelente forma de modelar e entender os processos sociais, pois pode-se analisar aspectos de emergência relacionada a atividades simples. É uma área de pesquisa interdisciplinar, que busca construir dinamicamente a teoria científica, enquanto compreendemos melhor o problema através de simulação multiagente [Gilbert and Troitzsch 2005].

O uso de ferramentas para auxiliar no processo de modelagem e desenvolvimento das simulações é muito importante, pois agiliza todo processo, bem como tende a minimizar erros. Na área de MABS, diversos grupos de pesquisa desenvolvem e disponibilizam

ferramentas. Neste artigo, o objetivo principal é apresentar uma comparação entre três ferramentas de simulação, utilizando um mesmo "problema" para testes: "Pegue o Porco", onde dois agentes fazendeiros tentam capturar um terceiro agente denominado porco.

As ferramentas comparadas neste artigo são:

- O JADE[1] é uma API (*Application Programming Interface*) da linguagem Java para desenvolvimento de agentes baseada nas especificações da FIPA (*Foundation for Intelligent Physical Agents*);
- O MESA[2] é uma ferramenta de modelagem baseada em agentes na linguagem Python, projetada para realizar simulações customizadas que podem ser visualizadas em navegador web;
- CORMAS[3] é uma plataforma de simulação baseada no ambiente de programação VisualWorks, para linguagem Smalltalk, com foco no desenvolvimento de aplicações para gestão de recursos naturais.

O método proposto para realizar a comparação entre as ferramentas citadas acima, refere-se à especificação de cinco parâmetros para a avaliação das ferramentas e a verificação em termos do desempenho obtido por meio dos resultados da média e desvio padrão da simulação do problema "pegue o porco", descrito na seção 3.

O artigo está estruturado da seguinte forma: na seção 2 são descritas as ferramentas utilizadas. Na seção 3 é apresentado o problema "pegue o porco" e o processo de desenvolvimento em cada ferramenta. Na seção 4 são apresentadas as vantagens e desvantagens entre as ferramentas, como também, os resultados da simulação e por fim, na seção 5 são apresentadas as conclusões desta análise comparativa entre as ferramentas de simulação baseadas em agentes.

2. Descrição das Ferramentas Utilizadas

De forma geral existem dois tipos de ferramentas disponíveis para o desenvolvimento de modelos baseados em agentes: kits de ferramentas e softwares [Castle and Crooks 2006].

Os kits de ferramentas, também conhecidos por *frameworks*, fornecem bibliotecas com classes e funções predefinidas, projetadas especificamente para o desenvolvimento de simulações baseadas em agentes. Apesar de apresentarem a desvantagem da alta curva de aprendizagem de uma linguagem, os *frameworks* reduzem esforços com a programação de interfaces gráficas com usuário, importação e exportação de dados e visualização do modelo [Castle and Crooks 2006]. Outro aspecto positivo é de que o paradigma orientado à objetos permite estender a capacidades dos *frameworks*, integrando novas funcionalidades particulares às bibliotecas.

Em contraste com os kits de ferramentas, os softwares simplificam o processo de implementação, por não exigirem conhecimento em linguagens de programação. Softwares são úteis para o desenvolvimento de modelos básicos ou protótipos em ambientes mais limitados ou restritos à funcionalidade fornecida pelo mesmo [Castle and Crooks 2006].

Neste artigo, são comparados os kits de ferramentas MESA, JADE e CORMAS.

[1] http://jade.tilab.com/

[2] https://github.com/projectmesa/mesa

[3] http://cormas.cirad.fr/indexeng.htm

2.1. JADE

JADE (*Java Agent Development Framework*) é um *framework* Java completo que trata da comunicação, do ciclo de vida do agente, do monitoramento da execução, entre outras atividades e que tem como objetivo facilitar o desenvolvimento de aplicações multi-agentes em conformidade com as especificações FIPA (*Foundation for Intelligent Physical Agents*) [Bellifemine et al. 2000].

FIPA é uma associação internacional sem fins lucrativos de empresas e organizações que compartilham o esforço para produzir especificações para tecnologias de agentes genéricos. O FIPA-ACL é uma linguagem que descreve a codificação e semântica de mensagens, mas não exige mecanismos específicos para o transporte de mensagens sendo essa linguagem utilizada por diferentes SMA incluindo o JADE.

O JADE é uma das ferramentas mais utilizadas para o desenvolvimento de sistemas baseados em agentes. De acordo com [Bellifemine et al. 2002], JADE foi escrito em Java devido a características particulares da linguagem, especificamente pela programação orientada a objetos em ambientes distribuídos heterogêneos. Foram desenvolvidos tanto pacotes Java com funcionalidades prontas pra uso, quanto interfaces abstratas para se adaptar de acordo com a funcionalidade da aplicação de agentes.

Embora pareça uma entidade única para o mundo externo, uma plataforma de agente JADE é, por si só, um sistema distribuído. Um sistema JADE compreende um ou mais contêineres de agentes, cada um numa máquina virtual Java de forma independente [Bellifemine et al. 2001].

2.2. MESA

O MESA é um *Framework* em Python de código aberto que permite aos usuários criar rapidamente modelos baseados em agentes. A partir dos conceitos do paradigma de programação orientada a objetos, o MESA distribui seus componentes dentro de três módulos que podem facilmente ser combinados e ampliados para construir diferentes tipos de simulações [Masad and Kazil 2015]. Os módulos são divididos em três categorias gerais: modelagem, análise e visualização. Em cada módulo, existem conjuntos de classes com atributos e funções predefinidos, que podem ser herdados durante o desenvolvimento de uma simulação.

No módulo de modelagem encontram-se classes destinadas a definição de parâmetros das simulações, são elas; modelo, agentes, agendador e espaço. De forma geral, uma simulação consiste da instância da classe modelo que representará o funcionamento da simulação; uma ou mais instâncias da classe de agente; uma instância de agendador, que gerencia a ativação de agentes de forma totalmente personalizável; e um espaço que possibilita que os agentes se movimentem e interajam com vizinhos.

A análise de informações após uma simulação ou um lote de simulações é facilitada, através dos componentes fornecidos no módulo de análise. Nesta categoria, encontram-se os coletores de dados, usados para registrar dados de cada execução do modelo ou de um lote de execuções com registros detalhados sobre variáveis do modelo, bem como dos agentes [Masad and Kazil 2015].

Um modelo baseado em agente não é particularmente útil se não houver maneira de ver os resultados produzidos pelas simulações. No módulo de visualização, o *fra-*

mework oferece a visualização dos resultados em interface gráfica ou através de coleta de dados quantitativos. Para facilitar a última opção, são fornecidas classes genéricas que podem armazenar e exportar dados de variáveis do modelo, variáveis do agente e tabelas que são um resumo da simulação [Masad and Kazil 2015].

A interface gráfica pode ser construída livremente utilizando *Javascript, Hypertext Markup Language (HTML) e Cascading Style Sheets (CSS)* e sua exibição ocorre em um navegador web que recebe dados através de um servidor criado pelo MESA.

O MESA é o primeiro *framework* para MABS na linguagem Python. Por estar sendo construído do zero, tem-se a viabilidade de incorporar novos recursos que ampliem a diversificação de representação das simulações baseadas em agentes.

2.3. CORMAS

O ambiente de simulação multiagente CORMAS (*Common-Pool Resources and Multi-agent Systems*) [Bousquet et al. 1998] foi desenvolvido pelo grupo Green-CIRAD, principalmente para simulação de recursos naturais [Adamatti 2007], sendo útil para o melhor entendimento das interações complexas entre a dinâmica natural e social, quando se estuda o gerenciamento de recursos renováveis [Bousquet et al. 1998].

CORMAS foi desenvolvida a partir do ambiente Visualworks, e para o desenvolvimento de modelos, é utilizada a linguagem de programação orientada a objetos Smalltalk. Além disso, é possível programar e importar, para a plataforma CORMAS, os códigos na linguagem dinâmica e reflexiva Pharo[4], inspirada na linguagem de programação Smalltalk.

A plataforma CORMAS é completa, permite vários agentes, comunicação, movimentação e também possibilita o monitoramento e análise de simulações, bem como a observação de simulações com a noção de ponto de vista (*pov*) que permite ao modelador do sistema multiagente, definir a observação de uma parte do espaço ou de um agente em diferentes pontos de vista [Bousquet et al. 1998]. Além disso, possui integração com R[5], permitindo a geração de gráficos e análises estatísticas complexas.

3. Desenvolvimento do Cenário Pegue o Porco

Para efetuar a comparação das ferramentas, foi desenvolvido o mesmo problema em cada uma das ferramentas apresentadas. Esse problema, denominado “Pegue o Porco”, é composto por um tabuleiro 5x5, conforme Figura 1. Nele, existe um agente denominado porco e dois agentes fazendeiros. O objetivo é fazer com que os agentes fazendeiros colaborem e capturem o porco, da maneira mais eficiente possível.

O problema proposto tem como especificações os seguintes requisitos:

1. A posição inicial de cada um dos três agentes deve ser aleatória;
2. O ambiente é completamente observável, ou seja, todos os agentes sabem a posição de todos;
3. O número máximo de movimentos (N) é um parâmetro definido pelo usuário;
4. Cada agente se move um quadrado de cada vez e só pode se movimentar para os quadrados adjacentes a sua posição atual. Não é permitido andar na diagonal;

[4]https://pharo.org/

[5]https://www.r-project.org/

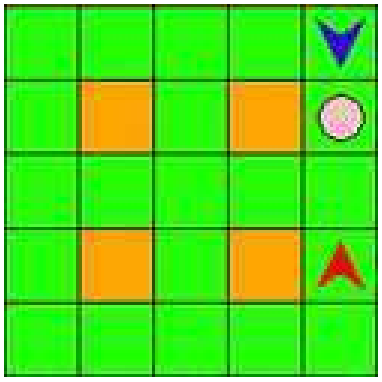


Figura 1. Cenário Proposto

5. A movimentação é feita em turnos, uma vez para cada;
6. O agente porco é um agente autônomo que deve sempre fugir do fazendeiro mais próximo. Caso o porco não possa se afastar, ele fica imóvel;
7. Os agentes fazendeiros também são autônomos e devem colaborar, a fim de cercar o porco para que ele possa ser capturado;
8. A simulação acaba quando um dos agentes fazendeiros captura o porco.

A seguir, é detalhado o processo para modelagem e desenvolvimento em cada ferramenta e suas características.

3.1. JADE

O cenário foi desenvolvido a partir de uma classe principal Java chamada Tabuleiro, que gera uma matriz 5x5. A posição inicial dos agentes é gerada de forma randômica. Essa classe também contém diferentes funções que são utilizadas pelos agentes para mapeamento e movimentação dos mesmos. Além disso, uma interface gráfica 2d foi desenvolvida utilizando uma biblioteca Java para visualização do ambiente.

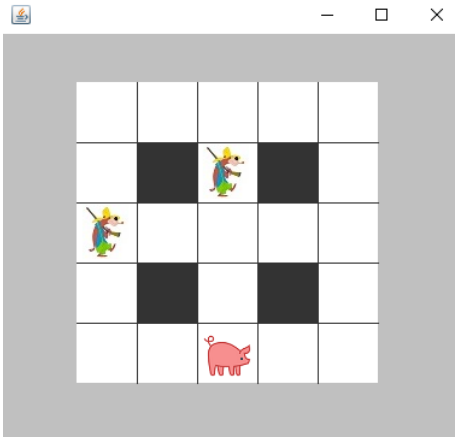


Figura 2. Cenário no JADE

A Figura 2 mostra o ambiente desenvolvido com dois fazendeiros e o porco. Nela, pode-se visualizar o tabuleiro, que contém quatro obstáculos formados pelos quadrados pretos que os agentes não podem entrar. Quanto a estratégia utilizada para capturar o porco: os fazendeiros, ao iniciar seus respectivos turnos calculam a distância em relação

ao porco e também verificam qual o melhor caminho a seguir com base na localização do outro fazendeiro no sentido de encurralar o porco. Já o porco escolhe sempre o caminho que contém a maior distância em relação aos dois fazendeiros. Nesse exemplo, não foi necessária a utilização da comunicação entre os agentes, porém essa é uma das futuras melhorias que serão realizadas no projeto.

Em JADE, cada tipo diferente de agente é desenvolvido a partir de uma classe própria. Nesse exemplo, foi criado uma classe para cada agente JADE e elas são responsáveis pelo comportamento cíclico do agente e chamamento dos métodos que foram desenvolvidos na classe Tabuleiro. Também foi necessário criar um agente "genérico", que inicializa um tabuleiro único e pode ser estendido pelos outros agentes evitando assim problemas de redundância dos dados [6].

3.2. MESA

O cenário no MESA pode ser entendido como um elemento espacial, onde os agentes podem ocupar posições e realizar interações com outros agentes e outros objetos [Masad and Kazil 2015]. Esses espaços podem ser representações abstratas, como um autômato celular utilizando modelos toroidais, por exemplo; ou escalar, com representações de cidades ou regiões do mundo.

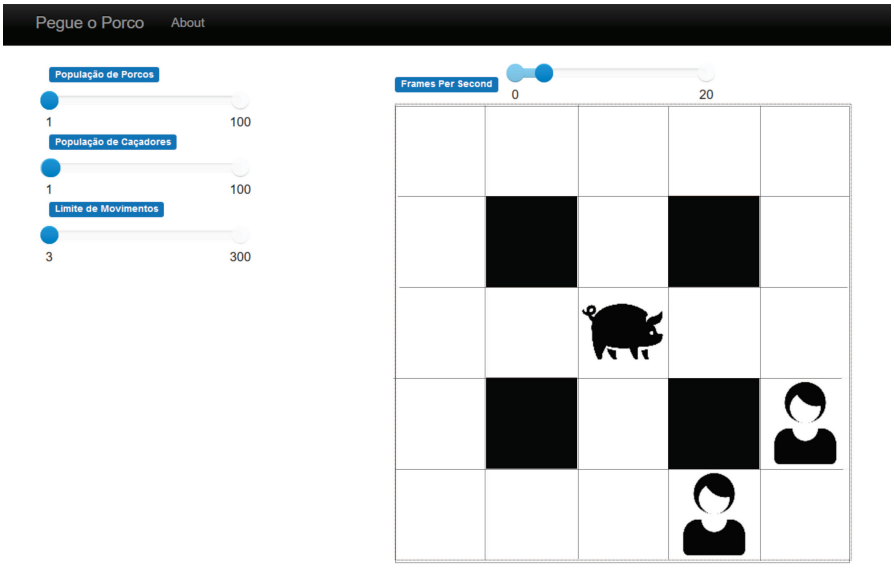


Figura 3. Cenário em MESA

O MESA atualmente implementa duas grandes classes de espaço: o espaço grade e o espaço contínuo. No espaço grade, os agentes e outros objetos só podem estar em uma célula particular (podendo abranger várias células); já no espaço contínuo podem ocupar qualquer posição arbitrária [Masad and Kazil 2015].

Existem várias classes de espaços específicas, todas herdadas de uma classe mãe. No núcleo da classe mãe os espaços são representados por uma matriz bidimensional que

[6]Todo o código fonte do projeto aqui descrito está disponibilizado para visualização via github: https://github.com/ricardoarend/CatchPig

possui métodos para obtenção de vizinhos, adição e remoção de agentes. Para definir o conteúdo das células da matriz podem ser utilizadas as classes *Multigrid* ou *Singlegrid* [Masad and Kazil 2015]. No *MultiGrid*, vários objetos podem compartilhar uma célula, enquanto no *Singlegrid* no máximo um objeto pode estar contido em uma célula.

Para modelagem do problema "Pegue o Porco", instanciou-se um cenário de espaço contínuo, de forma que os agentes tenham posicionamento inicial randômico e movimentação arbitrária entre as células vizinhas. Utilizou-se a ocupação espacial *MultiGrid* pois, o porco é capturado quando existe uma célula com um agente do tipo fazendeiro e um agente do tipo porco.

A posição dos agentes é armazenada duas vezes, como uma tupla: uma vez é registrada no gráfico da célula e outra nos atributos do agente. Para cada agente definir sua melhor coordenada para movimentação, de acordo com sua estratégia, criou-se um raio de percepção do ambiente.

Por fim, utilizando os recursos do *framework* para visualização da simulação, criou-se uma interface para o problema do porco, com botões para parametrizar a população de agentes, a quantidade de movimentos e a quantidade de *steps* de execução. A interface gráfica da simulação pode ser observada na Figura 3.

3.3. CORMAS

A Figura 4 exibe a implementação do cenário na ferramenta CORMAS.

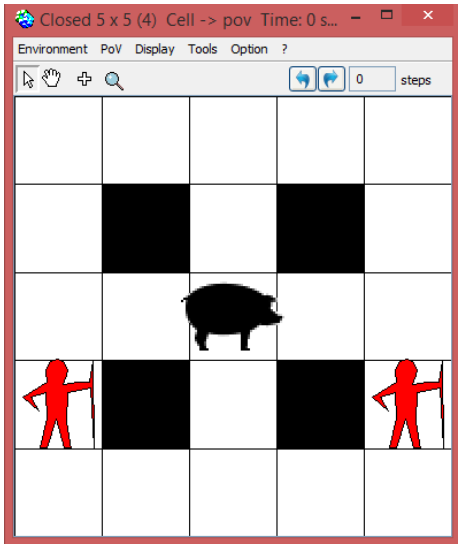


Figura 4. Cenário no CORMAS

Na plataforma CORMAS existem basicamente três tipos de entidades genéricas: "agente social", "entidade espacial"e "entidade passiva"[Bommela et al. 2017].

- Na classe da entidade social são definidos os agentes que se comunicam e interagem com outros agentes, nesse caso, os agentes fazendeiros e o porco;
- A classe da entidade espacial define o ambiente e os elementos que estão localizados nele e dá suporte topológico para simulações, arbitrando na alocação de recursos naturais de acordo com protocolos pré-estabelecidos [Sichman et al. 2003].

Nesse modelo, na entidade *cell*, está definido o *grid* 5x5, onde os obstáculos são definidos e os agentes são situados aleatoriamente;

- E na classe da entidade passiva estão as mensagens de comunicação e objetos. No modelo, os obstáculos situados no ambiente para os agentes.

As células são organizadas hierarquicamente. Desse modo, a entidade *cell*, definida na entidade espacial, os agentes dos tipos fazendeiro e porco da classe social e os objetos obstáculos definidos na entidade passiva, são subclasses que descrevem o estado atual das células. Estes estados correspondem a implementação dos métodos dos agentes, isto é, as ações que cada agente poderá realizar no ambiente.

Os autores [Bousquet et al. 1998] definem um *patch* como uma "classe pré-programada e proposta para a herança". Esta classe inclui atributos que definem a vizinhança (ordenada ou não, em um raio de percepção variável). Cada *patch* possui um atributo *lesOccupants* que contém automaticamente a lista de agentes localizados nele, classificados por tipo de agente". Assim, para efetuar a captura do porco, o agente fazendeiro escolhe uma célula dentro do raio de percepção que contenha um agente porco.

3.4. Resultados

Para cada ferramenta foram realizadas 50 simulações, com dois agentes fazendeiros e um agente porco. A Tabela 1 apresenta a média e o desvio padrão do número de movimentos realizados até a captura do porco em cada ferramenta. O número de movimentos contados é resultado da quantidade de turnos onde cada um dos agentes fez uma movimentação.

Tabela 1. Resultados das simulações

Ferramenta	Média	Desvio Padrão
JADE	5,54	1,98
MESA	5,82	5,05
CORMAS	25,28	24,74

Percebe-se que a implementação com melhor desempenho e maior estabilidade é o JADE, onde a média de movimentos dos agentes foi de aproximadamente 5, com um desvio padrão próximo de 2.

No caso do MESA, o número médio de movimentos foi também próximo a 5, como nas simulações realizadas do JADE, porém com um desvio padrão mais alto (próximo de 5), o que indica uma maior variação no número de movimentos necessários para encerrar a simulação.

No caso do CORMAS, houveram simulações muito eficientes e outras muito ruins, já que o desvio padrão ficou próximo a média.

4. Vantagens e Desvantagens entre as Ferramentas Analisadas

Com a finalidade de comparar as três ferramentas de simulação baseadas em agentes, foram estabelecidos os seguintes parâmetros:

1. Material disponível para consulta;
2. Complexidade de modelagem da interface;
3. Complexidade de programação (métodos pré-prontos/ métodos genéricos

4. Ambiente de modelagem/desenvolvimento;
5. Interface gráfica para visualização das simulações.

A Tabela 2 apresenta esses parâmetros para cada uma das ferramentas. Abaixo são apresentados os motivos que levaram a tais classificações.

Tabela 2. Comparação entre as ferramentas

Parâmetro	JADE	MESA	CORMAS
1	✓	✓	✓
2	Difícil	Fácil	Difícil
3	Médio	Fácil	Difícil
4	Diversos	Diversos	Um
5	×	✓	✓

Esses parâmetros foram aplicados com base em um nível acadêmico de conhecimento e não servem como comparação para profissionais da área que tenham maior conhecimento de determinada linguagem.

4.1. JADE

A ferramenta JADE possui um bom material de apoio para iniciantes. Em sua página é possível encontrar todo tipo de tutoriais, inclusive um em português, além de algumas publicações.

No quesito interface, a principal dificuldade foi a ausência de referências para serem utilizadas como base para sua modelagem. Foi preciso utilizar mecanismos para fazer com que a interface do ambiente ficasse sempre atualizada independente de qual agente estivesse se movimentando.

Quanto a complexidade de programação, um conhecimento de java é o suficiente para conseguir desenvolver o sistema. Porém, não existem métodos prontos e tudo tem que ser feito do zero (movimentação, cenário, interação).

O ambiente de desenvolvimento foi o NetBeans, porém também é possível utilizar o Eclipse, ou qualquer outro ambiente para Java.

No JADE não existe uma interface gráfica já pronta. Porém, nada impede que seja criado usando Java, como no caso desse exemplo.

4.2. MESA

Nos últimos anos, o Python tornou-se uma linguagem cada vez mais popular para computação científica [Pérez and Granger 2007], apoiado por crescente conjunto de ferramentas para análise e modelagem. Python se destaca em relação ao Smalltalk por ser *mainstream*, o que é uma vantagem no aprendizado e documentação.

A interface gráfica de MESA é totalmente personalizável, podendo-se criar botões para iniciar ou encerrar uma simulação, campos para definir parâmetros de quantidade de agentes predadores instanciados e quantidade de movimentos que os agentes realizarão, por exemplo. Para criar ou realizar modificações na interface são necessários conhecimentos em programação web, contudo a ferramenta oferece interfaces de modelos de

simulações que podem demandar pouca ou nenhuma modificação, como no problema tratado neste artigo.

O MESA nativamente oferece classes completas para o desenvolvimento de uma MBA, além de possibilitar a importação de bibliotecas disponíveis, deixando para o desenvolvedor apenas a implementação de funções específicas, como a estratégia de fuga do porco. Além disto, disponibiliza um pacote com diversos exemplos de simulações com abordagens em diferentes cenários.

O ambiente de desenvolvimento utilizado foi PyCharm[7], mas a escolha para a IDE ideal é realizada pelo o usuário e suas preferências de desenvolvimento.

4.3. CORMAS

No ambiente de modelagem da plataforma CORMAS, em sua interface principal, está disponibilizado um glossário com informações sobre os modelos implementados na plataforma e também um glossário dos métodos de CORMAS. Além disso, não possui flexibilidade de modelagem da interface.

CORMAS possui uma estrutura de alto nível com uma vasta implementação de funções prontas e métodos gerados automaticamente que precisam ser compreendidos para serem aplicados. Pela complexidade da plataforma CORMAS e do não conhecimento da linguagem Smalltalk, o processo de desenvolvimento de modelos pode ser demorado.

Como CORMAS é uma plataforma, ela integra o ambiente de simulação e o desenvolvimento, bem como, disponibiliza uma interface gráfica para visualização das simulações e propícia recursos para análise de simulações.

5. Conclusões

O artigo apresentou uma breve explanação a respeito de MABS, além do estudo e uma comparação das ferramentas de simulação baseadas em agentes: JADE, MESA e CORMAS, e sua aplicação para o problema “Pegue o porco”.

Conforme os resultados apresentados na Tabela 1, cada ferramenta apresentou um comportamento diferente em relação a média e o desvio padrão do número de movimentos para os agentes fazendeiros capturarem o agente porco. Mostrando assim a diferença entre as características específicas de cada ferramenta na implementação de uma mesma solução para um mesmo problema.

Observa-se que as ferramentas JADE e MESA apresentaram resultados semelhantes. Neste contexto, essas ferramentas mostraram-se eficazes para o problema específico “Pegue o Porco”. No entanto, a plataforma CORMAS teve um destaque negativo, visto que o propósito da plataforma CORMAS é facilitar o desenvolvimento de modelos baseados nas interações entre as dinâmicas naturais e sociais no domínio da gestão de recursos naturais, sem preocupação com desempenho.

Em relação a simulação na Tabela 1, foi verificado que a ferramenta JADE teve um melhor desempenho por ter a melhor média e com um baixo desvio padrão. Já em relação a comparação das ferramentas em si, onde foram apresentados alguns parâmetros,

[7]Disponível em: https://www.jetbrains.com/pycharm/

a Tabela 2 sumarizou o comparativo, demonstrando que cada ferramenta tem vantagens e desvantagens, porém a ferramenta MESA teve uma melhor avaliação ao somar todos os parâmetros analisados.

Como trabalhos futuros, sugere-se explorar a comunicação entre os agentes, aumentar a complexidade do ambiente, realizar comparações entre mais ferramentas ou adicionar outros tipos de agentes.

Esse artigo mostrou um comparativo para três ferramentas para um problema específico porém fica a cargo do desenvolvedor da simulação a escolha da ferramenta correta para o estudo a ser realizado.

Referências

Adamatti, D. F. (2007). Inserção de jogadores virtuais em jogos de papéis para uso em sistema de apoio a decisão em grupos: um experimento no domínio da gestão dos recursos naturais. *Universidade de São Paulo, São Paulo*.

Adamatti, D. F., Sichman, J. S., and Coelho, H. (2007). Utilizaç ao de rpg e mabs no desenvolvimento de sistemas de apoioa decisao em grupos. *Anais do IV Simpósio Brasileiro de Sistemas Colaborativos SBSC 2007*, page 15.

Bellifemine, F., Caire, G., Trucco, T., and Rimassa, G. (2002). Jade programmer's guide. *Jade version*, 3:13–39.

Bellifemine, F., Poggi, A., and Rimassa, G. (2000). Developing multi-agent systems with jade. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 89–103. Springer.

Bellifemine, F., Poggi, A., and Rimassa, G. (2001). Jade: a fipa2000 compliant agent development environment. In *Proceedings of the fifth international conference on Autonomous agents*, pages 216–217. ACM.

Bommela, P., Becuc, N., Le Pageb, C., Bousquetb, F., and Leclercb, G. (2017). Cormas, una plataforma multiagente para la modelización interactiva.

Bousquet, F., Bakam, I., Proton, H., and Le Page, C. (1998). Cormas: common-pool resources and multi-agent systems. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 826–837. Springer.

Castle, C. J. and Crooks, A. T. (2006). Principles and concepts of agent-based modelling for developing geospatial simulations.

Gilbert, N. and Troitzsch, K. G. (2005). *PSimulation for the Social Scientist*. Open University Press Milton Keynes, UK.

Masad, D. and Kazil, J. (2015). Mesa: an agent-based modeling framework. In *14th PYTHON in Science Conference*, pages 53–60.

Pérez, F. and Granger, B. E. (2007). Ipython: a system for interactive scientific computing. *Computing in Science & Engineering*, 9(3):21–29.

Sichman, J. S., Bousquet, F., and Davidsson, P. (2003). Multi-agent-based simulation ii: Third international workshop, mabs 2002. *Lecture Notes in Artificial Intelligence. Springer*.

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.

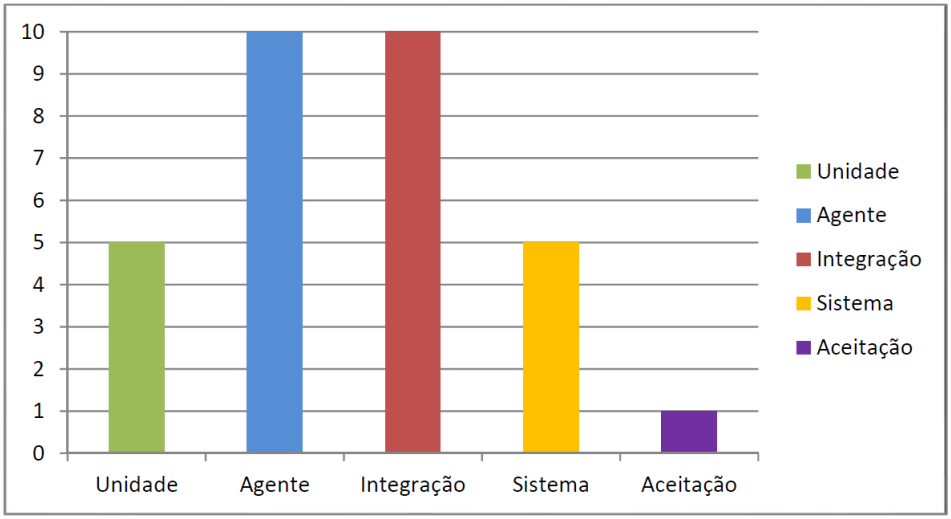


Figura 1. Níveis de Teste

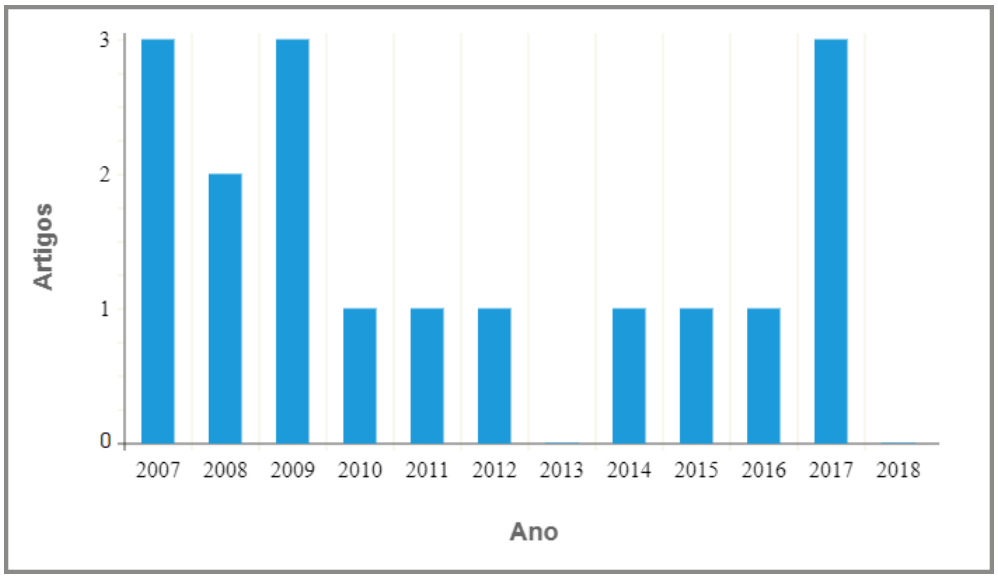


Figura 2. Linha do tempo

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.

Scrumie: Jogo orientado a agentes para ensino de Scrum

Bruna Costa Cons[1], Leonardo Lima Marinho[1], Suelen Regina C. dos Santos[1], Marcelo Schots[2], Vera Maria B. Werneck[1,2]

[1]Programa de Mestrado em Ciências Computacionais, Universidade do Estado do Rio de Janeiro (UERJ), Brasil

[2]Departamento de Informática e Ciência da Computação, Universidade do Estado do Rio de Janeiro (UERJ), Brasil.

{suelen_cordeiro,leonardomarinho_10}@hotmail.com, brunacons94@gmail.com, {schots, vera}@ime.uerj.br

Abstract. *The use of agile methods has become essential in software development at the present time. Among the existing methods, Scrum is one of the major ones, and is used to manage processes in companies, even outside of the scope of software systems development. Considering the relevance of this subject and the success usually obtained in learning through educational games, it was proposed Scrumie, an adaptation of 2TScrum (serious board game). Both games seek to teach Scrum project management. Scrumie added intelligence in multiagent architecture being developed with Agile Passi methodology. This article contains a proposal, modeling and implementation of the Scrumie game.*

Resumo. *O uso de métodos ágeis se tornou imprescindível no desenvolvimento de software na atualidade. Dentre os métodos existentes, o Scrum é um dos principais utilizados para gerenciar processos em empresas, mesmo fora do escopo de desenvolvimento de sistemas. Considerando a relevância deste assunto e o sucesso geralmente obtido no aprendizado por meio de jogos educacionais, foi proposto Scrumie, uma adaptação do 2TScrum (serious game de tabuleiro). Ambos os jogos buscam ensinar o gerenciamento de projetos Scrum. Scrumie adicionou inteligência numa arquitetura multiagentes, sendo desenvolvido com a metodologia Agile Passi. Este artigo apresenta a proposta, modelagem e implementação do jogo Scrumie.*

1. Introdução

O modelo em cascata (*waterfall*), caracterizado por exigir especificações do sistema bem definidas e processos de trabalho longos e sequenciais [Soares 2004] tem sido considerado um padrão para o desenvolvimento de *software*. Com a crescente complexidade dos sistemas e uma maior busca por eficiência nas empresas, este modelo começou a ser considerado inadequado em projetos de pequeno ou médio porte, ou que possuam requisitos dinâmicos. Portanto, surgiu a necessidade de incluir conceitos baseados no manifesto ágil [Beck 2001], com etapas reduzidas e mais curtas, nas quais há entrega incremental para o cliente por meio de versões do sistema.

Os métodos ágeis possuem maior foco na entrega de código do que na documentação, havendo maior integração da equipe, interação com o cliente e aceitação para mudanças nas funcionalidades do sistema. Como o *Scrum* é o *framework* ágil mais utilizado no mercado atualmente [Sille 2013], considerou-se relevante buscar a disseminação e aprendizado de seus conceitos.

Entre as formas de ensino mais eficazes, pode-se citar o uso de jogos educativos, pois, segundo Grando (2001), estes proporcionam uma participação ativa de aprendizado pelo jogador, de forma que conceitos de difícil compreensão possam ser aprendidos e sedimentados. Considerando também que os jogos possam permitir a exploração do conhecimento de forma prática e divertida, através do aspecto lúdico [Legey 2012], foi proposto um jogo de tabuleiro com o intuito de ensinar a prática de gerenciamento de projetos utilizando *Scrum*, o 2TScrum [Brito e Vieira 2017] que obteve bons resultados em relação a sua avaliação.

Neste sentido, o presente trabalho propõe jogo Scrumie, uma adaptação do 2TScrum incorporando o uso de agentes em sua implementação. O jogo 2TScrum tem como limitação separar as atividades entre os participantes da equipe *Scrum*, o que pode ser melhor abordado utilizando a arquitetura multiagentes. No contexto de ensino-aprendizagem, o emprego dos agentes, tem como propósito gerar maior qualidade e flexibilidade, tanto do ponto de vista do aluno quanto do professor. Além disso, a adaptação do jogo usando agentes pôde torná-lo mais acessível e prático, por não exigir o uso de um tabuleiro e peças físicas. O uso de agentes viabilizou a inclusão de um conceito inteligente, em que um agente provê auxílio a jogadores que precisem de dicas do jogo.

Além da introdução, este artigo é composto por mais 5 seções. A seção 2 aborda o método ágil *Scrum*, como este funciona, como a equipe é formada e quais são os seus eventos. A seção 3 apresenta o jogo Scrumie, definindo o modo de funcionamento e suas regras. A seção 4 descreve a metodologia utilizada no processo de desenvolvimento deste trabalho, o Agile PASSI, além de listar as etapas realizadas neste contexto, mostrando em seguida a modelagem realizada no desenvolvimento do jogo, enquanto a seção 5 relata a implementação do jogo. Por fim, a seção 6 apresenta as considerações finais.

2. *Scrum*

O *Scrum* é um *framework* de gestão e planejamento de projetos que segue os ideais ágeis. Ele recomenda a auto-organização da equipe, envolvimento constante do cliente no projeto, prazos de entrega curtos e flexibilidade de adaptação a mudanças.

As atividades básicas em um processo *Scrum* são: pré-planejamento, desenvolvimento e pós-planejamento. No pré-planejamento, as funcionalidades do sistema são relatadas em um documento chamado *backlog*. Cada requisito possui uma prioridade e um custo para ser implementado, e isto também é definido nesta fase, além dos riscos do projeto, ferramentas que serão utilizadas, definição da equipe e proposta de uma arquitetura de desenvolvimento. A fase de desenvolvimento é iterativa, na qual ocorrem as *sprints* (unidades de tempo de um mês ou menos que compartimentalizam o trabalho de um incremento) e onde se observa e controla possíveis mudanças nas variáveis já estabelecidas previamente. Já a fase de pós-planejamento engloba reuniões

realizadas para avaliar o progresso do projeto, testes finais e integração do sistema [Devmedia 2008].

A equipe *Scrum* é formada pelos seguintes papéis: *Product Owner,* o responsável pelo gerenciamento do *backlog* do produto; o *Scrum Master,* que garante a compreensão do *Scrum* pela equipe, além de resolver problemas que impedem o sucesso da *sprint*; e a equipe de desenvolvimento, responsável pela criação dos incrementos utilizáveis do produto.

O *Scrum* define a realização de reuniões para manter a equipe integrada junto ao cliente. A reunião de planejamento da *sprint* possui o intuito de definir os itens a serem considerados na *sprint* atual. A reunião diária é uma reunião curta, realizada para acompanhar as atividades atuais e planejar as atividades do dia. Há também duas reuniões mais informais, a revisão e a retrospectiva da *sprint*, com o objetivo de apresentar e avaliar os resultados obtidos ao fim da *sprint*. A partir das reuniões, artefatos são gerados, como o *backlog* do produto – uma lista de requisitos do sistema – e o *backlog* da *sprint*, que relata as funcionalidades selecionadas do *backlog* do produto que precisarão ser desenvolvidas na *sprint* atual [Brito e Vieira 2017].

3. O jogo Scrumie

O Scrumie possui mecanismo de jogo semelhante ao 2TScrum, no qual um jogador deve seguir um fluxo que o permite gerenciar projetos de *software*, aplicando seus conhecimentos teóricos sobre *Scrum* e estimulando novos conhecimentos. Para isto, o jogo simula situações cotidianas de desenvolvimento de *software*, motivando o jogador a ter autonomia para tomar decisões a respeito de acontecimentos no projeto que devem ser gerenciados [Brito e Vieira 2017].

Com o objetivo de instigar o aprendizado no jogo, o jogador precisa finalizar o desenvolvimento do projeto dentro do prazo e orçamento estabelecidos pelo cliente no início da partida. No decorrer do jogo, o desempenho do jogador será avaliado para que dicas possam ser dadas a ele, auxiliando-o no gerenciamento do projeto.

Igualmente ao 2TScrum, o Scrumie apresenta os mesmos elementos que contribuem para a interação do jogador com o jogo: a carta do cliente, as cartas de *backlog*, a carta de validação do *backlog*, as cartas com perfil do desenvolvedor, as cartas surpresa e as cartas de eventos [Brito e Vieira 2017].

Os componentes que integram o jogo Scrumie são representados por quase todos os papéis da equipe *Scrum*, sendo o jogador responsável pelo papel do gerente de projeto e do *Product Owner;* o agente Cliente responsável pelo papel do cliente; e o agente *Scrum Master* responsável pelo papel do *Scrum Master*. No entanto, novas adaptações foram feitas em relação aos papéis dos agentes Cliente e *Scrum Master*. O agente Cliente é o encarregado de criar cartas do cliente, cartas de *backlog* e cartas de validação do *backlog,* de acordo com o cenário elaborado para a partida. Já o *Scrum Master* é o encarregado de apresentar dicas ao jogador mediante o seu desempenho, e selecionar aleatoriamente cartas das respectivas reuniões presentes no *Scrum*.

Dois novos componentes foram adicionados com o objetivo de auxiliar a dinâmica do jogo. Um deles é o agente Gerenciador de Progresso, responsável pela avaliação de desempenho do jogador, que mantém comunicação com o agente *Scrum Master* a respeito do desempenho apresentado e gerenciando a pontuação do jogador ao

longo do jogo. Há também o agente Executor de Regras, responsável por tratar as violações de regras.

3.1. Etapas do jogo

A interação entre o jogador e o jogo Scrumie, nessa primeira versão é realizada por meio de uma interface de linha de comando sob a forma de sucessivas linhas de texto. Baseado em Brito e Vieira (2017), a seguir são detalhadas as etapas do fluxo principal que o jogador percorre durante uma partida.

Antes de dar início ao jogo, o agente Cliente apresenta para o jogador a seguinte narrativa [Brito e Vieira 2017]: "Um novo projeto chega à empresa e você é alocado para gerenciá-lo utilizando o *framework Scrum.* Este projeto se trata de um sistema *web* para uma biblioteca, com um orçamento e prazo fixos que você deve cumprir, mas no decorrer do tempo há diversos acontecimentos que você deve gerenciar. Além desses acontecimentos, durante a construção do produto, o seu cliente acaba entendendo melhor o que necessita e mudanças podem acontecer. Você deve manter o projeto no prazo e no orçamento estabelecidos, mas será que é possível? Você aceita esse desafio?".

Após a leitura da narrativa, o jogador inicia a partida inserindo seu nome no sistema. Em seguida, o agente Cliente apresenta a carta do cliente, informando o produto desejado, o orçamento e o prazo para o projeto. Posteriormente, o jogador segue para a criação do *backlog* do produto, onde o agente Cliente apresenta trinta e seis itens do *backlog*, exibidos com um identificador, descrição do item e estimativa de tempo para o desenvolvimento. O jogador deve escolher apenas os itens esclarecidos na carta do cliente.

Assim que o jogador montar o *backlog* do produto, o agente Gerenciador de Progresso passa a calcular a pontuação do jogador com base no prazo e no custo divulgados nos itens selecionados. Este agente mantém o gerenciamento da pontuação do jogador ao longo do jogo, a partir dos acontecimentos e modificações presentes nas cartas das próximas etapas.

O jogador segue para a reunião com o cliente e, neste momento, o agente Cliente passa a validar os itens do *backlog* do produto segundo a carta de validação do *backlog* concedida por ele. O agente ajusta o *backlog* do produto de acordo com essa carta, incluindo os itens dela que não estão no *backlog* do produto e descartando os itens que não estão presentes nela. Se o agente verificar que o jogador acertou menos de oito itens no *backlog* do produto, ele aplica uma penalidade ao jogador. Caso contrário, oferece uma recompensa.

Depois de validado o *backlog* do produto, o jogador deve escolher três perfis de desenvolvedores para sua equipe de desenvolvimento. No caso, é exibido para o jogador cartas com os perfis dos desenvolvedores, informando a classificação, o custo monetário por *sprint* e o bônus de cada um. O agente Executor de Regras é comunicado para verificar a quantidade de desenvolvedores escolhidos pelo jogador.

Com a equipe definida, o jogador faz o planejamento da *sprint*, onde cria o *backlog* da *sprint* escolhendo no máximo sete itens do *backlog* do produto. O agente Executor de Regras é comunicado para verificar a quantidade de itens escolhidos para o *backlog* da *sprint*. Com a verificação feita e aprovada, assim que o *backlog* da *sprint* é

criado, o agente Gerenciador de Progresso atualiza o prazo do projeto, decrementando o total de tempo estimado para os itens do *backlog* da *sprint*.

Ainda no planejamento, o agente *Scrum Master* sorteia aleatoriamente duas cartas de planejamento da *sprint* e as apresenta para o jogador. Após a leitura das cartas, o jogador deve tomar decisões sobre os acontecimentos descritos nelas, considerando o prazo e o custo. Com as escolhas feitas, o agente Gerenciador de Progresso avalia o desempenho do jogador com base na média de desempenho aceitável de um jogador na partida. Caso este verifique que a atuação do jogador atual está ruim, comunica isso ao agente *Scrum Master*, que prontamente apresenta dicas ao jogador em relação às cartas de planejamento da *sprint*. Depois, o agente *Scrum Master* sorteia aleatoriamente duas cartas surpresa, aplicando suas alterações e apresentando as cartas para o jogador.

Finalizada a etapa do planejamento, segue-se para a etapa de desenvolvimento, na qual o agente *Scrum Master* sorteia aleatoriamente duas cartas de reunião diária e apresenta as cartas para o jogador. Novamente, o jogador deve considerar as informações nas cartas para tomar decisões. Feito isso, o Gerenciador de Progresso faz mais uma avaliação de desempenho, repetindo o mesmo processo descrito acima.

Ao ter gerenciado os acontecimentos presentes na reunião diária, o jogador segue para a revisão da *sprint*, onde o agente *Scrum Master* sorteia aleatoriamente uma carta de revisão da *sprint* e a apresenta para o jogador. Esta carta mostra dificuldades e soluções identificadas pelo Cliente ao expor o incremento do sistema na reunião. Baseado na informação recebida, o jogador deve realizar uma escolha. Em seguida, o Gerenciador de Progresso atua novamente da forma já descrita.

Depois da reunião de revisão, o jogador segue para a retrospectiva da *sprint*, onde o agente *Scrum Master* sorteia aleatoriamente uma carta de retrospectiva da *sprint* e a exibe para o jogador. Esta carta mostra acontecimentos positivos ou negativos da *sprint* que devem ser lidados pela equipe na próxima *sprint*. Da mesma forma que antes, o jogador deve tomar uma decisão e o agente Gerenciador de Progresso irá avaliá-la, tomando as medidas necessárias.

Finalmente, o agente Cliente é notificado para validar se todos os itens do *backlog* do produto foram desenvolvidos. Caso sobrem itens no *backlog* do produto, o agente Cliente informa ao agente Executor de Regras que o jogo não pode ser finalizado. Este comunica ao jogador sobre o retorno para o planejamento da *sprint*, pois o produto não foi concluído, iniciando mais uma *sprint*. Caso contrário, o Cliente informa para o jogador que o jogo foi finalizado. Em seguida, são apresentadas ao jogador sua pontuação final e sua posição no ranking. O ranking indica um *feedback* de desempenho para o jogador, levando em consideração o prazo e o orçamento resultantes no final do projeto em relação aos outros jogadores. Por último, o agente Gerenciador de Progresso atualiza a pontuação média de desempenho de um jogador com a pontuação final da partida.

4. Modelagem do sistema

O Agile PASSI é a metodologia ágil orientada a agentes utilizada no desenvolvimento do jogo Scrumie. Ela surgiu a partir da metodologia PASSI, criada inicialmente para desenvolver um sistema robótico, incluindo processos ágeis oriundos do método *Extreme Programming*, onde há a liberação de versões iterativas do sistema [Chella 2006]

É uma metodologia orientada a código, que valoriza mais o desenvolvimento do código do que a criação da documentação do mesmo. Há a identificação de agentes como um conjunto de funcionalidades expressas em casos de uso e a utilização de ontologias para descrever o domínio do sistema [Chella 2004]. A maioria das etapas de desenvolvimento são auxiliadas por automatizações e reutilização de código, por meio do *framework* disponibilizado, *Agent Factory*. Ele realiza um trabalho de engenharia reversa, construindo automaticamente um esboço das classes dos agentes a partir dos diagramas que os identificam [Chella 2006] ,[Cons 2018].

Originalmente, a metodologia recomenda a reutilização de padrões no código através da ferramenta *Agent Factory,* além da criação de ontologias e de histórias de usuário ao planejar as funcionalidades do sistema. No entanto, estas especificidades não foram utilizadas no desenvolvimento do trabalho.

A princípio, foi criado um fluxo principal do jogo, como uma lista de atividades, posteriormente utilizado para a criação dos diagramas. Algumas adaptações foram realizadas em relação às recomendações do Agile PASSI: manteve-se a sugestão de criação do diagrama de caso de uso, enquanto os diagramas de identificação de agentes e o diagrama de identificação de papéis foram incluídos. Os estereótipos usados vêm do padrão UML [Larman 2002].

Nesta etapa também foi elaborada uma lista contendo as principais regras de negócio do jogo. Isso porque estas regras detalham as funcionalidades particulares do *software* presentes nos diagramas, para satisfazer o cliente e o objetivo do negócio [Dallavalle e Cazarini 2000].

4.1. Diagrama de Caso de Uso

Este diagrama auxilia no levantamento dos requisitos funcionais do sistema Scrumie. A Figura 1 apresenta parte do diagrama que descreve um conjunto de funcionalidades do jogo.

Assim como o diagrama exemplificado na Figura 1, outros diagramas de caso de uso também foram elaborados para estabelecerem restrições que permitem o funcionamento correto do jogo.

4.2. Diagrama de Identificação de Agentes

De acordo com Henderson-Sellers (2005), a elaboração do diagrama de identificação de agentes começa a partir do diagrama de casos de uso já produzido. Isto é, ele é formado com base no agrupamento de um ou mais casos de uso em pacotes estereotipados. E, ao fazer isso, cada pacote passa a definir as funcionalidades de um agente específico. Além disso, os nomes dos pacotes são os nomes dos agentes representados.

Uma das principais características deste diagrama é o seu relacionamento, onde relacionamentos entre casos de uso de diferentes agentes são estereotipados como “communicate”, enquanto relacionamentos entre casos de uso do mesmo agente são modelados usando “include” e “extend” [Henderson-Sellers 2005]. Foram identificados 4 agentes: (i) Cliente, (ii) Executor de Regras, (iii) Scrum Master e (iv) Gerenciador de Progresso. A Figura 2 representa um exemplo deste diagrama no cenário do jogo Scrumie para os agentes Cliente e Executor de Regras.

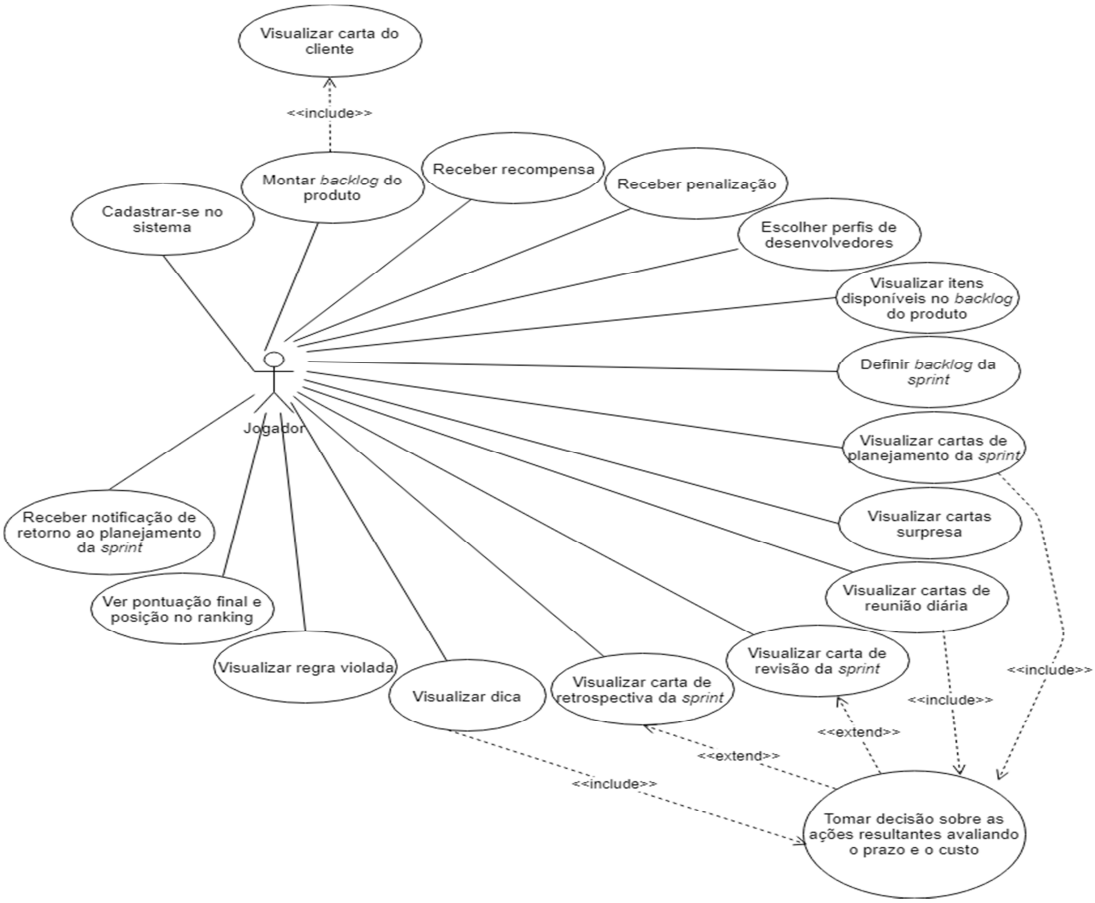


Figura 1. Parte do diagrama de caso de uso voltada às atividades do jogador

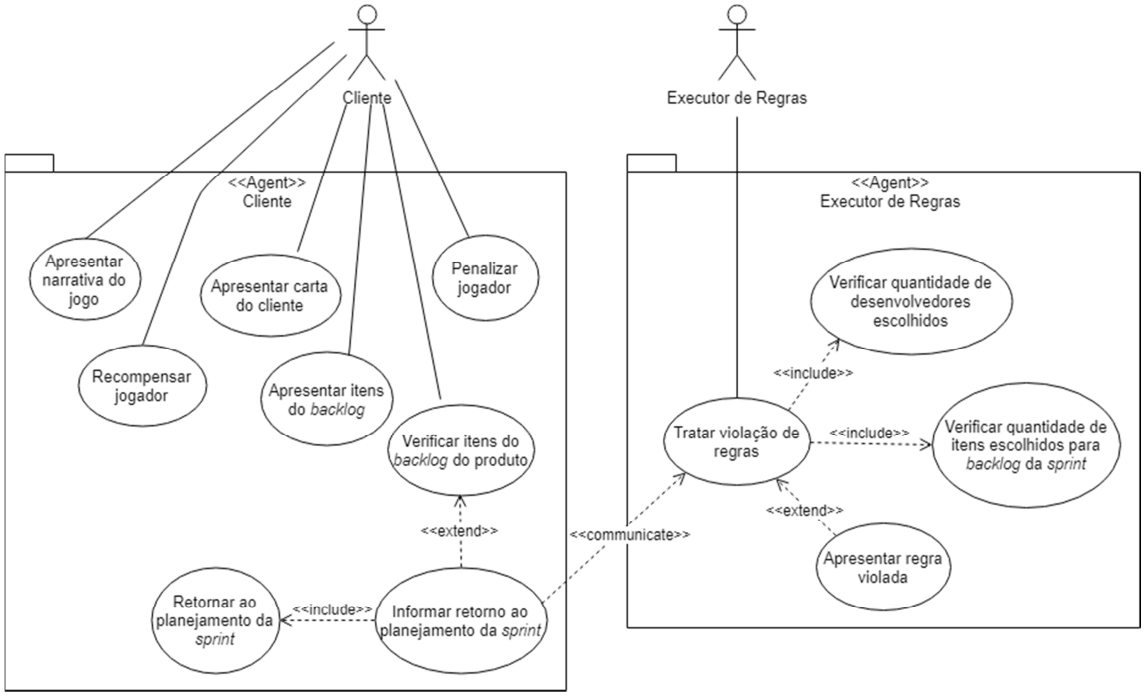


Figura 2. Parte do diagrama de identificação de agentes

4.3. Diagrama de Identificação de Papéis

Esse é um diagrama de sequência UML, onde cada objeto é utilizado para simbolizar o papel do agente, cuja sintaxe é <nome-do-papel>: <nome-do-agente>. Um agente pode desempenhar papéis distintos dentro do mesmo diagrama [Henderson-Sellers 2005].

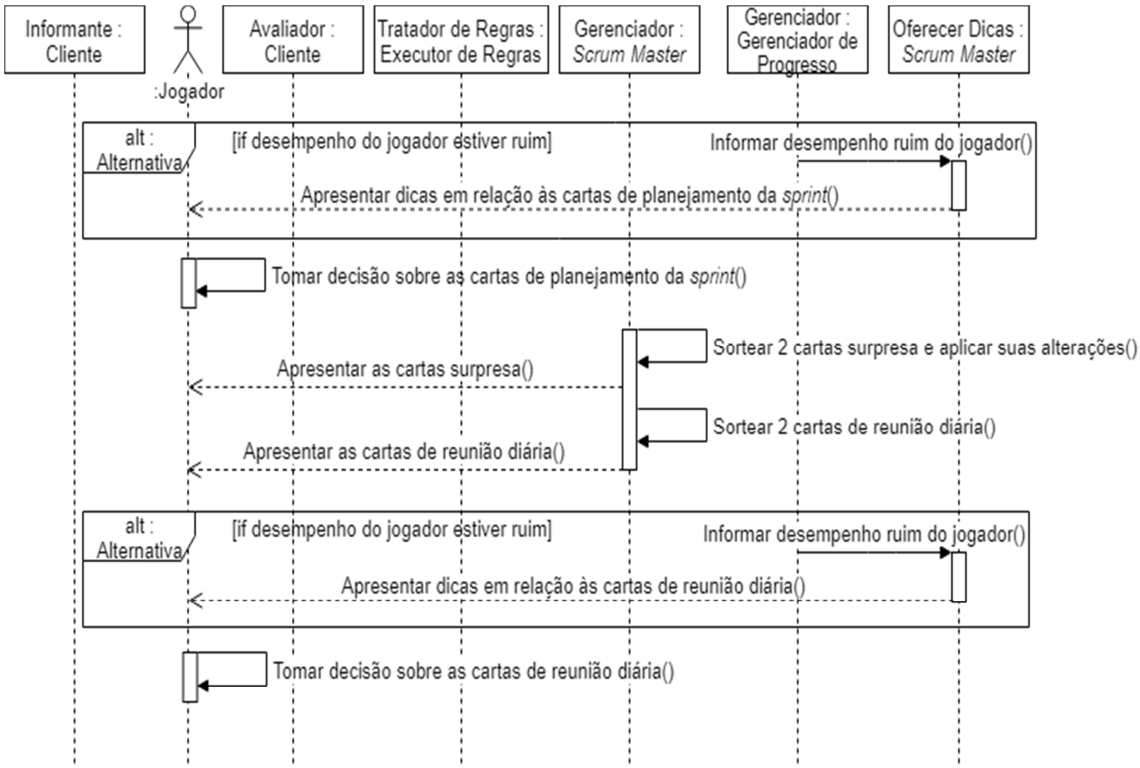


Figura 3. Diagrama de identificação de papéis

A Figura 3 mostra um exemplo do diagrama, que explora a troca de mensagens ao longo de parte do cenário do jogo Scrumie, envolvendo comunicação entre agentes e seus papéis. A elaboração deste diagrama também parte do princípio de que o diagrama de identificação de agentes já foi desenvolvido.

5. Implementação do jogo

Após a modelagem, deu-se início à implementação do jogo. Nesta etapa, primeiramente foram escolhidas as tecnologias utilizadas. Em seguida, foi definida a organização dos componentes do sistema, como arquivos de configuração e agentes, a partir das informações sobre o jogo 2TScrum e da modelagem. Com estas informações estabelecidas, o sistema foi desenvolvido.

Ao decidir a partir de quais tecnologias o jogo seria construído, levou-se em conta a necessidade da velocidade de desenvolvimento e da capacidade da tecnologia em representar bem a arquitetura multiagentes. Neste ponto, optou-se por não usar um *framework*, uma vez que isso demandaria um período de estudos e compreensão do mesmo, podendo atrasar a entrega da aplicação, e indo além da simplicidade proposta na modelagem.

Com essas ideais em consideração, optou-se pela utilização da plataforma Node.js[1], que é capaz de executar a linguagem de programação JavaScript em aplicações *desktop*. Dessa forma, o código foi escrito em TypeScript[2], uma linguagem que é compilada para JavaScript e que fornece a versão mais recente do mesmo, agregando funcionalidades, como um melhor suporte à tipagem de dados e à orientação a objetos. Como o JavaScript é a linguagem de programação utilizada em todos os navegadores modernos e uma das mais usadas no mercado [Salles 2018], o código já escrito pode ser facilmente portado para uma plataforma web, podendo ser executado em praticamente qualquer dispositivo com acesso à mesma.

Depois das tecnologias terem sido definidas, a organização da aplicação foi pensada com os objetivos de facilitar a utilização de boas práticas de desenvolvimento e de agilizar a implementação. Para tal, separou-se o sistema nos seguintes agentes: (i) Cliente, responsável pela inicialização e por apresentar o cenário da jogabilidade; (ii) Gerenciador de Progresso, responsável pelo controle do fluxo de interações entre o jogador e o sistema; (iii) Scrum Master, que gerencia todas as cerimônias do Scrum; (iv) Executor de Regras, que valida as atividades do jogador

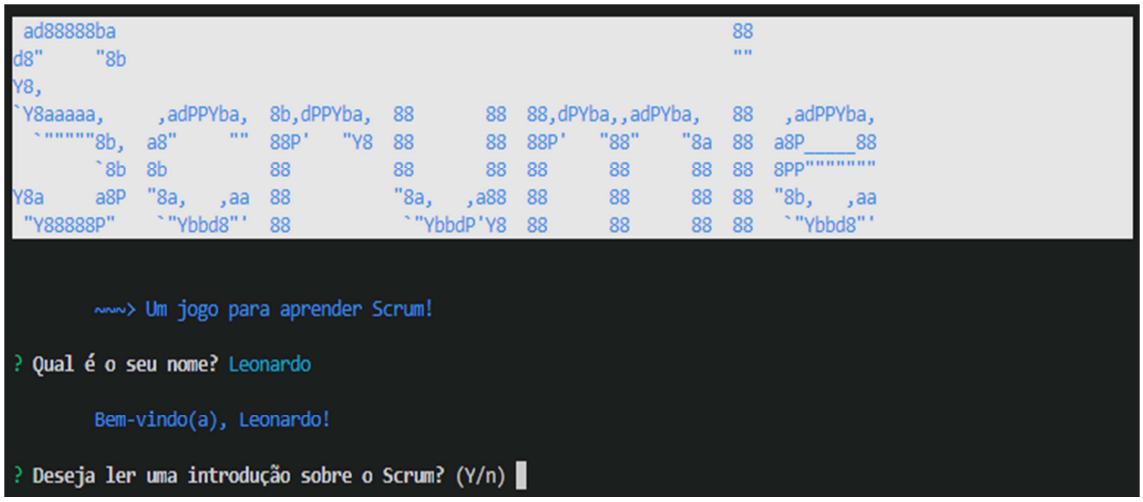


Figura 4. Tela inicial do jogo Scrumie

Além disso, optou-se por não utilizar banco de dados, já que nesta versão apenas as informações sobre a classificação dos jogadores são armazenadas. As configurações, do jogo são carregadas e armazenadas durante a execução como objetos. Como modo de

[1] Disponível em: http://nodejs.org/en/about/

[2] Disponível em: https://www.typescriptlang.org/

interação com o jogador, a interface de linha de comando foi a escolhida, devido à sua simplicidade e eficiência.

Durante a implementação, algumas das técnicas utilizadas foram: reuso de código através de herança e de classes auxiliares, nomes bem definidos, formatação padronizada e funções pequenas e objetivas. Deste modo, eventuais versões futuras do jogo terão sua manutenção e seu desenvolvimento facilitados.

A implementação dos agentes proporcionou a adição de inteligência ao jogo, além de ter facilitado a separação das responsabilidades e dos papéis dos módulos do sistema, permitindo uma melhor organização.

6. Conclusões

O jogo Scrumie representa a maior contribuição deste artigo, como proposta de investir no aprendizado prático por meio do uso de jogos educativos em *software*. Considera-se um estudo relevante devido à busca crescente por eficiência em empresas no desenvolvimento de sistemas e à consequente importância dada ao conhecimento de como aplicar métodos ágeis.

O uso de agentes facilitou a separação de papéis envolvidos na equipe *Scrum*, apesar de trazer maior complexidade para a implementação. Por ser uma arquitetura diferente da convencional, utilizar agentes foi um desafio, no sentido de saber inserir a autonomia e a inteligência geralmente associadas aos agentes da maneira correta. Esse primeiro protótipo foi desenvolvido de forma bastante simples e no futuro pretende-se a incorporação de uma arquitetura BDI por meio de um framework orientado a agentes. Assim, a abordagem de ter agentes representando pessoas de uma equipe serviu como uma solução para este problema abstrato, gerando bons resultados. No futuro, esses agentes podem evoluir, estendendo o jogo para uso de múltiplos papéis.

Alguns pontos negativos originados no 2TScrum não foram abordados para melhoria no Scrumie, como o fato de possuir apenas um cenário de desenvolvimento simulado – o sistema *web* para uma biblioteca; e possuir um limite de participantes no jogo, sendo que o Scrumie admite apenas um jogador. Sendo assim, esses pontos ficam sugeridos como melhorias futuras. Além deles, em relação à implementação, próximas versões do jogo poderiam criar uma interface gráfica, para torná-lo mais atraente aos jogadores; e migrá-lo para uma implementação *web*, para que tenha mais acessibilidade.

Referências

Beck, Kent. Embracing change with extreme programming. Computer, v. 32, n. 10, p. 70-77, 1999.

Beck, Kent; et al. Manifesto for agile software development. Disponível em: <http://agilemanifesto.org/>. Acesso em 12 de dez. de 2018.

Brito, A.; Vieira, J. '2TScrum': A Board Game to Teach Scrum. In: Proceedings of the 31st Brazilian Symposium on Software Engineering. ACM, 2017. p. 279-288.

Chella, Antonio; et al. Agile PASSI: An agile process for designing agents. International Journal of Computer Systems Science & Engineering, vol. 21, no. 2, p. 133-144, 2006.

Chella, Antonio; et al. From passi to agile passi: Tailoring a design process to meet new needs. In: Intelligent Agent Technology, (IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on. IEEE, p. 471-474, 2004.

Cons, Bruna. Comparação entre Metodologias Ágeis Orientadas a Agentes. 2018. 71 f. Monografia (Graduação em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2018.

Dallavalle, Silvia Inês; Cazarini, Edson Walmir. Regras do Negócio, um fator chave de sucesso no processo de desenvolvimento de Sistemas de Informação. Encontro Nacional de Engenharia de Produção. São Paulo, 2000.

De Paula, Felipe. MAS Ontology: uma ontologia de métodos orientados a agentes. 2014. 160 f. Dissertação (Mestrado em Ciências Computacionais) - Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro. 2014.

Devmedia, Equipe. Processos Ágeis para desenvolvimento de Software – Parte 02. DevMedia. Disponível em: <https://www.devmedia.com.br/processos-ageis-para-desenvolvimento-de-software-parte-02/9209>.

Ferreira, Vitor. Comparação de Desenvolvimento Orientado a Agentes para Jogos Educacionais: Um estudo de caso. 2015. 163 f. Dissertação (Mestrado em Ciências Computacionais) - Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro. 2015.

Grando, Regina. O jogo na educação: aspectos didático-metodológicos do jogo na educação matemática. Unicamp, 2001 <www.cempem.fae.unicamp.br/lapemmec/cursos/el654/2001/jessica_e_paula/JOGO.doc> Acesso em 18/maio/2003.

Henderson-Sellers, Brian; Giorgini, Paolo. Agent-oriented Methodologies. Hershey: Idea Group, 2005. 429 p.

Knublauch, Holger. Extreme programming of multi-agent systems. Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 2 - AAMAS '02, 2002.

Larman, Craig. Utilizando UML e padrões. Bookman Editora, 2002.

Legey, Ana Paula et al. Desenvolvimento de Jogos Educativos Como Ferramenta Didática: um olhar voltado à formação de futuros docentes de ciências. Alexandria: Revista de Educação em Ciência e Tecnologia, v. 5, n. 3, p. 49-82, 2012.

Moratori, Patrick. Por que utilizar jogos educativos no processo de ensino aprendizagem. UFRJ. Rio de Janeiro, 2003.

Pressman, Roger. Engenharia de Software. Edição 6. Porto Alegre: MCGrawHill, 2010. 711 p.

Salles, Filipe. Top 10 linguagens de programação mais usadas no mercado. DevMedia. Disponível em: <https://www.devmedia.com.br/top-10-linguagens-de-programacao-mais-usadas-no-mercado/39635>.

Shehory, Onn; Sturm, Arnon. Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks. Berlim: Springer-Verlag Berlin Heidelberg, 2014. 331 p.

Sille, Felipe; Braga, Juliana Cristina. Software educacional para prática do scrum. In: Anais dos Workshops do Congresso Brasileiro de Informática na Educação. 2013.

Soares, Michel. Comparação entre metodologias Ágeis e tradicionais para o desenvolvimento de software. INFOCOMP, v. 3, n. 2, p. 8-13, 2004.

Sommerville, Ian. Engenharia de Software. Edição 9. Tradução Ivan Bosnic; Kalinka Oliveira. São Paulo: Pearson, 2013. 529 p.

Decisão por parceiros de Transferência de Tecnologia: uma abordagem baseada em agentes

Adriana Neves dos Reis[1,2]

[1]Instituto de Ciências Criativas e Tecnológicas (ICCT) – Universidade Feevale
ERS-239, 2755 – 93.525-075 – Novo Hamburgo – RS – Brasil

[2]Escola de Arquitetura, Engenharia e TI – Centro Universitário Ritter dos Reis
Orfanotrófio, 555 – 90.840-440 – Porto Alegre – RS – Brasil

adriana.anreis@gmail.com

Abstract. *Technology Transfer (TT) is the result of the interaction between actors who agree to exchange ownership, knowledge and value. However, the decision involved in the selection by these partners is open in the literature. This research proposes an artifact for decision-making support that considers this transaction as a Complex Adaptive System (CAS). Thus, the proposal provides a construct vocabulary that serves as an analytical framework for modeling both the structure and behavior of the interaction between abstracted TT actors and agents. The results show that the artifact assists in the identification of decision criteria and in the observation of emerging patterns resulting from the dynamics of agents.*

Resumo. *Transferência de Tecnologia (TT) é o resultado da interação entre atores que concordam em trocar posse, conhecimento e valor. Porém, a decisão envolvida na seleção por esses parceiros é pouco explorada. Esta pesquisa propõe um artefato para suporte à tomada de decisão que considera essa transação como um Sistema Adaptativo Complexo (SAC). Assim, a proposta oferece um vocabulário de construtos que servem de referencial analítico para modelar tanto a estrutura quanto o comportamento da interação entre atores de TT abstraídos como agentes. Os resultados evidenciam que o artefato auxilia na identificação de critérios de decisão e na observação de padrões emergentes resultantes da dinâmica de agentes.*

1. Introdução

Um processo de tomada de decisão envolve a seleção da melhor opção entre um conjunto de muitas alternativas [Bulling 2014]. No contexto de formação de parcerias para a realização de Transferência de Tecnologia (TT), essa escolha implica não apenas decidir como e/ou com quem negociar, mas também em optar por não fazer uma dada transferência [Speser 2006].

Além disso, de acordo com a Teoria da Decisão, é necessário caracterizar o que o decisor entende como a melhor alternativa, pois o adjetivo melhor pode assumir diferentes significados [Parsons e Wooldridge 2002]. Assim, para TT, essa ideia é ajustável ao conceito de utilidade e de valor esperado desta transação [Speser 2006].

Para que uma transferência dessa natureza ocorra, as organizações envolvidas realizam a negociação de uma tecnologia, usualmente em estágio recente, que resulta em transação firmada por meio de um acordo [Speser 2006]. Além disso, deve-se considerar que esse tipo de relação envolve, de modo geral, diferentes classes de atores, entre elas: universidade, empresa e governo.

Contudo, elas não possuem os mesmos objetivos em uma operação de TT. A universidade normalmente almeja resultados científicos, como publicações em revistas científicas, doutorados, etc.; a empresa, por sua vez, tem por objetivo comum o lucro; e o governo, geralmente, dá maior relevância para a transparência dos procedimentos financeiros, a fim de assegurar o uso adequado do dinheiro dos contribuintes [Hilkevics 2014].

Atores que pertencem a estas classes interagem em um ambiente dinâmico e complexo em que se dá o desenvolvimento tecnológico e, sendo assim, a interface entre eles é considerada significante. Além disso, deve-se considerar que os decisores responsáveis por estabelecer uma relação de TT avaliam os cenários para a tomada de decisão com racionalidade limitada. A Figura 1 traz uma visão esquemática de como os decisores interagem nesse contexto.

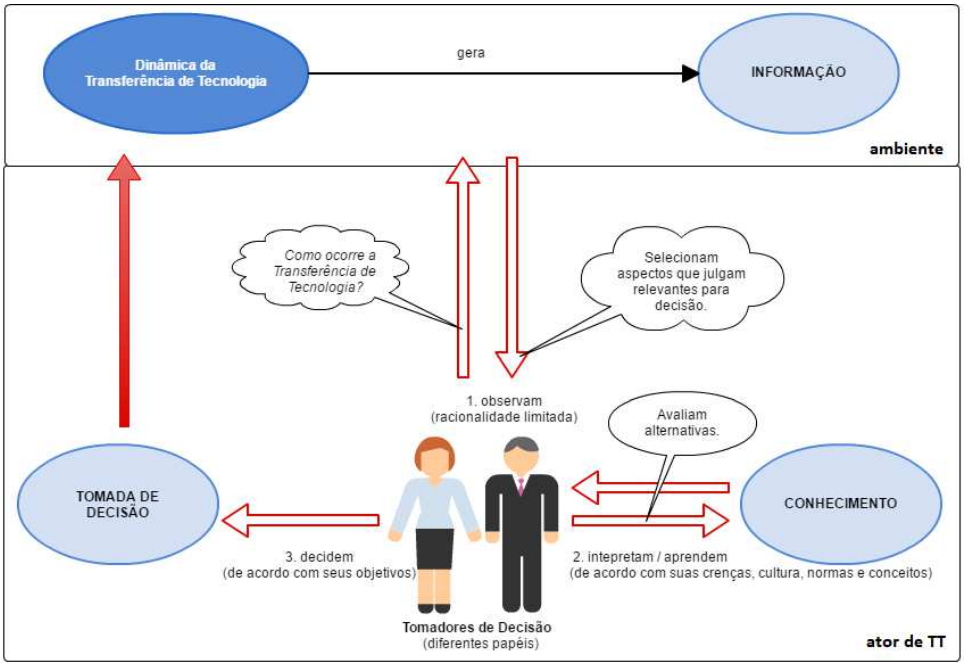


Figura 1. Esquema do Contexto de Decisão dos Atores de TT

Na literatura não é identificada uma estruturação consolidada sobre os elementos especificadores de uma relação de TT. Além disso, o ponto de vista dinâmico desta interação também é pouco descrito, fato que dificulta a concepção de modelos que permitam avaliar qual o real impacto de uma tomada de decisão neste processo na prática.

Uma possível alternativa é compreender a dinâmica dessas relações e descrevê-la como uma instância inserida nos Sistemas Adaptativos Complexos (SAC) [Watts e Gilbert 2014], em que cada ator é entendido como um agente. SAC é uma abordagem útil para entender a realidade [Furtado et al. 2015]. Nela, duas características se

destacam: a emergência, ou seja, a percepção de que as atitudes individuais das partes ao interagirem geram resultados emergentes, e a realimentação, como a constatação de que esses resultados retornam afetando as atitudes individuais [Wilensky e Rand 2015].

Desse modo, esta pesquisa tem como objetivo construir um artefato de suporte à abstração dos atores de TT e o ambiente dinâmico em que eles interagem de acordo com um SAC. Almeja-se, desse modo, contribuir para a elaboração de cenários de experimentação mais realísticos para tomada de decisão, capazes de reproduzir as dinâmicas complexas resultantes das decisões de firmar acordos de TT.

Este artigo está organizado em 7 seções. A próxima seção descreve o método de trabalho empregado na pesquisa. A seção 3 trata da problematização do processo decisório em TT, seguida da seção 4, a qual faz um apanhado dos trabalhos relacionados a este estudo. A seção 5 aborda o desenvolvimento do artefato propriamente dito. Seção 6 reúne os experimentos de validação, seguida da seção 7 com as considerações finais.

2. Procedimentos Metodológicos

Para o desenvolvimento desta pesquisa foi adotado o método *Design Science Research* (DSR). Esse método está inserido na *Design Science*, ou ciência do projeto, a qual é uma alternativa para pesquisas científicas que não são atendidas de forma satisfatória pelas ciências tradicionais [Dresch et al. 2015].

De acordo com [Van Aken 2004], o objetivo desta abordagem é trabalhar com o desenvolvimento de novos conhecimentos para a construção de artefatos. [Bax 2014] ainda destaca que este método de pesquisa envolve a construção, validação e avaliação dos artefatos gerados pela pesquisa. Esse artefato pode ser entendido como algo artificial, como um modelo. Na Figura 2 é representado esquematicamente o fluxo de DSR empregado neste trabalho.

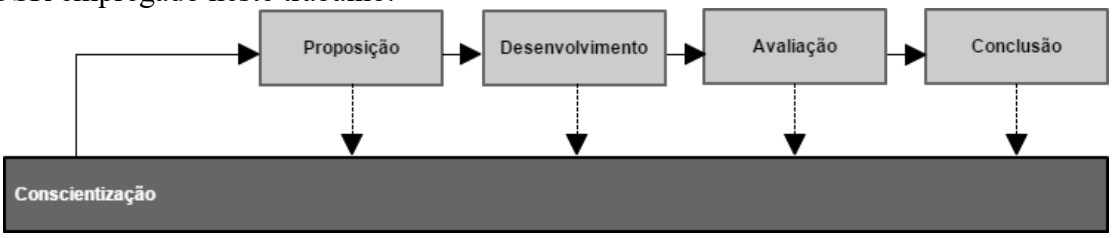


Figura 2. Método de Trabalho utilizando *Design Science Research*.

Cada uma das etapas do DSR foi planejada conforme segue:

- Conscientização: com base na revisão da literatura de trabalhos que abordassem a sistematização de informações sobre as parcerias entre atores de TT, esta etapa teve por objetivo consolidar elementos das interações entre atores de TT como um SAC.

- Proposição: a partir da análise dos resultados da revisão da literatura, neste ponto foi verificada a viabilidade de adotar os elementos de modelagem identificados para abstrair a dinâmica de parcerias de TT, assim como investigados possíveis tipos de artefatos para solucionar o problema.

- Desenvolvimento: construção do artefato propriamente dito, um vocabulário dos construtos base para entender TT como um SAC, permitindo a criação de modelos simuláveis de TT.

- Avaliação: a partir das abstrações que compõem o artefato desenvolvido, experimentações foram realizadas empregando simulações baseadas em agentes.

- Conclusão: especificação do fluxo de modelagem e simulação para o uso do artefato para suporte à modelagem de tomada de decisão por parceiros de TT.

A próxima seção trata da visão geral de decisão por parceiros de TT, para, posteriormente, descrever a condução do estudo de acordo com as etapas descritas.

3. Decisão em TT

Em função das diferentes alçadas em que seus impactos são observados [Bozeman 2000], TT tem sido do interesse tanto do meio acadêmico e científico, quanto das organizações empresariais e decisores políticos [Pagani 2016]. Do ponto de vista prático, um decisor tem informação incompleta, o que torna as consequências resultantes de sua decisão repletas de incertezas [Bulling 2014]. Do mesmo modo, os atores decidem no contexto de TT com racionalidade limitada. Neste sentido, a área de Teoria da Decisão preocupa-se em reunir técnicas que permitam o decisor analisar cenários em um ambiente imprevisível [Parsons e Wooldridge 2002]. Esses cenários estão geralmente inseridos em problemas de alta complexidade, representando diferentes ações que podem ser tomadas [Ragsdale 2011], tendo como suporte técnico diferentes propostas de modelagem.

Neste sentido, identifica-se a possibilidade de adotar a perceptiva de SAC para modelar o processo de decisão por parcerias em TT. De acordo com [Chwif e Medina 2015], esse paradigma consiste na proposição de um modelo de sistema baseado em um conjunto de entidades com decisão autônomas, denominadas agentes, que residem em um ambiente, um mundo virtual discreto ou contínuo, e que são capazes de interagir tanto entre si quanto com o seu ambiente.

Os autores ainda destacam que se trata de uma abordagem *bottom-up* (de baixo para cima), uma vez que é a partir do comportamento individual dos agentes que se obtém o comportamento global do sistema. Assim, um modelo baseado em agentes tem como base a construção de agentes com funções simples. Quando estes indivíduos interagem, considerando suas funções, comportamentos específicos emergem [Bordini et al. 2007].

Assim, na abstração de um SAC, os agentes simulam os elementos básicos do processo da tomada de decisão [Nan et al. 2014]. Em outras palavras, um agente é uma representação de um indivíduo com características específicas interagindo com outros indivíduos em um contexto compartilhado. Além disso, o modelo baseado nesta abstração caracteriza-se como um sistema reativo que exibe uma dada autonomia para determinar o "quão bem" ele é capaz de executar uma tarefa para ele delegada [Bordini et al. 2007].

A abordagem baseada em agentes tem sido adotada em estudos para permitir a definição de modelos, os quais analisados em cenários reais, consequentemente, servem de base para a proposição de recomendações de ação nestes contextos [Kiesling et al.

2012; Watts e Gilbert 2014]. [Kiesling et al. 2012] apresentam um levantamento do SAC como estratégia em estudos de difusão de tecnologia. Nele, os autores destacam a relevância desta abordagem como uma ferramenta para promover suporte à decisão baseada em dados empíricos. Do ponto de vista da dinâmica, um SAC pode abstrair qualquer sistema que tem agentes interagindo localmente com regras simples, sem um controle centralizado [Furtado et al. 2015]. Assim, ele é um tipo de sistema que evolui constantemente, revelando seu comportamento ao longo do tempo [Ajzental 2015]. A modelagem de um problema como um SAC, portanto, contempla cinco componentes: agentes, interação, emergência, ambiente e realimentação [Furtado et al. 2015].

Para adotar uma abordagem baseada em agente em TT, é necessário questionar "Quais são os elementos (atores, contexto e interação) a serem considerados para compor a análise de decisão dos agentes para concretização de negócios de TT?". A resposta a essa questão traz como resultado um meta-modelo e sua semântica correspondente para descrever os agentes e simular a dinâmica dos cenários de TT, a fim de oferecer suporte à decisão.

Também deve-se considerar que os modelos não ditam o quê e como o decisor deve pensar, mas eles oferecem uma forma de reduzir a complexidade de uma situação, concentrando a atenção no que realmente é importante [Krogerus e Tschäppeler 2017]. Os mesmos autores propõem uma série de critérios para que um modelo seja útil para decisão, dos quais se destaca: fornecer uma estrutura e ter utilidade como método.

Em busca de conhecimento para tratar destes aspectos, a próxima seção contempla os trabalhos relacionados considerados na fase de conscientização desta pesquisa.

4. Trabalhos Relacionados

A revisão da literatura a respeito da modelagem de TT revela que as tentativas neste sentido não convergem quanto ao propósito, e não exploram em detalhes a importância da formalização do conhecimento sobre seus elementos e dinâmica de interação para compor cenários específicos de tomada de decisão em firmar relações de transferência. Além disso, as relações de causa e efeito das interações da dinâmica de TT, necessárias para a construção de modelos de decisão, não são encontradas de forma sistemática e formalizada.

Uma das propostas mais promissoras neste sentido é a abstração da TT como um jogo [Speser 2006]. Nela, três elementos constituem o processo de transferência tecnológica: peças, as quais podem ser entendidas como os atores; tabuleiro, consistindo do ambiente ou contexto em que ocorre a transferência; e estratégia, como os mecanismos de dinâmica que guiam as tomadas de decisão nestas negociações.

Ainda da revisão realizada, seis trabalhos requerem destaque: [Pagani 2016], [Battistella et al. 2015], [Bozeman et al. 2015], [Reisman 2005], [Bozeman 2000] e [Robinson 1989]. A Tabela 1 resume a contribuição de cada trabalho selecionado para fins desta pesquisa.

Tabela 1. Contribuições dos Trabalhos Relacionados

Fonte	Propósito	Principais Características	Principal Resultado
[Robinson 1989]	Reunir em um modelo o que é conhecido sobre Transferência Internacional de Tecnologia para identificar relações de influência em decisões das organizações envolvidas.	- Trata exclusivamente de Transferência Internacional de Tecnologia; - Descreve a tecnologia em 13 dimensões; - Considera 2 tipos de atores envolvidos na transferência: o fornecedor e o demandante;	Um modelo geral do fluxo da tecnologia internacional, destacando que nenhuma transferência envolveria todos os fatores considerados.
[Bozeman 2000]	Revisar, sintetizar e criticar a literatura multidisciplinar em TT.	- Avalia impacto e eficácia; - Foca em TT entre universidade e laboratórios governamentais; - Considera a perspectiva de definição de políticas públicas;	Análise de forças e fraquezas da pesquisa e teoria de Transferência de Tecnologia segundo o critério de eficácia do modelo proposto.
[Reisman 2005]	Propor uma taxonomia como uma abordagem meta-interdisciplinar para estudar TT, uma vez que essa é por natureza multifacetada e multidisciplinar.	- Organiza a taxonomia em elementos chave; - Elenca um conjunto de atributos para cada um dos elementos chave; - Considera quatro disciplinas envolvidas com TT: economia, antropologia, sociologia, administração e engenharia;	Uma taxonomia capaz de definir e circunscrever a área de TT, e, ao mesmo tempo, identificar cada um de seus principais constituintes.
[Bozeman et al. 2015]	Revisar o modelo proposto por Bozeman (2000), considerando os estudos empíricos nos EUA ao longo dos 15 anos subsequentes à publicação do mesmo.	- Avalia impacto e eficácia; - Foca em TT entre universidade e laboratórios governamentais; - Considera a perspectiva de definição de políticas públicas; - Acrescenta ao modelo revisado o interesse crescente em TT orientada pelo valor público e social;	Série de recomendações para alcançar efetividade em Transferência de Tecnologia.
[Battistella et al. 2015]	Fornecer, por meio de uma análise da literatura, uma fundamentação teórica sólida que permita identificar os fatores críticos para Transferência de Tecnologia/Conhecimento.	- Considera tanto a transferência de tecnologia quanto de conhecimento, pois trata ambos como um objeto com suas próprias características; - Descreve fatores que representam os principais parâmetros e alavancas que devem ser considerados para projetar e implementar uma atividade de Transferência de Tecnologia/Conhecimento;	Descrição de fatores positivos e negativos para Transferência de Tecnologia, sendo estes relacionados a atores e ao processo.
[Pagani 2016]	Propor um modelo que sirva de ferramenta didática para compor cenários e situações envolvendo o processo de TT por completo.	- Oferece uma compreensão rápida dos principais aspectos envolvendo TT; - Pode ser usado como um guia geral por qualquer organização envolvida no processo de TT, para que nenhuma parte essencial do processo seja desconsiderada.	Um modelo simples e genérico que pode ser adotado por qualquer organização interessada em transferir ou receber tecnologia.

A proposta de [Robinson 1989] destaca-se por ser a única a ter a decisão em TT como foco de investigação. No seu estudo, o autor propõe um modelo genérico de fatores que influenciam as decisões relacionadas à Transferência Internacional de Tecnologia, tanto do lado do fornecedor quanto do demandante deste processo. Mesmo com a ênfase na transferência entre países, as decisões caracterizadas no seu modelo se mostram pertinentes para qualquer tipo de TT.

A partir dessas decisões, [Robinson 1989] considera que um dos sujeitos da interação da TT quer maximizar o seu lucro. Ambos lados possuem as mesmas decisões, sendo as diferenças relacionadas exclusivamente à expectativa do mesmo em relação à transação. Além disso, o autor dá ênfase à caracterização da tecnologia em negociação, considerando para a mesma treze dimensões: maturidade, dinamismo, importância relativa, especialidade ambiental, fator de substituição, especificidade de escala, disponibilidade, complexidade, centralidade, continuidade de produção, susceptibilidade para engenharia reversa, produto/processo, e especificidade para firma. Segundo o autor, a tecnologia é negociada por meio de dois tipos de transferência: a interna e a externa.

Para fins de desempenho, é indicado, pelo mesmo autor, que a medição desse relacionamento entre fornecedor e comprador tem dependência das expectativas dos mesmos. Sendo que três fatores exercem influência no processo de transferência: o custo percebido, o risco percebido e o benefício antecipado. O autor ainda destaca que esses fatores, por sua vez, são influenciados por políticas governamentais.

A proposta de [Robinson 1989], porém, não descreve a dinâmica de interação entre atores propriamente dita. Logo, trata-se de um modelo descritivo e explicativo, com restrições para entender como os elementos considerados atuam dinamicamente. [Reis e Vaccaro 2015] estudam a aplicação de agentes para a dinâmica de TT em um contexto específico, mas sem sistematizar uma generalização de suporte à decisão.

Assim, a viabilidade de reunir informações de outros modelos selecionados da literatura e relacioná-las com a proposta de [Robinson 1989], para consolidar conhecimento suficiente que sirva de base para a elaboração de modelos para a visão dinâmica de TT para tomada de decisão, foi foco da etapa de proposição e de desenvolvimento da pesquisa descritas a seguir.

5. TT como um SAC

A partir da análise dos trabalhos relacionados, na etapa de proposição, foi realizado o mapeamento das características levantadas nos modelos selecionados. Para isso, as dimensões propostas por [Speser 2006] foram relacionadas à organização de um SAC, resultando em 3 dimensões de caracterização da decisão em TT: Ator, agregando a concepção de peça e agente; Ambiente, associando as percepções de tabuleiro e ambiente complexo; Interação, considerando a estratégia como parte das interações entre agentes.

Como resultado, obteve-se um vocabulário consolidado apresentado na Tabela 2. Os termos selecionados referem-se à primeira terminologia usada na literatura.

Mesmo considerando o objetivo de estruturar conhecimento sobre a dinâmica de TT, tratando-a como um SAC, proposto para o vocabulário consolidado, para que sua utilidade seja percebida na tomada de decisão, um processo de modelagem é requerido.

Tabela 2. Mapeamento de Construtos para o Vocabulário Consolidado

	[Robinson 1989]	[Bozeman 2000] e [Bozeman et al. 2015]	[Reisman 2005]	[Battistella et al.2015]	[Pagani 2016]	Vocabulário Consolidado
Ator	fornecedor	agente de transferência	ator	Ator	cedente	fornecedor
	demandante	destinatário da transferência	ator	ator	cessionário	demandante
	intermediário			ator	intermediário agente	intermediário
Ambiente	fatores externos	ambiente de demanda	motivação			fatores externos
	tecnologia	objeto de transferência		objeto		tecnologia
	fatores de influência		motivação		barreiras pontos de sucesso	fatores de influência
				contexto		contexto
Interação	transferência	meio de transferência	tipo de transação	processo		transferência
	decidir transferir					decidir transferir
	decidir transferir interna ou externamente					decidir transferir interna ou externamente
	escolher a tecnologia					escolher a tecnologia
	escolher o mecanismo de transferência			escolher mecanismos		escolher o mecanismo de transferência
	escolher o vínculo organizacional					escolher o vínculo organizacional
		indicardor de eficácia			resultados	indicador de eficácia
				escolher canais		escolher canais
				conexão		conexão
					avaliar resultados	avaliar resultados

De acordo com abordagem para resolução de problemas, uma descrição genérica para o processo de tomada de decisão é apresentada na Figura 3.

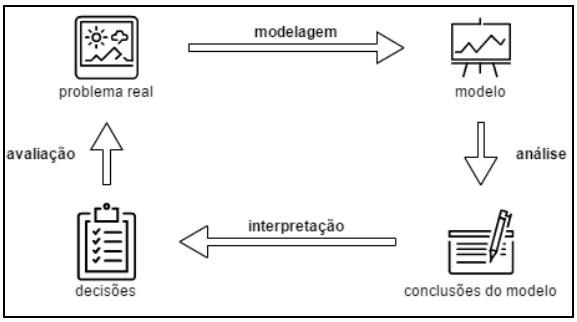


Figura 3. Processo para Resolução de Problemas adaptado de [Morabito 2008].

Nele, inicia-se pela definição do escopo do problema em estudo. Na fase seguinte, esse escopo é descrito na forma de um modelo. Na sequência, um método é empregado para resolver o problema, para que, na última fase, seus resultados sejam interpretados para uma decisão ser tomada. Essa, por sua vez, tem seu desempenho observado no mundo real. Para fins de metodologia de tomada de decisão para uso de um vocabulário consolidado de TT, esse processo de modelagem é adaptado, conforme especificado na Figura 4.

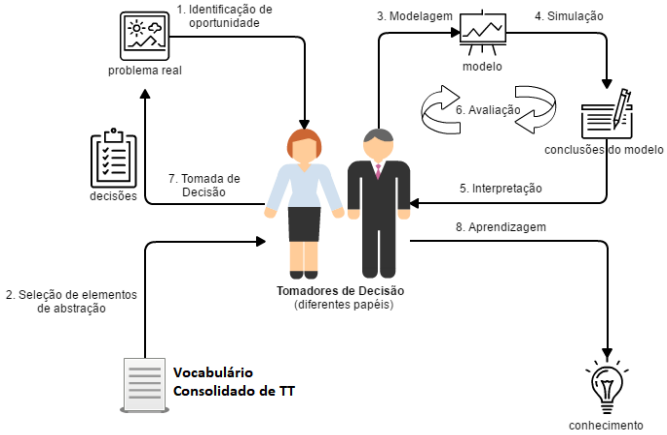


Figura 4. Processo de Modelagem com o Vocabulário Consolidado de TT

A modelagem proposta é composta de 8 passos básicos, os quais contemplam da identificação de uma oportunidade em TT até a tomada de decisão propriamente dita. O vocabulário consolidado, neste procedimento serve como base de unidades de abstração adequadas ao problema na forma de um modelo de agentes, bem como para validação se o modelo possui uma estrutura consistente com a realidade. A próxima seção descreve a modelagem e a simulação realizadas.

6. Experimentos

A avaliação da utilidade do vocabulário consolidado de TT como um SAC e o método de modelagem foi realizada com base nas percepções de um decisor real, proprietário de uma empresa do ramo de eletrônicos e produtora de soluções em tecnologia, a qual é referenciada como A neste artigo. Este caso foi selecionado, pois o perfil desse decisor foi levantado e analisado em [Kayser e Dusan 2013], o que oferece uma base para a modelagem realizada.

Os autores relatam que a empresa A tem na colaboração com parceiros externos a sua visão estratégica para inovar. Entre os aspectos descritos, destaca-se uma preferência da empresa em fazer parcerias com outras empresas. Mas o decisor também demonstra interesse por ampliar suas relações com universidades. Assim, a modelagem realizada para fins de validação, se propõe a apoiar a decisão de que classe de ator a empresa A deve escolher para realizar TT, considerando que o ambiente de negociação é influenciado pela disponibilização de recursos financeiros governamentais. Logo, foi elaborado o modelo baseado em agentes com base no vocabulário consolidado para este caso e implementada a sua simulação em NetLogo [Wilensky 1999], conforme apresentado na Figura 5.

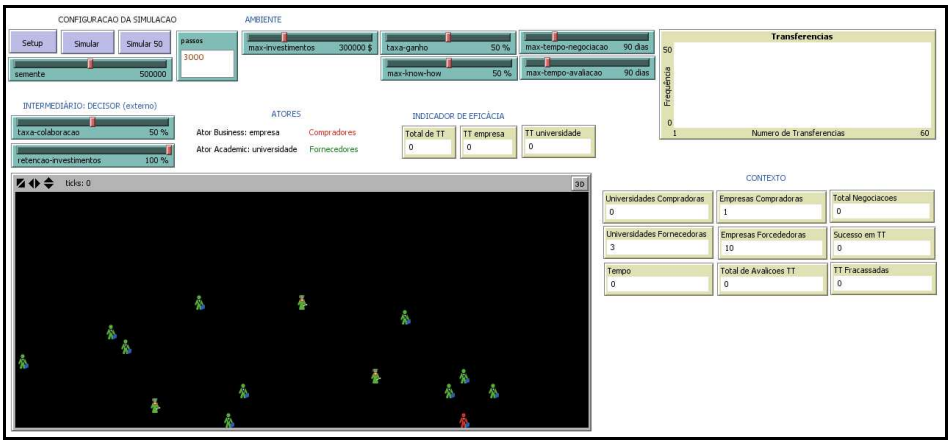


Figura 5. Interface do Modelo de Agentes para a Empresa A

Neste simulador, a métrica de interesse não é apenas o número de TT's realizadas, mas também quantos destes acordos são efetuados com empresa ou com universidade. Para o ambiente, assumiu-se um entorno com 10 empresas e 3 universidades, todos possuidores da tecnologia de interesse da empresa A. Além disso, o decisor age de acordo com a crença de que as universidades: possuem maior volume de recursos financeiros para acordos; possuem maior conhecimento a oferecer; são lentas tanto para fechar acordos de TT quanto para conduzir a transferência propriamente dita.

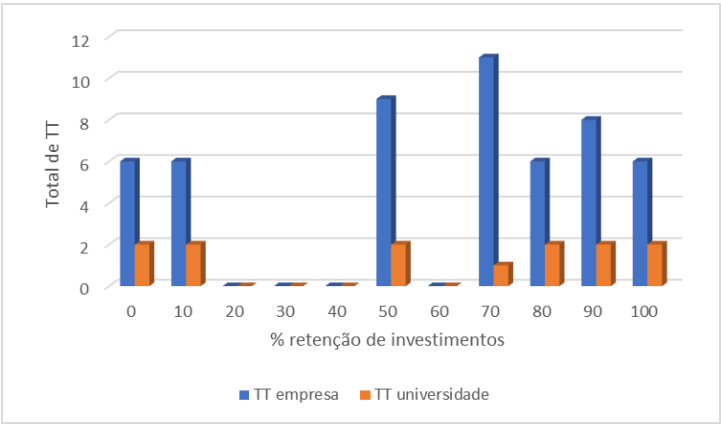


Figura 6. Efetivação de TT em Cenários de Retenção de Investimentos

Para fins de apoio à decisão, foram simulados cenários com variação na 'taxa-retencao" de 0 até 100%, em intervalos de 10 em 10. No gráfico da Figura 6, os

resultados médios de 10 rodadas são apresentados. Observe que em todos os cenários, com as crenças do decisor, a empresa A tende a realizar de TT com empresas. Ainda que a retenção dos investimentos cresça, o decisor opta por fazer mais transferências com empresas.

É interessante destacar que essa recomendação de privilegiar interações com outras empresas evidencia-se como razoável para o contexto real da empresa A. Na descrição realizada por [Kayser e Dusan 2013], é relatado que o decisor teve a experiência de realizar um acordo com uma universidade, com suporte de recursos federais, mas em dez meses após a aprovação do projeto pela instância governamental, o dinheiro liberado ainda não tinha sido entregue.

7. Considerações Finais

A formalização de um vocabulário consolidado para TT a partir das dimensões ator, ambiente e interação foi o artefato construído neste trabalho. Na análise de sua composição, é possível perceber uma contribuição em relação à visão macro dos elementos e fatores que participam de uma negociação de TT. Até porque um modelo de decisão não tem o propósito de fornecer uma resposta direta, e sim oportunizar estímulos mentais sobre uma dada situação [Krogerus e Tschäppeler 2017].

Porém, mesmo reunindo trabalhos relevantes da revisão da literatura, observa-se a necessidade de ampliar o detalhamento estrutural e dinâmico de como os atores fornecedor e receptor tomam a decisão de efetivamente firmar a relação de transferência. Assim, o artefato proposto é o ponto de partida para uma concepção de TT que busca compreender como a combinação de comportamento dos atores de TT pode fazer emergir fenômenos como competição, colaboração e cooperação.

Referências

Ajzental, A. (2015), Complexidade Aplicada à Economia, Editora FGV.

Battistella, C. et al. (2015) “Inter-organizational technology/knowledge transfer: a framework from critical literature review”, The Journal of Technology Transfer, p. 1-40.

Bax, P. M. (2014) “Design Science: Filosofia da Pesquisa em Ciência da Informação e Tecnologia”, Em: *XV Encontro Nacional de Pesquisa em Ciência da Informação – ENANCIB*, p. 3883-3903.

Bordini, R. H. et al. (2007), Programming multi-agent systems in AgentSpeak using Jason, John Wiley & Sons.

Bozeman, B. (2000) “Technology transfer and public policy: a review of research and theory.”, Research Policy, v. 29, n. 4, p. 627-655.

Bozeman, B. et al. (2015) “The evolving state-of-art in technology transfer research: Revisiting the contingent effectiveness model.”, Research Policy, v. 44, n. 1, p. 34-49.

Bulling, N. (2014) “A Survey of Multi-Agent Decision Making.”, KI – Künstliche Intelligenz, v. 28, n. 3, p. 147-158.

Chwif, L. e Medina, A. C. (2015), Modelagem e simulação de eventos discretos: teoria & aplicações, Elsevier.

Dresch, A. et al. (2015), Design Science Research, Bookman.

Furtado, B. A. et al. (2015) "Abordagens de Sistemas Complexos para Políticas Públicas", Em: *Modelagem de Sistemas Complexos para Políticas Públicas*.

Hilkevics, A. (2014) "Technology Transfer Models and Innovation Business Development", In: *Social Sciences for Regional Development*, p. 36-44.

Kiesling, E. et al. (2012) "Agent-based simulation of innovation diffusion: a review", Central European Journal of Operations Research, v. 20, n. 2, p. 183-230.

Kayser, A. C. e Schreiber, D. (2013) "Inovação nas empresas a partir de projetos colaborativos.", Gestão e Desenvolvimento, v. 10, n. 2, p. 69-78.

Krogerus, M. e Tschäppeler. (2017), O livro da decisão, Best Business.

Morabito, R. (2008), Introdução à Engenharia de Produção, Elsevier.

Nan, N. (2014) "A complex adaptative systems perspective of innovation diffusion: an integrated theory and validated virtual laboratory.", Computational and Mathematical Organization Theory, v. 20, n. 1, p. 52-88.

Pagani, R. N. (2016), Modelo de Transferência de Conhecimento e Tecnologia entre Universidades Parceiras na Mobilidade Acadêmica Internacional, Tese de Doutorado na Universidade Tecnológica Federal do Paraná.

Parsons, S e Wooldridge, M. (2002) "Game theory and decision theory in multi-agent systems", Autonomous Agents and Multi-Agent Systems, v. 5, n. 3, p. 243-254.

Ragsdale, C. T. (2011), Modelagem e análise de decisão, Cengage.

Reis, A. N. e Vaccaro, G. L. R. (2015) "Um Modelo Baseado em Agentes para Transferência de Tecnologia", Em: *XVIII Simpósio de Pesquisa Operacional e Logística da Marinha - SPOLM*.

Reisman, A. (2005) "Transfer of technologies: A cross-disciplinary taxonomy.", Omega, v. 33, n. 3, p. 189-202.

Robinson, R. D. (1989) "Toward creating an international technology transfer paradigm.", The International Trade Journal, v. 4, n. 1, p. 1-19.

Speser, P. L. (2006), The art and science of technology transfer, John Wiley & Sons.

Van Aken, J. E. (2004) "Management Research on the Basis of the Design Paradigm: the Quest for Field-tested and Grounded Technological Rules.", Journal of Management Studies, v. 41, n. 2, p. 219-246.

Watts, C. e Gilbert, N. (2014), Simulating Innovation: Computer-based Tools for Rethinking Innovation, Edward Elgar Publishing.

Wilensky, U. (1999) "NetLogo: Center for connected learning and computer-based modeling", Nothwestern University.

Wilensky, U. e Rand, W. (2015), An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with Netlogo, MIT Press.

Slavery in Material Agent Societies

Antônio Carlos da Rocha Costa[1]

[1]Programa de Pós-Graduação em Computação
Universidade Federal do Rio Grande - FURG
96.203-900 Rio Grande, RS, Brazil.

ac.rocha.costa@gmail.com

Abstract. *This paper formally characterizes* slavery systems *in material agent societies. The concepts of* master-slave social relationship*,* master-slave economic relationship*,* slavery-based economic system*,* slavery-supporting legal system*, and* slavery-based material agent society *are formally defined. A case study formally revisits North & Thomas' model stating the objective conditions that justify the rational choice between* slavery-based economic systems *and* free labor-based economic systems*, in material agent societies.*
Keywords: *material agent societies, slavery, slavery-based economic system, rational choice between slavery and free labor.*

Resumo. *Este artigo caracteriza formalmente* sistemas de escravatura *em sociedades de agentes materiais. Os conceitos de* relacionamento social mestre-escravo*,* relacionamento econômico mestre-escravo*,* sistema econômico baseado em escravatura*,* sistema jurídico endossador de escravatura *e* sociedade de agentes materiais baseada em escravatura *são formalmente definidos. Um estudo de caso revista formalmente o modelo de North & Thomas que estabelece as condições objetivas que justificam a escolha racional entre* sistemas economicos baseados em escravidão *e* sistemas econômicos baseados em trabalho livre*, em sociedades de agentes materiais.*
Palavras-chave: *sociedade de agentes materiais, escravidão, sistema econômico baseado em escravidão, escolha racional entre escravidão e trabalho livre.*

1. Introduction

In this paper, we introduce a conceptual framework supporting the formal modeling of *slavery* in material agent societies[1].

The *social* and *economic master-slave relationships* that characterize *slavery systems*, the *economic systems* that arise from them, and the *legal systems* that support them, are informally explained and formally defined. A formal general definition is given of *slavery-based material agent societies*.

So called *chattel slavery* is assumed as the basic form of slavery, and is formally characterized. A case study formally revisits North & Thomas' model stating the objectives conditions for the rational choice between *slavery-based economic systems* and *free labor-based economic systems*, in material agent societies.

The aim of the present paper, in line with the work that we have been doing concerning *agent societies* in general [Costa 2017c, Costa 2019], and *material agent societies*

[1]See, e.g., [Wikipedia 2018] for a general account of the history and variety of slavery.

in particular [Costa 2017b, Costa 2018], is to provide an *agent-based semantical model* for formal social and political theories of *slavery-based societies*. However, we investigate no single theory of slavery-based societies, regarding such applicability. In particular, we do not go beyond the general level of detail of the concepts mentioned above.

2. Material Agent Societies and Elementary Economic Exchanges

We take a general *agent society* to be any multiagent systems with a full-fledged social organization [Costa 2017c].

As in [Costa 2017b] and [Costa 2018], we say that an agent is a *material agent* whenever that agent has a *material body*, that is, a body that requires *energy* for its operation. And we call *material agent society* any agent society whose agents are all material agents.

We call *energy producer* any material agent that is capable of producing *energy objects*, with which energy is distributed for consumption in the society. All the other material agents of that society are said to be *energy consumers*.

Formally, we denote:

- $EnergProd$: the set of material agents that are *energy producers*;
- $EnergCons$: the set of material agents that are *energy consumers*.

Two types of *objects* are assumed to be exchangeable in an *elementary economic exchange* of two material agents:

- $EnergObj$, the type of the so-called *energy objects*, that is, objects that are carriers of the energy needed by the material agents for their functioning;
- $Chip$, the type of the so-called *chips*, that is, valuable objects that the material agents may be interested to acquire, possibly by exchanging some of the *energy objects* they have in their possession;

so that, in general, an *elementary economic exchange* is constituted by an exchange, between two material agents, of one or more *energy objects* for one or more *chips*.

3. Masters and Slaves in Chattel Slavery

Chattel slavery is slavery where *slaves* are considered personal belongings of their *masters*, which can dispose of them as they wish. Other forms of slavery also exist (see, e.g., [Wikipedia 2018]).

In this paper, we take into account only *chattel slavery*. However, we make an informal use of the term *property*, that is, we use property to mean both *informal*, non-legally supported ownership of objects, as well as *formal*, legally supported ownership of objects, *slaves* being a particular type of ownable objects.

Given a material agent society $MatAgSoc$, we formally define[2]

- $MatObj$: the set of *material objects* of $MatAgSoc$;
- $MatAg \in \wp(MatObj)$: the set of *material agents* of $MatAgSoc$;
- $Master \in \wp(MatAg)$: the set of *masters* of $MatAgSoc$;

[2] $\wp(X)$ is the powerset of set X.

- $Slave \in \wp(MatAg)$: the set of *slaves* of $MatAgSoc$.

For simplicity, we take that:

- $Master \cap Slave = \varnothing$: no master is a slave, and vice-versa;
- $Master \in \wp(EnergCons)$: all masters are *energy consumers*;
- $Slave \in \wp(EnergProd)$: all slaves are *energy producers*.

In the following sections, we introduce formal accounts of diverse aspects of the *property relationship* between masters and slaves.

4. Master-Slave Property Relationship

We call *system of master-slave property relationship* is the system of actions, norms and commands that empower the set of masters of a *slave-based material agent society*, maintaining slaves in their slavery condition.

Formally, we characterize the *property relationship* between masters and slaves in the following way:

- $owns \subseteq Master \times Slave$, the *property relation* between masters and slaves, so that $owns(master_i, slave_j)$ means that master $master_i$ owns $slave_j$.

The property relationship is supposed to allow that $master_i$ do any of the following actions on $slave_j$, whenever it happens that $owns(master_i, slave_j)$:

- $sell(slave_j, master_k)$
- $lend(slave_j, master_k)$
- $borrow(slave_j, master_k)$
- $lend(slave_j, master_k)$
- $rent(slave_j, master_k)$
- $kill(slave_j)$
- $free(slave_j))$
- $command(slave_j, cmd)$
- $punish(slave_j, cmd)\}$

meaning that:

- $master_i$ can *sell*, *lend*, *borrow*, *lend* and *rent* $slave_j$ to any other master $master_k$, besides *killing* or *freeing* it, and *commanding* it to perform any command cmd and *punishing* it for the way he performed such command.

In accordance with the possibility of $master_i$ performing $sell(slave_j, master_k)$, we take that master $master_k$ can *buy* slave $slave_j$ from master $master_i$. That is, we also have, as possible action:

- $buy(slave_j, master_i)$

In addition, the following (formal or informal) obligation is taken to be valid for slaves:

- $mustexec(slave_j, cmd, master_i)$

meaning that:

- $slave_j$ is supposed to peremptorily execute any command cmd issued by $master_i$.

For simplicity, we omit here any reference to the *conditions* under which those *actions* and *commands* may be effective, such as the explicit *connection* between *commands* and possible *punishments*. But, see Section 8 for some of the *legal* types of such conditions.

Other legal forms of *acquisition* of slaves (such as by having them *born* from parents that are already slaves, or by *capturing* them in certain specified conditions) are also considered in Section 8.

5. Master-Slave Economic Exchanges

Let time be given by the linearly ordered set $T = \{0, 1, 2, ...\}$, ranged over by variable t. For the purpose of the present paper, we call *elementary economic exchange* any exchange of the form (cf. [Costa 2018]):

$$e2exch^t = \langle mag/obj \rangle^t \ {}_{c'}\!\rightleftarrows^{c} \langle mag'/obj' \rangle^t$$

meaning that material agents mag and mag' exchange objects obj and obj', a pair of such objects at each time t, under the assumption that mag provides operational condition c for mag' to produce and delivery obj', and mag' provides operational condition c' for mag to produce and delivery obj.

We call *master-slave elementary economic exchange* any elementary economic exchange of the form:

$$mse2exch^t = \langle master/\perp \rangle^t \ {}_{\perp}\!\rightleftarrows^{c} \langle slave/obj \rangle^t$$

meaning that, at each t, the slave $slave$ sends an object obj to his master $master$, without receiving no object in exchange ($\perp$ is the *null* object), while the master $master$ is required to provide condition c for $slave$ to produce and delivery obj, and the slave $slave$ is not required to provide any condition ($\perp$) for the master to produce and delivery nothing ($\perp$).

Notice that in any *master-slave elementary economic exchange* like $mse2exch^t$, masters accumulate a set of received objects, up to time t, in the form:

$$accumobj[master/mse2proc^t] = \{obj^0, obj^1, \ldots, obj^t\}$$

while slaves accumulate nothing, because we take that a set of *null* objects is an empty set. That is:

$$accumobj[slave/mse2proc^t] = \{\perp^0, \perp^1, \ldots, \perp^t\} = \varnothing$$

In general, *masters* are allowed to have a *group of slaves* with more than one slave in it. In such a case, the *master-slave group elementary economic exchange* that the *master* and the *group of slaves* perform has the form:

$$msge2exch^t = \langle master/\perp \rangle^t \ {}_{\perp}\!\rightleftarrows^{c} \langle Slave/Obj \rangle^t$$

where:

- $Slave$ is the *group of slaves* that belong to master $master$;
- Obj is the *set of objects* that the set of slaves $Slave$ produce and deliver, at each time, to their master $master$.

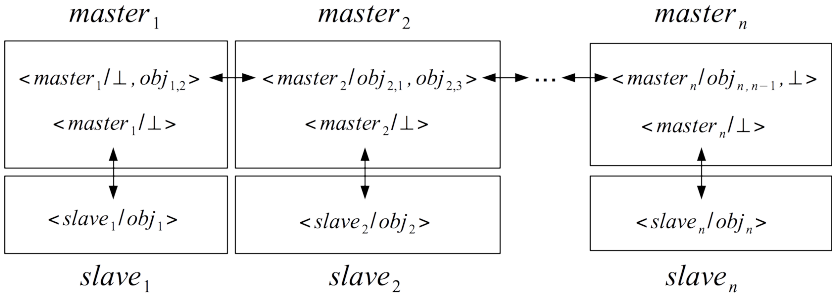


Figure 1. Sketch of a slavery-based elementary economic process.

6. Slavery-Based Elementary Economic Processes

In general, *individual elementary economic processes* have the form (see [Costa 2018]):

$$ie2proc^t = \langle mag_1/\perp, obj_{1,2}\rangle^t \; {}_{c_{2,1}}\!\rightleftarrows^{c_{1,2}} \langle mag_2/obj_{2,1}, obj_{2,3}\rangle^t \; {}_{c_{3,2}}\!\rightleftarrows^{c_{2,3}} \ldots \\ \ldots \; {}_{c_{n,n-1}}\!\rightleftarrows^{c_{n-1,n}} \langle mag_n/obj_{n,n-1}, \perp\rangle^t$$

where:

- mag_i is the i-th material agent participating in $ie2proc^t$;
- each $\langle mag_i/obj_{i,i-1}, obj_{i,i+1}\rangle^t \; {}_{c_{i+1,i}}\!\rightleftarrows^{c_{i,i+1}} \langle mag_{i+1}/obj_{i+1,i}, obj_{i+1,i+2}\rangle^t$ is an *elementary economic exchange*;
- $obj_{i,k}$ is the object that mag_i produces and deliveries to its k-th partner, for $k \in \{i-1, i+1\}$;
- mag_1 has no left partner, so $obj_{1,0} = \perp$;
- mag_n has no right partner, so $obj_{n,n+1} = \perp$.

In the general case of *slavery-based elementary economic processes*, we have the form:

$$mse2proc^t = \langle master_1/\perp, obj_{1,2}\rangle^t \; {}_{c_{2,1}}\!\rightleftarrows^{c_{1,2}} \langle master_2/obj_{2,1}, obj_{2,3}\rangle^t \; {}_{c_{3,2}}\!\rightleftarrows^{c_{2,3}} \ldots \\ \ldots \; {}_{c_{n,n-1}}\!\rightleftarrows^{c_{n-1,n}} \langle master_n/obj_{n,n-1}, \perp\rangle^t$$

where one can notice that only *masters* participate in the society's economic processes, *slaves* being restricted to private economic exchanges with their *masters*, as sketched in Figure 1. Notice that a *master* participates in *two* elementary economic exchanges (one local, with his *slave*, the other global, with other *masters*), while a *slave* participates only in *one* elementary economic exchange, the local exchange with his *master*.

Notice also that this model of slavery-based elementary economic processes naturally extends to the cases where the masters may have *groups of slaves*, instead of just individual slaves.

7. Slavery-Based Elementary Economic Systems

Let the term *elementary economic material agent* denote material agent of the population of a material agent society that can participate in an elementary economic exchange. We define the general notion of *elementary economic system* of a material agent society as follows (cf. [Costa 2018]):

Definition 7.1 *The* elementary economic system $E2Sys$ *of a material agent society* $MAgSoc$ *is a time-indexed structure:*

$$E2Sys^t_{MAgSoc} = (E2Ag^t, Objs^t, E2Beh^t, E2Exch^t, E2Proc^t)$$

where, for each time t:

- $E2Ag^t$ *is the set of* elementary economic material agents*, which participate in the elementary economic processes of $MAgSoc$, at the time t;*
- $Objs^t$ *is the set of* objects *that the elementary economic agents can exchange during the performance of their elementary economic exchanges, at that time;*
- $E2Beh^t$ *is the set of* elementary economic behaviors *that the elementary economic agents can perform during the performance of the elementary economic exchanges, at that time;*
- $E2Exch^t$ *is the set of* elementary economic exchanges *that the elementary economic agents can perform during the performance of the elementary economic processes, at that time;*
- $E2Proc^t$ *is the set of* elementary economic processes *that the elementary economic agents can perform in $MAgSoc$, at that time.*

In the case of a slavery-based elementary economic system, we have that, for any time t:

- $E2Ag^t = Master^t \cup Slave^t$, meaning that the set of elementary economic material agents of $E2Sys^t_{MAgSoc}$ is partitioned into *masters* and *slaves*;
- $Obj^t = MasterSlaveObj^t \cup MasterMasterObj^t$, meaning that the objects exchanged between elementary economic agents are either of the *master-slave* type or of the *master-master* type, with (see [Costa 2018]):
  - $MasterSlaveObj^t \in \wp(EnergObj)$, that is, objects exchanged between masters and slaves (in fact, just from slaves to masters) are *energy objects*, resulting from the labor of the slaves;
  - $MasterMasterObj^t \in \wp(EnergObj) \cup \wp(Chip)$, that is, objects exchanged between masters are either *energy objects* or *chips*, with chips exchanged in return for energy objects;
- $E2Beh = MasterBeh^t \cup SlaveBeh^t$, that is, both masters' and slaves' elementary economic behaviors may participate in the elementary economic exchanges of $E2Sys^t_{MAgSoc}$;
- $E2Exch = MasterSlaveExch^t \cup MasterMasterExch^t$, that is, the elementary economic exchanges are either of the *master-slave* type or of the *master-master* type;
- $E2Proc = MasterSlaveProc^t \cup MasterMasterProc^t$, that is, the elementary economic processes are either of the *master-slave* type or of the *master-master* type.

Notice that no particular requirement is established concerning the types of *conditions* that may be imposed on the elementary economic exchanges.

8. Slavery-Supporting Legal Systems

8.1. Legal Systems of Agent Societies

We define the general notion of *legal system* situated in an agent society $AgSoc$ as follows (cf. [Costa 2015]):

Definition 8.1 *A* legal system *is a* time-indexed structure*:*

$$LegalSys^t_{AgSoc} = (LOrd^t, LOrg^t, RLFact^t, LegalOps)$$

where:

- $LOrd^t$ *is the* legal order *at time* t*;*
- $LOrg^t$ *is the* system of legal organs *at time* t*;*
- $RLFact^t$ *is the* record of legal facts *at time* t*;*
- $LegalOps$ *is the set of* legal operations*, like:*
  - $createlnrm$*, the* creation *of legal norms;*
  - $deroglnrm$*, the* derogation *of legal norms;*
  - $createlauth$*, the* creation *of authorizations to perform* legal operations*;*
  - $cancellauth$*, the* cancellation *of such authorizations;*
  - $recordlfct$*, the* recording *of a legal fact in the record of legal facts;*
  - $deletelfct$*, the* deletion *of one such record.*

with:

- $LegalSys^t_{AgSoc}$ *supposed to contain a* public record of legal facts*, to be freely accessed by the agents of the society;*
- $LOrd^t$ *and* $LOrg^t$ *such that* $LOrd^0 \neq \varnothing \neq LOrg^0$.

8.2. Legal Systems of Slave-Based Societies

In the context of slavery-based material agent societies endowed with legal systems, we are particularly interested in the types of legal norms that support the masters in their maintenance of the slave-based social and economic relationships.

Cleraly, the most fundamental of such legal norms are:

- $owns(master_i, slave_k) \Rightarrow Auth(master_i, command(master_i, slave_j, anyact))$, which legally authorizes master $master_i$ to *command* that slave $slave_j$ does any act the master wishes;
- $owns(master_i, slave_k) \Rightarrow Auth(Master, punish(Master, Slave, AnyAct))$, which legally authorizes master $master_i$ to *punish* slave $slave_j$ for not doing properly any act the master has commanded it to do.

Typically, the following conditional legal authorization is also formally adopted by the legal systems of slave-based material agent societies, so that the initial condition of being a slave, and which is its initial master, is established:

- $owns(master_i, slave_k) \wedge mother(slave_k, slave_j) \Rightarrow Auth(master_i, owns(master_i, slave_j))$, which legally authorizes master $master_i$ to *own* slave $slave_j$ if the mother of slave $slave_j$ is itself owned by master $master_i$.

Legal norms as the above ones, are sufficient for material agent societies whose only means to produce slaves is through their parental reproduction. Some material agent societies, however, adopt the legal procedure of allowing slaves to be produced by their capture from other societies (either in the context of war between the two societies, or in the context of commercial exploitation of the second society by the first one).

That type of legal norm may be formally sketched as follows:

- $autorized(mag_i, slavecapture) \wedge captured(mag_j, mag_i) \Rightarrow Auth(mag_i, owns(mag_i, mat_j))$, which states that if material agent mag_i is legally authorized to capture slaves, and it happened that mag_i capture mag_j then mag_i is legally authorized to own mag_j as a slave, effectively making of mag_i a *master* and of mag_j a *slave*.

9. Slavery-Based Material Agent Societies

A detailed formal account of the architecture of material agent societies that are organized on the basis of slavery is out of the scope of the present paper. Here, we can only leave implicit the amount of details that would have to be provided in order to properly instantiate, as a *slavery-based material agent society*, the following general structure of an *agent society* (see [Costa 2017c]):

- $AgSoc = (Pop, Org, MEnv, SEnv, IMP, ACC)$

where:

- Pop is the *population* of $AgSoc$;
- Org is the *organizational structure* of $AgSoc$;
- $MEnv$ is the *material environment* of $AgSoc$;
- $SEnv$ is the *symbolic environment* of $AgSoc$, where the society's system of *legal norms* is embedded;
- IMP is the collection of *implementation relations* between Pop and Org;
- ACC is the collection of *access relations* between Org and the environments $MEnv$ and $SEnv$.

Moreover, considering the case of slavery-based material agent societies that *do not* produce slaves by capturing them in another society, but that, besides producing them by parental reproduction, also *import* them from *slave capturing societies*, the full account of the details of that slavery-based society would require placing it in an *inter-societal context* (see [Costa 2017a]), which is also out of the scope of the present paper.

10. A Case Study: North & Thomas' Model of Rational Choice between Slavery and Free Work in Material Agent Societies

Douglass C. North and Robert Paul Thomas develop in [North and Thomas 1971] an institutional dynamical model for the rise and fall of manorial systems, that encompasses both *serfdom* and *slavery*. They contrast *serfdom* an *slavery* by telling the first to be "*essentially a contractual arrangement where labor services were exchanged for the public good of protection and justice*" (p.778), with a "*contractual relationship which could be changed only by both parties*" (p.779), while in the second a slave "*has no legal control over decision-making with respect either to his labor or to his income stream*" (p.779).

Clearly, by basing the distinction on the notion of *contract*, North & Thomas' model presupposes the existence of some sort of (formal or informal) *legal system* that is effective in the society and capable of enforcing the compliance with valid contracts.

The core of their rational model for the choice between *slavery-based* and *free labor-based* economic system is the following:

> "*Slavery was always more profitable than free labor <..> when the following conditions existed: (1) a market economy, (2) profitable opportunities to produce those types of economic activities where the costs of supervision to reduce shirking were low, and (3) where the costs of enforcing property rights in human beings were low.*" (p.779)

Notice that:

1. Condition (1) means the possibility of freely *selling* and *buying* slaves.
2. Let $costsuperv(slave_j, master_i)$ denote the *cost of the supervision*, for master $master_i$, of the operation of the slave $slave_j$. Analogously, denote the cost of the corresponding supervision, concerning the free laborer $freelaborer_k$, by $costsuperv(freelaborer_k, master_i)$. Then, condition (2) means that it is rational to adopt a *slavery-based economic system*, instead of an *free labor-based economic system*, in a given society, if and only if, for most of the masters ($mstr$) it holds that:

$$\sum_{j=1}^{j=n} costsuperv(slave_j, mstr) \leq \sum_{k=1}^{k=m} costsuperv(freelaborer_k, mstr)$$

   where:
   - n is the *average number of slaves* owned by most of the masters, in the alternative of the *slavery-based economic system*;
   - m is the *average number of free laborers* hired by most of the masters, in the alternative of the *free labor-based economic system*.
3. Let $costenforc(slave_j, master_i)$ denote the *cost*, for master $master_i$, of *enforcing property rights* in the slave $slave_j$ (both regarding the slave itself and the other competing masters). Analogously, let $costenforc(freelaborer_k, master_i)$ denote the cost, for master $master_i$, of the corresponding enforcement, concerning the *product* of the labor of the free laborer $freelaborer_k$. Then, condition (3) means that it is rational to adopt a *slavery-based economic system*, instead of an *free labor-based economic system*, in a given society, if and only if, for most of the masters ($mstr$) it holds that:

$$\sum_{j=1}^{j=n} costenforc(slave_j, mstr) \leq \sum_{k=1}^{k=m} costenforc(freelaborer_k, mstr)$$

   where:
   - n is the *average number of slaves* owned by most of the masters, in the alternative of the *slavery-based economic system*;
   - m is the *average number of free laborers* hired by most of the masters, in the alternative of the *free labor-based economic system*.

In summary, one can see, by this brief account of the elements of North & Thomas' model, that a rational choice is possible between a *slavery-based economic system* and a *free labor-based economic system*, in any given *material agent society*, at any time of its history.

11. Discussion

This paper introduced a formal model for *slavery* in material agent societies. Clearly, we adopted the term *slave* to refer to material agents that can be bought and sold in a material agent society, while we adopted the term *free laborer* to material agents that can be *hired* for the realization of specified jobs.

The basic elements of North and Thomas' [North and Thomas 1971] model for the rational choice between *slavery-based* and *free labor-based* economic systems was

briefly reviewed. One sees that the analysis of their model opens the possibility for a *mixed* type of economic systems of material agent societies, namely, that which combines *slave material agents* and *free laborer material agents*.

Two criteria arise for a rational choice between *slavery* and *free laboring*, in such mixed situations:

- first, a choice at the level of the *type of economic activity*: choose between *slavery* and *free laboring* according to the costs of *work supervision* and *property rights enforcement* peculiar to each activity;
- second, a choice at the level of the *particular situation of the master*: choose between *slavery* and *free laboring* according to the costs of *work supervision* and *property rights enforcement* for each particular master.

A situation where the full combination of all such possibilities are adopted would certainly introduce extra complexity in the legal system of the society, because legal provisions would have to be established for each such possibility, including different legal norms applying to the same master, in accordance with the particular type of economic relation he has with each of his workers.

A more sensible choice would be that the material agent society chooses, for each *type of economic activity*, either *slavery* or *free labor*. In such case, the legal norms of the legal system, concerning the way work is performed in the material agent society, could be specialized for the different types of economic activities.

Finally, notice that even if the legal system of the society adopts only free labor based economic activities, it may be rational for some particular economic activity, or for some particular master, to establish a slavery-based form of economic exchange with its workers, giving rise to *illegal* slavery forms of economic activities in the society.

12. Conclusion

This paper introduced the notion of *slavery* in material agent societies. Formal account was given of its basic aspects. North & Thomas' model for the rational choice between *slavery-based* and *free labor-based* elementary economic systems was shown to be applicable to material agent societies.

The particular possibility of material agent societies with *mixed modes* of economic system (slavery + free labor) was discussed.

Future work, in this line of investigation, could explore the *details of the legal systems* regulating slavery and free labor, characterizing the *basic differences* between the legal norms that apply to each of them.

References

Costa, A. C. R. (2015). Situated legal systems and their operational semantics. *Artificial Intelligence & Law*, 43(1):43–102.

Costa, A. C. R. (2017a). Ecosystems as agent societies, landscapes as multi-societal agent systems. In Adamatti, D. F., editor, *Multiagent Based Simulations Applied to Biological and Environmental Systems*, pages 25–43. IGI Global, Hershey.

Costa, A. C. R. (2017b). Energy systems in material agent societies. *RITA - Revista de Informática Teórica e Aplicada*, 24(2):130–144.

Costa, A. C. R. (2017c). SML - a society modeling language. Technical report, Tutorial presented at WESAAC 2017, São Paulo. Available on http://wesaac.c3.furg.br.

Costa, A. C. R. (2018). Elementary economic systems in material agent societies. In *Anais do WESAAC 2018*, pages 12–24, Fortaleza. UECE.

Costa, A. C. R. (2019). *A Variational Basis for the Regulation and Structuration Mechanisms of Agent Societies*. Springer, Cham.

North, D. C. and Thomas, R. P. (1971). The rise and fall of manorial systems: A theoretical model. *The Journal of Economic History*, 31(4):777–803.

Wikipedia (2018). *Slavery*. Available at https://en.wikipedia.org/wiki/Slavery.

Instituições em Sistemas Multiagentes à luz da Teoria da Construção da Realidade Social

Rafhael R. Cunha[1], **Jomi F. Hübner**[1], **Maiquel de Brito**[2]

[1]PGEAS – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

[2]Departamento de Engenharia de Controle, Automação e Computação
Universidade Federal de Santa Catarina (UFSC)
Blumenau – SC – Brasil

rafhael.cunha@posgrad.ufsc.br,jomi.hubner@ufsc.br,maiquel.b@ufsc.br

Resumo. *Este artigo apresenta uma revisão sobre alguns trabalhos que tratam de instituições virtuais no âmbito de sistemas multiagentes, analisando-os à luz da teoria da construção da realidade social proposta pelo filósofo John Searle [Searle 1995]. O objetivo desta pesquisa é comparar os trabalhos selecionados com pontos cruciais da teoria citada. Como resultado, o artigo elenca limitações e vantagens dos trabalhos analisados, indicando direções para trabalhos futuros.*

1. Introdução

Sistemas multiagentes (SMA) abertos podem ser considerados como uma extensão tecnológica de sociedades humanas [Fornara et al. 2004], uma vez que agentes são autônomos para entrarem no sistema, atingir seus objetivos e eventualmente saírem [De Brito et al. 2018, De Brito et al. 2012, Dastani et al. 2009a, Cardoso and Oliveira 2007]. Contudo, tais características podem gerar problemas no sistema, como objetivos conflitantes ou interações que não levarão para resultados coerentes.

As ciências sociais têm inspirado propostas para coexistência de agentes em suas sociedades, sendo um meio interessante de regular SMAs abertos. O filósofo John Searle [Searle 1995] apresentou a teoria da construção da realidade social que introduz alguns conceitos, dentre eles o de realidade institucional, segundo a qual instituições humanas habilitam elementos concretos do mundo a desempenharem funções em uma sociedade independente de suas virtudes físicas. Esta teoria tem sido adotada (e adaptada) por diferentes pesquisas relacionadas a SMAs.

Em instituições humanas, o fato de uma pessoa ser o presidente do Brasil é efetivado por um conjunto de características. A primeira refere-se à necessidade de membros da sociedade aceitarem que esta pessoa possui o status de presidente, habilitando-o a executar tarefas. A segunda diz respeito as obrigações - ser chefe do poder executivo, conduzir a política econômica, aplicar leis aprovadas, etc - que estão relacionadas ao status de presidente. A primeira e a segunda características são denominadas *acordo coletivo* na teoria de Searle. A terceira está relacionada ao indivíduo possuir status anteriores ao status de presidente, como o de cidadão, eleitor, vencedor das eleições presidenciais, etc, gerando um padrão iterativo de status. Tal propriedade é intitulada *iteratividade de status*. Por último, os status devem possuir um tempo de validade - o status de presidente

não é vitalício e está relacionado a um tempo de ocorrência em virtude da legislação. Esta característica é nomeada *manutenção de instituição*. Todos esses aspectos estão contidos na teoria de Searle. Contudo, alguns parecem ainda inexplorados em trabalhos da área de SMAs, sendo o problema investigado neste trabalho.

Ao desenvolver instituições artificiais - instituições computacionais que simulam instituições humanas dentro do SMA - é especialmente relevante considerar todos aspectos citados para que os SMAs incorporem as características relevantes do funcionamento de sociedades humanas. A *aceitação coletiva* é importante porque o fato das pessoas aceitarem ou reconhecerem um status viabiliza as ações executadas e corrobora para que as pessoas envolvidas à respeitem. Caso contrário, qual será a eficácia de uma exigência emitida pelo presidente no sistema? Ou qual será sua motivação para desempenhar obrigações inerentes ao status de presidente? Um conceito social semelhante encontrado nas organizações multiagentes é o *papel*. Todavia, o papel pode ser atribuído a qualquer elemento presente no sistema e não depende de nenhuma atitude mental ou acordo para ocorrer. A *iteratividade de status* promove benefícios como rastreabilidade de status e possibilidade de atribuição automática de status hierarquicamente inferiores. Por último, a *manutenção da instituição* é também importante, pois instituições sociais permanecem válidas enquanto um estado social ainda é satisfeito, caso contrário, são descontinuadas. Mecanismos que oportunizem tais adaptações em aparatos computacionais são necessários para uma representação íntegra de instituições.

Propostas de instituições em SMA foram analisadas, por exemplo em [Brito and Hübner 2014, Brito et al. 2016, De Brito et al. 2018, Dastani et al. 2009b, Campos et al. 2008], etc, considerando as implementações efetivadas de instituições em sistemas computacionais. Contudo, as características apresentadas acima - aceitação coletiva, iteratividade de status e manutenção da instituição - fundamentadas na teoria de Searle não foram discutidas nessas propostas. Portanto, o objetivo desta pesquisa é identificar alguns pontos da teoria de Searle e avaliar sua cobertura na área de SMAs. Para isto, utilizamos o método revisão sistêmica que inclui etapas para definição dos pontos a serem avaliados, busca por trabalhos relacionados, critérios para inclusão/exclusão de trabalhos, extração de informações, sumarização dos resultados e análise final.

O artigo está organizado da seguinte maneira: A seção 2 apresenta uma breve introdução da teoria da construção da realidade social proposta por John Searle. A seção 3 descreve como instituições tem sido tratadas no âmbito de SMAs. A seção 4 discorre sobre o processo de revisão, sumarizando e analisando os resultados encontrados. Por fim, na seção 5 são feitas as conclusões sobre a pesquisa e indicações para trabalhos futuros.

2. Instituições de acordo com John Searle

Searle argumenta que existem fatos que são explicados pela ciência básica, como a água ser composta de hidrogênio e oxigênio, que não dependem de nenhuma atitude mental dos agentes para existirem. Elementos como um pedaço de papel, uma montanha, gelo, etc, são considerados fatos brutos, pois sua existência é livre de qualquer percepção ou estado mental. A Figura 1 ilustra de forma simplificada a teoria proposta.

Contudo, existem funções que elementos do mundo não conseguem realizar em virtude de seus atributos físicos. Para tanto, esses elementos precisam de *funções de status*, que são status que os habilitam a desempenhar as funções requisitadas. To-

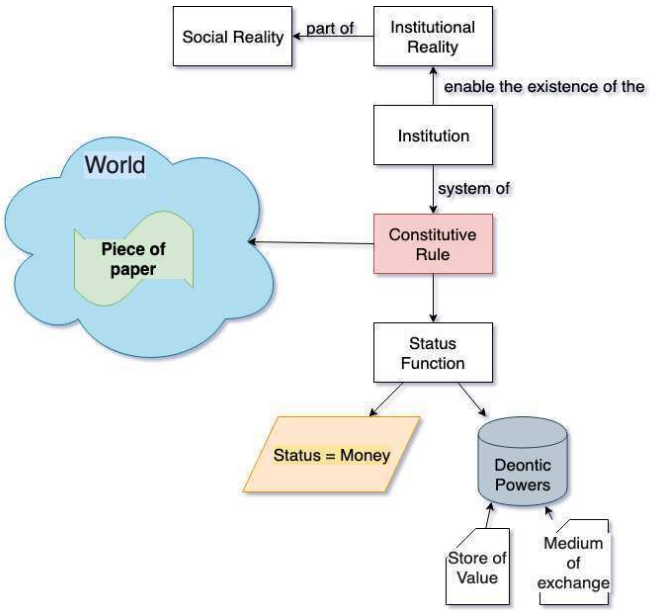


Figure 1. Sintetização da Teoria da Realidade Social proposta por John Searle

davia, para um elemento possuir um status (e desempenhar a função correspondente) é necessário que a sociedade concorde a respeito disso, ou seja, estabeleça um acordo coletivo [Searle 1995, p. 44]. Por exemplo, o status *presidente do Brasil* qualifica uma pessoa a desempenhar funções que não seriam possíveis apenas por suas características físicas. Além disso, para a pessoa ser considerada presidente, é necessário que pessoas aceitem e, ou, reconheçam coletivamente a validade deste ato.

As *funções de status* carregam consigo *poderes deônticos* [Searle 2010, p.8]. Eles são os direitos, deveres, obrigações, permissões, etc, que direcionam o comportamento esperado das pessoas nas sociedades. Por exemplo, um pedaço de papel, por carregar a função *Money*, permite que seu proprietário o troque por algum outro bem. A atribuição de *funções de status* é determinada por *regras constitutivas* através da forma *X conta como Y em C*, significando que um elemento físico X tem a *funções de status* Y no contexto C. Por exemplo, uma regra pode definir que um *Piece of paper* (X) conta como *Money* (y) no Brasil (C). Um outro exemplo é o jogo de xadrez. As *funções de status* atribuídas a jogadas no tabuleiro criam propriamente dito o jogo. Sem elas, aquilo que se compreende como o jogo de xadrez seria apenas uma movimentação de peças sobre o tabuleiro.

Searle afirma em [Searle 2010, p.9 - tradução livre] que o sistema de *funções de status* constituídas a partir de *regras constitutivas* é "a cola que mantêm a sociedade unida". Isto porque elas fornecem razões para pessoas agirem independentemente de seus desejos e de restrições físicas implementadas na realidade bruta e também porque elas constituem os elementos nos quais os *poderes deônticos* estão baseados. Portanto, *instituições* são sistemas de *regras constitutivas* que habilitam a existência da *Realidade Institucional* [Searle 2010] - interpretação, por parte da *instituição*, dos fatos ocorridos no ambiente. Por sistemas de *regras constitutivas*, Searle quer dizer que *instituições* definem alguma interpretação particular da realidade bruta, expressada em termos de *funções de status* atribuídas, para basear os *poderes deônticos*. Estes existem dentro de *instituições* e somente quando a *função de status* está atribuída. Por exemplo, a norma "o presi-

dente deve decretar uma nova lei no tempo de seu mandato", é efetiva somente quando a instituição constitui presidente e lei. Portanto, além de *instituições* ser sistemas de *regras constitutivas* que habilitam a *realidade institucional*, elas também define o comportamento esperado de indivíduos. A *Realidade Institucional* é parte da *Realidade Social*.

3. Instituições Situadas em Sistemas Multiagentes

Uma sociedade de agentes é um sistema multiagente *aberto*, *organizado*, *persistente* e *situado*[da Rocha Costa 2014]. Abertura representa a possibilidade dos agentes entrarem e saírem da sociedade livremente. Organização possibilita que subconjuntos de agentes possam se misturar em subsistemas que possivelmente se inter-relacionem de maneira sistemática. Persistência é a característica da organização da sociedade persistir no tempo, independentemente da entrada ou saída de agentes ou de suas mudanças de comportamento ou interações. Por último, situado refere-se a particularidade da sociedade operar em um ambiente físico definido.

Por sistemas abertos apresentarem tais características, informações como o número, comportamento ou até mesmo o meio que os agentes interagem ou acessam recursos compartilhados podem ser desconhecidos em tempo de projeto [Piunti 2009], podendo casualmente gerar conflitos para o sistema. Em SMAs abertos, as interações entre agentes são tipicamente projetadas e implementadas por diferentes partes, podendo resultar em conflitos de interesse (objetivos individuais divergentes, metas globais inatingíveis, etc) comportamentos sociais inesperados [De Brito et al. 2018], confiança limitada e não conformidade com a especificação [Artikis et al. 2002].

As ciências sociais têm estudado meios de habilitar a coexistência de sociedades humanas [Brito et al. 2016]. Esses meios tem inspirado propostas para coexistência de agentes em sociedades artificiais. [Boella and Van Der Torre 2004, Campos et al. 2008] observam que conceitos da realidade social (normas, instituições, etc) têm sido introduzidos em SMAs e parecem ser um caminho para coordenar o comportamento dos diferentes e possivelmente heterogêneos agentes, além de controlar o comportamento emergente de sua interação. Entre estes conceitos, *normas* desempenham um papel importante no projeto e no gerenciamento de sistemas abertos porque expressam o comportamento esperado dos agentes em muitos meios diferentes [De Brito et al. 2018]. Independentemente de qualquer regulação, o cenário de atuação dos agentes é por definição o ambiente [De Brito et al. 2018]. Tratando-se de sistemas abertos, a dinâmica do ambiente pode mudar ao longo da execução do sistema [Helleboogh et al. 2007]. Especificar normas que se adaptem a mudanças e permaneçam estáveis parece ser uma tarefa complexa nesse tipo de sistema. Em um cenário em que compradores devem pagar por suas compras, a norma *"bob é obrigado a fazer um pagamento via depósito bancário"* está restrita ao agente *bob* e ao método de pagamento *depósito bancário*. Ao especificar normas com conceitos abstratos como *comprador* e *pagamento*, as normas podem permanecer estáveis mesmo variando todos os agentes considerados compradores e todas as formas de pagar que são consideradas pagamento.

Normas e outros meios de regulação (organização, *commitments*) são apenas um componente da instituição - inspirados na teoria de Searle - e são adotados por alguns trabalhos [Fornara et al. 2008, De Brito et al. 2018]. Elas representam os *poderes deônticos* que devem ser associados as *funções de status*. De acordo com Searle [Searle 1995], a re-

alidade institucional é composta também por outros aspectos relevantes, como a atribuição de *funções de status* a elementos ambientais, que são constituídas a partir da especificação de *regras constitutivas*, produzindo o estado institucional. Regras constitutivas são importantes porque elas definem significados, na instituição, para os elementos concretos (agentes, mensagens etc) que efetivamente compõem os SMAs. Por exemplo, enquanto uma norma define que um comprador é obrigado a pagar por suas compras, regras constitutivas definem que determinados agentes constituem compradores e determinadas ações constituem pagamentos. Nas sociedades humanas a relação entre elementos concretos e estado institucional surge naturalmente e é analisado pelas ciências sociais. Mas em SMAs, essa relação não é natural e deve ser modelada, definindo como fatos ambientais afetam os diferentes componentes da instituição. Essa interpretação tem inspirado trabalhos na área de SMA e diversas soluções tem sido propostas.

Em [Brito and Hübner 2014] classificou-se trabalhos concebendo instituições situadas em ontológicas e funcionais. A primeira, refere-se a trabalhos que modelam instituições em conceitos abstratos e posteriormente relacionam tais conceitos com elementos concretos do ambiente ou do estado normativo. A segunda conecta elementos do mundo aos diferentes estados do ciclo de vida das normas. A classificação funcional foi subdividida em *conceitual* e *interface*. A subdivisão *conceitual* abrange trabalhos que definem como o ciclo de vida de uma abstração especifica, representa por conceitos institucionais, pode ser afetado por fatos ocorrendo no ambiente. A outra subcategoria, *interface*, não inclui conceitos institucionais nos modelos. Nessas abordagens, uma interface monitora o ambiente e, interpretando a especificação situada, produz informações sobre o que deve acontecer na instituição.

Uma nova categorização foi apresentada em [Brito et al. 2016]. Neste trabalho, os artigos são divididos em três grupos i) realidade institucional como uma questão de interoperabilidade; ii) realidade institucional como estado normativo e iii) realidade institucional como estado constitutivo. O primeiro grupo refere-se a trabalhos que fazem uma interface entre fatos que ocorrem no ambiente e o estado normativo. A plataforma normativa é responsável por interpretar o dado e considerá-lo na regulação. Isto significa que a plataforma normativa é responsável por construir a realidade institucional baseada nos dados recebidos. O próximo grupo considera a realidade institucional como sendo o estado normativo, isto é, fatos ocorrendo no ambiente desencadeiam a ativação, violação, satisfação, etc, de normas presentes no estado normativo. Por fim, a categoria de realidade institucional como estado constitutivo é composta por contrapartidas institucionais de elementos ambientais. Por exemplo, uma *regra constitutiva* pode definir que algum fato ambiental conta como pagar e algum agente conta como devedor. Essas relações entre as contrapartidas institucionais e elementos ambientais compõem o estado constitutivo - que é a interpretação das *regras constitutivas* - da instituição.

É importante destacar que todas essas classificações e agrupamentos de trabalhos relacionados à concepção de instituições artificiais analisaram aspectos objetivos relativos ao modo de como a realidade institucional foi de fato implementada em soluções computacionais. Todavia, a teoria de Searle possui alguns aparatos conceituais que até onde sabemos não foram explorados explicitamente por trabalhos na área de SMA, sendo portanto, o foco desta pesquisa.

4. Revisão Sistemática

A revisão sistemática (RS) é um método de síntese de evidências que avalia criticamente e interpreta pesquisas relevantes disponíveis para uma questão particular, área do conhecimento ou fenômeno de interesse [Brasil 2012]. Em [Kitchenham 2004] é proposto um método de revisão com três fases: i) planejando a revisão, ii) conduzindo a revisão e iii) reportando a revisão. Na primeira etapa, a partir da descoberta da necessidade de se realizar uma revisão acerca de um assunto específico, deve-se elaborar um conjunto de questões que objetivam ser respondidas no decorrer da revisão. Para tanto, elaboramos as seguintes questões:

1. As regras constitutivas, que definem a imposição de status a um elemento X, são desenvolvidas via acordo coletivo ou algum outro mecanismo de reconhecimento [Searle 1995, p.41, 44, 47,51]?
2. As regras constitutivas, que definem a imposição de status a um elemento X, estão sendo compostas por um conjunto de poderes deônticos (direitos, obrigações, etc)[Searle 1995, p.74] ?
3. As regras constitutivas podem ser iterativas, isto é, pode ser atribuída uma nova função de status condicionada a uma entidade do ambiente já possuir previamente outra função de status [Searle 1995, p.80]. Ex. Cidadão Brasileiro (X) pode tornar-se presidente (Y). Todavia, para ser considerado cidadão brasileiro (X), também é necessário possuir um status, demandando uma atribuição de status anterior - "indivíduo nascido em território brasileiro (X) é considerado um cidadão brasileiro (Y)". Como as implementações tem desenvolvido esse mecanismo na prática ?
4. Um função de status atribuído a um elemento do mundo concreto pode ser resultado da combinação de outras funções de status atribuídas anteriormente a elementos do mundo concreto ou a vigência de alguns estados ambientais. Ex. A função de status "todos saírem da sala E apagarem as luzes (ambos estados ambientais) (X) conta como evacuar a sala (Y). Os modelos atuais comportam isso? De que forma?
5. Searle afirma que uma função pode ser imposta via ato de fala [Searle 1995, p.82]. Por exemplo, o ato de alguém pronunciar um número (X) conta como uma oferta em um leilão (Y) se o número anunciado for maior do que a oferta anterior (C). Nos SMAs, existe a possibilidade de impor status a elementos/pessoas através de atos de fala (isto é, troca de mensagens entre agentes)?
6. Cada uso da instituição é uma expressão renovada do comprometimento dos usuários com a instituição [Searle 1995, p.57]. Dessa forma, entende-se que a instituição tem um prazo de validade e precisa ser constantemente renovada para ser mantida. Como esse aspecto tem sido implementado nos SMAs?

As perguntas foram baseadas na teoria de Searle. Quanto ao planejamento da revisão, optou-se por utilizar alguns dos trabalhos discutidos em [Brito and Hübner 2014], [Brito et al. 2016] e [De Brito et al. 2018] que já foram filtrados por critérios semelhantes. Consequentemente, a definição das palavras-chaves, critérios de inclusão/exclusão de trabalhos e bases de dados para a realização da pesquisa não foram necessários. A revisão avança para a próxima fase, conduzindo a revisão, especificamente para as tarefas de sumarização e avaliação dos dados presentes nos artigos.

4.1. Sumarização e Avaliação dos dados

A Tabela 1 apresenta a síntese dos resultados a partir da análise dos trabalhos selecionados. A coluna de número 1 refere-se à necessidade das *funções de status* serem validadas na instituição via acordo coletivo ou algum outro mecanismo de aceitação. Esta coluna refere-se à pergunta de número 1 apresentada na seção 4. Em [Dastani et al. 2012, Cardoso and Oliveira 2007], fatos institucionais são criados via atos de fala (trocas de mensagens entre agentes) que posteriormente transformam-se em contratos - *commitments* - caso os agentes aceitem as condições propostas no contrato. Esses contratos possuem relações com fatos brutos porque estão relacionados a ações que os agentes devem executar no ambiente para ter um significado na dimensão normativa. Contudo, ao estipular um contrato, os agentes se comprometem em realizar determinada ação em um instante de tempo. O comprometimento entre as partes é opcional, ou seja, o agente pode recusar a proposta de estabelecimento do contrato. Todavia, a possibilidade de reconhecer o fato institucional está restrita aos agentes que estão participando daquela interação, sendo desconhecida pelos demais agentes que compõem o sistema. Os demais trabalhos especificam fatos institucionais pré-definidos em tempo de projeto, não existindo mecanismos de aceitação no sistema para tais fatos.

A coluna número 2 diz respeito a obrigatoriedade das *funções de status* constituídas carregarem consigo um conjunto de *poderes deônticos* que representam direitos, obrigações, proibições, etc. Esta coluna refere-se à pergunta de número 2 apresentada na seção 4. [Boella and van der Torre 2006] mistura fatos brutos com fatos institucionais e normas em sua proposta de arquitetura. [Aldewereld et al. 2009, Aldewereld et al. 2010] criam regras *count-as* que relacionam fatos brutos a *funções de status* e regras *count-as* que relacionam *funções de status* a normas. [Piunti 2010, De Brito et al. 2012] criam regras *count-as* que entregam à dimensão normativa a informação de acontecimentos na dimensão ambiental. Todavia, é a dimensão normativa que interpreta as informações recebidas e propõe um significado especial à elas. A informação entregue não contêm nenhum poder deôntico associado. Para [Dastani et al. 2009b, Dastani et al. 2009a] fatos

Table 1. Síntese da análise dos trabalhos relacionados. As colunas 1 a 6 correspondem a respostas às perguntas da seção anterior. O símbolo "$\surd$" significa plenamente atingido, "∂" parcialmente atingido e "–" não atingido.

	1	2	3	4	5	6
[Boella and van der Torre 2006]	–	$\surd$	–	–	–	∂
[Aldewereld et al. 2009, Aldewereld et al. 2010]	–	$\surd$	–	–	$\surd$	∂
[Piunti et al. 2010]	–	–	–	–	$\surd$	∂
[De Brito et al. 2018]	–	$\surd$	$\surd$	∂	$\surd$	$\surd$
[Dastani et al. 2009a, Dastani et al. 2009b]	–	$\surd$	$\surd$	–	–	∂
[Dastani et al. 2012]	∂	$\surd$	$\surd$	–	$\surd$	$\surd$
[Cardoso and Oliveira 2007]	∂	$\surd$	–	–	$\surd$	$\surd$
[Cliffe et al. 2006]	–	$\surd$	–	–	–	∂
[Campos et al. 2008]	–	–	–	–	–	∂
[Fornara et al. 2008]	–	$\surd$	$\surd$	–	$\surd$	$\surd$
[De Brito et al. 2012]	–	–	–	–	$\surd$	∂
[Viganò and Colombetti 2006]	–	$\surd$	$\surd$	–	$\surd$	$\surd$

institucionais representam o estado normativo da organização. Também são especificados regras *count-as* para relacionar fatos brutos a fatos institucionais e possui um mecanismo de monitoramento que objetiva garantir que as normas sejam cumpridas ou penalizar em caso de violação. Em [Dastani et al. 2012, Cardoso and Oliveira 2007] as obrigações são especificadas na sintaxe do compromisso - *commitment* - e portanto, criam fatos institucionais e relacionam-os com normas. Em direção parecida, [Viganò and Colombetti 2006] especifica *funções de status* em termos de posições deônticas, que representam quais ações são autorizadas, obrigadas, proibidas ou permitidas para um agente. Em [Cliffe et al. 2006] normas fazem referências a eventos, podendo ser eventos observáveis e eventos institucionais (constituídos a partir do ambiente). [Campos et al. 2008] propõe uma instituição situada que contem agentes responsáveis por monitorar alguns elementos do ambiente, propiciando que a instituição controle estes elementos ambientais supervisionados. Usa-se normas para representar convenções sociais em relação a interação dos agentes. Ações permitidas seguem normas e não permitidas podem ser eventualmente penalizadas. Todavia, em direção semelhante a [Piunti 2010, De Brito et al. 2012], a informação lida do ambiente não está associado a poderes deônticos, ficando a cargo da instituição interpretá-la e executar ações associadas a tais interpretações. Nos trabalhos de [Fornara et al. 2008, De Brito et al. 2018] ao assumir um papel implementado dentro da instituição, o agente se compromete com alguns compromissos que são regulados por meio de normas.

A coluna de número 3 é relativa a iteratividade das regras constitutivas, isto é, a possibilidade de atribuir um novo status a uma entidade ambiental que já possui um status prévio. Esta coluna refere-se à pergunta de número 3 apresentada na seção 4. [Boella and van der Torre 2006] só admite fatos brutos como dado de entrada de sua arquitetura, impossibilitando essa iteração. [Aldewereld et al. 2009, Aldewereld et al. 2010] separa o fato institucional em contextos, isto é, um carro de policia pode ter uma determinada *função de status* no cenário 2 e não representar nada no cenário 3, não sendo possível atingir também essa propriedade. Em [Cardoso and Oliveira 2007] as regras constitutivas têm como um de seus propósitos estabelecer compromissos por meio de contratos. Todavia, esses contratos não contém em sua especificação meios de descrever pré-condições para que eles ocorram. [Campos et al. 2008] os fatos institucionais acontecem por meio da interpretação de informações coletas por agentes em propriedades ambientais. Todavia, não é possível realizar a iteração. Em [Piunti et al. 2010, Brito et al. 2012] são descritas regras *count-as* para dar significado a eventos ou a eventos e estados que ocorrem no ambiente. Especificar mais de um significado para o mesmo evento acarretará conflitos na dimensão normativa. No trabalho de [Cliffe et al. 2006] a ocorrência de um fato institucional associado a um conjunto de regras transacionais - regras que são descritas para modificar o estado do sistema - modificam o estado atual da instituição. Entretanto, não é mencionado a possibilidade de especificar fatos institucionais a elementos que já contêm um fato institucional atribuído. [Dastani et al. 2009a, Dastani et al. 2009b] garante a iteratividade de regras constitutivas por meio de regras *count-as* que podem descrever pré-condições (sendo acontecimentos no ambiente ou fatos institucionais) e pós condições para serem executadas. Uma pré-condição pode ser a validade de um determinado fato institucional para que outro fato institucional (associado a aquele fato) possa ocorrer. Em direção similar, [Viganò and Colombetti 2006, Fornara et al. 2008, Dastani et al. 2012] especificam

mecanismos que garantem pré-condições (fatos institucionais) para que ações ocorram modificando a instituição. [De Brito et al. 2018] essa característica chama-se constituição de segunda ordem, em que é possível definir que uma função de status conta como outra função de status.

A coluna de número 4 refere-se à combinação de duas ou mais *funções de status* para a geração de uma nova *função de status*. Esta coluna refere-se à pergunta de número 4 apresentada na seção 4. De todos os trabalhos investigados, [De Brito et al. 2018] oportuniza tal associação em relação a funções de status relacionadas a estados, todavia, a funções de status atribuídas a eventos não é uma tarefa trivial. Os demais trabalhos não oportunizam e, ou, não descrevem há possibilidade desta combinação. A maioria dos trabalhos evolui o estado constitutivo/normativo através de pré-condições que normalmente são a realização de uma ação no ambiente, gerando um fato institucional. Todavia, não é relatado nos trabalhos há viabilidade de combinar dois ou mais fatos institucionais para constituição de um inédito.

A coluna de número 5 é relativa à viabilidade de criar *funções de status* através de atos de fala, que em SMAs, geralmente são trocas de mensagens entre os agentes atuando no sistema. Esta coluna refere-se à pergunta de número 5 apresentada na seção 4. No trabalho de [Boella and van der Torre 2006] o dado de entrada da arquitetura é um acontecimento no ambiente e não é elucidado no trabalho a possibilidade desse fato ser a troca de mensagens entre agentes atuando no ambiente. Em [Aldewereld et al. 2009, Aldewereld et al. 2010] é descrita uma espécie de ontologia que impõe um significado para cada coisa do ambiente. É permitido esboçar atos de fala com um significado especial na plataforma regulativa. [Piunti et al. 2010, Brito et al. 2012] entregam um significado especial para a plataforma regulativa de eventos e eventos e estados que acontecem no ambiente. Pode-se especificar que um ato de fala é um evento ocorrido, portanto, a propriedade é contemplada. [Dastani et al. 2009a, Dastani et al. 2009b, Cliffe et al. 2006, Campos et al. 2008] retratam ações no ambiente que tem algum efeito normativo, desconsiderando a troca de mensagens. Em [Dastani et al. 2012, Cardoso and Oliveira 2007, Fornara et al. 2008] é criado fatos institucionais via atos de fala que resultam em contratos e [De Brito et al. 2018, Viganò and Colombetti 2006] relacionam ações no ambiente (troca de mensagens) com um significado especial na interpretação institucional/normativa.

Por fim, a coluna de número 6 aborda a manutenção das instituições condicionada a sua usabilidade. Esta coluna refere-se à pergunta de número 6 apresentada na seção 4. A instituição conceitualmente definida por Searle, é formada por conjuntos de *regras constitutivas*, carregando uma ligação entre um status e um elemento ambiental e um conjunto de poderes deônticos. Os poderes deônticos são considerados por modelos normativos em SMAs, todavia, fazer um link entre os status/fatos institucionais e as normas é essencial para a formação da dinâmica constitutiva do sistema. Diante do exposto, alguns trabalhos apresentam parcialmente a manutenção de instituições como em [Boella and van der Torre 2006, Aldewereld et al. 2009, Aldewereld et al. 2010] por exibirem um ciclo de vida relacionado apenas ao estado normativo, não considerando quando um fato institucional ocorre ou quanto tempo ele é mantido. Outros trabalhos preocupam-se com a condição de satisfazer uma determinada situação para que um fato institucional ocorra [Dastani et al. 2009b, Dastani et al. 2009a, De Brito et al. 2012,

Cliffe et al. 2006, Viganò and Colombetti 2006, Piunti 2010] mas não é considerado aspectos relacionados ao tempo que estará vigorando. Em [De Brito et al. 2018] a própria sintaxe da especificação de *regras constitutivas* leva em consideração o motivo para que a regra seja ativada e quais circunstâncias manterão ela ativa. Em [Dastani et al. 2012, Cardoso and Oliveira 2007, Fornara et al. 2008] os *Commitment* são as condições de ativação para um fato institucional e em sua sintaxe é especificado o seu tempo de vida.

5. Conclusões e Trabalhos Futuros

O objetivo desta pesquisa foi comparar alguns trabalhos que tratam de instituições virtuais com pontos cruciais da teoria de Searle. Observa-se que algumas questões já estão bem exploradas, como o caso de uma função de status ter interpretações normativas (questão 2) ou as plataformas considerarem a troca de mensagens como uma possibilidade de se criar funções de status (questão 5). Algumas questões estão parcialmente exploradas, como é o caso da questão 3 (iterações em funções de status). A maioria das soluções atingidas apresentaram soluções simplistas que apenas garantem a realização de uma função de status ao passo que a outra está ativa. Todavia, realizar o mapeamento de funções de status parece ser uma questão em aberto neste tipo de sistema. Nesta mesma direção, apenas [De Brito et al. 2018] propicia a combinação de duas ou mais funções de status para a geração de uma inédita. Esta característica é particularmente importante quando a combinação resulta em uma nova função de status que promove a satisfação de um estado no ambiente ou a possibilidade de realização de um evento proibido até o momento.

O assunto abordado na questão 6 é tratado por todos os trabalhos analisados. Todavia, vários conceituaram apenas o ciclo de vida da dimensão normativa, desconsiderando aspectos importantes como o motivo da ocorrência de uma função de status ou seu tempo de duração. Tais características são concebidas em trabalhos que tratam de *Commitments* ou em [De Brito et al. 2018] onde é possível descrever regras constitutivas considerando tais perspectivas. A necessidade de um mecanismo de reconhecimento coletivo ou de aceitação de funções de status recomendado por Searle também é uma questão aberta nas sociedades artificiais. Nenhum trabalho analisado leva em consideração a viabilidade de reconhecimento da função de status imposta a um elemento perante aos demais membros da instituição.

Como trabalhos futuros, é recomendado a concepção de um mecanismo que possibilite que os demais agentes que compõem a instituição consigam aceitar a atribuição de novas funções de status a elementos ambientais. Tal característica aproximará ainda mais as instituições artificiais de sociedades humanas. Indica-se também a necessidade de um método capaz de mapear as funções de status. Ao mapear um conjunto de funções de status, pode-se automaticamente atribuir funções de status hierarquicamente inferiores a atual atribuída ao agente. Por exemplo, antes de um agente ser presidente, ele deve ser político, eleitor, cidadão, empregado, etc. Um mecanismo de mapeamento possibilitará que ao ser declarado político, o agente receba automaticamente as funções de eleitor, cidadão, empregado e também seu conjunto de responsabilidades.

6. Agradecimentos

Os autores agradecem ao Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS) pelo fomento desta pesquisa.

References

Aldewereld, H., Alvarez-Napagao, S., Dignum, F., and Vázquez-Salceda, J. (2009). Engineering social reality with inheritance relations. In *International Workshop on Engineering Societies in the Agents World*, pages 116–131. Springer.

Aldewereld, H., Álvarez-Napagao, S., Dignum, F., and Vázquez-Salceda, J. (2010). Making norms concrete. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 807–814. International Foundation for Autonomous Agents and Multiagent Systems.

Artikis, A., Pitt, J., and Sergot, M. (2002). Animated specifications of computational societies. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, pages 1053–1061. ACM.

Boella, G. and Van Der Torre, L. (2004). An agent oriented ontology of social reality. *Procs. of FOIS*, 4:199–209.

Boella, G. and van der Torre, L. (2006). An architecture of a normative system: counts-as conditionals, obligations and permissions. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 229–231. ACM.

Brasil, Ministério da Saúde (MS), S. d. C. T. e. I. E. D. d. C. e. T. (2012). Diretrizes metodológicas: elaboração de revisão sistemática e metanálise de ensaios clínicos randomizados.

Brito, M. d. et al. (2012). Uma linguagem para especificação da dinâmica dos fatos institucionais em sistemas multiagentes.

Brito, M. d. et al. (2016). *A model of institucional reality supporting the regulation in artificial institutions*. PhD thesis, Universidade Federal de Santa Catarina.

Brito, M. d. and Hübner, J. F. (2014). Institutional situatedness in multi-agent systems. *wesaac*, page 12.

Campos, J., López-Sánchez, M., Rodríguez-Aguilar, J. A., and Esteva, M. (2008). Formalising situatedness and adaptation in electronic institutions. In *International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems*, pages 126–139. Springer.

Cardoso, H. L. and Oliveira, E. (2007). Institutional reality and norms: Specifying and monitoring agent organizations. *International Journal of Cooperative Information Systems*, 16(01):67–95.

Cliffe, O., De Vos, M., and Padget, J. (2006). Answer set programming for representing and reasoning about virtual institutions. In *International Workshop on Computational Logic in Multi-Agent Systems*, pages 60–79. Springer.

da Rocha Costa, A. C. (2014). Proposal for a notion of modularity in multiagent systems. In *Pre-proceedings of Engineering Multi-Agent Systems Workshop at AAMAS, Paris*, pages 21–40.

Dastani, M., Grossi, D., Meyer, J.-J. C., and Tinnemeier, N. (2009a). Normative multi-agent programs and their logics. In *Knowledge Representation for Agents and Multi-Agent Systems*, pages 16–31. Springer.

Dastani, M., Tinnemeier, N. A., and Meyer, J.-J. C. (2009b). A programming language for normative multi-agent systems. In *Handbook of research on multi-agent systems: semantics and dynamics of organizational models*, pages 397–417. IGI Global.

Dastani, M., Van Der Torre, L., and Yorke-Smith, N. (2012). Monitoring interaction in organisations. In *International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems*, pages 17–34. Springer.

De Brito, M., Hübner, J. F., and Boissier, O. (2018). Situated artificial institutions: stability, consistency, and flexibility in the regulation of agent societies. *Autonomous Agents and Multi-Agent Systems*, 32(2):219–251.

De Brito, M., Hübner, J. F., and Bordini, R. H. (2012). Programming institutional facts in multi-agent systems. In *International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems*, pages 158–173. Springer.

Fornara, N., Viganò, F., and Colombetti, M. (2004). Agent communication and institutional reality. In *International Workshop on Agent Communication*, pages 1–17. Springer.

Fornara, N., Viganò, F., Verdicchio, M., and Colombetti, M. (2008). Artificial institutions: a model of institutional reality for open multiagent systems. *Artificial Intelligence and Law*, 16(1):89–105.

Helleboogh, A., Vizzari, G., Uhrmacher, A., and Michel, F. (2007). Modeling dynamic environments in multi-agent simulation. *Autonomous Agents and Multi-Agent Systems*, 14(1):87–116.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26.

Piunti, M. (2009). *Situating agents and organisations in artifact-based work environment*. PhD thesis, Univerist di Bologna.

Piunti, M. (2010). *Designing and programming organizational infrastructures for agents situated in artifact-based environments*. PhD thesis, alma.

Piunti, M., Boissier, O., Hübner, J. F., and Ricci, A. (2010). Embodied organizations: a unifying perspective in programming agents, organizations and environments. *COIN10@ MALLOW*, pages 98–114.

Searle, J. (2010). *Making the social world: The structure of human civilization*. Oxford University Press.

Searle, J. R. (1995). *The construction of social reality*. Simon and Schuster.

Viganò, F. and Colombetti, M. (2006). Specification and verification of institutions through status functions. In *International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems*, pages 115–129. Springer.

Efeitos de Estratégias de Distribuição de Recompensas em Coalizões Baseadas em Agentes: Resultados Preliminares

Luís Gustavo Ludescher[1], **Jaime Simão Sichman**[1]

[1]Laboratório de Técnicas Inteligentes (LTI)
Escola Politécnica (EP)
Universidade de São Paulo (USP)

{lgludescher, jaime.sichman}@usp.br

Abstract. *In a typical political system in which leaders are responsible for distribution of benefits among people, coalitions are usually formed in which some individuals, seeking to serve their own interests, support the emergence of certain leaders. The purpose of this work is to evaluate the dynamics of coalition formation and distribution of rewards in such a context. Using agent-based simulation, we evaluate a model in which three leader profiles are considered and individuals seek to maximize their earnings either by participating or not in coalitions. The preliminary results indicate that egoist leaders encourage the formation of more coalitions and promote more imbalances.*

Resumo. *Num sistema político típico em que líderes são responsáveis pela distribuição de benefícios à população, geralmente verifica-se a formação de coalizões em que alguns indivíduos, buscando atender a interesses próprios, apoiam a surgimento de certas lideranças. O objetivo deste trabalho é avaliar a dinâmica de formação de coalizões e distribuição de recompensas em tal contexto. Utilizando-se simulação baseada em agentes, avalia-se um modelo no qual são considerados três perfis de líderes e os indivíduos tentam maximizar seus ganhos participando ou não de coalizões. Os resultados, ainda preliminares, indicam que líderes egoístas encorajam a formação de mais coalizões e promovem maiores desequilíbrios.*

1. Introdução

Na versão clássica do *Public Goods Game*, cada indivíduo escolhe secretamente o quanto deseja contribuir para o bem público. O montante coletado é então multiplicado por um fator de ganho (maior do que 1 e menor do que o número total de indivíduos) e distribuído igualmente por toda a população. Evidentemente, do ponto de vista coletivo, o cenário mais favorável é aquele em que todos contribuem o máximo disponível, maximizando o bem coletivo. Entretanto, considerando que o objetivo de cada indivíduo é maximizar o seu próprio ganho e que a escolha de cada um é desconhecida pelos demais, impossibilitando acordos ou retaliações, a melhor estratégia individual é não contribuir com nada, pois quem a adota se beneficia igualmente da partilha do bem público sem precisar oferecer nada em troca. Adicionalmente, experimentos mostram [Fischbacher et al. 2001] que alguns indivíduos ainda cooperam sob determinadas condições e suas decisões sobre contribuir ou não são influenciadas pelas escolhas dos demais, mas um eventual comportamento cooperativo que se apresente inicialmente tende a declinar ao longo do tempo.

Já num cenário político mais realista, estruturado em torno de um governo central, verifica-se duas distinções fundamentais em relação ao *Public Goods Game*: as contribuições (de parte) dos indivíduos são compulsórias, na forma de impostos; e a distribuição de benefícios (bens, serviços, políticas econômicas, etc) não é necessariamente igualitária, cabendo aos representantes políticos as decisões sobre quais grupos, localidades e interesses devem ser priorizados. Embora tal estrutura, em tese, inviabilize que os indivíduos adotem a estratégia de não contribuição, o fato de a distribuição de benefícios estar a cargo de representantes políticos abre caminho para outro tipo de estratégia: indivíduos podem formar coalizões para apoiar o surgimento de certos líderes com a expectativa de serem recompensados pelo apoio oferecido caso tais líderes sejam escolhidos como representantes políticos. Tal possibilidade pode levar a distorções na maneira como os governos distribuem os benefícios à população [Gilens and Page 2014], fazendo com que apenas determinados grupos tenham suas demandas atendidas.

Este trabalho se propõe a avaliar a dinâmica de formação de coalizões e distribuição de recompensas em um cenário simplificado em que indivíduos tentam maximizar seu próprio ganho, optando por participar ou não de coalizões. Em cada ciclo de simulação, a coalizão vencedora, cujo líder é o responsável pela administração do bem público, emerge a partir do conjunto de coalizões formadas.

2. Coalizões Baseadas em Agentes

Coalizões são formadas quando agentes podem se beneficiar através da cooperação, de modo que um agente pertencente a uma coalizão potencialmente obtenha mais benefícios do que obteria fora dela [Shehory and Kraus 2005]. Portanto, a forma de distribuição dos benefícios é fundamental na estabilidade de uma coalizão: cada membro avalia o benefício recebido e então decide se vale a pena continuar cooperando ou não. A melhor forma conhecida de divisão de benefícios em uma coalizão é o *Shapley Value*: a recompensa de um indivíduo é calculada proporcionalmente a quanto a sua participação agrega ao desempenho do conjunto [Ieong and Shoham 2005]. A rigor, portanto, uma coalizão só se justifica quando o ganho do conjunto supera a soma dos ganhos individuais de seus membros.

Em [Nardin et al. 2014] é apresentado um modelo de coalizão baseada em agentes que ilustra bem sua dinâmica: os agentes, distribuídos em um *grid*, competem individualmente por recompensas e, a cada rodada, avaliam o desempenho de seus vizinhos. Caso um agente identifique um vizinho com melhor desempenho, ele pode decidir participar da coalizão do vizinho (se existir uma) ou formar uma nova coalizão em que o agente de melhor desempenho torna-se o líder. Um líder é responsável por coletar os ganhos de cada membro de sua coalizão e redistribuí-los, cobrando uma taxa por exercer tal papel. Por fim, um membro de uma coalizão deve comparar seus benefícios recebidos com os obtidos por seus vizinhos, podendo abandonar sua coalizão caso a considere desfavorável.

É interessante notar que a determinação de quanto cada membro de uma coalizão agrega ao conjunto é frequentemente subjetiva, bem como a percepção de um membro sobre o quanto ele está sendo recompensado de maneira justa ou não. Portanto, a determinação do *Shapley Value* nem sempre é possível, podendo-se, em cenários mais complexos, fazer apenas estimativas de seu valor. Por exemplo, num caso em que a distribuição de benefícios em uma coalizão seja responsabilidade de um líder e que não

necessariamente haja uma única forma justa (conhecida) de distribuição, cabe ao líder tentar fazê-la de modo a manter certa estabilidade.

Em [Schreiber 2014] é apresentado um modelo baseado em agentes que utiliza o conceito de coalizão para simular a formação de partidos políticos em um cenário simplificado. Neste trabalho, demonstra-se que é possível obter essencialmente os mesmos resultados presentes nos modelos clássicos de dinâmica de partidos utilizando-se apenas um conjunto de regras simples: os agentes, posicionados tanto em cenários unidimensionais quanto bidimensionais, buscam a cada iteração formar coalizões com agentes próximos de posições políticas semelhantes até que suas coalizões consigam maioria; a posição política de uma coalizão a princípio reflete a posição política intermediária entre seus membros, podendo ser alterada conforme agentes entram e saem dela; além disso, coalizões perdedoras podem tentar ajustar suas posições políticas na tentativa de atrair novos membros, bem como membros de coalizões podem deixá-las caso não se considerem mais representados.

3. Modelo

Utilizando-se técnicas de simulação baseada em agentes [Sichman 2015] foi criado e simulado um modelo em que os agentes representam indivíduos de uma sociedade e são interconectados através de uma rede. Cada indivíduo possui inicialmente uma mesma quantia de recursos (capital), parte da qual deve ser utilizada para o pagamento de um imposto fixo (por rodada) e pode também ser utilizada para apoiar (financiar) um líder de coalizão, se for o caso. Tais indivíduos então interagem entre si definindo lideranças e formando coalizões. A cada rodada é eleita uma coalizão vencedora e seu respectivo líder é responsável por redistribuir o total arrecadado via imposto, após multiplicá-lo por um fator de ganho. Ao final de cada rodada, os indivíduos recebem suas recompensas e tomam decisões com base no seu grau de satisfação.

3.1. Cenário

Os agentes são interconectados através de uma rede do tipo livre de escala [Barabási and Bonabeau 2003], na qual a maior parte dos nós têm poucas conexões enquanto alguns outros, denominados *hubs*, acabam concentrando grandes quantidades de conexões. Entre exemplos de redes livre de escala estão as redes de interação entre proteínas e a rede formada pelas linhas de transporte aéreo, além das redes sociais em geral, motivo pelo qual foi selecionado tal tipo de rede para este modelo. Para a formação da rede livre de escala, foi escolhido um método amplamente utilizado baseado no modelo Barabási-Albert [Albert and Barabási 2002].

3.2. Fator de Ganho

Conforme citado anteriormente, o *Public Goods Game* clássico utiliza um fator de ganho que amplifica o total coletado de contribuições, o que foi incorporado neste modelo. Este fator pode ser entendido como um ganho promovido pela cooperação entre indivíduos. Adicionalmente, considera-se que diferentes líderes têm (possivelmente) diferentes aptidões para administrar o bem público, o que é representado no modelo através da atribuição de diferentes fatores de ganho a diferentes indivíduos.

3.3. Perfis

No modelo, um líder possui uma estratégia, que é definida como o quanto ele está disposto a recompensar a sua coalizão em caso de vitória. Tal estratégia é modelada como um fator de recompensa que define o quanto do total da recompensa a ser distribuída o líder oferecerá à sua coalizão. Para efeito comparativo, foram criados três perfis de líder: (i) *egoísta*, que oferece uma fatia maior de recompensa à sua coalizão, e portanto diminuindo a recompensa dos demais membros da população; (ii) *altruísta*, que oferece uma fatia menor de recompensa à sua coalizão, tendendo a uma distribuição mais igualitária; (iii) *intermediário*, cuja estratégia é intermediária entre as duas anteriores.

3.4. Ciclos

Cada ciclo da simulação é composto por 5 etapas:

1. Formação de coalizões;
2. Investimentos nas coalizões;
3. Definição da coalizão vencedora;
4. Atuação do líder escolhido;
5. Avaliação de recompensas.

O pseudo-algoritmo, apresentado no apêndice A, mostra em maiores detalhes o funcionamento da simulação.

3.4.1. Formação de coalizões

Primeiramente, cada indivíduo decide se pretende ou não ser líder. Tal decisão é aleatória e baseada em uma probabilidade parametrizada. Em seguida, com base em uma probabilidade variável, cada indivíduo decide se pretende ou não participar de uma coalizão. Em caso positivo, ele busca o vizinho com maior fator de ganho para formar uma coalizão (ou entrar em uma já formada) e a confiança em seu novo líder é inicializada com um valor padrão.

3.4.2. Investimentos nas coalizões

Membros de coalizões decidem o quanto pretendem investir nas respectivas coalizões. Tal investimento é definido como um percentual do capital que o indivíduo tem disponível.

3.4.3. Definição da coalizão vencedora

A probabilidade de uma coalizão vencer uma rodada é dada pelo total de investimentos que ela recebeu dividido pelo total de investimentos recebidos por todas as coalizões.

3.4.4. Atuação do líder escolhido

Primeiramente, um imposto fixo (parametrizado) é cobrado de todos os indivíduos. Em seguida, o líder da coalizão vencedora amplifica o total arrecadado com seu fator de ganho, gerando um valor de recompensas a ser distribuído. O fator de recompensa da coalizão vencedora é definido pelo perfil do seu líder, bem como a estratégia de distribuição.

Cada membro da coalizão recebe uma recompensa proporcional ao seu investimento; já o restante da população, composta por membros de outras coalizões e indivíduos independentes, recebe um mesmo valor correspondente às recompensas restantes.

3.4.5. Avaliação de recompensas

Cada indivíduo então avalia a recompensa recebida:

1. Membros da coalizão vencedora comparam a recompensa recebida com o investimento acumulado na coalizão. Caso o retorno do investimento tenha sido superior a um limiar parametrizado, a confiança do indivíduo no seu líder aumenta; em caso contrário, ela cai drasticamente, levando-o a provavelmente abandonar a coalizão, se a confiança ficar abaixo de determinado limiar;
2. Membros de outras coalizões (não vencedoras) têm a confiança em seus líderes decrementadas. Caso a confiança de um membro seja inferior a determinado limiar, ele também sai da coalizão;
3. Indivíduos independentes avaliam se a recompensa recebida está acima de certo limiar (parametrizado) de satisfação. Em caso positivo, a probabilidade de o indivíduo querer participar de uma coalizão na rodada seguinte é decrementada, ou incrementada em caso contrário. Além disso, baseado na mesma probabilidade, o indivíduo pode procurar o vizinho que recebeu a maior recompensa e se juntar à sua coalizão, caso esta exista.

4. Experimentos

A simulação foi desenvolvida utilizando-se a ferramenta Repast[1] [North et al. 2013], voltada a simulação baseada em agentes, e a linguagem de programação ReLogo [Ozik et al. 2013].

4.1. Descrição

Os experimentos iniciais foram feitos com o objetivo de comparar os diferentes perfis de líder. Assim, foram realizadas 3 baterias de experimentos, sendo que em cada uma só havia um único perfil de líder. Cada experimento foi configurado com 1000 indivíduos, e consistiu na execução da simulação por 1000 ciclos. Os experimentos foram repetidos 10 vezes, totalizando 30 experimentos. Os resultados apresentados refletem a média, por perfil, dos resultados obtidos.

4.2. Resultados Obtidos

Os resultados obtidos foram analisados sob duas perspectivas distintas, detalhadas nas subseções seguintes: análise de distribuição e análise temporal. Para a análise de distribuição, foram coletados dados pontualmente ao final de cada simulação, mostrando o capital de que cada indivíduo dispunha e quantos membros cada coalizão possuía. Tais dados foram utilizados para verificar respectivamente a distribuição do capital pelos indivíduos e a distribuição do número de membros pelas coalizões. Já no caso da análise temporal, foram coletados os dados ao longo de cada ciclo da simulação, de modo a indicar a evolução temporal dos valores das variáveis, tais como o número de coalizões e o total de investimentos.

[1] `https://repast.github.io/`.

4.2.1. Análise de Distribuição

As figuras 1 e 2 mostram, respectivamente, a distribuição do capital e da quantidade de membros por coalizão ao final dos 1000 ciclos de simulação para cada perfil de líder. Pode-se notar que líderes egoístas promovem maior desigualdade na distribuição do capital: a figura 1 aponta dois picos, respectivamente ao redor dos valores 100 e 500, e mostra que há uma grande probabilidade de encontrar indivíduos com capital abaixo de 500. Deste modo, perfis egoístas de líder estimulam a busca dos indivíduos pela participação em coalizões, provocando inclusive a formação de coalizões maiores (contendo em torno de 23 e 33 membros, como mostra a figura 2) que tendem a monopolizar as recompensas, beneficiando continuamente um número restrito de indivíduos. Já no cenário com líderes altruístas, verifica-se uma distribuição mais equilibrada do capital (com grande parte dos indivíduos possuindo capital entre 1000 e 1200, como mostra a figura 1) e pouco incentivo à formação de coalizões que, quando formadas, permanecem pequenas (contendo em torno de 3 membros, como mostra a figura 2) e instáveis.

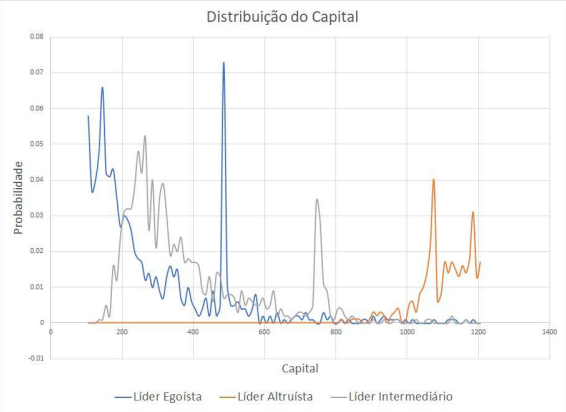


Figura 1. Distribuição do Capital

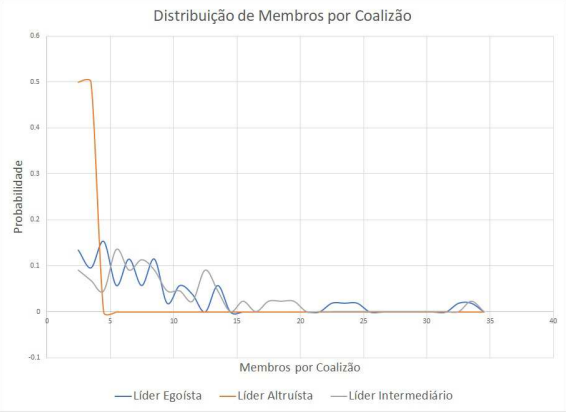


Figura 2. Distribuição de Membros por Coalizão

4.2.2. Análise Temporal

As figuras 3 e 4 apresentam a evolução temporal do número de coalizões e do número total de indivíduos participantes de alguma coalizão ao longo dos ciclos da simulação. Tais

gráficos apresentam dados consistentes em relação à análise anterior: líderes egoístas estimulam a busca pela participação em coalizões enquanto líderes altruístas tornam desinteressante (pouco rentável) investir nelas. Resultado análogo pode ser inferido em relação ao total de investimentos em coalizões ao longo da simulação, exibido na figura 5: líderes egoístas estimulam tais investimentos, enquanto que no caso de líderes altruístas os valores são sensivelmente menores.

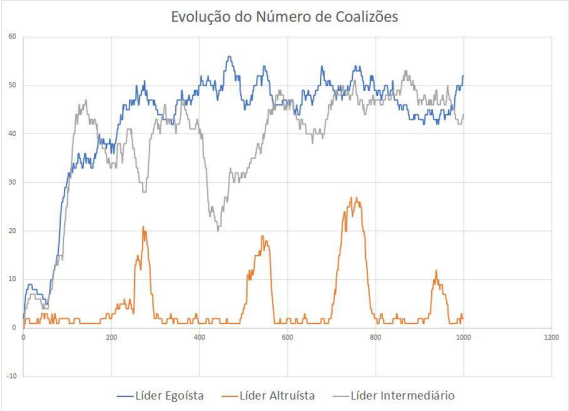


Figura 3. Evolução do Número de Coalizões

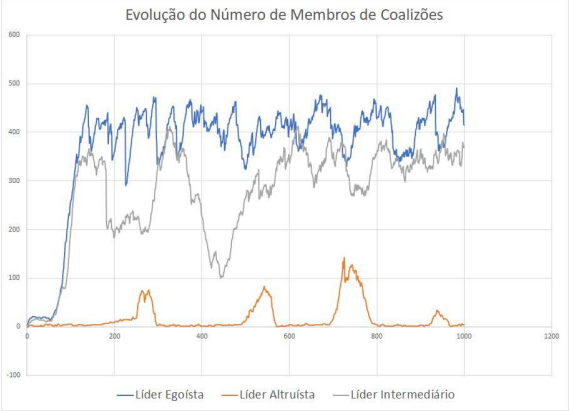


Figura 4. Evolução do Número de Membros de Coalizões

A figura 6 apresenta o total de recompensas distribuídas em cada ciclo da simulação. Pode-se observar que os valores permanecem relativamente estáveis, sendo que as oscilações se devem aos diferentes fatores de ganho dos líderes escolhidos em cada ciclo. Como o objetivo dos experimentos era comparar os diferentes perfis de líder, os indivíduos foram configurados inicialmente com fatores de ganho semelhantes, para que isso não influenciasse a interpretação dos resultados: tal fato explica a relativa estabilidade observada.

Por fim, a figura 7 mostra a evolução do total de capital acumulado por todos os indivíduos ao longo da simulação. É interessante verificar como o desempenho coletivo é favorecido no cenário com líderes altruístas. Isso acontece devido ao fato de poucas coalizões serem formadas e, portanto, os indivíduos quase não investirem parte de seus capitais em coalizões: consequentemente, o capital total praticamente só aumenta, de acordo com o fator de ganho dos líderes. Já no cenário com líderes egoístas, parece haver

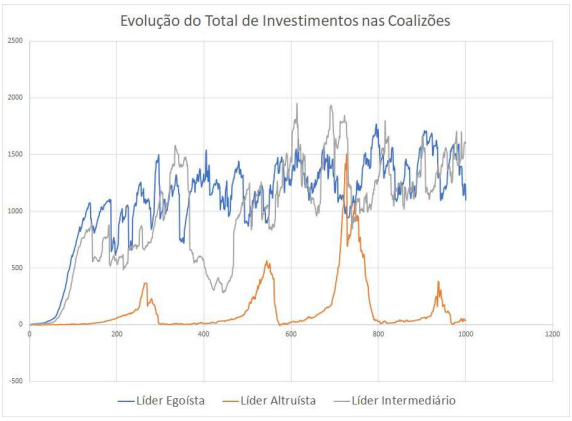


Figura 5. Evolução do Total de Investimentos nas Coalizões

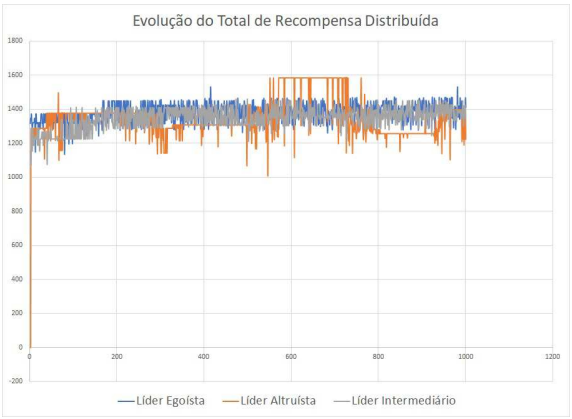


Figura 6. Evolução do Total de Recompensa Distribuída

uma tendência de equilíbrio entre o aumento do capital promovido pelo fator de ganho dos líderes e a perda de capital resultante dos investimentos em coalizões.

5. Conclusões

Os resultados obtidos neste trabalho, ainda que preliminares, mostram que num sistema político típico, em que líderes são responsáveis pela distribuição de benefícios à população e que podem ainda receber investimentos dos indivíduos para que sua coalizão seja vencedora, podem emergir coalizões que comprometam uma distribuição de recompensas mais homôgenea, caso não haja mecanismos eficazes de regulação que as evitem. Apesar disso, a presença de líderes altruístas pode minimizar tais distorções por diminuir o incentivo ao investimento em coalizões, e gerando uma distribuição de recompensas mais homôgenea.

Nas próximas etapas do trabalho, prevê-se a execução de novos experimentos combinando diferentes perfis de líder em uma mesma simulação, complementando os experimentos já realizados com perfis isolados. Outra ideia é fazer experimentos que envolvam diferentes distribuições de fator de ganho entre os indivíduos, permitindo avaliar as relações entre fator de ganho e perfil de líder. Adicionalmente, o modelo deverá ser aprimorado para incluir também diferentes perfis de indivíduos, com diferentes graus de tolerância e expectativa em relação aos ganhos individuais obtidos. Por exemplo,

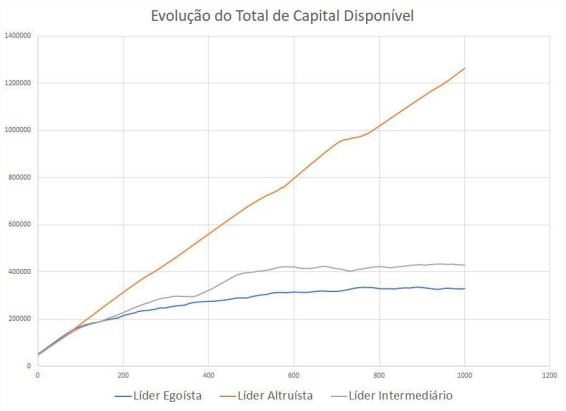


Figura 7. Evolução do Total de Capital Disponível

indivíduos pacientes aceitariam recompensas menores e teriam maior pré-disposição a confiar em líderes, enquanto indivíduos impacientes buscariam maiores ganhos e teriam menos tolerância ao mau desempenho de uma coalizão.

Agradecimentos

Luís Gustavo Ludescher é financiado pelo CNPq, auxílio 132317/2016-8.

Referências

Albert, R. and Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97.

Barabási, A.-l. and Bonabeau, E. (2003). Scale-free networks. *Scientific American*, 3(1):50–59.

Fischbacher, U., Gächter, S., and Fehr, E. (2001). Are people conditionally cooperative? Evidence from a public goods experiment. *Economics Letters*.

Gilens, M. and Page, B. I. (2014). Testing theories of American politics: Elites, interest groups, and average citizens. *Perspectives on Politics*.

Ieong, S. and Shoham, Y. (2005). Marginal contribution nets: A compact representation scheme for coalitional games. In *Proceedings of the ACM conference on Electronic Commerce*.

Nardin, L. G., Rosset, L. M., and Sichman, J. S. (2014). Scale and Topology Effects on Agent-Based Simulation.

North, M. J., Collier, N. T., Ozik, J., Tatara, E. R., Macal, C. M., Bragen, M., and Sydelko, P. (2013). Complex adaptive systems modeling with repast simphony. *Complex adaptive systems modeling*, 1(1):3.

Ozik, J., Collier, N. T., Murphy, J. T., and North, M. J. (2013). The ReLogo agent-based modeling language. In *2013 Winter Simulations Conference (WSC)*, pages 1560–1568.

Schreiber, D. (2014). The Emergence of Parties: An Agent-Based Simulation. *Political Research Quarterly*.

Shehory, O. and Kraus, S. (2005). Coalition formation among autonomous agents: Strategies and complexity (preliminary report).

Sichman, J. S. a. (2015). Operationalizing complex systems. In *Modeling complex systems for public policies*, pages 85–123. IPEA.

A. Pseudo-Algoritmo

Algorithm 1 Pseudo-Algoritmo da Simulação

```
1:  CriaIndivíduos(n)
2:  CriaRedeLivreDeEscala()
3:  for Agente a = 1 to n do
4:     InicializaIndivíduo()
5:  end for
6:  loop
7:     for Agente a = 1 to n do
8:        DecideSePretendeSerLíder(probabilidade)
9:     end for
10:    for Agente a = 1 to n do
11:       if QuerParticiparDeCoalizão(probabilidade) then
12:          vizinho = ProcuraVizinhoComMaiorFatorDeGanho()
13:          if PertenceCoalizão(vizinho) or PretendeSerLíder(vizinho) then
14:             EntraOuFormaCoalizão(vizinho)
15:          end if
16:       end if
17:    end for
18:    for Agente a = 1 to n do
19:       if MembroDeCoalizão() then
20:          InvesteNaCoalizão(%capital)
21:       end if
22:    end for
23:    for Coalizão c = 1 to n do
24:       probabilidadeVitória = Investimento(c) / Investimento(todas as coalizões)
25:    end for
26:    coalizãoVencedora = SelecionaCoalizãoVencedora()
27:    impostoTotal = 0
28:    for Agente a = 1 to n do
29:       PagaImposto(valor)
30:       impostoTotal = impostoTotal + valor
31:    end for
```

```
32:     for Líder l = Líder(coalizãoVencedora) do
33:       recompensaTotal = FatorDeGanho(l) * impostoTotal
34:       recompensaCoalizãoVencedora = FatorDeRecompensa(l) * recompensaTotal
35:       recompensaOutros = recompensaTotal - recompensaCoalizãoVencedora
36:       for Membro m in Membros(coalizãoVencedora) do
37:         recompensa = Investimento(m) / Investimento(coalizãoVencedora) * recompensaCoalizãoVencedora {distribui as recompensas proporcionalmente aos investimentos de cada membro}
38:       end for
39:       númeroOutros = totalPopulação - númeroMembrosCoalizãoVencedora
40:       for OutrosIndivíduos do
41:         recompensa = recompensaOutros / númeroOutros {o que resta da recompensa é igualmente dividido entre os demais indivíduos}
42:       end for
43:     end for
44:     for Agente a = 1 to n do
45:       if Membro(a, coalizãoVencedora) then
46:         if recompensa / investimentoAcumulado is greater than limiar then
47:           AumentaConfiançaNoLíder()
48:         else
49:           DiminuiConfiançaNoLíder()
50:         end if
51:         if confiançaNoLíder is lower than limiar then
52:           SaiDaCoalizão()
53:         end if
54:       end if
55:       if Membro(a, outraCoalizão) then
56:         DecrementaConfiançaNoLíder()
57:         if confiançaNoLíder is lower than limiar then
58:           SaiDaCoalizão()
59:         end if
60:       end if
61:       if IndivíduoIndependente(a) then
62:         if recompensa / imposto greater than limiarDeSatisfação then
63:           DiminuiProbabilidadeDeParticiparDeCoalizão()
64:         else
65:           AumentaProbabilidadeDeParticiparDeCoalizão()
66:         end if
67:         if QuerParticiparDeCoalizão(probabilidade) then
68:           vizinho = ProcuraVizinhoComMaiorRecompensa()
69:           if Existe(vizinho) and PertenceCoalizão(vizinho) then
70:             EntraCoalizão(vizinho)
71:           end if
72:         end if
73:       end if
74:     end for
75:   end loop
```

Protocolo de Interação Entre SMA Embarcados Bio-Inspirado na Relação de Predatismo

Vinicius Souza de Jesus[1], **Fabian Cesar P. B. Manoel**[1], **Carlos Eduardo Pantoja**[1,2]

[1]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)
Av. Maracanã, 229 - Maracanã – 20271-110 – Rio de Janeiro – RJ – Brazil

[2]Universidade Federal Fluminense (UFF)
Niterói – Rio de Janeiro – RJ – Brazil.

{souza.vdj,fabiancpbm}@gmail.com, pantoja@cefet-rj.br

Abstract. *Approaches inspired by biological concepts are commonly found in the area of Artificial Intelligence and help to solve some problems related to the interaction between entities in societies and groups. Thus, the objective of this work is to present a protocol of interaction between open MAS and embedded agents in robotic platforms based on a model adapted from the ecological relations of predation. This protocol aims to preserve the integrity of its knowledge when the hardware of a platform is damaged, by transferring all the agents and knowledge obtained to a known MAS and embarked on another similar platform. The proposed protocol will be implemented using the Jason framework and a proof of concept in a real environment will be presented with two prototypes of land vehicles assuming each the role of predator and prey in the relationship.*

Resumo. *Abordagens inspiradas em conceitos biológicos são comumente encontradas na área da Inteligência Artificial e contribuem para auxiliar na resolução de alguns problemas relacionados com a interação entre entidades em sociedades e grupos. Sendo assim, o objetivo deste trabalho é apresentar um protocolo de interação entre agentes integrantes de SMA abertos e embarcados em plataformas robóticas baseado em um modelo adaptado da relação ecológica de predatismo. Este protocolo tem como objetivo preservar a integridade de seus conhecimentos quando o hardware de uma plataforma estiver danificado, através da transferência de todos os agentes e conhecimentos obtidos para um SMA conhecido e embarcado em outra plataforma similar. O protocolo proposto será implementado utilizando o framework Jason e uma prova de conceito em um ambiente real será apresentada com dois protótipos de veículos terrestres assumindo cada um o papel de predador e presa na relação.*

1. Introdução

Na biologia, o conceito de relações ecológicas [Begon et al. 2005] classifica as interações que ocorrem entre os seres vivos de acordo com a posição que cada ser vivo assume na relação. A relação ecológica de predatismo, por exemplo, é estabelecida quando um ser vivo por meio de seus instintos de sobrevivência domina outro ser vivo para se manter vivo. Similarmente, agentes inteligentes são entidades autônomas capazes de interagir com outros agentes em um Sistema Multi-Agente (SMA) [Huynh et al. 2006]. Nos SMA,

a composição dos agentes integrantes do sistema, a forma como eles interagem, e a possibilidade deles poderem migrar de sistema permitem classificar estes SMA como abertos ou fechados.

Um SMA fechado é caracterizado por seus agentes serem restritos a um sistema de origem e não poderem se locomover para outros sistemas e, normalmente, estes agentes somente interagem com agentes dentro de seu sistema ou esfera de influência [Wooldridge 2009]. Já um SMA aberto [Huynh et al. 2006] é caracterizado pela capacidade dos agentes poderem migrar de um sistema para outro, permitindo que a forma de interação exceda a esfera de influência de origem de um agente. A capacidade de um agente se mover de um sistema a outro depende da existência de agentes móveis por meio de um ambiente aberto. Um agente móvel [Chen et al. 2009] é um agente que possui a capacidade de se transferir de um SMA para outro e interagir com outros agentes em um ambiente compartilhando artefatos e recursos comum a eles.

Diversas técnicas e soluções bio-inspiradas utilizam a abordagem de agentes para a resolução de problemas relacionados com a interação entre entidades em sociedades e grupos [Zeghida et al. 2018]. Existem trabalhos que são inspirados pelos conceitos de biologia para elaborar técnicas e algoritmos que buscam aumentar o nível de interação e comunicação entre sistemas que estão em um mesmo ambiente [Günay et al. 2015]. Por exemplo, [Ferri et al. 2006] apresenta um algoritmo bio-inspirado para SMA para identificar concentrações de gás em um ambiente; [Zedadra et al. 2016] propõe uma situação de busca e exploração de recursos naturais em que um SMA aberto precisa trabalhar em conjunto com outro e executar sub-tarefas simples para concluir uma tarefa complexa comum a todos os SMA. Embora estes trabalhos considerarem SMA abertos e aplicações bio-inspiradas, eles interagem somente com um ambiente aberto onde agentes de diferentes SMA podem se conectar e trocar conhecimentos. Sendo assim, estes trabalhos não levam em consideração o fato de um agente móvel poder se transferir de um SMA para outro e, com isso, não focam na interação entre agentes de diferentes SMA e caso ocorra algum dano na parte física onde está aplicado o SMA destes trabalhos, seus agentes não são capazes de se transferir para outro SMA e preservar a integridade de seus conhecimentos.

Portanto, o objetivo deste trabalho é apresentar um protocolo de transferência de agentes baseado em um modelo adaptado da relação ecológica de predatismo, onde será possível uma plataforma robótica controlada por um SMA embarcado assumir o controle de outra plataforma através da movimentação de todos os agentes de um SMA para o SMA da plataforma de destino. O protocolo de transferência de agentes proposto tem como objetivo preservar a integridade dos conhecimentos dos agentes de um SMA embarcado em situações onde o hardware de sua plataforma robótica estiver comprometida. Sendo assim, o SMA ativa o protocolo para transferir todos os agentes e seus respectivos conhecimentos para um outro SMA embarcado em uma plataforma robótica similar, assumindo assim, o controle desta plataforma. O protocolo é considerado um predatismo adaptado uma vez que o SMA de origem não realiza uma predação escolhendo uma presa aleatória, pois necessita conhecer e ter a permissão do SMA de destino para poder assumir o controle da plataforma.

A implementação dos SMA, o protocolo utilizará o *framework* Jason [Bordini et al. 2007], que possui um interpretador de uma linguagem orientada

a agentes *AgentSpeak* [Rao 1996] e implementa o modelo *Belief-Desire-Inention* (BDI) [Bratman 1987] para programação de agentes cognitivos. O Jason foi escolhido por ser um *framework* de código aberto, com uma interface facilitadora para a programação de SMA. Para permitir SMA abertos e a comunicação entre agentes de SMA distintos, o Jason foi estendido para permitir a integração de um *middleware* [Endler et al. 2011] para a *IoT* e serviços de contexto, que visa aplicações colaborativas, coordenação de atividades entre entidades móveis e compartilhamento de informações. Este *middleware* permite trabalhar com a conexão de milhares de nós moveis conectados ao mesmo tempo [David et al. 2012] e garante escalabilidade.

Como prova de conceito, foram desenvolvidos dois protótipos de veículos terrestre com SMA embarcados para promover testes em ambientes reais. Um dos protótipos é considerado o líder e contém planos, crenças e intenções importantes para uma hipotética missão coletiva. Contudo, é simulado um dano induzido a esta plataforma considerado irreparável e o protocolo de predatismo é ativado. O SMA como um todo migra para uma plataforma de hardware similar previamente conhecida, tomando o seu controle. Testes foram realizados para poder medir a velocidade de acionamento do protocolo de transferência de agentes e verificar se houve a preservação de integridade dos conhecimento dos agentes durante a transmissão assim como se houve a devida tomada de controle.

O trabalho está estruturado da seguinte maneira: na seção 2, o referencial teórico é apresentado; na seção 3, os trabalhos relacionados são discutidos; na seção 4, a metodologia utilizada é explorada e apresentada; na seção 5, a avaliação experimental é discutida; na seção 6 é levantada as discussões do trabalho; na seção 7, as considerações finais são mostradas e; por fim, as referencias são apresentadas.

2. Referencial teórico

Esta seção tem o objetivo de apresentar e descrever os principais conceitos utilizados no trabalho para prover uma melhor compreensão do protocolo de transferência de agentes proposto.

O predatismo é uma relação ecológica da biologia em que um ser vivo, denominado predador, caça outro ser vivo, denominado presa. Esta ação é motivada pelo instinto de sobrevivência da espécie predadora, que captura e mata a espécie de nível inferior da cadeia alimentar, e com isso, se fortalece. No contexto geral, o predatismo é uma relação necessária para manter o equilíbrio do ambiente ecológico. Na IA, SMA agrupam entidades chamadas de agentes para realizarem atividades coletivamente e atingirem seus objetivos. Alguns destes sistemas podem se manter fechados; já em outros cenários, pode haver mais de um SMA interagindo entre si para melhorar a capacidade de resolver problemas.

Os SMA fechados [Wooldridge 2009] são SMA onde seus agentes ficam restritos a se relacionar somente com os agentes de seu SMA para realizar um objetivo comum. Ou seja, em um SMA fechado, os agentes possuem restrições para adquirir novos conhecimentos já que são capazes de interagir somente com os agentes do próprio SMA. Sendo assim, quando são implementados em uma plataforma robótica, estes agentes ficam restritos a esta plataforma. Além disso, caso a plataforma robótica sofra danos físicos, o SMA fechado aplicado ficará impossibilitado de preservar os conhecimentos de seus agentes, que serão perdidos. Já os SMA abertos [Huynh et al. 2006] são SMA que possuem a

capacidade de interagir com agentes de outros sistemas por meio de agentes móveis ou através de um ambiente aberto onde agentes de diferentes sistemas podem interagir e trocar conhecimentos. Com isso, um SMA aberto possui a capacidade de transferir seus agentes para outros SMA ou para um ambientes aberto. Quando aplicado em uma plataforma robótica, este SMA pode preservar os conhecimentos de seus agentes mesmo que esta plataforma sofra danos físicos, pois a transferência dos seus agentes pode acontecer a qualquer momento para um ambiente aberto ou para outro SMA.

Para o desenvolvimento de SMA, este trabalho está utilizando o *framework* Jason [Bordini et al. 2007], que além de ser interpretado pela linguagem *AgentSpeak* [Rao 1996], possui implementação do modelo BDI [Bratman 1987] para agentes cognitivos e uma interface facilitadora para a programação de SMA. Além disso, o *framework* Jason possui uma arquitetura customizada de agentes chamada ARGO [Pantoja et al. 2016] que permite programar agentes cognitivos com a capacidade de se comunicar e controlar plataformas robóticas. Com a utilização do *framework* Jason em conjunto com agentes ARGO, é possível realizar somente aplicações com SMA fechados e, portanto, caso a plataforma robótica fosse danificada, os conhecimentos dos agentes seriam perdidos. Com isso, em [Pantoja et al. 2018] foi proposta uma arquitetura customizada de agentes com a capacidade de se comunicar com agentes de outros SMA chamada de *Communicator*, que visa permitir a programação de SMA abertos utilizando o *framework* Jason e os agentes ARGO.

Para os agentes *Communicator* se comunicarem com outros SMA, estes SMA abertos utilizam o *middleware* ContextNet criado para IoT. O ContextNet é capaz de oferecer um ambiente para aplicações colaborativas entre dispositivos moveis, permitindo que eles compartilhem informação ao estar conectados em uma rede. Por ser um *middleware* feito para abordagens em *IoT*, o ContextNet permite a conexão de múltiplos nós móveis ao mesmo tempo em uma rede de forma estável [David et al. 2012]. No entanto, a utilização dos agentes *Communicator* nesta conexão possibilita a troca de conhecimento dos agentes de um SMA apenas entre agentes de outros SMA. Mesmo assim, caso a plataforma robótica de um SMA sofresse algum tipo de dano físico, os agentes deste sistema ainda não poderiam se transferir para outro SMA para preservar a integridade de seus conhecimentos.

Com a capacidade do *middleware* ContextNet de prover a troca de informações de forma estável, foi possível também obter um canal de movimentação de agentes. Com isso, tornou-se possível a criação de SMA abertos e, consequentemente, a proposta do protocolo de transferência de agentes. Este protocolo consiste em desenvolver SMA abertos com a capacidade não só de se comunicar com agentes de diferentes SMA, mas também de transferir agentes entre eles, formando assim uma rede de movimentação de agentes e seus conhecimentos fornecida pelo *middleware* ContextNet.

3. O Protocolo de Predatismo

Esta seção tem como objetivo apresentar o protocolo inspirado nos conceitos de predatismo da biologia, mostrando sua ideia, onde e porque é indicado, como funciona e quais são os detalhes da implementação que o compõe. Além disso, será apresentada a prova do conceito utilizando prototipagem em um ambiente real e serão discutidos os pontos fortes e fracos deste trabalho.

Assim como as relações ecológicas são necessárias para garantir o equilíbrio e a evolução dos os seres vivos do ambiente biológico, os SMA abertos também podem se relacionar entre si para garantir a integridade de um sistema. Um ponto ainda não muito explorado neste cenário para garantir tal integridade é a possibilidade de preservar os conhecimentos de um SMA importante que está prestes a ser destruído. Sendo assim, este trabalho se inspira na relação ecológica de predatismo para criar um protocolo que visa explorar a interação entre dois SMA com o intuito de garantir a integridade do sistema.

O protocolo de predatismo envolve dois SMA, onde um deles é o predador e o outro é a presa. O SMA predador é aquele que possui conhecimentos valiosos para atingir os objetivos do grupo, porém, controla um hardware que passou a apresentar inconsistências. Ao ser identificado o problema físico, este SMA realiza a transferência de todos os agentes para outro destino fisicamente semelhante e mais seguro. Já o SMA presa é aquele que ainda não possui conhecimentos significativos sobre as situações que o cerca no ambiente real, mas por controlar um hardware consistente e semelhante ao hardware do predador, ele serve como o destino dos agentes que foram transferidos pelo SMA predador. Após o processo de predatismo, o SMA predador passará a controlar o hardware que pertencia ao SMA da presa e este deixará de existir.

Para que o processo de transferência seja iniciado, é necessário que haja um agente responsável por realizar percepções sobre as condições do ambiente e identificar se as anomalias causam danos ao hardware. Em caso positivo, o agente responsável pela comunicação entre SMA deverá avaliar, de acordo com suas crenças sobre a relação de seu SMA com os demais da rede, se o plano de transferência pode ser iniciado. No momento em que a transferência começa, é feita a clonagem de todos os agentes do sistema e suas características, como crenças iniciais e adquiridas, planos e os objetivos. Após isso, o SMA de origem localiza o SMA de destino, e este recebe os clones criados na origem. Quando os agentes clonados chegarem no SMA de destino, haverá uma tentativa de iniciá-los e depois uma mensagem será enviada para o SMA de origem informando sobre a recepção destes agentes. Caso os agentes clonados do SMA de origem tenham sido inicializados no SMA de destino, os agentes originais do SMA de origem e aqueles que já pertenciam ao SMA de destino antes da interação serão finalizados e apagados. Ao apagar os agentes originais do SMA de origem, garante-se que de fato ocorreu uma transferência de agentes, e não apenas uma cópia; além disso, ao apagar os agentes originais do SMA de destino, garante-se que não haverá interferência nos planos do predador, garantindo que este predador de fato possua controle sobre o hardware. Por outro lado, os agentes clonados serão deletados do SMA de destino caso não tenham sido inicializados corretamente. Essa sequência de operações da interação, que está representada na Figura 1 através do diagrama de sequência da AUML, pode ocorrer quantas vezes forem necessárias enquanto o problema de *hardware* não interferir nas atividades do SMA.

Este protocolo foi feito para ser aplicado nos problemas que demandam o uso de mais de um SMA trabalhando em conjunto onde o foco da aplicação preza o conhecimento, e com isso, as interações entre SMA colaboram para que os objetivos da implementação não sejam comprometidos. Além disso, o fato do protocolo predador permitir que um SMA se transfira para outro destino cria uma oportunidade para este SMA de obter conhecimentos mais variados, dependendo do ambiente externo que o SMA da presa se encontrava.

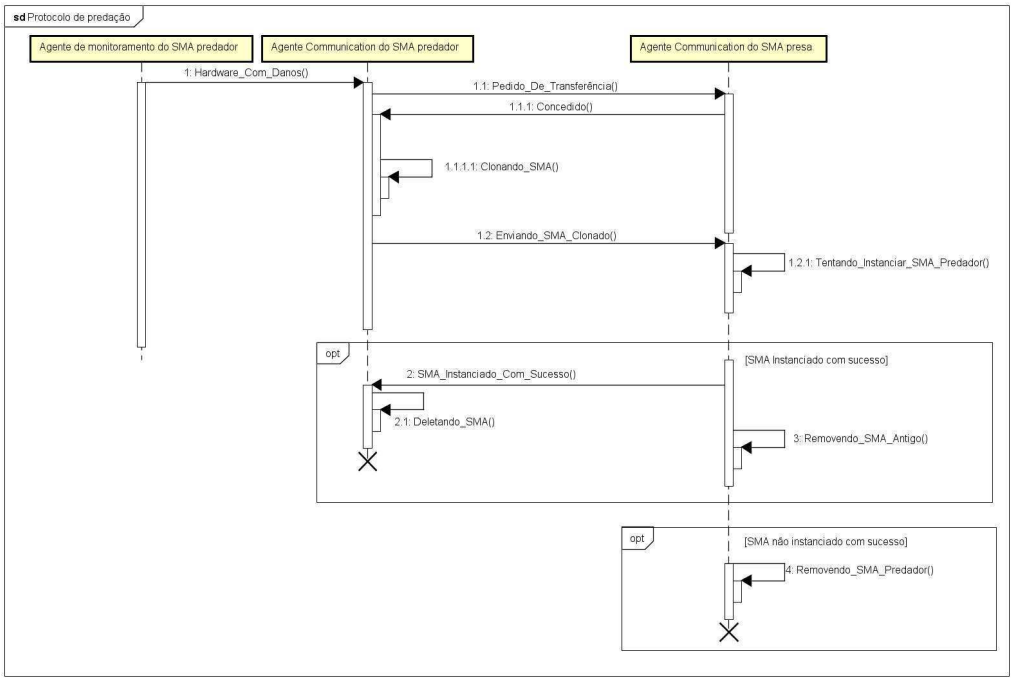


Figura 1. Diagrama de sequência da AUML do processo de interação do protocolo de predatismo entre o predador e a presa.

3.1. A implementação do Protocolo de Predatismo

O protocolo de transferência proposto está embasado na relação de predatismo que ocorre em um determinado ecossistema. Para fornecer este ecossistema, é preciso criar uma infraestrutura que permita que agentes de SMA abertos se comuniquem entre si. Para implementar essa infraestrutura será utilizado a *middleware* ContextNet, que oferece conexão a nível de servidor utilizando uma *middleware* para camada de distribuição de dados escalável (SDDL) [David et al. 2012] e que utiliza o padrão de assinante para para endereçar aplicativos em tempo real e sistemas embarcados estendendo o serviço de distribuição de dados (DDS) padrão do OMG [Pardo-Castellote 2003]. Com isso, o ContextNet permite conectar múltiplos dispositivos tratando os problemas de reconexão e escalabilidade.

Para implementar um SMA que seja capaz de trocar informações nessa infraestrutura de comunicação, é necessário desenvolver um tipo de agente customizado para ter condições de executar esta tarefa. Para criar SMA com essa característica, será utilizado o *framework* Jason que, além de ser uma plataforma que já é bem explorada, também oferece arquiteturas customizáveis para exploração.

O agente responsável por realizar trocas de mensagem entre os SMA através da arquitetura de rede fornecida pelo ContextNet é o agente *Communicator*. Ao entrar nessa rede, o agente *Communicator* assume o papel de cliente, além de possuir uma identificação única que o distingue de outras entidades dentro da mesma rede. Através deste tipo de agente, será possível customizar um padrão de troca de mensagens para que possuam como conteúdo as informações de todos os agentes de um SMA e assim, possibilitar a transferência de agentes de um SMA para outro. Ao embutir o protocolo de transferência na implementação do agente *Communicator*, o SMA ganha autonomia para fazer transferência entre SMA sempre que um plano é executado.

Para adaptar o protocolo de predatismo no agente *Communicator* é necessário alterar o seu ciclo de raciocínio. Inicialmente, o ciclo de raciocínio do agente tradicional era capaz de receber percepções e mensagens vindas de outros agentes do SMA, que eram passadas para a base de crenças do agente e eram mapeados como um evento externo. Cada evento adicionado neste mapa dispara um ação de verificação dos planos da biblioteca de planos. Caso haja um plano a ser executado após a chegada do novo evento externo, ele será designado como uma intenção e com isso, o agente executa o plano, que pode desencadear em uma ação ou uma mensagem a outro agente do SMA. O agente *Communicator* foi resultado da customização deste ciclo de raciocínio, que passou a poder receber e enviar mensagens vindas de outro agente aberto que esteja dentro da rede ContextNet. Para implementar o protocolo de predatismo, o agente *Communicator* foi customizado para realizar três operações que irão variar de acordo com o papel do agente *Communicator* na relação e de acordo com o momento em que cada ação precisa ser tomada. A primeira operação é realizada no agente *Communicator* do SMA predador, que, após fazer a solicitação para efetuar a transferência e ter essa permissão concedida, realiza a clonagem de todo o SMA e este é enviado para o SMA presa. A segunda operação é realizada no agente *Communicator* do SMA presa, que é a substituição de todo SMA da presa pelo SMA do predador. A terceira operação ocorre no agente *Communicator* do SMA predador, que é a remoção do SMA após ter recebido a mensagem que indique que o SMA já foi criado com sucesso no destino.

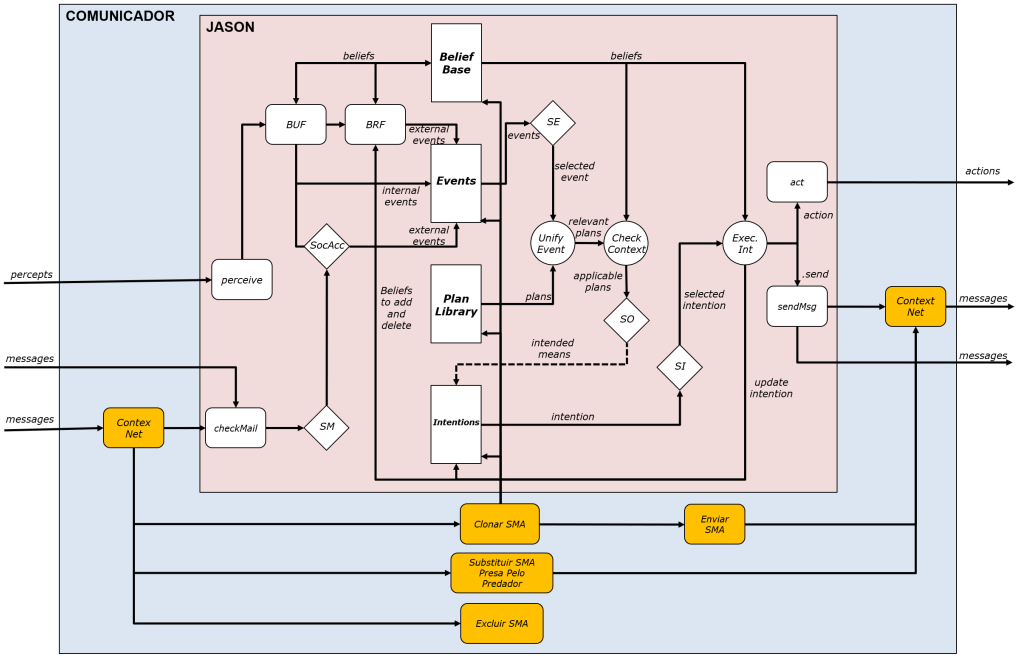


Figura 2. Ciclo de raciocínio do agente *Communicator* com o protocolo de predatismo.

3.2. Prova de Conceito

Como prova de conceito, foram criados dois protótipos de veículos terrestres utilizando plataformas robóticas e cada um deles tem embarcado um SMA programado utilizando o protocolo proposto nesse trabalho. Esses SMA são programados de forma diferente —

para que um contenha mais informações do que o outro — a fim de que seja feita a devida transferência de conhecimento no momento da ativação do protocolo. Os veículos possuem hardware idênticos: são compostos por três sensores: de temperatura, luminosidade e um de distância; dois atuadores: os motores das duas rodas traseiras; um controlador *Arduino*; e uma *Raspberry* onde fica situado o SMA embarcado. Na Figura 3 a seguir é possível observar os protótipos dos veículos terrestres.

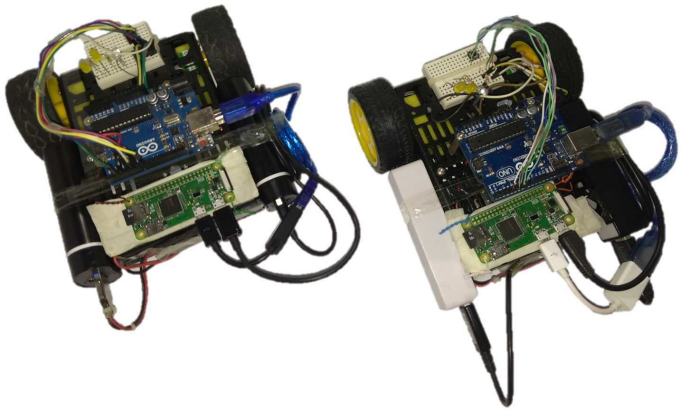


Figura 3. Protótipo dos Veículos Terrestres.

Os sensores dos protótipos denotam a forma de percepção do ambiente para o SMA. Através destas percepções, os agentes do SMA são capazes de identificar riscos de danos físicos para a plataforma robótica. Para isso, a temperatura ambiente de um dos protótipos foi elevada para que o SMA interprete que a plataforma robótica está tendo superaquecimento e ativar o protocolo de transferência de agentes. Na figura 4 é apresentado o código fonte do agente remetente que é responsável por ativar o protocolo de transferência de agentes, porém, para a ativação, este agente necessita receber a crença de que o veículo está superaquecendo do agente que monitora o ambiente.

```
!start.

+!start : true <-
    .print("Sou o remetente e vou enviar uma mensagem ao destinatário.");
    .sendOut("788b2b22-baa6-4c61-b1bb-01cff1f5f878", tell, oi).

+!ola : true <-
    .print("Sou o remetente e recebi uma resposta do destinatário").

+superAquecimento <-
    .moveOut("788b2b22-baa6-4c61-b1bb-01cff1f5f878", predator).
```

Figura 4. Código fonte do agente remetente.

Esse teste tem o objetivo de verificar o grau de confiabilidade do processo de transferência envolvido na ativação do protocolo e na verificação da integridade dos conhecimentos dos agentes após o fim do processo. A aferição da velocidade com que a transferência dos agentes também é realizada. Para isso, os testes foram realizados alterando a quantidade de agentes do SMA predador. Com isso, o SMA predador que está embarcado na plataforma robótica chegou a possuir 10, 30, 50 e 100 agentes.

Com isso, foram realizados 10 repetições para cada uma das variações de quantidade de agentes — somando no total 40 testes — e concluiu-se que o protocolo foi ativado

e finalizado corretamente em 100% dos casos, garantindo a preservação e a integridade dos conhecimentos do SMA predador. Essa verificação foi constatada pois o veiculo de origem era capaz de perceber o ambiente de forma simultânea enquanto que o veiculo que foi predado não possui essa habilidade. Ao fim do processo, o hardware do veiculo de origem ficou inativo e o veiculo predado começou realizar as funções que inicialmente não faziam parte de sua programação.

3.3. Discussões

Os benefícios do uso dos protocolos de predatismo giram em torno da preservação do conhecimento dos agentes, da possibilidade de transportar agentes para outros SMA e possibilitar que os agentes adquiram e transmitam novos conhecimentos por meio de interações com agentes de outros SMA. Além disso, a implementação do protocolo de predatismo é bastante abrangente, levando em consideração aplicações de SMA em ambientes físicos ou simulados, lidando com questões de segurança do SMA e agentes maliciosos através da implementação de conceitos de relações ecológicas.

Levando em consideração a segurança do SMA, os protocolos de predatismo lida com questões de segurança através da aplicação de conceitos de relações ecológicas em todos os casos de transporte de agentes. Além disso, para que a transferência de agentes seja concluída, o SMA de destino deve garantir que a transferência seja realizada somente com o seu consentimento. Já levando em conta agentes maliciosos, as implementações de segurança precisam ser aprimoradas com um critério mais rigoroso de aceitação da transferência, uma vez que o predatismo substitui completamente o SMA presa pelo SMA predador e a falha nessa segurança pode por em risco todo o sistema onde o SMA interage.

Outra questão relacionada à implementação do protocolo de predatismo gira em torno do agente *Communicator*. A responsabilidade deste agente de transferir SMA poderia ter sido atribuída a outra entidade não agentificada. Porém, a vantagem de utilizar o agente *Communicator* para este fim está na autonomia que o SMA ganha para executar este protocolo. Por outro, essa autonomia permitirá ao agente, por exemplo, decidir por deixar de se relacionar com o SMA danificado para reiniciar uma outra relação com um sistema íntegro. Neste caso, a implementação do protocolo não deve permitir que este evento interrompa uma relação caso ela já tenha sido iniciada; caso contrário, as intenções do agente *Communicator* devem ser executadas normalmente.

Ainda sobre agentes *Communicator*, quando o SMA predador domina a SMA presa, o agente *Communicator* do SMA predador é clonado e transferido junto com os demais agentes e em seguida, o SMA de origem é apagado e o antigo SMA da presa também é. No entanto, caso seja necessário reativar o protocolo de predatismo, o agente *Communicator* tentaria novamente dominar a presa que foi inicialmente definida para este agente *Communicator*, porém este agente já não existe mais e um erro ocorreria. Portanto, antes que o *Communicator* do SMA predador seja instanciado e o agente comunicador da SMA presa seja excluído, o agente *Communicator* do SMA predador deve remover a referência que identifica o SMA presa que deixará de existir nas redes de SMA e adicionar outra referência com a ajuda do próprio agente *Communicator* do SMA presa.

Considerando os agentes ARGO no protocolo de predatismo, há a necessidade de atualizar a interface de comunicação entre o agente ARGO e o controlador do *hardware* onde o SMA presa está embarcado, porque os agentes ARGO precisam conhecer o iden-

tificador da porta de comunicação à qual o controlador está conectado. Para resolver esse problema, antes que os agentes ARGO do SMA predador sejam inicializados, é necessário realizar a atualização dos seus identificadores de porta com a ajuda dos agentes ARGO do SMA de presa antes que eles sejam excluídos.

4. Trabalhos Relacionados

Alguns trabalhos, como [Ferri et al. 2006], já utilizam as técnicas bio-inspiradas em Sistemas Multi-Agentes para tarefas de localização em mundo real, nas quais é necessário acessar o *hardware* implementado. Além disso, novas técnicas e metodologias surgiram, tais como [Paes et al. 2005, Zeghida et al. 2018], e, tendo elas ou não inspiração em biologia, todas visam melhorar a forma de trabalhar com Sistemas Multi-Agente abertos e agentes móveis. Consequentemente, trabalhos como [Zedadra et al. 2016] aparecem como uma tentativa de assegurar que diferentes agentes interajam e cooperem entre si em um dado ambiente simulado.

Em [Ferri et al. 2006], é apresentado um algoritmo inspirado em biologia, para um sistema multi-agente robótico cooperativo no qual existe um computador central que localiza todos os robôs e armazena os dados coletados. Sua aplicação é focada na identificação de fontes de gás em um ambiente real, onde cada robô possui sensores e atuadores que servem como uma interface interativa entre este ambiente e o Sistema Multi-Agente. Além disso, o algoritmo de cooperação do robô é biologicamente inspirado no comportamento *Bombox mori* e é dividido em duas fases: pesquisa individual de gás de alta concentração feita por cada robô de forma independente e uma pesquisa cooperativa que envolve todo o enxame de robôs. No entanto, implementações em ambientes reais estão sujeitas a vários comportamentos que podem ocorrer e que podem colocar em risco a integridade do sistema. Se um robô fosse danificado, a eficiência do sistema ficaria comprometida. Na pior das hipóteses, se o computador central estiver danificado, todo o sistema parará de funcionar.

Em [Zedadra et al. 2016], o assunto discutido é a robótica cooperativa e a exploração de recursos naturais, que é uma tarefa complexa, dividida em pequenas subaéreas e caracteriza o ato de busca e captura de alimento em um local de armazenamento particular. Este trabalho apresenta um algoritmo de exploração de recursos naturais bio-inspirado envolvendo agentes simples que são capazes de realizar tarefas complexas em grupo.

Esses trabalhos reforçam a importância de explorar sistemas abertos como uma alternativa para cooperar e interligar diferentes sistemas de um ambiente. A necessidade de adotar os paradigmas biológicos nas aplicações de SMA também pode ser vista em [Zeghida et al. 2018], pois, apesar dos desafios relacionados à sobrevivência dos seres, os organismos biológicos têm capacidade de evolução, autocorreção e controle. No entanto, as inspirações biológicas estudadas no SMA estão limitadas à cooperação e interação entre estes SMA por meio de troca de mensagens. Além disso, de todas as técnicas que foram estudadas, uma maneira de preservar a integridade dos sistemas em situações críticas ainda não foi explorada. Os trabalhos relacionados mostram que as técnicas bio-inspiradas auxiliam na implementação de sistemas abertos para aumentar a eficiência de algoritmos colaborativos. Porém, as técnicas bio-inspiradas também podem auxiliar na formação de um sistema que preserve a existência de um SMA assim como a biologia é capaz de fazer com os seres vivos dentro do ecossistema.

Em [Jesus et al. 2018], foram apresentadas ideias de protocolos inspirados em relações ecológicas da biologia, como o predatismo, mutualismo e inquilinismo. Porém, este trabalho relacionado apresentou brevemente as definições e o funcionamento dos protocolos. Sendo assim, este trabalho focará em explorar as ideias, os detalhes de implementação e as aplicações do protocolo de predatismo. A escolha por este protocolo foi feita devido ao fato de que as implementações que devem ser feitas para o seu funcionamento podem ser reaproveitadas, com algumas adaptações, para os demais protocolos.

5. Considerações Finais

Este trabalho apresentou um protocolo de interação entre SMA embarcados inspirado na relação ecológica de predação da biologia, mostrando sua ideia com o objetivo de preservação de conhecimento e principalmente, sua implementação, onde foi reaproveitada a arquitetura do agente *Communicator* para customizar o seu ciclo de raciocínio e assim, permitir transportar todos os agentes de um SMA predador para a presa. Como prova do conceito, dois protótipos de veículos terrestres foram criados utilizando plataformas robóticas que possuem, cada uma, um SMA embarcado. Na prova do conceito, os dois protótipos possuíam o *hardware* semelhante, porém, um SMA possuía mais conhecimento do que outro. Com este cenário, foi possível mostrar a situação em que o hardware de um veículo era danificado e então, este aplicava o protocolo de predação no SMA que estava embarcado no veículo com o hardware que apresentava boas condições.

Como trabalhos futuros serão tratados os problemas que envolvem a conversão de portas de comunicação com a parte física onde o SMA está embarcado, garantindo que o novo sistema conseguirá se instalar por completo no novo *hardware* de forma automática. Também será implementado o mecanismo que reajusta a lista de SMA disponíveis para predação sempre que o protocolo for utilizado dentro da rede do ContextNet, evitando assim, erros de transferência. Além disso, é visado o ajuste do mecanismo de comunicação para reforçar a segurança quanto aos agentes suspeitos durante a transferência do SMA. Com isso, o processo de transferência passará a ser mais rigoroso, porém, a predação ocorrerá de forma segura e sem prejudicar na conclusão dos objetivos do sistema.

Referências

Begon, M., Townsend, C. R., and Harper, J. L. (2005). *Ecology: from individuals to ecosystems*, volume 51. Freshwater Biology - FRESHWATER BIOL.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons Ltd.

Bratman, M. E. (1987). *Intention, Plans and Practical Reasoning*. Cambridge Press.

Chen, B., Cheng, H. H., and Palen, J. (2009). Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems. *Transportation Research Part C: Emerging Technologies*, 17(1):1–10.

David, L., Vasconcelos, R., Alves, L., André, R., Baptista, G., and Endler, M. (2012). A communication middleware for scalable real-time mobile collaboration. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on*, pages 54–59. IEEE.

Endler, M., Baptista, G., Silva, L., Vasconcelos, R., Malcher, M., Pantoja, V., Pinheiro, V., and Viterbo, J. (2011). Contextnet: context reasoning and sharing middleware for large-scale pervasive collaboration and social networking. In *Proceedings of the Workshop on Posters and Demos Track*, page 2. ACM.

Ferri, G., Caselli, E., Mattoli, V., Mondini, A., Mazzolai, B., and Dario, P. (2006). A biologically-inspired algorithm implemented on a new highly flexible multi-agent platform for gas source localization. In *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*, pages 573–578. IEEE.

Günay, A., Winikoff, M., and Yolum, P. (2015). Dynamically generated commitment protocols in open systems. *Autonomous Agents and Multi-Agent Systems*, 29(2):192–229.

Huynh, T. D., Jennings, N. R., and Shadbolt, N. R. (2006). An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 13(2):119–154.

Jesus, V. S., Manoel, F. C. P. B., Pantoja, C. E., and Viterbo, J. (2018). Transporte de agentes cognitivos entre sma distintos inspirado nos princıpios de relaç oes ecológicas. *Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCações — XII WESAAC*, pages 179–187.

Paes, R., Carvalho, G. d., Lucena, C. d., Alencar, P., Almeida, H. d., and Silva, V. d. (2005). Specifying laws in open multi-agent systems. *Agents, Norms and Institutions for Regulated Multi-agent Systems (ANIREM), AAMAS2005*.

Pantoja, C. E., Jesus, V. S., Manoel, F. C. P. B., and Viterbo, J. (2018). A heterogeneous architecture for integrating multi-agent systems in ami systems. *The Thirtieth International Conference on Software Engineering and Knowledge Engineering (SEKE 2018)*.

Pantoja, C. E., Stabile Jr, M. F., Lazarin, N. M., and Sichman, J. S. (2016). Argo: A customized jason architecture for programming embedded robotic agents. *Fourth International Workshop on Engineering Multi-Agent Systems (EMAS 2016)*.

Pardo-Castellote, G. (2003). Omg data-distribution service: Architectural overview. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 200–206. IEEE.

Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In de Velde, W. V. and Perram, J. W., editors, *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world (MAAMAW'96)*, volume 1038 of *Lecture Notes in Artificial Intelligence*, pages 42–55, USA. Springer-Verlag.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Zedadra, O., Seridi, H., Jouandeau, N., and Fortino, G. (2016). A cooperative switching algorithm for multi-agent foraging. *Engineering Applications of Artificial Intelligence*, 50:302–319.

Zeghida, D., Meslati, D., and Bounour, N. (2018). Bio-ir-m: A multi-paradigm modelling for bio-inspired multi-agent systems. *Informatica*, 42(3).

Modelagem Baseada em Agentes para Análise de Recursos Hídricos

Giovani P. Farias[1], Bruna S. Leitzke[1], Míriam B. Born[2], Marilton S. de Aguiar[2], Diana F. Adamatti[1]

[1]Programa de Pós-Graduação em Modelagem Computacional (PPGMC)
Universidade Federal do Rio Grande (FURG) – Rio Grande/RS – Brasil

[2]Programa de Pós-Graduação em Computação (PPGC)
Universidade Federal de Pelotas (UFPel) – Pelotas/RS – Brasil

brunaleitzke@hotmail.com

{dianaada,giovanifarias}@gmail.com

{marilton,mbborn}@inf.ufpel.edu.br

Abstract. *The paper aims to present a river basin modelling using GAMA platform for water resources analysis. Currently, several models based on multiagent systems (MAS) are used for natural resources management and they present satisfactory results for this type of scenario. GAMA is agents based and widely used in this context with several studies already published. In this study, the São Gonçalo and Lagoa Mirim basins were considered from georeferenced data. In the modelling, regions and rivers are agents on the system where rivers water can flow among neighbours regions.*

Resumo. *O artigo tem por objetivo apresentar a modelagem de uma bacia hidrográfica para a análise do uso dos recursos hídricos utilizando a plataforma GAMA. Na atualidade, diversos modelos baseados em sistemas multiagente (SMA) são utilizados para a gestão de recursos naturais e apresentam resultado satisfatório neste tipo de cenário. A ferramenta GAMA é baseada em agentes e amplamente utilizada neste contexto, com diversos estudos já publicados. Neste estudo, considerou-se a bacia hidrográfica São Gonçalo e Lagoa Mirim a partir de dados georreferenciados. Na modelagem, regiões e rios são agentes no sistema, sendo que a água dos rios pode fluir entre regiões vizinhas.*

1. Introdução

Os recursos naturais compreendem os elementos da natureza aos quais os seres humanos utilizam para a sua sobrevivência, sendo estes renováveis ou não renováveis. Com a crescente demanda da população mundial, estes recursos tornam-se escassos em algumas regiões, além disso, o mau uso, a falta de gerenciamento e, às vezes, o compartilhamento dos mesmos acarretam graves conflitos. Os modelos baseados em agentes são ferramentas que surgiram da necessidade de capturar melhor as características de sistemas complexos, em particular, os sistemas ecológicos e sociais, pois abrangem várias áreas de estudo. Deste modo, a partir da simulação multiagente, é possível projetar políticas adequadas para resolver os problemas mencionados [Filatova et al. 2013]. A modelagem multiagente simula sistemas com base na tomada de decisões e ações de atores individuais ou grupos de atores, nas interações entre si e com o ambiente em que estão inseridos.

A gestão de recursos renováveis em sistemas complexos, principalmente o recurso hídrico no contexto da bacia hidrográfica, é um importante meio na busca de possíveis soluções aos problemas enfrentados pelos interessados em um determinado ecossistema [Adamatti 2007]. Neste trabalho, propõe-se como estudo de caso a bacia hidrográfica Mirim-São Gonçalo, localizada no sudeste do Rio Grande do Sul/RS, a qual possui abrangência nas províncias da planície costeira Uruguaio-Sul Riograndense. Os municípios que englobam a bacia são: Arroio Grande, Candiota, Canguçu, Capão do Leão, Chuí, Jaguarão, Pelotas, Rio Grande e Santa Vitória do Palmar, com uma área total de 25.961,04 km^2 [SEMA 2019].

A modelagem da bacia foi realizada na plataforma de código aberto GAMA (*GIS Agent-based Modeling Architecture*), sendo considerado os dados geográficos da região de estudo, onde cada região hidrográfica e seus rios foram considerados agentes no modelo. A simulação possibilita diversas análises do ambiente, tais como: o consumo de água e taxa de produção por região, o volume de água dos rios em cada região e o fluxo de água entre rios de regiões vizinhas.

O artigo encontra-se organizado da seguinte forma. Na Seção 2 são apresentados conceitos básicos sobre Sistemas Multiagente. Na Seção 3 é introduzida a plataforma GAMA com suas características e funcionalidades. Na Seção 4, é apresentado o estudo de caso, bem como, a modelagem de uma bacia hidrográfica com a utilização da plataforma GAMA. As análises das simulações, realizadas no modelo de bacia hidrográfica, estão na Seção 5. Finalmente, a Seção 6, apresenta as conclusões e trabalhos futuros.

2. Sistemas Mutiagente

O uso de Sistemas Mutiagente (SMA) atualmente abrange pesquisas direcionadas a diversos temas sobre o gerenciamento de ecossistemas. Com essa técnica é possível reproduzir o conhecimento e raciocínio de vários agentes heterogêneos que, juntos, precisam resolver problemas comuns de planejamento [Bousquet and Le Page 2004].

Segundo [Coppin 2010], os agentes de um sistema devem cooperar, aprender e agir de forma autônoma. Desta maneira, os classifica em: i) *agentes reativos*, os quais reagem a eventos no ambiente em que estão inseridos de acordo com regras/normas especificadas previamente; ii) *agentes de interface* que têm o intuito de auxiliar o usuário nas diversas aplicações; iii) *agentes de informação* que auxiliam o usuário a encontrar, classificar e filtrar informações provenientes de inúmeras fontes da internet; e, iv) *agentes colaborativos*, os quais cooperam entre si para alcançar objetivos/metas.

Os sistemas multiagente constituem-se de diversos agentes interagindo em um ambiente. Os SMA foram introduzidos na Computação na década de 80, entretanto, somente nos anos 90 tornaram-se populares [Wooldridge 2002]. Cada agente de um sistema possui comportamento individual, porém os mesmos devem ser capazes de interagir com os demais de forma organizada, desta maneira são relevantes características como [Bordini et al. 2001]: cooperação, coordenação, competição e negociação.

Tais características tornam-se importantes pois a maioria dos problemas a serem resolvidos buscam uma maneira distribuída de resolução. Além disso, muitos destes possuem elevada complexidade, tornando impossível encontrar a solução apenas com um agente. Em um SMA, os agentes podem cooperar em busca da solução de um objetivo

geral, sendo que, cada um destes possui seu próprio objetivo, mas que juntos, alcançam uma meta maior [Alvares and Sichman 1997, Bordini et al. 2001].

Os benefícios da utilização de um sistema multiagente são diversos, como: i) rapidez na resolução de problemas visto a inerência do processamento concorrente; ii) aumento da flexibilidade e escalabilidade através da conexão de vários sistemas; iii) aumento da capacidade de resposta a um determinado problema pelo fato de todos os recursos estarem localizados no mesmo ambiente.

No desenvolvimento de SMA, a arquitetura comumente utilizada é a BDI (*beliefs*, *desires* e *intentions*), baseadas em um modelo cognitivo que representam crenças, desejos e intenções [Hübner et al. 2004]. De acordo com [Wooldridge 2002], BDI é estruturada em: i) crenças, representando o que o agente sabe sobre si mesmo, sobre os demais agentes e sobre o ambiente ao qual está inserido; ii) desejos, representando os estados que o agente almeja atingir, geralmente são objetivos; e, iii) intenções, que são representadas pela sequência de ações que um determinado agente executa para alcançar um objetivo.

Para simular sistemas complexos é necessário compreender a dinâmica e o funcionamento característico deste tipo de sistema. Eles podem ser representados por uma grande variedade de interação entre os agentes envolvidos. Algumas ferramentas são específicas para a simulação de SMA. Para simular o estudo deste artigo, voltado para a bacia hidrográfica Mirim-São Gonçalo, foi escolhida a plataforma GAMA [Taillandier et al. 2018].

Outras ferramentas vêm sendo utilizadas para a simulação em diversas aplicações de SMA, os autores [Dos Santos et al. 2016] utilizaram a estrutura JaCaMo [Boissier et al. 2011] (Jason [Bordini et al. 2007], CArtAgO [Ricci et al. 2009] e $\mathcal{M}$OISE [Hübner et al. 2007]), que cobre alguns dos níveis de abstrações necessários para o desenvolvimento de SMA, para analisar um experimento de ecossistema urbano. Utilizando o modelo de simulação do consumo urbano no Netlogo, [Li et al. 2017] analisaram os sistemas de alocação e fluxo de água, energia e alimento. Com o intuito de desenvolver um RPG (*Role-Playing Game*) chamado ReHab, [Page et al. 2016] propuseram uma modelagem baseada em multiagente para harmonização entre regeneração de biomassa e habitat de reprodução de aves migratórias protegidas, onde a implementação foi realizada na plataforma CORMAS [Page et al. 2000].

3. Plataforma GAMA

A plataforma GAMA[1] (*GIS Agent-based Modeling Architecture*) é um ambiente de desenvolvimento integrado completo, que permite alternar de forma rápida e prática entre perspectivas de modelagem e simulação. GAMA é baseada na consolidada IDE (*Integrated Development Environment*) do Eclipse, utilizando os diversos recursos presentes neste ambiente de desenvolvimento, e é desenvolvida por várias equipes da unidade internacional de pesquisa UMMISCO (*Unité de Modélisation Mathématique et Informatique des Systèmes Complexes*) no IRD (*Institut de Recherche pour le Développement*) da UPMC (*Université Pierre et Marie Curie*) como um projeto *open source* desde o ano de 2007.

A plataforma apresenta um editor que visa facilitar o trabalho de modelagem e desenvolvimento do usuário (Figura 1), contendo ferramentas usuais de IDEs, tais como:

[1]https://gama-platform.github.io/

coloração da sintaxe, compilação e preenchimento automáticos e a possibilidade de formatar ou comentar linhas de código específicas. Além disso, a IDE está conectada a uma extensa documentação online, permitindo aos usuários obterem informações sobre as diversas palavras-chave, operadores e declarações disponíveis.

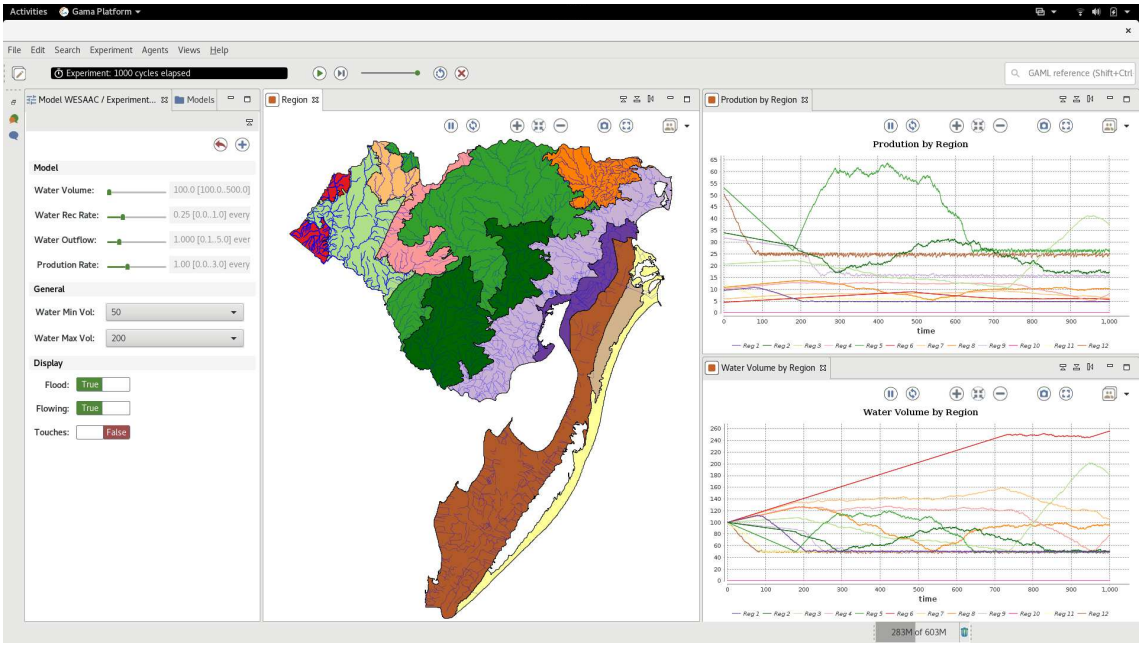


Figura 1. Interface da bacia Mirim-São Gonçalo na plataforma GAMA

O ambiente de desenvolvimento integrado permite formular e construir modelos baseados em agentes, a partir de diferentes conjuntos de dados, possuindo integração com sistemas de informações geográficas (*GIS - Geographic Information System*). Além disso, a flexibilidade de sua interface permite organizar os painéis de visualização com comandos simples de arrastar e soltar ou através de *layouts* predefinidos, apresentando uma ferramenta de inspeção/verificação de agentes, a qual permite obter informações sobre um ou vários agentes (visão tabular), e também possui um mecanismo de busca desenvolvido para a obtenção de informações e exemplos de uso dos diversos operadores presentes na plataforma.

GAMA fornece uma linguagem de modelagem completa GAML (*GAma Modeling Language*) e um ambiente de desenvolvimento integrado que permite formular e construir modelos de forma tão rápida e fácil quanto no NetLogo [Tisue and Wilensky 2004], indo além do que Repast (*Recursive Porous Agent Simulation Toolkit*) [North et al. 2006] ou que Mason (*Multi-Agent Simulator Of Neighborhoods*) [Luke et al. 2005]. Atualmente, GAMA é utilizada em vários modelos, como [Nguyen Vu et al. 2009, Taillandier and Buard 2009] e também em projetos, como sistemas de apoio a decisões ambientais [Chu et al. 2009], projetos urbanos [Amouroux et al. 2009], gerenciamento de recursos hídricos [Thérond et al. 2014], invasões biológicas [Amouroux et al. 2008] e adaptação às mudanças climáticas ou mitigação de desastres [Gaudou et al. 2014].

4. Modelagem do Problema

Uma bacia hidrográfica pode ser considerada como uma parte territorial importante para a hidrologia urbana, pois ela relaciona a geografia natural da região, a água, a civilização e a população. Dessa forma, existe a necessidade de gerir as bacias hidrográficas a partir de questões sociais, econômicas e ambientais. Neste trabalho, apresenta-se a modelagem de uma bacia hidrográfica com a utilização da ferramenta GAMA, que permite lidar facilmente com dados geoespaciais e vetoriais de sistemas de informação geográfica.

O modelo de dados GIS utilizado neste trabalho, representa uma base de dados com informações geográficas, que encontra-se em dois arquivos no formato shapefile, contendo informações geoespaciais da bacia hidrográfica Mirim-São Gonçalo. Esses arquivos, descrevem espacialmente qualidades de vetores (pontos, linhas e polígonos) para representar as regiões e os rios presentes na bacia, sendo que, cada um desses itens possui ainda atributos que o descreve, como nome, código, área ou comprimento.

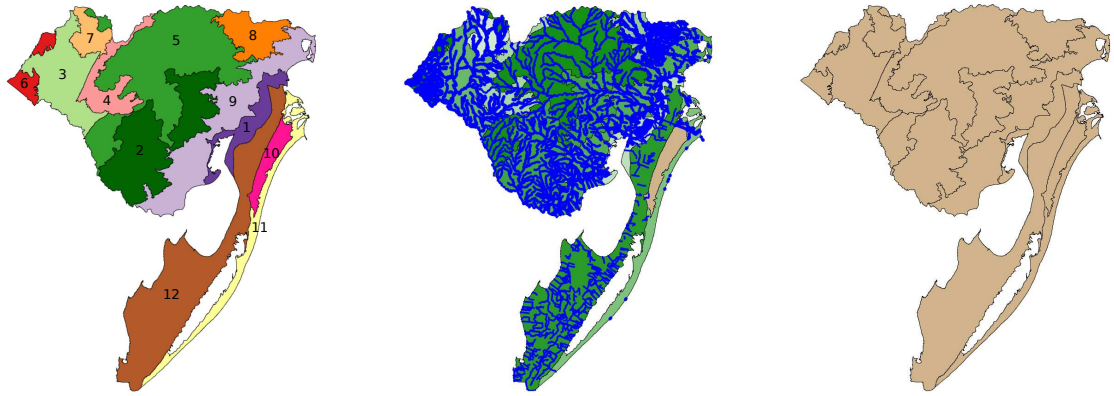
(a) regiões (b) rios (c) regiões e rios

Figura 2. Representação gráfica da bacia hidrográfica Mirim-São Gonçalo

A plataforma GAMA consegue separar esses itens em diferentes camadas temáticas e representá-los de forma independente, permitindo trabalhar com eles de modo rápido e simples. Deste modo, cada região ou rio pode ser considerado um agente específico com suas próprias características e atributos. Na Figura 2a, por exemplo, conseguimos visualizar o formato e localização de cada região, bem como acrescentar o atributo cor a cada uma delas com tonalidade própria. GAMA permite ao usuário relacionar as informações existentes através da posição e topologia dos objetos, gerando assim novas informações. Neste caso, considerando a junção das informações na Figura 2a (regiões) e na Figura 2b (rios), obtemos a representação presente na Figura 2c (regiões e rios), da qual, a plataforma consegue extrair novas informações e atributos para cada objeto, como, por exemplo, associar um rio a uma determinada região de acordo com sua localização.

O modelo hidrográfico, neste caso, é composto por 12 regiões (Figura 3a), com características distintas, conforme apresentado na Tabela 1. Cada região possui valores específicos para taxa de produção (*prd*) de bens/serviços e consumo de água (*cons*), os quais, estão relacionados com o tamanho da sua área (*area*), ou seja, regiões com uma área maior apresentam uma taxa de produção e consumo de água maiores. O ambiente também é composto por 2294 rios que estão distribuídos entre as diversas regiões de

acordo com sua posição geoespacial. Todos os rios apresentam os mesmos valores para o volume de água (*vol*) inicial, a taxa de recuperação de água (*rec*) e o fluxo de água (*fluxo*).



(a) regiões numeradas (b) rios alagados (c) rios secos

Figura 3. Representação das regiões, cenário de alagamento e seca extrema

Tabela 1. Atributos básicos de cada região.

região	*area* km^2	*cons*	*prd*	nº rios	vizinhos
1	956,55	0.09	9.5	161	[9, 12]
2	3.402,83	0.34	34.0	358	[5, 9]
3	2.069,33	0.20	20.6	255	[4, 5, 6, 7]
4	1.016,72	0.10	10.1	24	[3, 5, 7]
5	5.306,11	0.53	53.0	216	[2, 3, 4, 7, 8, 9]
6	448,17	0.04	4.4	66	[3]
7	590,02	0.05	5.9	22	[3, 4, 5]
8	1.081,65	0.10	10.8	213	[5, 9]
9	3.178,26	0.31	31.7	503	[1, 2, 5, 8]
10	564,70	0.05	0.0	0	[11, 12]
11	1.199,01	0.11	11.9	24	[10, 12]
12	5.047,57	0.50	50.4	452	[1, 10, 11]

O volume de água dos rios diminui com o passar do tempo de acordo com a taxa de consumo de água da região na qual o rio está inserido. Os rios recuperam parte do seu volume de água de acordo com a taxa de recuperação de água, que é um valor global do ambiente igual para todos os rios. Cada rio pertence a uma única região e todos os rios de uma mesma região possuem o mesmo volume de água. Se o consumo de água da região é menor que a taxa de recuperação de água, a tendência é que o volume dos rios cresçam no decorrer do tempo e a região fique alagada (Figura 3b), caso contrário, quando o consumo de água da região é maior que a taxa de recuperação de água, a tendência é que o volume dos rios cheguem a zero, isto é, os rios secam (Figura 3c), tornando a região totalmente improdutiva até o final da simulação. Regiões vizinhas podem compartilhar água de seus rios quando o valor do fluxo de água for maior que zero. É importante observar que uma região só pode obter água quando atingir um volume mínimo (*min_vol*) estabelecido no ambiente, da mesma forma, uma região só pode "ceder" água a um vizinho quando seu volume de água for maior que este volume mínimo. O ambiente também estabelece um valor de volume máximo de água (*max_vol*), neste caso, quanto mais o volume de água da região ultrapassar este valor (alagamento), menor será sua taxa de produção.

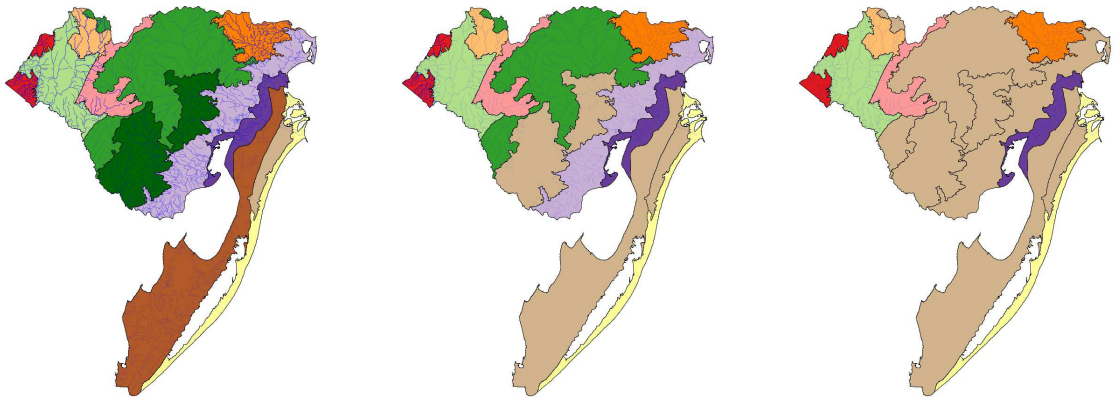
5. Simulações e Análises

A interface de simulação do modelo hidrográfico da bacia Mirim-São Gonçalo (Figura 1), desenvolvido na plataforma GAMA, possibilita a representação gráfica (mapa), da variação dos volumes dos rios e alteração das cores das regiões que secam, bem como, a visualização em gráfico de linhas da variação da taxa de produção e do volume de água em cada região. Além disso, permite atribuir diferentes valores aos diversos parâmetros de configuração do ambiente.

Neste trabalho, são apresentados resultados e análises de duas simulações com os seguintes valores globais: volume de água inicial vol = 100; taxa de recuperação da água rec = 0.15; volume mínimo de água min_vol = 25; volume máximo de água max_vol = 200. A única diferença entre as duas simulações é que na primeira simulação (Seção 5.1) o valor do fluxo de água é 1.5 ($fluxo > 0$), ou seja, é permitido o compartilhamento de água entre regiões vizinhas. Ao contrário, na segunda simulação (Seção 5.2) o valor do fluxo de água é zero ($fluxo = 0$), ou seja, não há compartilhamento de água entre as regiões. Os demais valores específicos para cada região estão representados na Tabela 1.

5.1. Simulação com partilha de água entre regiões vizinhas

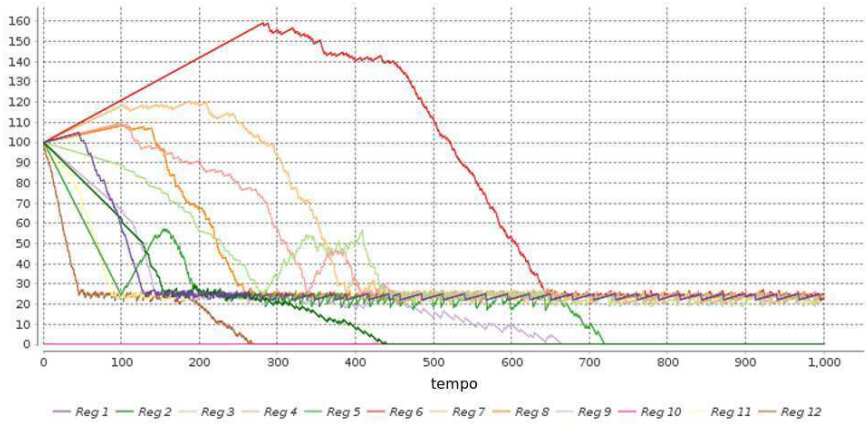
Na simulação com partilha de água entre regiões, podemos observar que regiões com áreas maiores, quando atingem o volume mínimo de água, passam a "consumir" água dos vizinhos (Figura 5a), pois podemos observar que regiões com áreas menores têm volume de água crescente no começo da simulação, o qual começa diminuir a medida que regiões maiores atingem o volume mínimo de água (min_vol = 25). Neste caso, regiões com consumo de água maiores que 0.15 ($cons > 0.15$) tendem a secar de forma gradual no decorrer do tempo (visualmente, ficam na cor cinza), conforme apresentado na Figura 4.



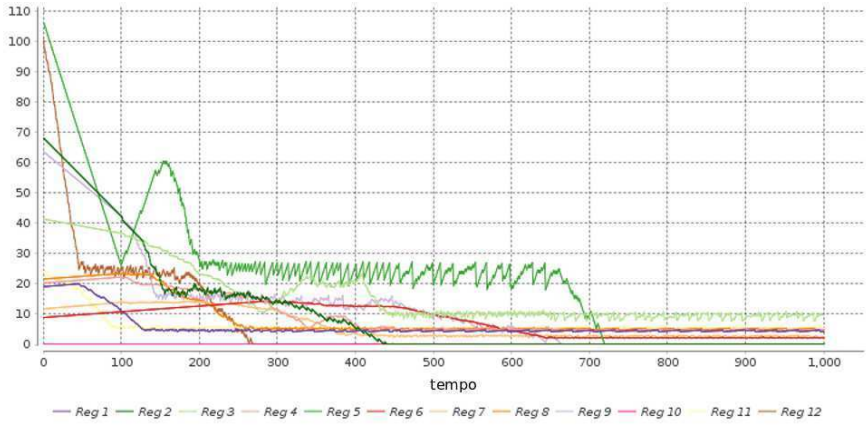
(a) 100 (b) 500 (c) 1000

Figura 4. Alterações no ambiente no decorrer do tempo (com partilha de água)

Nesta simulação também podemos observar que: i) a região 10, mesmo possuindo $cons < 0.15$, acaba secando, pois ela não possui nenhum rio; ii) a região 3, apesar de possuir $cons > 0.15$, não seca, pois está cercada de vizinhos aptos a partilhar água; iii) após o passo 700, tanto o consumo de água (Figura 5a) quanto a produção (Figura 5b) por região, se estabilizam.



(a) volume de água por região



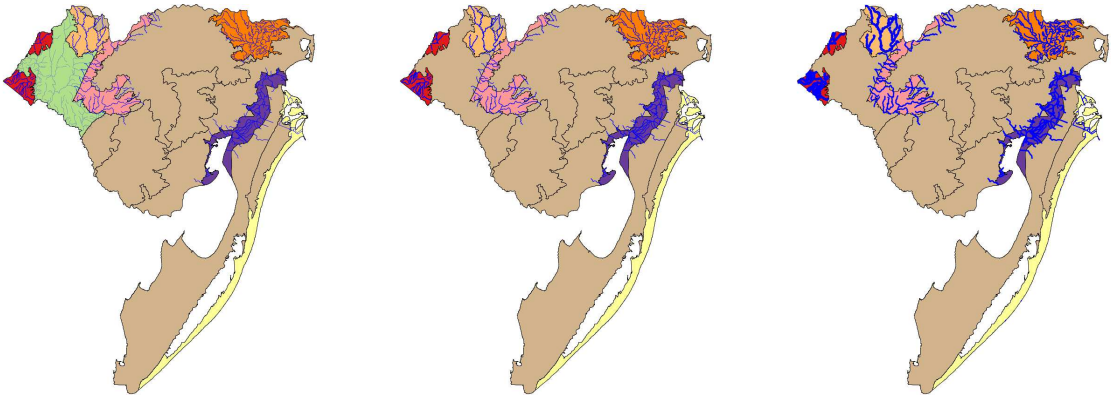
(b) produção por região

Figura 5. Variação do volume de água e produção (com partilha de água)

5.2. Simulação sem partilha de água entre regiões vizinhas

Na simulação sem partilha de água, podemos observar que regiões com áreas maiores, quando atingem o volume mínimo de água, não "consomem" água dos vizinhos, permanecendo o comportamento de queda linear do volume de água (Figura 7a) das regiões com consumo de água maiores que 0.15 ($cons > 0.15$), as quais, tendem a secar de forma mais abrupta no decorrer do tempo, conforme apresentado na Figura 6. Regiões com consumo de água menor que 0.15 ($cons < 0.15$) apresentam um crescimento linear do volume de água durante toda simulação, porém este crescimento linear não se reflete em um crescimento contínuo da produção (Figura 7b).

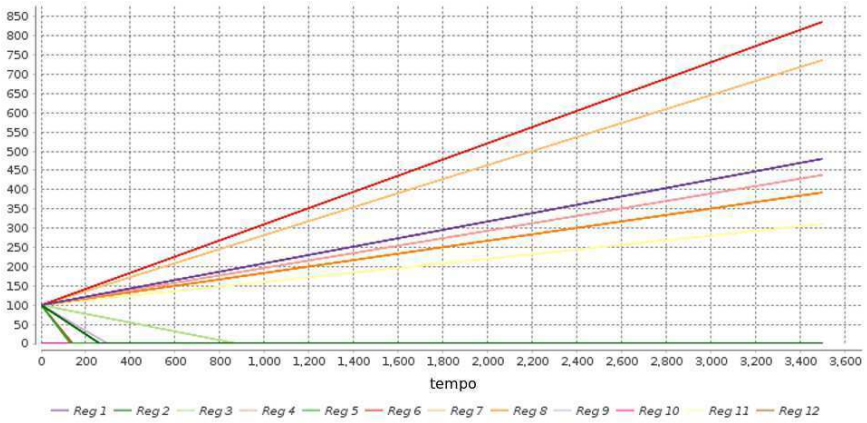
Podemos observar que a produção das regiões com maiores áreas diminui conforme a queda no volume de água, até atingir o valor zero, caso este, onde os rios secaram e a região tornou-se totalmente improdutiva. As regiões com menores áreas apresentam um crescimento linear do volume de água e de produção até atingirem o volume máximo de água $max\_vol = 200$, a partir deste ponto a produtividade diminui porque a região começa a sofrer com alagamento cada vez maior no decorrer do tempo, conforme podemos observar nas Figuras 6b e 6c.



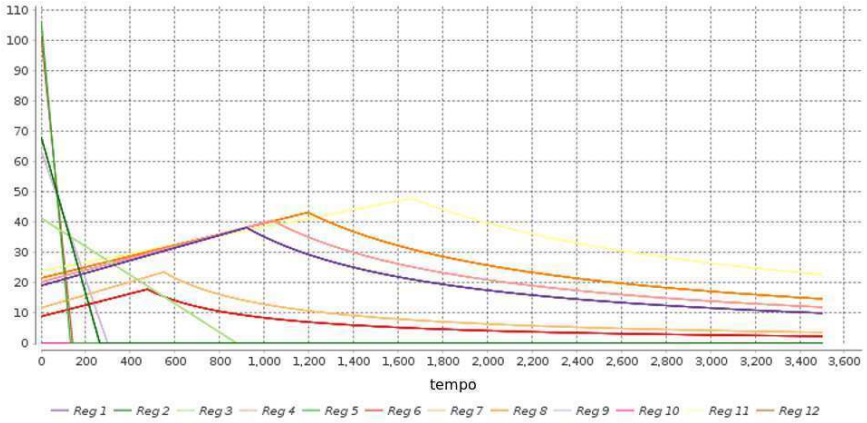
(a) 500 (b) 1000 (c) 3500

Figura 6. Alterações no ambiente no decorrer do tempo (sem partilha de água)

Nesta simulação também podemos observar que: i) todas as regiões com $cons < 0.15$ acabam sofrendo com alagamento, exceto a região 10, que acaba secando por não possuir rios; ii) a região 3, apesar de possuir vizinhos aptos a partilhar água, ao contrário do que ocorre na simulação com partilha de água, acaba secando e se tornando improdutiva; iii) a produtividade das regiões está diretamente relacionada com o volume de água, que não deve ser muito alto (alagamento) nem muito baixo (seca).



(a) volume de água por região



(b) produção por região

Figura 7. Variação do volume de água e produção (sem partilha de água)

6. Conclusões e Trabalhos Futuros

Com o uso da ferramenta GAMA, a gestão participativa pode ser simulada sem gerar consequências efetivas. Dessa forma, para a modelagem deste trabalho, foi possível simular algumas situações entre as regiões hidrográficas determinadas. Assim, pode-se concluir que cada agente interfere de forma significativa no ambiente dos demais. Quando a tomada de decisão é realizada de forma participativa, as partes envolvidas tendem a melhorar suas condições ou pelo menos permanecer estáveis diante de possíveis problemas. Já a situação inversa apresenta situações preocupantes, pois neste caso, quando os agentes não compartilhavam seus recursos, a maioria deles obteve grande perda ou perda total da sua produção.

Neste trabalho foi apresentada uma abordagem inicial da simulação de uma bacia hidrográfica, mas desconsiderando seu real comportamento com relação ao fluxo de rios entre as regiões. Para trabalhos futuros, um dos objetivos principais será acrescentar uma modelagem matemática de transporte de água nessa bacia. Sendo assim, podem ser abordados, por exemplo, assuntos como qualidade de água, distribuição real de água entre as regiões de estudo e análise da poluição e seu impacto ambiental, como tratado em [Yu et al. 2016].

Agradecimentos

Os autores deste artigo agradecem ao Programa de apoio ao Ensino e à Pesquisa Científica e Tecnológica em Regulação e Gestão de Recursos Hídricos – Pró-Recursos Hídricos Chamada N°16/2017, pelo auxílio financeiro no desenvolvimento desta pesquisa.

Referências

Adamatti, D. F. (2007). *Inserção de jogadores virtuais em jogos de papéis para uso em sistemas de apoio à decisão em grupo: um experimento no domínio da gestão de recursos naturais*. PhD thesis, Escola Politécnica – Universidade de São Paulo, São Paulo, Brasil. doi:10.11606/T.3.2007.tde-07012008-154915.

Alvares, L. O. and Sichman, J. S. (1997). Introduçao aos sistemas multiagentes. In *XVII Congresso da SBC-Anais JAI'97*.

Amouroux, E., Chu, T.-Q., Boucher, A., and Drogoul, A. (2009). GAMA: An environment for implementing and running spatially explicit multi-agent simulations. In Ghose, A., Governatori, G., and Sadananda, R., editors, *Agent Computing and Multi-Agent Systems*, pages 359–371, Berlin, Heidelberg. Springer Berlin Heidelberg.

Amouroux, E., Desvaux, S., and Drogoul, A. (2008). Towards virtual epidemiology: An agent-based approach to the modeling of h5n1 propagation and persistence in north-vietnam. In Bui, T. D., Ho, T. V., and Ha, Q. T., editors, *Intelligent Agents and Multi-Agent Systems*, pages 26–33, Berlin, Heidelberg. Springer Berlin Heidelberg.

Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., and Santi, A. (2011). Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons.

Bordini, R. H., Vieira, R., and Moreira, A. F. (2001). Fundamentos de sistemas multiagentes. In *Anais do XXI Congresso da Sociedade Brasileira de Computação (SBC2001)*, volume 2, pages 3–41.

Bousquet, F. and Le Page, C. (2004). Multi-agent simulations and ecosystem management: a review. *Ecological modelling*, 176(3-4):313–332.

Chu, T.-Q., Drogoul, A., Boucher, A., and Zucker, J.-D. (2009). Interactive learning of independent experts' criteria for rescue simulations. *J. UCS*, 15:2701–2725.

Coppin, B. (2010). *Inteligência Artificial*. Rio de Janeiro: LTC, 3a edition.

Dos Santos, F. P., Adamatti, D., Rodrigues, H., Dimuro, G., Jerez, E. D. M., Dimuro, G., et al. (2016). A multiagent-based tool for the simulation of social production and management of urban ecosystems: a case study on san jerónimo vegetable gardenseville, spain. *Journal of Artificial Societies and Social Simulation*, 19(3):1–12.

Filatova, T., Verburg, P. H., Parker, D. C., and Stannard, C. A. (2013). Spatial agent-based models for socio-ecological systems: Challenges and prospects. *Environmental modelling & software*, 45:1–7.

Gaudou, B., Sibertin-Blanc, C., Therond, O., Amblard, F., Auda, Y., Arcangeli, J.-P., Balestrat, M., Charron-Moirez, M.-H., Gondet, E., Hong, Y., Lardy, R., Louail, T., Mayor, E., Panzoli, D., Sauvage, S., Sánchez-Pérez, J.-M., Taillandier, P., Van Bai, N., Vavasseur, M., and Mazzega, P. (2014). The MAELIA multi-agent platform for integrated analysis of interactions between agricultural land-use and low-water management strategies. In Alam, S. J. and Parunak, H. V. D., editors, *Multi-Agent-Based Simulation XIV*, pages 85–100, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hübner, J. F., Bordini, R. H., and Vieira, R. (2004). Introdução ao desenvolvimento de sistemas multiagentes com jason. *XII Escola de Informática da SBC*, 2:51–89.

Hübner, J. F., Sichman, J. S., and Boissier, O. (2007). Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. *Int. J. Agent-Oriented Software Engineering*, 1(3/4):370–395.

Li, G., Wang, Y., Huang, D., and Yang, H. (2017). Water-energy-food nexus in urban sustainable development: an agent-based model. *International Journal of Crowd Science*, 1(2):121–132.

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and Balan, G. (2005). Mason: A multiagent simulation environment. *SIMULATION*, 81(7):517–527.

Nguyen Vu, Q. A., Gaudou, B., Canal, R., and Hassas, S. (2009). Coherence and robustness in a disturbed mas. In *2009 IEEE-RIVF International Conference on Computing and Communication Technologies*, pages 1–4.

North, M. J., Collier, N. T., and Vos, J. R. (2006). Experiences creating three implementations of the repast agent modeling toolkit. *ACM Trans. Model. Comput. Simul.*, 16(1):1–25.

Page, C. L., Bousquet, F., Bakam, I., Bah, A., and Baron, C. (2000). CORMAS : A multiagent simulation toolkit to model natural and social dynamics at multiple scales. In *Wageningen : Resource Modeling Association*.

Page, C. L., Dray, A., Perez, P., and Garcia, C. (2016). Exploring how knowledge and communication influence natural resources management with rehab. *Simulation & Gaming*, 47(2):257–284.

Ricci, A., Piunti, M., Viroli, M., and Omicini, A. (2009). Environment programming in CArtAgO. In *Multi-Agent Programming: Languages, Tools and Applications*, Multiagent Systems, Artificial Societies, and Simulated Organizations, chapter 8, pages 259–288. Springer.

SEMA (2019). Secretaria do Meio Ambiente e Infraestrutura - L040 - Bacia Hidrográfica da Lagoa Mirim e do Canal São Gonçalo. https://www.sema.rs.gov.br/l040-bacia-hidrografica-da-lagoa-mirim-e-do-canal-sao-goncalo [Online; accessed 31 mar. 2019].

Taillandier, P. and Buard, E. (2009). Designing agent behaviour in agent-based simulation through participatory method. In Yang, J.-J., Yokoo, M., Ito, T., Jin, Z., and Scerri, P., editors, *Principles of Practice in Multi-Agent Systems*, pages 571–578, Berlin, Heidelberg. Springer Berlin Heidelberg.

Taillandier, P., Gaudou, B., Grignard, A., Huynh, Q.-N., Marilleau, N., Caillou, P., Philippon, D., and Drogoul, A. (2018). Building, composing and experimenting complex spatial models with the GAMA platform. *GeoInformatica*.

Thérond, O., Sibertin-Blanc, C., Lardy, R., Gaudou, B., Balestrat, M., Hong, Y., Louail, T., Nguyen, V. B., Panzoli, D., Sanchez-Perez, J.-M., Sauvage, S., Taillandier, P., Vavasseur, M., and Mazzega, P. (2014). Integrated modelling of social-ecological systems: The MAELIA high-resolution multi-agent platform to deal with water scarcity problems. In *7th International Environmental Modelling and Software Society (iEMSs 2014)*, page pp. 1, San Diego, California, United States.

Tisue, S. and Wilensky, U. (2004). Netlogo: A simple environment for modeling complexity. In *International Conference on Complex Systems*, volume 21, pages 16–21, Boston, MA.

Wooldridge, M. (2002). An introduction to multi agent systems, department of computer science, university of liverpool, uk.

Yu, S., He, L., and Lu, H. (2016). An environmental fairness based optimisation model for the decision-support of joint control over the water quantity and quality of a river basin. *Journal of Hydrology*, 535.

Finding new routes for integrating Multi-Agent Systems using Apache Camel*

Cleber J. Amaral[1,2], Sérgio P. Bernardes[1], Mateus Conceição[1]
Jomi F. Hübner[1], Luis P. A. Lampert[1], Otávio A. Matoso[1], Maicon R. Zatelli[1]

[1]Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil

[2]Instituto Federal de Santa Catarina (IFSC)
São José – SC – Brazil

cleber.amaral@ifsc.edu.br, {sergiopb1998,lp.lampert}@gmail.com
mateusconceicao1@hotmail.com, {jomi.hubner,maicon.zatelli}@ufsc.br
otaviomatoso@yahoo.com.br

Abstract. *In Multi-Agent Systems (MAS) there are two main models of interaction: among agents, and between agents and the environment. Although there are studies considering these models, there is no practical tool to afford the interaction with external entities with both models. This paper presents a proposal for such a tool based on the Apache Camel framework by designing two new components, namely* camel-jason *and* camel-artifact*. By means of these components, an external entity is modelled according to its nature, i.e., whether it is autonomous or non-autonomous, interacting with the MAS respectively as an agent or an artifact. It models coherently external entities whereas Camel provides interoperability with several communication protocols.*

1. Introduction

MAS literature has plenty of research about agents' interactions, i.e., agents sending and receiving messages to and from other agents (A-A). Many approaches model almost any entity as an agent and thus the interaction remains something among agents. However there are new approaches that questioned the *agentification* method proposing an MAS where non-autonomous entities are conceived as artifacts in the environment. In these approaches, the development of an MAS considers the design of both agents and artifacts. The environment is not simply what is outside the system (the exogenous environment), but it is designed accordingly to the system purpose (the endogenous environment) [Ricci et al. 2006, Omicini et al. 2008]. In this sense, we have two models of interactions: agent-to-agent (A-A) and agent-to-environment (A-E). In the former, an agent *communicates* with another agent using an Agent Communication Language (ACL) and in the latter, an agent *perceives and acts* upon artifacts in the environment.

When we consider the integration with other applications, those two models are adopted by the current development platforms. On the one hand, we have approaches that use ACL for that purpose and other applications are seen by the agents as other agents (having a mental state, implied by the ACL semantics). On the other hand, we

*Supported by Petrobras project AG-BR, IFSC and UFSC.

have approaches where other applications are seen as part of the environment and agents perceive and act on them. Some platforms provide an A-A approach while others an A-E approach, but, as far as we know, no platform provides both. The designer is forced to conceive some other application either as an agent or as an artifact, despite the application properties.

In this paper, we propose to apply the same argument as [Omicini et al. 2008] for the integration of MAS and external applications: some external applications are autonomous and should be modelled as agents while others are non-autonomous and should be modelled as artifacts. For instance, in the Industry 4.0 context, it is expected the interaction of many entities such as an autonomous planner sending commands to a non-autonomous machine, which signalises what was done. Later, the planner must choose a supplier after an auction to hire a freight to take the product to the destiny, which is usually an human. This short example gives an idea of how comprehensive and challenging the integration can be. We can notice that both models of integration are required: the *autonomous* planner and the human are better modelled as agents, performing A-A interactions, and the *non-autonomous* machine should be integrated as an artifact which when communicating with and agent performs and A-E interaction. Following this concept, we have developed two components for JaCaMo platform, for integration among agents, and between agents and the environment. The referred components are used to set communication routes for the MAS and external entities, using the framework Apache Camel [Ibsen and Anstey 2010], a mediation tool to provide interoperability with many technologies.

2. MAS integration approaches

MAS are being applied as a core technology for distributed systems that needs cooperation and negotiation [Roloff et al. 2016]. The integration of MAS and external entities, i.e. any entity which was not defined its totality within the MAS itself, regards concerns such as compatibility with standards, interoperability and portability. We have found two main forms of integration: (i) among agents (A-A); and, (ii) between agents and the environment (A-E).

2.1. Integration among agents (A-A)

The communication among agents is usually done by speech-acts which considers utterances as actions, usually intending to change the mental state of recipients. The utterance can inform beliefs, desires and intentions of rational agents that attempt to influence other agents. The Knowledge Query and Manipulation Language (KQML) is the first speech-act based language providing high level communication in the distributed artificial intelligence applications [Vieira et al. 2007]. In fact, once speech acts became widely accepted in MAS community, the integration among different agent's platforms was facilitated.

Currently FIPA-ACL, which is very similar to KQML, is the main standard for agents communication. FIPA-ACL uses performatives to make explicit an agent's intention for each sent message, for instance, *inform* is used to influence the recipient to believe in something, and *request* to influence the recipient to add something as a goal [Vieira et al. 2007].

Another communication aspect is related to the expected sequence of messages. Conversations among agents usually follow some patterns which are often referred to

as interaction protocols. Typical patterns such as negotiation, auction, and task delegation are defined using FIPA standards [Bellifemine et al. 2005]. In addiction, there are communication infrastructures that allow agents to be distributed over a network. The challenge in A-A is the integration between an agent, for instance, using FIPA-ACL, and another agent using another language, for instance, an human. This situation leads to the necessity of some tool to make both end-points compatible.

2.2. Integration between agents and the environment (A-E)

There are systems or parts of a system which are better seen as resources or tools that can be used by agents to achieve their goals. These entities, called artifacts, have no internal goals, they are not autonomous and neither proactive, but they supply useful functionalities for agents. An analogy for agents and artifacts is the interaction between humans, as autonomous entities, and tools they exploit in their activities [Ricci et al. 2006]. For instance, a blackboard shared by agents would be modelled as an artifact, being predictable and deterministic, if not, it would perform undesirable autonomous behaviour.

Artifacts are placed in workspaces which represent areas of the MAS environment. Agents can perceive changes and act within the workspaces they are occupying, and on artifacts they are watching, i.e., being aware of events and signals due to interactions with non-autonomous entities held inside of virtual boundaries. The environment can reflect the effects of agents' actions and other phenomena. It is being treated as a first-class programming abstraction with similar importance of agents programming abstraction [Ricci et al. 2006].

Besides modelling non-autonomous entities, the artifacts can also be used for other purposes: (i) for agents coordination using shared artifacts such as organisational boards or coordination marks; (ii) for indirect communication among agents, for instance, by blackboard artifacts; (iii) for implementing the user interface of a system; (iv) for controlling transactions over environment elements through distribution and synchronisation facilities; and (v) for integration between the MAS and external entities [Boissier et al. 2019].

Regarding the use of artifacts to integrate external entities, the integration is done usually through specific Application Programming Interfaces (APIs). The main concern with this approach regards to the high programming effort when there are different protocols in scenarios of heterogeneous devices.

3. Integrating A-A and A-E using Camel

Back to our example in the Industry 4.0 context, Figure 1 shows a process that begins with a packed product, in a production line, up to its delivery to the customer. On the first step, there is an industrial device, a non-autonomous entity, that communicates using an industrial protocol. Once the device signalises the end of the production, the order is checked out on the Enterprise Resource Planning (ERP) software, another non-autonomous entity. The supplier should choose the best offer for a freight, which may be done by accessing suppliers' systems to then interact with the winner, an autonomous entity. Later, the delivery should be tracked by a monitoring system, which is non-autonomous. Finally, when the product is near the destination, a message must keep the client, an autonomous entity, informed.

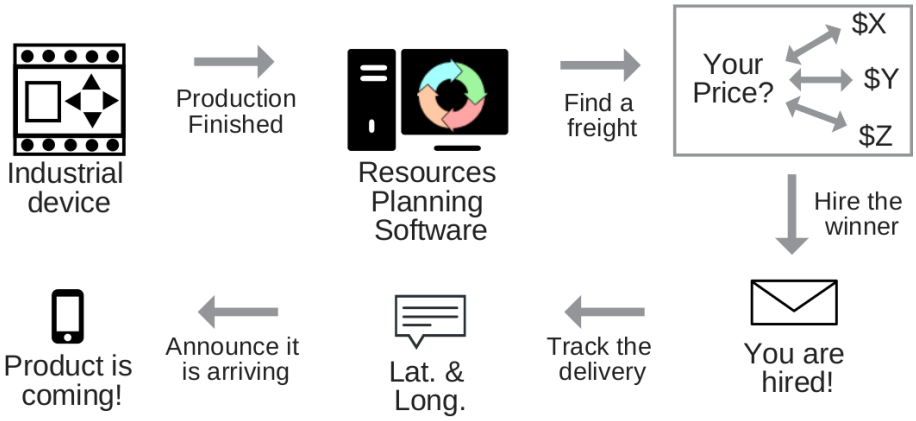


Figure 1. Finishing production and delivering the product in Industry 4.0 context

This scenario illustrates the requirements for the integration: heterogeneous endpoints with different kinds of interactions due to the autonomous and non-autonomous nature of entities. We propose the use of the framework Apache Camel for both integration models, A-A and A-E. We have thus two components: *camel-jason* for integrating MAS's internal agents with external entities modeled as agents, and *camel-artifact* for integrating the former agents with external entities modeled as environmental artifacts.

3.1. Apache Camel

Apache Camel is a lightweight Java-based framework message routing and mediation engine [Ibsen and Anstey 2010]. Camel achieves high-performance processes handling multiple messages concurrently, and provides functions such as routing, exception handling, and testing. It uses structured messages and queues based on Enterprise Integration Patterns (EIP) [Hohpe and Woolf 2003], preserving loose coupling among the resources. Camel works as a middleware that can be incorporated into an application through the use of *components*. Communication among Camel components is defined in so-called *routes*, which set and manage how messages will be exchanged, possibly following sets of rules and using data manipulation.

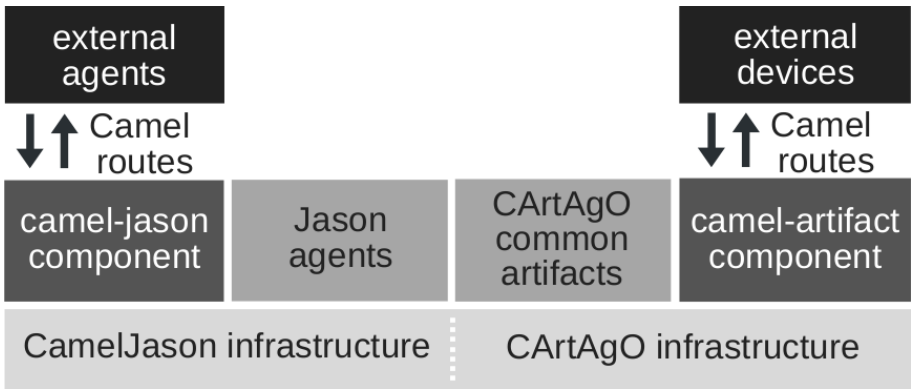


Figure 2. MAS architecture using *camel-jason* and *camel-artifact* components.

Routes define a single endpoint for each entity with an unique address. The defined endpoint may receive data, through a *producer* and send data through a *consumer*. Consumers are entities that admit data in specific formats and encapsulates it in a camel

exchange object, an item that any other producer understands and is able to decode. Producers are entities that receive encapsulated data from the consumer decoding it in its entity's message structure.

In our implemented components, Camel is being embedded in two slightly different manners regarding the models of integration, A-A and A-E, as shown in Figure 2. In the case of A-A, it works as a communication infrastructure that is used when the recipient is not found locally. In the case of A-E, the external device is usually modelled reflecting real operations and signals that it generates, typically having their individuals routes. In both cases, the components are able to define tuned integration, covering a range of endpoints features. Notice that, the complexity of each supported protocol is processed in a Camel component, which works as a bridge to Camel routes. There are more than two hundred components available on Camel's website[1] and many others on the community's repositories.

3.2. *camel-jason* component

The *camel-jason* component enables agents to communicate with external entities through ACL, whilst fulfilling the need of understanding those entities as agents when modeling the MAS. In our proposal, the external entity has a kind of virtual counterpart inside the MAS, a *dummy agent*. This counterpart is seen by the agents as an ordinary agent of the system. Doing so, agents can directly communicate with external entities assuming that they are other agents (as receivers and senders of ACL messages).

Camel-jason component provides a communication flow that is illustrated in Figure 3 where an agent interact with a service A (an external entity). Since the agent sees the service as another agent, it uses ACL for the communication. When the service wants to contact the agent, the camel-jason component translates the message into ACL and the agent receives it as if it comes from an agent.

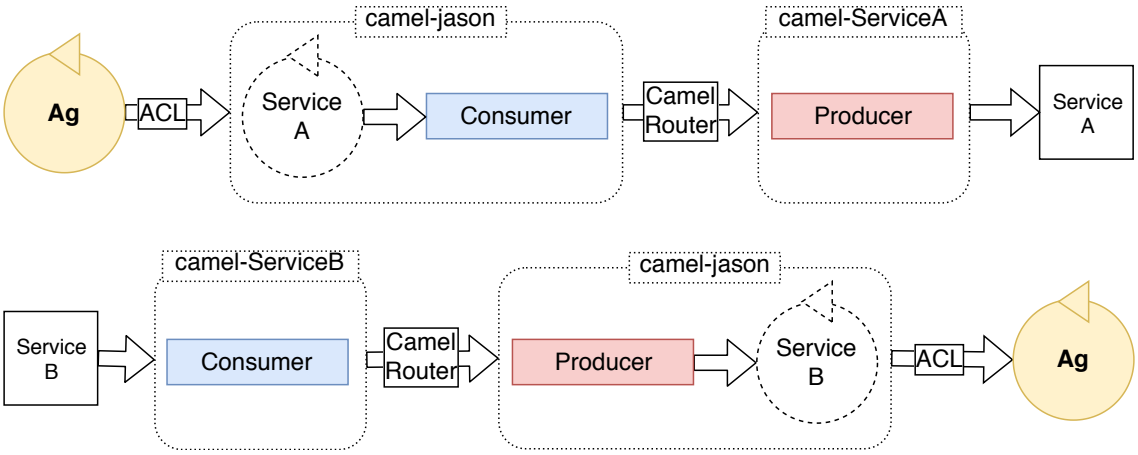


Figure 3. Communication flow using *camel-jason* component.

On the top communication flow showed in Figure 3, we have an agent sending a message to an external autonomous entity: (i) the agent send an ACL message addressed to a dummy agent, which is created by *camel-jason* component, referring to *Service A*;

[1]Supported Camel components are listed in http://camel.apache.org/components.html

(ii) the message is consumed by *camel-jason* component consumer; (iii) the message is exchanged to the other side of the route, possibly being transformed; and (iv) the message is processed by *Service A* component producer which prepares a service A compliance message, which will be sent to some network address to be effectively consumed by the *Service A*.

In the other way around, on the bottom of Figure 3, we have: (i) *Service B* sends some data through the network reaching *Service B* component consumer by its network address; (ii) the message is exchanged through Camel route, possibly being transformed; (iii) the message is processed by *camel-jason* component producer which generates an ACL message; and (iv) the receiver agent effectively consumes the ACL message.

The component uses a simplistic method to define the communication routes, in which for many cases no actual programming is required, only XML definitions. The user should know how to fill camel endpoint parameters according to the compatible endpoint of the application. In cases data transformation is required, camel brings some tools for simple transformation as well as complex ones, using embedded programming codes if needed.

3.3. *camel-artifact* component

In order to sustain the A-E model, the CArtAgO infrastructure is used, and the *camel-artifact* component was developed. This component allows agents to perceive and act upon artifacts that represent external entities inside the MAS.

Notwithstanding, *camel-artifact* also allows the definition of communication routes between CArtAgO artifacts and external entities. Routes for the *camel-artifact* component are implemented using the Java language. The user should be aware of regular camel routes and how to define endpoints and their respective parameters.

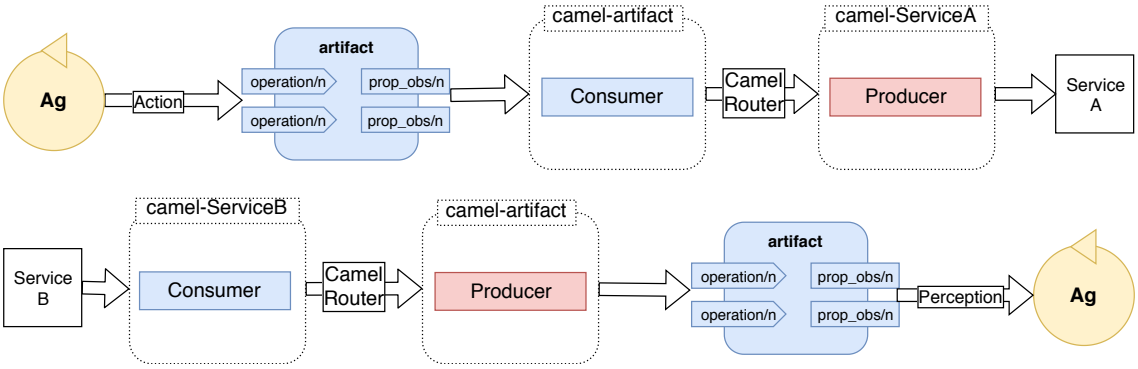


Figure 4. Communication flow using *camel-artifact* component.

On the top communication flow showed in Figure 4, we have an artifact integrated with some service A, an external non-autonomous entity. The interaction between them is as follows: (i) the artifact sends a message to to *Service A*, through specific methods provided by *camel-artifact*; (ii) the message, in form of an operation request, is consumed by *camel-artifact* component consumer which generates a camel standardised message to be sent to the external entity; (iii) the message is exchanged to the other side of the route, possibly being transformed; and (iv) the message is processed by *Service A* compatible component producer which prepares a compliance final message, with the proper format

and structure, which will be sent to some network address to be effectively consumed by the *Service A*.

In the other way around, on the bottom of Figure 4, we have: (i) *Service B* sends some data through the network reaching *Service B* component consumer by its network address; (ii) the message is exchanged through Camel route, possibly being transformed; (iii) the message is processed by *camel-artifact* producer which generates an artifact operation request; and (iv) the recipient artifact effectively consumes the operation request executing the referred method.

4. Illustrative application

For a better understanding of how the *camel-jason* and *camel-artifact* components can be used, we will resume the example of Industry 4.0, presented in Figure 1, and will build an implementation of this system.

The Figure 5 shows the MAS fully designed, with agents, external entities and the camel components used to implement the integration. These components are represented in the middle layer as artifacts and dummy agents. This hypothetical scenario implements an MAS to integrate the production and distribution stages of a product. The whole course can be divided in five stages: (i) a Programmable Logic Controller (PLC) finishes the product manufacturing, (ii) the information about the product is uploaded to an Enterprise Resource Planning (ERP) software, (iii) a research starts in order to contract the best freight company, (iv) the hired company starts transporting the product, providing its tracking information, and (v) warns the client via chat when it is near the final destination. The MAS is designed to unify those stages and to be responsible for managing each process. Moreover, Camel components are used as middleware between the MAS and external entities to integrate them.

One common question when designing the MAS is how many agents should be used. This is not mandatory, but a natural thought is to divide the process into sections and designate a single agent to be responsible for each part. In this case, we will consider that PLC and ERP stages represent the production part of the process, so one agent, named *production_agent*, will be responsible for managing these processes. Next, there is the hiring stage, that comprehends searching and hiring the best delivery company, for which the *distribution_agent* will be designated. The final stage could be thought as the delivery process, where the last agent, named *delivery_agent*, will be responsible for consulting the tracking information and sending the message to the costumer.

Another part of the designing process is the identification of which model, agent or artifact, is more suitable to represent each external entity. A common way to decide is observing its nature, i.e. autonomous or non-autonomous. Following this idea, it could be decided that the PLC and ERP software would be modeled as artifacts, since they are non-autonomous entities; and the customer as an agent, an autonomous entity. Another possibility to decide is looking at which type of communication the agents will perform with each external entity, i.e. via message exchanging or perception-action. For example, the action of hiring the delivery company, informing the company about some new contract via email, seems natural to be modeled as an A-A interaction. For this situation, the best suitable performative for the message is *tell*, since the agent is telling the supplier about a new hired delivery. On the other hand, when the *distribution_agent* is searching

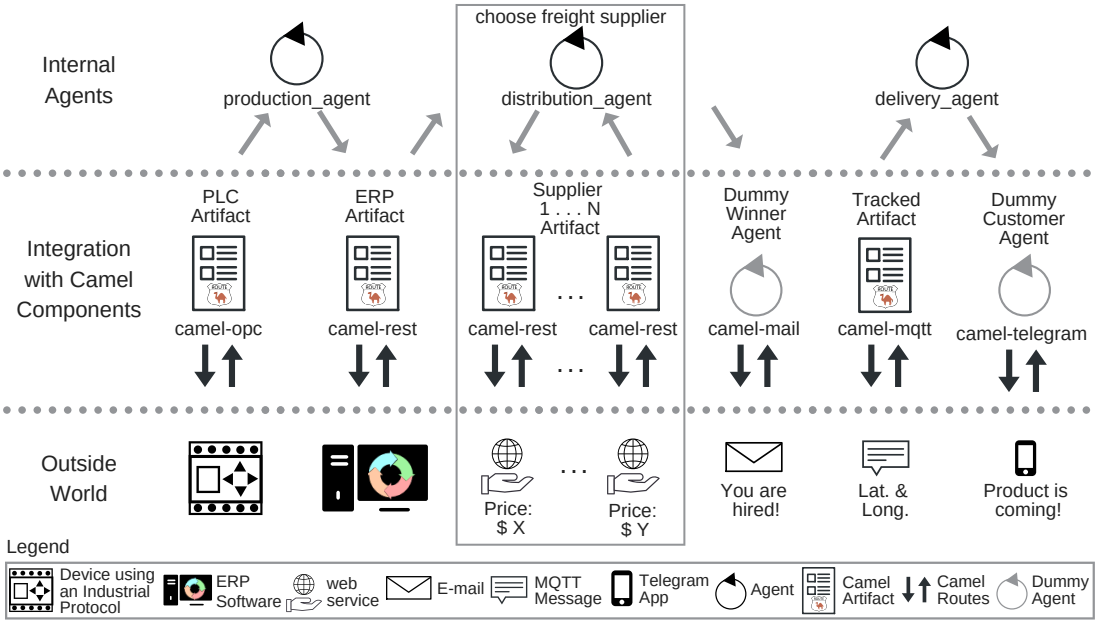


Figure 5. An industrial process illustrating the integration of an MAS with a manufacturing device, updating an enterprise management software, choosing a supplier for delivering a tracked product to the customer.

for the best delivery company, consulting their prices and conditions, it seems more suitable for this information to be perceived, like people do in a websearch. The same idea can be used when the *delivery_agent* tracks the position of the product, the information again is perceived by the agent, like looking at a screen. If message exchange was used in this case the agent would be flooded with unnecessary messages.

In fact, both interpretations, i.e., perception-action vs message exchanging and autonomous vs non-autonomous, could point to the same conclusion. This statement can be tested in the integration between the *delivery_agent* and the customer. The chat is done by message exchanging and it is performed by autonomous entities. Therefore, both interpretation reinforces the idea that the costumer should be modeled as an agent.

With the external entities modeled as MAS elements we could use *camel-artifact* and *camel-jason* to integrate the internal agents with the artifacts and agents, respectively. It is worth commenting that the external entities could be easily exchanged, for instance making the auction via email instead of using a web service. The interaction via email suggests modelling the participants as agents and use the *camel-jason* component for the integration.

Now the camel routes can be developed, depending on which type of technology the external entities use. As explained before, Camel have more than two hundred end-points available. In this example, OPC-DA, Rest, email, MQTT and Telegram end-points are being used to create the routes.

The code, in XML, for the route from the *delivery_agent* to Telegram can be seen in Listing 1. The `from` tag signalises the consumer part of the route, and `to` signalises the producer. In this case, the consumer address is the name of the dummy agent to which the message had been sent (in this example, `customer`), and the producer address is the

authorisation token followed by the `chatId` option.

```
<route>
  <from uri="jason:DummyCustomerAgent"/>
  <to uri="telegram:bots/sometoken?chatId=-364531"/>
</route>
```

Listing 1. Example of *camel-jason* route definition

In this example, the *delivery_agent* also uses the *camel-artifact* in order to obtain the delivery's position from a MQTT server. The route, in Java, is shown in Listing 2. When the position is published on the topic of interest (`latLong`) the route redirects it to the artifact by specifying its name (`TrackedArtifact`) and the operation (`giveDistance`) as headers. Here, we are assuming that the calculations will be done by the artifact, but they could be done in the route through a transformation, before sending to the artifact.

```
from( "mqtt : foo? host=tcp://broker & subscribeTopicName=latLong" )
  .setHeader( "ArtifactName" , constant ( "TrackedArtifact" ) )
  .setHeader( "OperationName" , constant ( "giveDistance" ) )
.to( "artifact : cartago" );
```

Listing 2. Example of *camel-artifact* route definition

5. Related research

In this section, we went over works that have addressed agent technology in an integrating context. Maturana and Norrie [Maturana and Norrie 1996] have proposed a mediation and coordination tool for MAS. They have used mediator agents as manufacturing coordinators. Following similar idea, Olaru et al. [Olaru et al. 2013] have developed an agent-based middleware, which creates a sub-layer of application layer that allows agents to mediate context-aware exchange of information among entities. We think an autonomous entity as middleware may increase complexity and compromise performance. Instead of creating some kind of hierarchy, our approach gives connectivity power to MAS entities.

Leading industrial suppliers are also providing solutions using agents such as the Agent Development Environment (ADE), designed by Rockwell Automation [Tichý et al. 2012]. It provides connectivity with common shop floor devices and supports the development of agents. The limitation we have seen regards especially connectivity with all sorts of entities (e.g. IoT sensors and mobile devices, ERP and other software etc), which in our case is provided by Camel.

Other research address the combination of MAS and Service-Oriented Architecture (SOA). One way to achieve this merge is based on the creation of a proxy function to provide interoperability between MAS and SOA, as found in [Nguyen and Kowalczyk 2005, Shafiq et al. 2005, Greenwood and Calisti 2004, Fayçal et al. 2010]. Another way is by implementing services as agents as we found in [Mendes et al. 2009, Tapia et al. 2009, Carrascosa et al. 2009, Argente et al. 2011]. The approaches using SOA are more mature to be applied in practice. The ones that *agentified* the services have also the advantage to use MAS background, i.e., using ACL messages they are able to use interaction protocols. In these studies integration is usually done through specific APIs and they lack differentiation over autonomous and non-

autonoumous entities, and interoperability with heterogeneous entities, both aspects increase development complexity.

Vrba et al. [Vrba et al. 2014] propose a gateway for wrapping an MAS as a service to be used as a loosely coupled software component into the Enterprise Service Bus (ESB). This gateway transforms agent messages to ESB messages and vice versa, enabling communication between agents and ESB services. This solution is closely to ours, only lacking support to the A-E approach, where interaction is based on agents perceiving and acting upon artifacts in the environment. The indiscriminately *agentification* may increase complexity and affect performance.

Cranefield and Ranathunga [Cranefield and Ranathunga 2013] developed a *camel-agent* component for *Jason* agents. It is very similar to our developed *camel-jason* component. Essentially, the difference is that we are embedding Apache Camel since our component works as an infrastructure, being transparent to the agents. In their work, Jason was actually embedded in an Apache Camel project where agents were smoothly placed in containers.

6. Conclusion

In this paper, we introduced two Camel components aiming the integration of MAS with external entities: *camel-jason* and *camel-artifact*. The former integrates agents with external entities modelled as agents. The latter integrates agents and external entities modelled as artifacts. The decision of which component to adopt for each entity depends on the characteristics of the external entity and the MAS developer *can* choose the most suitable component. For instance, he/she is not obliged to "agentify" every external entity, even those that do not have agent properties.

The two components introduced in this paper, along with the communication infrastructure provided by Camel and its existing components, makes the integration between MAS and different entities simpler. Issues related to interoperability, routing, and data transformations are partially solved in the camel routes. Another advantage of using such components is that the agent program does not need to deal with integration issues. Agents continue to interact only with another agents and artifacts.

Finally, this is an ongoing work. In a future step we intend to compare our approach with related works and to evaluate other aspects of using the developed Camel components to integrate MAS and external entities, such as the impact on the performance, security, openness, scalability, among others.

References

Argente, E., Botti, V., Carrascosa, C., Giret, A., Julian, V., and Rebollo, M. (2011). An abstract architecture for virtual organizations: The thomas approach. *Knowledge and Information Systems*, 29(2):379–403.

Bellifemine, F., Bergenti, F., Caire, G., and Poggi, A. (2005). *Jade — A Java Agent Development Framework*, pages 125–147. Springer US, Boston, MA.

Boissier, O., Bordini, R. H., Hübner, J. F., and Ricci, A. (2019). Dimensions in programming multi-agent systems. *The Knowledge Engineering Review*, 34:e2.

Carrascosa, C., Giret, A., Julian, V., Rebollo, M., Argente, E., and Botti, V. (2009). Service oriented mas: an open architecture. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 1291–1292. International Foundation for Autonomous Agents and Multiagent Systems.

Cranefield, S. and Ranathunga, S. (2013). Embedding Agents in Business Processes Using Enterprise Integration Patterns. pages 97–116.

Fayçal, H., Habiba, D., and Hakima, M. (2010). Integrating legacy systems in a SOA using an agent based approach for information system agility. *2010 International Conference on Machine and Web Intelligence, ICMWI 2010 - Proceedings*, pages 338–343.

Greenwood, D. and Calisti, M. (2004). Engineering web service-agent integration. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 2, pages 1918–1925. IEEE.

Hohpe, G. and Woolf, B. (2003). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman, USA.

Ibsen, C. and Anstey, J. (2010). *Camel in Action*. Manning Publications, USA, 1st edition.

Maturana, F. P. and Norrie, D. H. (1996). Multi-agent mediator architecture for distributed manufacturing. *Journal of Intelligent Manufacturing*.

Mendes, M., Electric, S., Restivo, F., Colombo, A. W., and Electric, S. (2009). Service-oriented Agents for Collaborative Industrial Automation and Production Systems. 2744(August).

Nguyen, X. T. and Kowalczyk, R. (2005). Enabling agent-based management of web services with WS2JADE. *Proceedings - International Conference on Quality Software*, 2005:407–412.

Olaru, A., Florea, A. M., and El Fallah Seghrouchni, A. (2013). A context-aware multi-agent system as a middleware for ambient intelligence. *Mobile Networks and Applications*, 18(3):429–443.

Omicini, A., Ricci, A., and Viroli, M. (2008). Artifacts in the A&A meta-model for multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 17(3):432–456.

Ricci, A., Viroli, M., and Omicini, A. (2006). Programming MAS with artifacts. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3862 LNAI:206–221.

Roloff, M., Amaral, C., Stivanello, M., and Stemmer, M. (2016). Mas4ssp: A multi-agent reference architecture for the configuration and monitoring of small series production lines. In *INDUSCON*.

Shafiq, M. O., Ali, A., Ahmad, H. F., and Suguri, H. (2005). Agentweb gateway-a middleware for dynamic integration of multi agent system and web services framework. In *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)*, pages 267–268. IEEE.

Tapia, D. I., Rodríguez, S., Bajo, J., and Corchado, J. M. (2009). Fusion@, a soa-based multi-agent architecture. In *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)*, pages 99–107. Springer.

Tichý, P., Kadera, P., Staron, R. J., Vrba, P., and Mařík, V. (2012). Multi-agent system design and integration via agent development environment. *Engineering Applications of Artificial Intelligence*.

Vieira, R., Wooldridge, M., and Bordini, R. H. (2007). On the Formal Semantics of Speech-Act Based Communication in an Agent-Oriented Programming Language. 29:221–267.

Vrba, P., Fuksa, M., and Klima, M. (2014). JADE-JBossESB Gateway: Integration of Multi-Agent System with Enterprise Service Bus. *2014 Ieee International Conference on Systems, Man and Cybernetics (Smc)*, pages 3663–3668.

Exposing agents as web services: a case study using JADE and SPADE

Henrique Donâncio[1], Arthur Casals[2], Anarosa A. F. Brandão[2]

[1]Instituto de Matemática e Estatística - Universidade de São Paulo (IME-USP)
R. do Matão, 1010 - 05508-090 - São Paulo - SP - Brazil

[2]Escola Politécnica - Universidade de São Paulo (EPUSP)
Av. Prof. Luciano Gualberto, 158 - trav. 3 - 05508-900 - São Paulo - SP - Brazil

donancio@ime.usp.br, {arthur.casals,anarosa.brandao}@usp.br

Abstract. *Agents are autonomous software components that can be combined into multiagent systems (MAS). They possess capabilities related to distributed systems. For this reason, agents have been studied along with web services and Web technologies in Service-Oriented Architectures (SOA). With the advent of new Web-related paradigms, the need of reviewing existing work in this area arises. In this paper we briefly review some of the existing work relating agents and web services. For this purpose, we perform a comparison between two existing multiagent platforms (JADE and SPADE) considering their ability to support agents as web services. This comparison is done using an existing implementation of a simple MAS in JADE and its re-implementation in SPADE.*

1. Introduction

Agents are autonomous entities that can be organized in communities and work together to solve problems of different complexity degrees [Ferber 1999]. As such, multiagent system (MAS) are systems composed of multiple agents that interact among themselves in a single environment [Russell and Norvig 2003]. Also, due to their inherent interactive capabilities, agents can be used as a paradigm when designing complex distributed systems [Wooldridge 2009]. Due to their inherent interoperability, each agent can act to maximize the expected output of the system and to adapt to unexpected contingencies [Jennings 2000].

Part of the research related to MAS and complex systems involves integrating agents and web services [Jennings et al. 1998, Lieberman et al. 1995, Etzioni 1996, Ardissono et al. 1999, Greenwood and Calisti 2004]. Designing inter-operable complex systems involves not only creating systems with distributed capabilities, but also systems capable of using existing communication protocols and mechanisms in order to share and reuse knowledge.

In particular, the idea of using agents *as* web services appeared in the early 2000s [Hendler 2001, Huhns 2002]. Subsequent work on agents exposed as web services was also related to web-based MAS [Muldoon 2007, Thiele et al. 2009, Tapia et al. 2009]. However, while SOAP web services are still largely used to implement SOA systems [Keen et al. 2004], new web paradigms use different technologies, from hypermedia-controlled RESTful web services (HATEOAS [Alarcon et al. 2010]) to real-time protocols (such as WebSockets [1]) and event-driven architecture [Michelson 2006]

[1]https://tools.ietf.org/html/rfc6455

elements.

Our research is focused on inter-operable agents capable of using Internet communication protocols. For this reason, we need to explore how existing MAS platforms can be used in conjunction with technologies related to the existing web paradigms. Different MAS platforms were developed in the last years [Kravari and Bassiliades 2015], with different levels of compliance with inter-operable systems development standards. Since the BDI agent architecture (explained below) is of particular interest for our research, we focused our comparison in MAS platforms that support this architecture. Also, due to the fact that we are interested in inter-operable agents, we explicitly analyze the communication language used by agents in the compared platforms.

This paper is organized as follows: in Section 2, we provide some background on web services, intelligent agents, and agent communication languages. Section 3 contains some of the existing work related to using agents in conjunction with Web-related technologies. In Section 4, we briefly present some multiagent platforms as well as the criteria used to choose them. These platforms are compare in Section 5. In addition, Section 6 is dedicated to illustrate how an agent can be deployed as a web service using two of the evaluated platforms. Finally, Section 7 discusses and concludes this paper with some perspectives for future work.

2. Background

In this section we present some concepts that will be used along this work. First we provide a brief description of web services, followed by an overview on intelligent agents (with emphasis on one particular architecture). After that we provide an overview on languages used by the agents to interact between themselves, referred to as *agent communication languages*.

2.1. Web services and SOA

A web service can be described as a software designed to interact with existing applications, enabling different applications to interact between themselves using Web technologies and protocols [Alonso et al. 2004]. It can be *discoverable* through the use of a directory service, and it can be *described* in terms of message formats, communication protocols, and data types it uses. Web services embody the concepts of a service-oriented architecture (SOA)[1] [Erl 2005], so they can be used in conjunction to obtain the functionalities of an equivalent larger system.

2.2. Intelligent agents

As mentioned before, agents are autonomous entities able to work together to solve determined problems. Agents can exist in both physical and virtual environments, and they are capable of proactively interacting with other agents. Such interactions can involve cooperation, negotiation, or coordination, and each agent is capable of creating their own goals and taking individual actions in order to satisfy them [Wooldridge 2009]. As a consequence, a MAS composed of goal-oriented agents is intrinsically adaptable, and can be used to solve problems in many different domains [Pěchouček and Mařík 2008].

[1] http://www.opengroup.org/subjectareas/soa

Intelligent agents are capable of reasoning, deciding which action to perform according (i) to the available information and (ii) to their consequences in the environment [Wooldridge 2009]. The belief-desire-intention (BDI) architecture [Rao and Georgeff 1991] is one of the software architectures used to model and implement intelligent agents, and it is based on the human practical reasoning model [Bratman 1987]. A BDI agent uses the concepts of belief, desire, and intention in a means-ends reasoning process, and its actions are organized in an execution plan built on top of (i) what the agent believes to be true, and (ii) what the agent desires to achieve as a goal [Konolige and Nilsson 1980].

2.3. Agent Communication Languages

An agent communication language (ACL) establishes a structure for the message exchanging between agents, both in type and meaning. However, agents don't simply exchange messages; they actually engage in *conversations* [Labrou 2001]. A conversation can be seen as a pre-arranged protocol or message exchanging pattern, oriented towards a specific task or objective.

We will limit the scope of the ACLs to the two most consistently used [Li and Kokar 2013, Kamdar et al. 2018]: FIPA-ACL and KQML. Both languages are based on the speech acts theory [Searle 1969] and are composed of different *performatives* [1]. They can be understood as sentences that describe and influence an environment at the same time. As such, the messages exchanged between agents can represent actions or communicative acts. Despite being relatively old, these standards are still being used to this date [Răileanu et al. 2018, Blos et al. 2018].

The FIPA protocol [2] is an effort to established guidelines to make platforms interoperable. FIPA-ACL is composed of 22 communication performatives. The most common performatives are:

- `inform`: The sender informs the receiver that a given proposition is true;
- `request`: The sender requests that the receiver execute some action;
- `agree`: The sender agrees to take some action, possibly in the future;
- `not understood`: The sender (eg, *i*) informs the recipient (eg, j) that it perceived the action performed by *j*, but did not understand it;
- `refuse`: The sender refuses to execute a particular action, explaining the reason for the refusal.

While FIPA-ACL only establishes interaction protocols between agents, there are also FIPA specifications for agent management used in conjunction with the communication standards. The basic components of these specifications are (i) the Agent Platform, where each agent has a unique identifier called AID (FIPA Agent Identifier); (ii) the Agent Management System (AMS), responsible for managing the Agent Platform; and (iii) the Directory Facilitator (DF), that allows the agents to publish services in a discoverable manner.

Similarly, the KQML language is also composed of communication performatives. It possesses three different layers:

[1] https://stanford.library.sydney.edu.au/entries/speech-acts/

[2] http://www.fipa.org/

- **Content layer:** where the actual content of the message resides, in the computer's own representation language;
- **Communication layer:** used to encode lower level communication parameters, such as the identity of both the sender and the recipient of the message;
- **Message layer:** provides the performatives used in the communication process, finding all possible interactions with KQML-compliant agents.

KQML also describes a special class of agents named *facilitators*. The *facilitator* is responsible for performing multiple communication services, such as maintaining a registry of services, routing messages to these services based on content, and providing mediation between communication parts. Differently from FIPA-ACL, these specifications are contained in the ACL definition, and do not depend on additional standards.

Despite being similar in multiple aspects, there are some characteristics particular to one or another ACL that make their use domain-dependent. We will not detail all the differences between the two ACLs at this point. These differences, however, can be crucial in choosing one or another ACL according to a specific set of requirements.

3. Related work

Integrating agents and the web services has been both proposed and experimented on since the late 1990s [Jennings et al. 1998, Lieberman et al. 1995, Etzioni 1996, Ardissono et al. 1999, Greenwood and Calisti 2004]. In particular, the idea of using agents *as* web services appeared in early 2000s [Hendler 2001, Huhns 2002]. Using BDI agents as web services was also proposed around that time by Dickinson and Wooldridge [Dickinson and Wooldridge 2005].

Using agents in conjunction with web services was further explores with platforms such as CArtAgO-WS [Ricci et al. 2010], which allowed the development of SOA applications populated by agents taking into account both the concepts of artifacts - "objects" or services that could be used by the agents - and the use of agent architectures (including BDI). Most of the existing work in this field, however, considers the Internet environment mostly as a means for communicating - which is from where our research motivation is originated.

Another related work [Radhakrishnan et al. 2018] presents a comparative performance study between SPADE and JADE, with focus on security in the cloud environment. The compared platforms were chosen due to reasons similar to our own, with the addition of considering knowledge transferring capabilities with the use of ontologies.

In our research, we would like to explore in detail how different Internet technologies could be used by agents and MAS - and not only for communication. Exposing agents as web services is an intuitive manner of taking advantage of the communication protocols already in place. However, the Web can be also used as a distributed environment *per se*. The existing interoperability provided by the SOA architecture can also be used for distributed reasoning and planning, for example, allowing a MAS to be fully implemented and deployed using technologies not necessarily particular to agents or MAS.

For this reason, it is important first to understand how the existing multiagent platforms could be used in this context, and what were their restrictions or limitations regarding the integration with the Web. The objective of the present work is to revisit

web-based agents considering the current state of the Internet. We focus particularly on agents deployed as web services and MAS that use them. For this purpose, we compared different existing multiagent platforms in terms of capabilities and web-related capabilities. This comparison is detailed in the next paragraphs.

4. Multiagent platforms

In order to compare existing multiagent platforms, our first criteria was to choose which ones were open-source and were actively used by the multiagent community. In addition, we also consider the platforms recently used in the annual Multiagent Programming Context event [1] to select the following ones to compare: JaCaMo [2], JIAC V [3], JADE [4], SPADE [5], and SARL [6].

Since the BDI architecture is of particular interest for our research and JIAC and SARL do not provide support for BDI agents, we did not include them in the comparison presented in this work. Both JaCaMo and SPADE provide native support for BDI agents, and JADE uses a complementary layer called BDI4JADE [7]. In terms of implementation, however, JADE and SPADE provide a communication middleware for agents. JaCamo, on the other hand, is a meta-framework for Multi-Agent Oriented Programming (MAOP) implemented in Java. It is composed of three different frameworks, and due to its nature it could be modified to be used on top of JADE or other communication middleware. For this reason, we focused our analysis in the JADE and SPADE platforms.

Before presenting the evaluated platforms, we will describe the communication standards they use. Understanding these standards is not crucial to comprehend the platforms or their evaluation. It is important, however, to understand how they take advantage of the Internet environment from a lower-level perspective. Such understanding is related to the motivation behind the present work, and it will play a steering role in our future research.

4.1. JADE

JADE is a distributed middleware written in JAVA that requires a minimal knowledge of agent theory to implement FIPA-compliant agents. Each agent in JADE is hosted by a *Container*. While a host can contain other Containers, ultimately there is a *Main Container* on top of them all, responsible for providing the essential FIPA-ACL functionalities for the agents.

JADE uses the FIPA protocol not only for agent communication, but also for management. The FIPA protocol specifies an agent management infrastructure (named *Agent Platform*) and a service publishing platform (named *Directory Facilitator*). Using JADE allows the identification of every agent through the use of an unique identifier (AID) in a distributed environment. The Directory Facilitator, on the other hand, allows the agents to publish services that are visible to other agents.

[1] https://multiagentcontest.org/
[2] http://jacamo.sourceforge.net/
[3] http://www.jiac.de/agent-frameworks/jiac-v/
[4] http://jade.tilab.com/
[5] https://pypi.org/project/SPADE/ - version 2.*
[6] http://www.sarl.io/
[7] http://www.inf.ufrgs.br/prosoft/bdi4jade/

The message delivering mechanism used by JADE depends on an agent's location, and it is optimized to minimize the delivery time of messages. If two communicating agents reside in the same host, JADE use an internal set of message transport protocols. Also, if both agents share the same container, the message will not be serialized, but cloned, and the new object reference is passed to the receiving agent. Otherwise, communication between different containers uses JAVA RMI (Remote Method Invocation) [1]. A study of the performance of the message transport layer system of the JADE platform is presented in [Cortese et al. 2002], where it was shown that the JADE internal communication protocol was more efficient for intra-platform communication.

4.2. SPADE

SPADE (Smart Python multi-Agent Development Environment) is an agent framework written in Python, and it is also FIPA-compliant. The BDI architecture used by SPADE is based on a distributed schema: plans used by the agents are represented in the form of services. Therefore, instead of possessing a library with multiple plans, the agents' actions are defined by services published in the Directory Facilitator. In that sense, a plan becomes a composition of offered services that are accessed by the agent so it can accomplish its intentions.

Similarly to JADE, SPADE implements the agent management infrastructure specified by the FIPA protocol. However, it uses a different message delivering mechanism - mostly due to differences in the programming language used in the implementation (Python [2]). It is important to notice that in this work we used SPADE version 2.*, which is built using Python version 2.* .

5. Multiagent Platforms Comparison

We established the comparison criteria between JADE and SPADE 2 from a Web-related perspective. For this reason, we focused on (i) communication protocols and associated characteristics; (ii) organization representation; (iii) support to content language; and (iv) support to ontologies. The latter is especially important since ontologies can be used to model and consume knowledge in an inter-operable manner [Bechhofer 2009]. The results of the comparison can be found in Table 1:

Table 1. Comparison: JADE and SPADE

	JADE	SPADE
BDI	✓	✓
Directory Facilitator (DF)	✓	✓
Agent Management System (AMS)	✓	✓
FIPA-ACL	✓	✓
KQML		
Multiplatform	✓	✓
Explicit organization representation		
Ontologies	✓	

[1] http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html
[2] https://www.python.org/

It is important to mention that SPADE can use ontologies according to the FIPA-ACL specifications, which state that ontologies can be passed as parameters within messages. However, SPADE does not possesses functionalities related to creating and manipulating ontologies within its own environment. JADE, on the other hand, possesses such functionalities - despite the fact that JADE-based ontologies cannot be reused since they are modeled as Java classes.

It is also important to notice that JADE and SPADE possess multiple points in common - especially considering the adherence to the FIPA protocol. While these specifications are not necessarily mandatory for integrating agents and the Web, they are certainly important in scenarios involving a huge number of services. The DF functions as a yellow pages for services, cataloging and listing all web services that can be used by the agents. The AMS, on the other hand, is crucial for the study of message passing and process management within the MAS platforms. The MAS architecture and some aspects of the implementation are detailed in the next section.

6. Deploying agents as web services

As part of our study process, we modeled a simple MAS that used agents deployed as web services. The objective of this step was to verify how this could be built and deployed using the current technologies and tools. As mentioned before, we chose JADE and SPADE as the target multiagent platforms. Part of the reason behind this choice was related to the framework's web-related functionalities previously described. While the modeled MAS is not complex (and we will not describe it in detail in the present work), we intend to explore and evolve this implementation as our research advances.

6.1. JADE

JADE agents have already been used in conjunction with web services [Nguyen and Kowalczyk 2007a, Nguyen and Kowalczyk 2007b, Liu et al. 2006]. The JADE platform uses an add-on called WSIG[1] for this purpose, which handles all requests coming from the Web and sending them to the MAS.

Also, a JADE agent must possess an *Ontology* and a set of *Actions* so that its capabilities can be exposed as web services. Ontologies define the vocabulary and semantics used in the communication between JADE agents. Actions are objects that correspond to a request: once a request is made, it results in an action object containing the request elements that is sent to the JADE agent.

6.2. SPADE

SPADE does not possess a specifically created component or add-on to expose agents as Web Services. For this reason, we used a Python-based library called Werkzeug [2] that works similarly to JADE's WSIG. Using this library made it possible to manipulate requests via the Web and send them to the agent platform.

Since exposing agents as web services is not an existing functionality in SPADE, it also does not have a established set of rules for *Ontologies* and *Actions*, present in the

[1]https://jade.tilab.com/doc/tutorials/WSIG_Guide.pdf
[2]http://werkzeug.pocoo.org/

JADE platform. Its messaging mechanism, however, allows matching specific actions with messages received through the use of a labeling mechanism: inbound messages are checked for this label and processed accordingly, resulting in different actions depending on the label.

6.3. Implementation

In order to perform the proposed comparison, we felt the need to use a single MAS model implemented using the two different MAS platforms. For this reason, we used SPADE to re-implemented an existing MAS that was originally implemented in JADE, called Smart Agenda [Casals et al. 2018] (see Figure 1). This MAS functions as an agent-based personal assistant, and it will be explained in the next paragraphs.

At this point, we would like to mention that the original MAS implementation depends on intrinsic multiagent platform components and functionalities (as seen below). For this reason we chose to use each of the platforms' functionalities as extensively as we could, as a manner of effectively comparing their built-in capabilities.

6.4. MAS Smart Agenda Requirements and Architecture

The original Smart Agenda MAS is a web-based, agent-based personal assistant. It is deployed on the web, and all interactions between the user and the system are performed through the use of a one-page web application. After creating an account in the system, the user can use it to schedule new events or to modify existing ones, with the option of including other registered users in these events. It works pretty much as a web calendar application, with the difference that it uses intelligent agents not only for event coordination among multiple users, but also for its built-in recommendation system [Casals et al. 2018].

The recommendation system used by Smart Agenda takes into account user preferences (i.e. accommodation, transportation means, and periods of the day in which events should be scheduled) for the scheduling of any new events. It also allows the automatic re-scheduling of existing events in the case of any unforeseen circumstances, without the intervention of the user. A video detailing the operation and use of SmartAgenda (made available by the original authors of the system) can be found at https://bit.ly/Emas18Demo .

From the MAS perspective, the system uses three different types of agents: *Coordinator*, *Manager* and *Agenda*. The Coordinator agent is responsible for handling all requests between the user and the system. In order to make the system scalable, every Manager is responsible only for a limited number or users. Every request made by an users is forwarded to a correspondent Manager agent, which is responsible for forward to respective Agenda agent. The Agenda checking if existing any restrictions at the moment. If everything is OK, then proceeds with the event scheduling according to the user's preferences. The Manager agent also plays a role in inter-mediating communications and scheduling conflicts when events involving more than one user are scheduled.

7. Discussion

In this work, we revisited some of the existing work on web-based agents and MAS. We also presented two different open source multiagent platforms: JADE and SPADE.

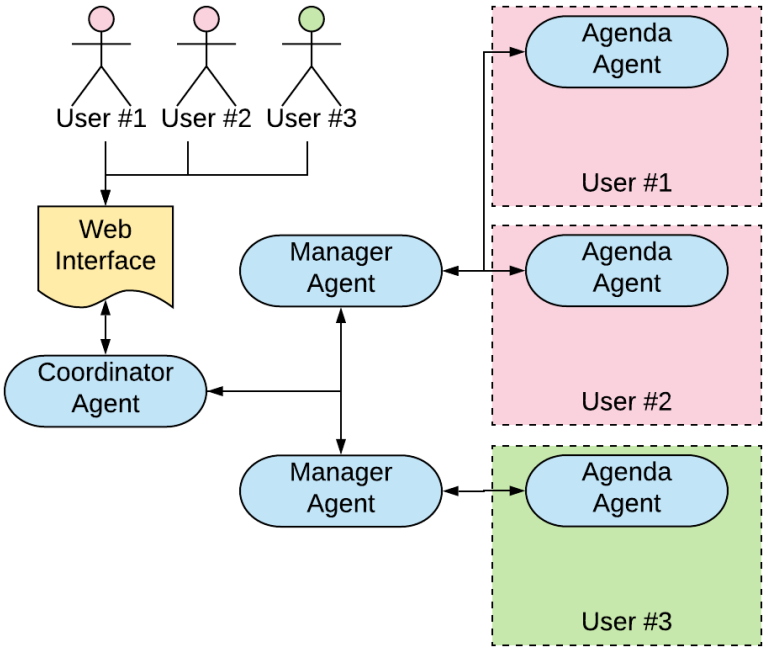


Figure 1. Implementation architecture for the Smart Agenda agents - adapted from [Casals et al. 2018]

Our objective was to compare them according to their support to developing agents to be exposed as web-services. The comparison was presented and a simple MAS was implemented using JADE and SPADE, with agents deployed as web services. Although not being complex, the idea behind the implementation was to create an MAS that could be further evolved and explored during the course of our research.

While the MAS architecture was simple enough to be modeled using a traditional BDI architecture, we found a few difficulties related to the implementation - mostly related to exposing the agents as web services. This is interesting because at this point we would expect that the libraries would be mature enough to allow a seamless implementation. Instead, we found problems varying from incomplete documentation to message passing between agents. This is somewhat worrisome considering that the first implementations exposing agents as web services (and specifically using JADE) are more than a decade old.

As a development ecosystem, Python features a variety of well-structured APIs (Application Programming Interface) as RESTful web services, thus providing multiple inter-operable functions that could be used by Python-based agent platforms. However, we encountered multiple difficulties while trying to establish agent communication within SPADE using web standards, even when using an auxiliary third-party library (Werkzeug). In comparison, we were able to do the same with the WSIG add-on for JADE, despite the fact that both JADE and SPADE are considered to be mature multiagent platforms. While this situation raises some concerns regarding the SPADE platform, it also validates our decision to evaluate MAS platforms considering simple SOAP web services, instead of using newer technologies. We intend to explore this point further in

future work.

This work is meant to be a stepping stone to our research interests. Our long-term objective is to study and explore MAS platforms from both the modeling and the implementation perspective, considering both existing agent-oriented software engineering methodologies and new Web paradigms.

At this point, it was meant only to verify possible bottlenecks in the chosen MAS platforms related to the technologies involved (which we successfully achieved). We also intend to evolve and consolidate both implementations, so they can be complex enough to be analyzed in terms of performance, scalability, and robustness. Once we have a consolidated model implemented in different MAS platforms, we will be able to establish a common baseline for future performance and scalability comparisons.

Nevertheless, we could use these results to define an architecture to delivery, for instance, web-bots that can chat with web users to assist them in several ways.

8. Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001. This work is partially supported by ANEEL's Research and Development (R&D) Program.

References

Alarcon, R., Wilde, E., and Bellido, J. (2010). Hypermedia-driven restful service composition. In *International Conference on Service-Oriented Computing*, pages 111–120. Springer.

Alonso, G., Casati, F., Kuno, H., and Machiraju, V. (2004). *Distributed Information Systems*, pages 3–27. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ardissono, L., Barbero, C., Goy, A., and Petrone, G. (1999). An agent architecture for personalized web stores. In *Proceedings of the third annual conference on Autonomous Agents*, pages 182–189. ACM.

Bechhofer, S. (2009). Owl: Web ontology language. In *Encyclopedia of database systems*, pages 2008–2009. Springer.

Blos, M. F., da Silva, R. M., and Wee, H.-M. (2018). A framework for designing supply chain disruptions management considering productive systems and carrier viewpoints. *International Journal of Production Research*, pages 1–17.

Bratman, M. (1987). *Intention, plans, and practical reason*, volume 10. Harvard University Press Cambridge, MA.

Casals, A., Seghrouchni, A. E. F., Negroni, O., and Othmani, A. (2018). Exposing agents as web services in jade. In *International Workshop on Engineering Multi-Agent Systems*. Springer.

Cortese, E., Quarta, F., Vitaglione, G., and Vrba, P. (2002). Scalability and performance of jade message transport system. In *AAMAS workshop on agentcities, Bologna*, volume 16, page 28.

Dickinson, I. and Wooldridge, M. (2005). Agents are not (just) web services: considering bdi agents and web services. In *Proceedings of the 2005 Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005), Utrecht, The Netherlands*.

Erl, T. (2005). *Service-Oriented Architecture – Concepts, Technology, and Design*. Prentice Hall, 1 edition.

Etzioni, O. (1996). Moving up the information food chain: Deploying softbots on the world wide web. In *Proceedings of the national conference on artificial intelligence*, pages 1322–1326.

Ferber, J. (1999). *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading.

Greenwood, D. and Calisti, M. (2004). Engineering web service-agent integration. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 2, pages 1918–1925. IEEE.

Hendler, J. (2001). Agents and the semantic web. *IEEE Intelligent systems*, 16(2):30–37.

Huhns, M. N. (2002). Agents as web services. *IEEE Internet computing*, 6(4):93–95.

Jennings, N. R. (2000). On agent-based software engineering. *Artificial intelligence*, 117(2):277–296.

Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1(1):7–38.

Kamdar, R., Paliwal, P., and Kumar, Y. (2018). A state of art review on various aspects of multi-agent system. *Journal of Circuits, Systems and Computers*, page 1830006.

Keen, M., Acharya, A., Bishop, S., Hopkins, A., Milinski, S., Nott, C., Robinson, R., Adams, J., and Verschueren, P. (2004). Patterns: Implementing an soa using an enterprise service bus. *IBM Redbooks*, 336:20–28.

Konolige, K. and Nilsson, N. J. (1980). Multiple-agent planning systems. In *AAAI*, volume 80, pages 138–142.

Kravari, K. and Bassiliades, N. (2015). A survey of agent platforms. *Journal of Artificial Societies and Social Simulation*, 18(1):11.

Labrou, Y. (2001). Standardizing agent communication. In *ECCAI Advanced Course on Artificial Intelligence*, pages 74–97. Springer.

Li, S. and Kokar, M. M. (2013). *Agent Communication Language*, pages 37–44. Springer New York, New York, NY.

Lieberman, H. et al. (1995). Letizia: An agent that assists web browsing. *IJCAI (1)*, 1995:924–929.

Liu, S., Küngas, P., and Matskin, M. (2006). Agent-based web service composition with jade and jxta. In *SWWS*, volume 6, pages 110–116.

Michelson, B. M. (2006). Event-driven architecture overview. *Patricia Seybold Group*, 2(12):10–1571.

Muldoon, C. (2007). *An agent framework for ubiquitous services*. PhD thesis, Citeseer.

Nguyen, X. T. and Kowalczyk, R. (2007a). Ws2jade: Integrating web service with jade agents. In Huang, J., Kowalczyk, R., Maamar, Z., Martin, D., Müller, I., Stoutenburg, S., and Sycara, K. P., editors, *Service-Oriented Computing: Agents, Semantics, and Engineering*, pages 147–159, Berlin, Heidelberg. Springer Berlin Heidelberg.

Nguyen, X. T. and Kowalczyk, R. (2007b). Ws2jade: Integrating web service with jade agents. In *International Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering*, pages 147–159. Springer.

Pěchouček, M. and Mařík, V. (2008). Industrial deployment of multi-agent technologies: review and selected case studies. *Autonomous Agents and Multi-Agent Systems*, 17(3):397–431.

Radhakrishnan, G., Chithambaram.V, and K.L., S. (2018). Comparative Study of JADE and SPADE Multi Agent System.

Răileanu, S., Anton, F. D., Borangiu, T., and Anton, S. (2018). Design of high availability manufacturing resource agents using jade framework and cloud replication. In *Service Orientation in Holonic and Multi-Agent Manufacturing*, pages 201–215. Springer.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. In Allen, J., Fikes, R., and Sandewall, E., editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

Ricci, A., Denti, E., and Piunti, M. (2010). A platform for developing soa/ws applications as open and heterogeneous multi-agent systems. *Multiagent and Grid Systems*, 6(2):105–132.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.

Searle, J. R. (1969). *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press.

Tapia, D. I., Rodríguez, S., Bajo, J., and Corchado, J. M. (2009). Fusion@, a soa-based multi-agent architecture. In *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)*, pages 99–107. Springer.

Thiele, A., Kaiser, S., Konnerth, T., and Hirsch, B. (2009). Mams service framework. In *International Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering*, pages 126–142. Springer.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Sistema de Recomposição Automática para Rede de Distribuição de Energia Desenvolvido em JADE

Lucas S. Melo[1], Raimundo F. Sampaio [1], Ruth P. S. Leão[1], Giovanni C. Barroso[2].

[1]Departamento de Engenharia Elétrica – Universidade Federal do Ceará (UFC)
Caixa Postal 6001 – Fortaleza – CE – Brazil

[2]Departamento de Física
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brazil

{rfurtado@dee.ufc.br, lucassmelo@dee.ufc.br, rleao@dee.ufc.br, gcb@fisica.ufc.br}

Abstract. *Traditionally, automatic recomposition systems (ARS) are developed with centralized intelligence integrated to SCADA. However, smart grids demand self-healing systems with distributed intelligence, in which Multiagent Systems are the most promising technique. This work aims to present a Multiagent Automatic Recomposition System (MARS) implemented and tested in four computers of Smart Grids Research Group (SGRG) - UFC laboratory. The results showed the potential of MAS for developing self-healing systems applied to smart grids.*

Resumo. *Tradicionalmente sistemas de recomposição automáticos (SRA) são desenvolvidos com inteligência centralizada integrados SCADA. No entanto, as Smart Grid demandam por sistemas self-healing com inteligência distribuída. Dentro deste contexto, a técnica de sistemas multiagentes (SMA) é indicada na literatura como mais promissora. Esse artigo tem como objetivo apresentar um Sistema Multiagente de Recomposição Automática (SMRA) implantado e testado em quatro computadores do Laboratório do Grupo de Redes Elétricas Inteligentes da UFC. Os resultados dos testes demonstraram o potencial da técnica de SMA para desenvolvimento de sistemas selfhealing aplicado as Smart Grid.*

1. Introdução

Nos últimos anos o setor elétrico mundial vem experimentando transformações norteadas a partir do conceito de Redes Elétricas Inteligentes (REI) que incorporam soluções tecnológicas em infraestrutura, sistemas de comunicação, controle e automação com o objetivo de tornar-se mais eficiente, confiável e ambiental e economicamente sustentável (Momoh, 2012). As principais características das REI são: integração de diferentes recursos renováveis, participação ativa de consumidores, uso eficiente dos ativos,

operação eficiente da rede elétrica, auto recomposição mediante distúrbio na rede, aplicação de controle pervasivo e ativo, e baixo impacto ambiental (Ma, Chen, Huang, & Meng, 2013).

A operação das REI demanda inovações nos recursos operacionais para isolar áreas afetadas por faltas e restabelecer o fornecimento de energia aos consumidores de forma automática, rápida e segura (Ma et al., 2013). A função de recomposição automática consiste na capacidade da rede elétrica, diante de uma falta, blackout, ou ação maliciosa, e com base no monitoramento de seu estado, executar ações para reconfigurar a rede elétrica e restabelecer o fornecimento de enrgia com segurança e sem violação das restrições operativas (Ma et al., 2013).

As REI têm proporcionado o crescente desenvolvimento de Sistema Multiagente (SMA) aplicados a sistemas elétricos (Zidan, El-Saadany, & El Chaar, 2011). SMA é uma técnica que por sua capacidade de modelagem, simulação e controle distribuído pode ser aplicada nos segmentos de geração, transmissão, distribuição, comercialização e consumo de energia elétrica, como por exemplo em: diagnóstico de falta (Davidson, McArthur, McDonald, Cumming, & Watt, 2006); monitoramento de transformador (Khamphanchai, Pipattanasomporn, Kuzlu, Zhang, & Rahman, 2015); controle de microrrede (Ansari, Gholami, & Kazemi, 2015); controle geração distribuída (Elkhatib, El-Shatshat, & Salama, 2011); recomposição automática de sistemas elétricos (Jayasinghe & Hemapala, 2015); proteção adaptativa (Do Nascimento & Rolim, 2013); qualidade de energia (Dominguez, Cerqueira, Dominguez, Frias, & Iglesias, 2015); automação residencial e industrial (Ruta, Scioscia, Loseto, & Di Sciascio, 2014), etc.

Neste artigo é apresentado um SMRA, desenvolvido em JADE que é uma plataforma de agentes desenvolvida em linguagem de programação Java. O SMRA foi implantado em quatro computadores do laboratório do Grupo de Redes Elétricas Inteligentes (GREI) da Universidade Federal do Ceará (UFC), integrados em rede local via comunicação sem fio, utilizando o protocolo de comunicação TCP/IP. Um simulador da rede elétrica de distribuição foi desenvolvido em também utilizando a linguagem de programação JAVA para testes e validação do SMRA.

O restante do artigo está organizado como se segue. Na Seção 2 são apresentadas as estratégias e técnicas para desenvolvimento de SMRA. Os conceitos de agentes e de SMA são apresentados na Seção 3. A Seção 4 descreve as funcionalidades do SMRA. O simulador para teste e validação do SMRA é apresentado na Seção 5. Nas Seções 6 e 7 são apresentados, respectivamente, os resultados das simulações e as conclusões do artigo.

2. Técnicas para Desenvolvimento de Sistemas de Recomposição Automático

Nas últimas décadas, pesquisadores do mundo inteiro vêm investigando diferentes métodos para desenvolvimento e implantação de Sistemas de Recomposição Automática (SRA) (Sudhakar & Srinivas, 2011). SRA é um aplicativo de suporte à operação de redes elétricas, capaz de detectar distúrbios, isolar os segmentos sem falta, reconfigurar a rede e restabelecer com segurança o fornecimento, segundo critérios de prioridade pré-definidos pela concessionária. Os benefícios proporcionados pelo SRA são: recomposição automática da rede; redução de custos de operação e manutenção para deslocamento de equipes a campo; redução do tempo de interrupção de energia; e

melhoria dos indicadores de qualidade de serviço, de satisfação do cliente e da imagem da concessionária de energia (Sampaio et al., 2012).

Existem várias técnicas para desenvolvimento de um SRA. As diferentes técnicas podem ser classificadas em doze grupos, a saber: base de conhecimento, sistemas especialistas, busca heurística, lógica fuzzy, otimização determinística, inteligência artificial, redes de Petri, algoritmos genéticos, algoritmo de busca por colônia de formigas, pesquisa tabu, redes neurais artificiais e modelos híbridos (Sudhakar & Srinivas, 2011). Praticamente todos esses métodos utilizam abordagem centralizada. Os SMA vêm sendo destacados como uma das técnicas mais promissoras para desenvolvimento da função recomposição automática de redes elétricas (Jayasinghe & Hemapala, 2015).

3. Sistemas Multiagentes

3.1. Definição e Características dos Agentes

Agentes são entidades, em hardware ou software, com alto nível de abstração, residentes em um determinado ambiente, com capacidade de interpretar dados e executar, de forma autônoma, ações que alteram o estado deste ambiente, caracterizados por seu comportamento e ontologia (FIPA, 2015). O ambiente compreende tudo que é externo aos agentes e que pode ser observado e alterado por eles. O ambiente pode ser físico, como por exemplo, o sistema elétrico (monitorado através de sensores) ou computacional como, por exemplo, base de dados e recursos computacionais. O agente pode alterar o ambiente sistema elétrico através de ações físicas como abrir e fechar equipamentos ou de ações não físicas como, por exemplo, o armazenamento de informações de diagnósticos em base de dados para outros acessos (Wooldridge, 2009).

Os agentes apresentam características como autonomia, reatividade, proatividade e sociabilidade (Wooldridge, 2009). A autonomia consiste na capacidade de o agente ter controle sobre suas ações, não dependendo da intervenção direta de humanos ou de outros agentes. A reatividade é a capacidade do agente, em tempo hábil, reagir a mudanças em seu ambiente e tomar decisão com base nessas mudanças e na função para a qual foi projetado. A proatividade é a capacidade do agente de mudar seu comportamento dinamicamente para alcançar suas metas e atingir seus objetivos, independente de estímulos externos. A habilidade social atribui ao agente a capacidade de interagir, cooperar e negociar uns com os outros a fim de atingir metas individuais e satisfazer um objetivo comum ou concorrente que sirva aos seus próprios interesses, satisfazendo o objetivo do SMA como um todo (Wooldridge, 2009). Para aplicação na engenharia elétrica, os agentes podem ser encapsulados, construídos em arquitetura aberta, distribuída e tolerante a falhas, consequentemente, flexível e expansível (Wooldridge, 2009). A flexibilidade refere-se à capacidade de um agente analisar várias alternativas possíveis para realizar determinada tarefa e escolher a alternativa mais adequada. A expansibilidade consiste na capacidade do sistema ou agente permitir a adição de novas funcionalidades ou atualização de funcionalidade existente, sem necessidade de novos desenvolvimentos. Enquanto sistema tolerante à falha, o SMA tem a capacidade de atingir as metas para o qual foi projetado, mesmo se uma parte do sistema falhar (Wooldridge, 2009).

3.2. Padronização dos Agentes

A FIPA é uma organização internacional vinculada à sociedade de computação do IEEE que define especificações para padronização de comunicação, gerenciamento e arquitetura de agentes. O padrão FIPA visa garantir a interoperabilidade entre agentes, desenvolvidos por diferentes companhias e organizações, presentes em uma mesma plataforma ou em diferentes plataformas (FIPA, 2015).

O modelo de protocolo definido pela FIPA é orientado a serviços em que a camada de aplicação possui múltiplas subcamadas, ao invés de somente uma como nos modelos OSI ou TCP/IP. A FIPA definiu um conjunto de protocolos de iteração entre os agentes, a exemplo do FIPA-Request-Protocol, FIPA-ContractNet e FIPA-Subscribe, utilizados neste trabalho. Para comunicação entre agentes a FIPA especificou a linguagem FIPA-ACL (Agent Communication Language). A FIPA-ACL descreve os requisitos necessários à troca de mensagens entre os agentes e utiliza ontologias para que agentes entendam o conteúdo das mensagens repassadas por outros agentes de um determinado domínio de conhecimento (Bellifemine, Caire, & Greenwood, 2007). Se o domínio é recomposição da rede elétrica, por exemplo, faz-se necessário o entendimento comum sobre trecho, carga, potência, estado de equipamentos, função de proteção, prioridade de recomposição.

3.3. Arquitetura de Sistemas Multiagentes

A FIPA define um padrão de plataforma, na qual os agentes residem e operam, que permite a criação, localização, remoção e a comunicação de agentes. Cada plataforma de SMA padrão FIPA provê containers distribuídos, onde residem os agentes desenvolvidos (Classe Agente) e os agentes públicos, Agente Gerenciador de Serviço (AMS), obrigatório, e Diretório Facilitador (DF), opcional, que estão em um container principal. Cada agente registrado na plataforma deve ter uma única Identificação de Agente (AID) (FIPA, 2015). O agente AMS fornece a cada agente uma identificação - AID (Agent Identifier). O agente DF provê o serviço de páginas amarelas para outros agentes. O SMRA proposto não utiliza o agente DF.

4. Formulação do Problema e Estrutura do SMRA

4.1 Formulação do SMRA

O SMRA possui uma função objetivo que pode realizar a recomposição de forma otimizada a partir da meta a ser alcançada, como por exemplo a quantidade de carga, número máximo de clientes, ou até mesmo número de eletro-dependentes recompostos. A expressão (1) representa a função objetivo do sistema.

$$\max \sum_{i \in A} w_i \quad (1)$$

em que w_i pode assumir valores de potência de carga, número de clientes ou número de eletrodependentes em cada trecho presente no conjunto A que representa todos os trechos da rede. A função objetivo escolhida vai depender da configuração do SMRA.

Durante o processo de recomposição, o SMRA analisa se o aumento de carga em decorrência da nova configuração do sistema não causa sobrecarga nos transformadores da subestação, ou se o aumento da corrente de carga no alimentador não ultrapassa os limites térmicos dos condutores. Essas restrições operativas são verificadas por:

$S_{j \in T} < S_{max}$ (2)

$I_{i \in A} < I_{max}$ (3)

em que j ∈ T é o conjunto dos transformadores das subestações, sendo S a capacidade do transformador; e i ∈ A, conjunto de todos os trechos da rede, sendo I a corrente que circula no trecho i.

O SMRA, conforme mostrado no fluxograma da Seção 4.4, analisa as restrições operativas, verifica se a carga transferida devido à reconfiguração do sistema não causa sobrecarga no transformador da subestação e não excede a capacidade atual dos condutores.

4.2 Arquitetura de Comunicação dos Agentes no SMRA

O SMRA proposto foi concebido com quatro tipos de agentes: agente subestação (AS), agente alimentador (AA), agente trecho (AT) e agente equipamento (AE), distribuídos em diferentes containers na plataforma JADE. Após a ocorrência de uma falta permanente, os agentes AS, AA, AE e AT, distribuídos ao longo da rede elétrica, cooperam entre si, compartilhando informações, negociando potência e tomando decisões com o objetivo de recompor a rede elétrica. Nesse processo, os agentes realizam as seguintes tarefas/funções: localizar a falta; identificar e isolar os trechos afetados pela falta; negociar potência de reserva; analisar as restrições operativas; e recompor trechos sãos. O diagrama de blocos apresentado na Figura1 ilustra a interação entre os diferentes tipos de agentes presentes no SMRA. O arquivo de dados XML (eXtensible Markup Language) é explicado a seguir.

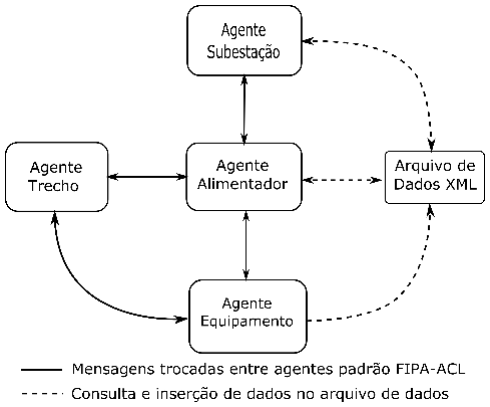


Figura 1. Diagrama de blocos representando comunicação entre agentes e com acesso ao arquivo de dados XML.

A base de dados XML é convertida em objeto JAVA, através da API JDOM, e transmitida como parâmetro de mensagem para outros agentes. Dessa forma a visão que cada agente tem da parcela do sistema sob sua supervisão é compartilhada entre os agentes interessados em atuar no ambiente mediante ocorrência de evento relacionado à sua área de atuação, implementando assim um comportamento de banco de dados distribuído.

Os AE atualizam a base de dados periodicamente, enquanto os AA consultam e modificam a base de dados XML no momento em que estão realizando negociação para recomposição da rede elétrica. Na Tabela 1 são apresentados os dados do arquivo XML acessados pelos agentes durante a recomposição do sistema de distribuição.

Tabela 1. Dados Armazenados pelos Agentes em Arquivo XML

Agentes	Dados/Atributos
Alimentadores	Nome, Estado e Recurso.
Trechos	Nome, Carregamento máximo e padrão, Número de Clientes, Número de Eletro-dependentes (consumidores com *home care* que dependem da energia elétrica para sobreviver) e Número de Cargas Prioritárias.
Equipamentos	Nome, Código, Estado, Porta de Comunicação.

4.4 Descrição e Modelagem dos Agentes do SMRA

O fluxograma na Figura 2 mostra a sequência de ações dos agentes do SMRA para recomposição automática de uma rede elétrica após uma falta.

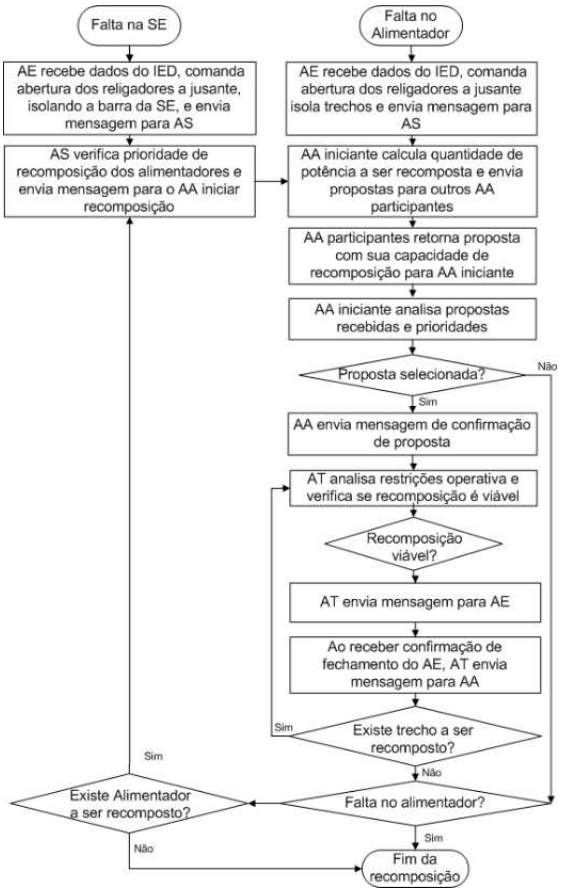


Figura 2. Sequência de ações do SMRA para recomposição da rede elétrica

5. Sistema Teste e Simulador de Sistema Elétrico para Validação do SMRA

5.1 Sistema Teste de Distribuição em Média Tensão

O SMRA foi testado e validado tomando como base o Sistema de Distribuição de Média Tensão (SDMT) da cidade de Aquiraz, no estado do Ceará, Brasil. A rede em 13,8 kV é suprida por 4 subestações: Aquiraz (SE AQZ), Jabuti (SE JAB), Messejana (SE MSJ) e Água Fria (SE AGF). Na Figura 3 é apresentado o diagrama unifilar do SDMT Aquiraz em que estão respresentados elementos como barramentos de 69,0kV e 13,8kV, trechos

de cada um dos alimentadores, chaves de seccionamento NF e de encontro NA e transformadores de força.

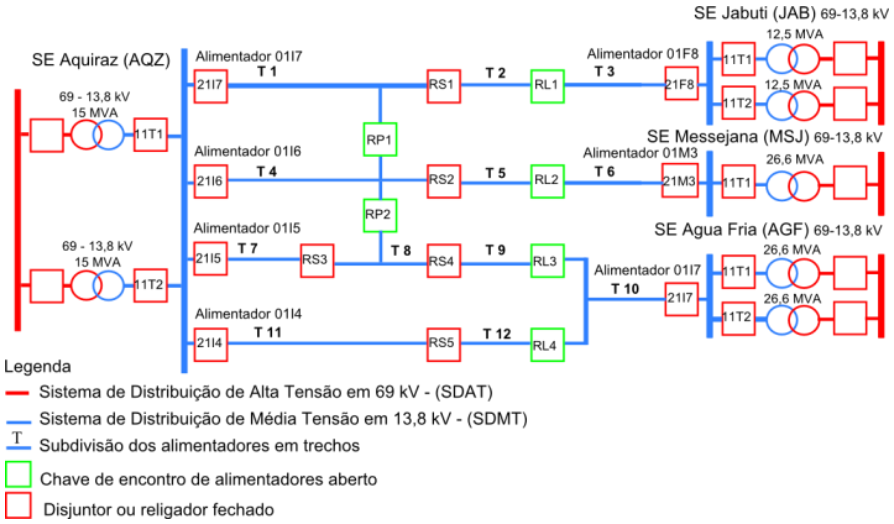


Figura 3. Topologia do SMDT de Aquiraz

A SE AQZ possui quatro saídas de alimentadores em 13,8 kV (01I4, 01I5, 01I6 e 01I7). Na ocorrência de uma falta permanente em algum trecho do SDMT Aquiraz, o SMRA possui a função de localizar o trecho afetado pela falta, isola-lo e recompor os demais trechos sãos a partir de outro alimentador da mesma subestação, em primeira instância, ou através de alimentadores das demais subestações em segunda instância, atendendo às restrições operacionais. Para viabilizar a recomposição automática, o SDMT Aquiraz possui nove religadores (21I4, 21I5, 21I6, 21I7, RS1, RS2, RS3, RS4 e RS5) para proteção e isolamento das regiões afetadas pelas faltas permanentes, e seis chaves de encontro de alimentadores (RP1, RP2, RL1, RL2, RL3 e RL4) utilizadas para transferência de carga entre alimentadores. Os alimentadores foram subdivididos em 12 trechos (T1-T12). Na Tabela 2 são apresentados os dados das correntes em cada trecho do alimentador indicando valores consumidos internamente e valores de corrente passante (valores essenciais para o cálculo das restrições de recomposição), também são mostrados quantidade de consumidores dos alimentadores e limites de corrente dos condutores.

Tabela 2. Dados das Correntes dos Trechos e dos Condutores

Trecho	Corrente Medida	Corrente Total do Trecho	Limite Térmico dos Condutores	Número de Consumidores
T1	12 A	244 A	475 A	300
T2	232 A	232 A	475 A	250
T4	33 A	183 A	438 A	350
T5	150 A	150 A	232 A	400
T7	54 A	331 A	475 A	350
T8	120 A	277 A	475 A	300
T9	157 A	157 A	242 A	250
T3	165 A	165 A	475 A	200
T6	263 A	263 A	438 A	300
T10	343 A	343 A	525 A	350
T11	164 A	203 A	475 A	200
T12	39 A	39 A	438 A	250

Os dados apresentados na Tabela 2, armazenados em arquivo XML, são acessados pelo SMRA durante o processo de recomposição automática da rede elétrica. A corrente medida corresponde à carga consumida em cada trecho. A corrente que circula no trecho corresponde à corrente total medida no início do trecho.

5.2 Desenvolvimento do Simulador e Integração ao SMRA

Para teste e validação do SMRA, foi desenvolvido em JAVA um simulador com a representação do SDMT Aquiraz. A integração do SMRA ao simulador foi implementada via porta de comunicação TCP/IP, através de bibliotecas do JAVA. A função principal do simulador é executada com o lançamento dos agentes no ambiente de execução JADE. Após lançados via simulador, os agentes são ativados, e o SMRA fica em modo de execução no ambiente JADE, pronto para isolar faltas e recompor o sistema. Na **Error! Reference source not found.** é apresentado o SMRA integrado ao simulador via comunicação TCP/IP e ao banco de dados XML por meio de uma biblioteca do JAVA denominada JDOM.

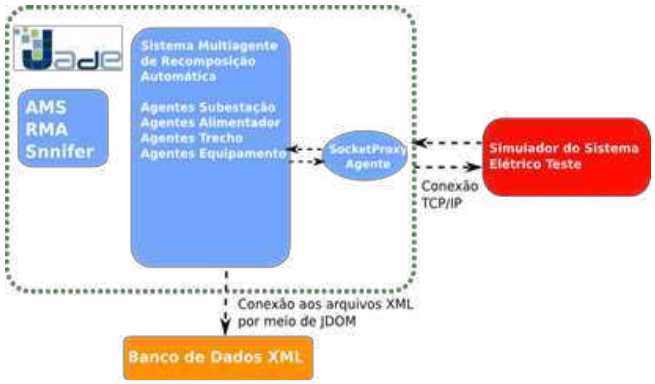


Figura 4. Arquitetura do SMRA integrada ao simulador de teste e validação

6. SMRA do SDMT Aquiraz

Nesta seção é apresentado o SMRA implantado, testado e validado em quatro computadores no Laboratório do GREI -UF, juntamente com os as simulações de falta e resultados obtidos.

6.1 Topologia do SMRA

A recomposição automática do SDMT Aquiraz é analisada, processada e controlada por meio de uma arquitetura computacional composta de um conjunto de agentes, distribuídos em quatro computadores conectados em rede. Nesta topologia, cada computador representa uma subestação com container composto de agentes AS, AA, AT e AE, configurando uma única plataforma. O computador que representa a SE Aquiraz, container principal, possui, além dos agentes supracitados, o Agente AMS responsável pelo gerenciamento de todos os agentes do SMRA e o simulador da rede elétrica.

Para realizar a recomposição automática, o SMRA analisa o carregamento dos condutores de cada trecho e da fonte (subestação) para evitar que o fechamento de uma chave de

encontro de alimentadores implique na violação do limite de carregamento dos condutores de trechos e da fonte. Durante o fechamento de chave de encontro de alimentadores, a característica radial da rede deve ser preservada.

6.2 Estudo de Caso

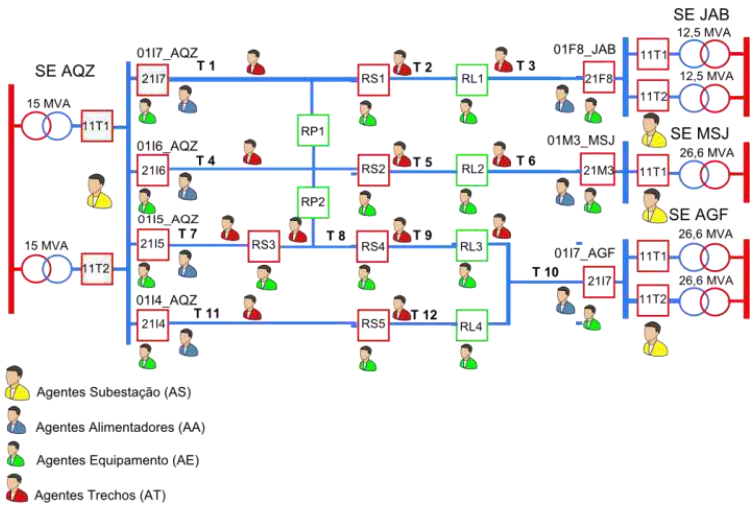
Ao simular uma falta, o simulador envia mensagens com os dados relacionados à falta para o SMRA. O SMRA analisa os dados recebidos e os agentes cooperam entre si com o objetivo de isolar o trecho afetado pela falta, comandando a abertura do religador a jusante à falta para restabelecer o fornecimento dos trechos sãos desenergizados, a partir do fechamento de equipamentos de encontros de alimentadores. Após análise das restrições operativas e das prioridades estabelecidas, o SMRA envia mensagens de comandos para o simulador abrir e fechar equipamento, visando o isolamento da área afetada pela falta e a transferência de cargas para outro alimentador.

Na Figura 5 é apresentado os agentes distribuídos ao longo das subestações e do SDMT Aquiraz e os dados das correntes dos trechos, armazenados em arquivo XML, que são acessados pelo SMRA durante o processo de recomposição automática da rede elétrica. A corrente consumida por trecho corresponde à diferença das correntes medidas no início e no final do trecho. A corrente que circula no trecho corresponde à corrente total medida no início do trecho.

A metodologia adotada na concepção do SMRA proposto leva em consideração o requisito de extensibilidade, o que implica que este sistema pode ser facilmente expandido e aplicado a outras redes de distribuição de energia.

Para validar o SMRA, foram realizadas simulações de faltas em todos os trechos do SDMT Aquiraz. Nesta seção serão apresentadas as ações do SMRA, a partir da simulação de uma falta no trecho T7 do alimentador (AQZ01I5) da SE AQZ, o isolamento dos trechos afetados e a recomposição dos demais trechos.

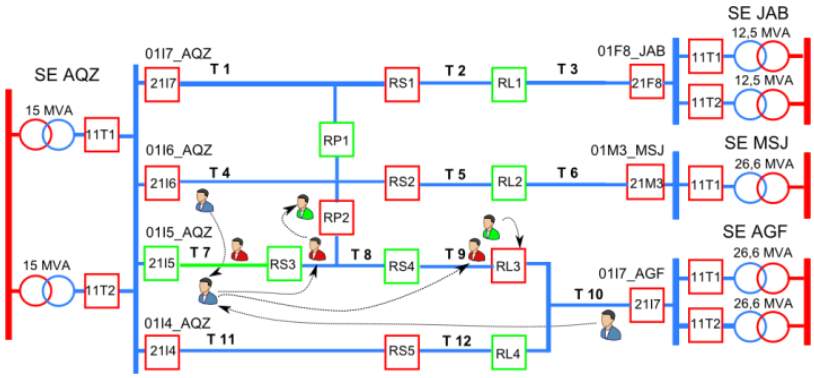
Figura 5 - SDMT Aquiraz no estado normal e agentes distribuídos observando o estado da rede.



Simulando a ocorrência de uma falta permanente no trecho T7, o relé associado ao religador AQZ21I5 atua, o religador abre desenergizando os trechos T7, T8 e T9 (cor

verde) (ver Figura 7). Os clientes em T8 e T9 são desenergizados embora não estejam em falta. Após a interrupção da falta, o SMRA inicia o processo de recomposição pelo agente AE (21I5_AQZ) que recebe do relé associado ao religador AQZ21I5 informações referentes à falta e as repassa ao agente AA (01I5_AQZ), conforme ilustrado na Figura 6. O agente AA 01I5_ AQZ imediatamente envia mensagens de comando de abertura para todos os agentes AE (RS3_01I5 e RS4_01I5) sob sua responsabilidade, realizando a abertura dos religadores RS3 e RS4 e ao mesmo tempo envia mensagens iniciando a negociação com o agente AA 01I6_AQZ da SE AQZ e com o agente AA 01I7_AGF da SE AGF, capazes de colaborar no processo de recomposição dos trechos sãos T8 e T9.

Figura 6 – Recomposição do trecho T8 pela SED Aquiraz



Para este processo de recomposição, definiu-se como primeiro critério de prioridade a transferência de carga entre alimentadores da mesma subestação. Tomando como base esse critério, o agente AA AQZ_01I5 analisa as propostas recebidas dos agentes AA AQZ_01I6 e AGF_01I3 e decide que o alimentador 01I6 da SE Aquiraz deve suprir o trecho T8 do SDMT Aquiraz. Em seguida, o agente AA AQZ_01I5 envia mensagens de autorização para o agente AT T8_01I5 iniciar o processo de recomposição. O agente AT T8_01I5 analisa as restrições operativas, e não encontrando qualquer problema, comanda o fechamento da chave de encontro de alimentadores RP2 (vermelho), suprindo o trecho T8 (azul), conforme ilustrado na Figura 6.

Após a recomposição do trecho T8, o AT T8_01I5 envia uma mensagem ao AA 01I5_AQZ informando que a recomposição foi realizada com sucesso. O agente AA 01I5_AQZ envia mensagem para o AT T9_01I5 autorizando a recomposição do trecho T9. Após análise, este agente constata que o alimentador 01I6_AQZ não possui capacidade para suprir o trecho T9. Conforme indicado na Tabela 2, o trecho T4 possui um fluxo de corrente de 183 A, e ao recompor T9 sua corrente elevar-se-ia para 303 A, elevando o fluxo de corrente no trecho T4 para 460 A, que está acima do limite de capacidade dos condutores do trecho que é de 438 A. Assim, 01I6 envia mensagem de falha na recomposição ao AA 01I5_AQZ. Este agente inicia nova negociação com o AA 01I7_AGF pertencente à subestação de Água Fria (SE AGF). O agente AA 01I7_AGF retorna mensagem para AA 01I5_AQZ informando disponibilidade de carga. O agente AA 01I5_AQZ autoriza o AT T9_01I5 analisar a recomposição do trecho T9, que não identifica restrições operativas e comanda o fechamento da chave de encontro de alimentadores RL3, ficando o trecho T9 suprido pela SE AGF, conforme ilustrado na Figura 6. Após a recomposição do trecho T9, o agente AT T9_01I5 envia uma mensagem para o agente AA informando que a recomposição foi realizada com sucesso e o agente

AA 01I5_AQZ verifica que não há outros trechos a serem reenergizados e finaliza o processo de recomposição.

7. Discussão de Resultados

Os resultados apresentados demonstram que o SMRA proposto foi capaz de localizar e isolar trecho submetido à falta permanente, negociar potência para recompor trechos sãos da rede elétrica de forma segura e eficaz, tomando como base a não violação das restrições operativas e atendendo prioridades pré-definidas, em um tempo máximo de três segundos. O trabalho apresenta as seguintes contribuições:

- Os agentes foram encapsulados em hardware independente da tecnologia do IED;
- Os agentes com inteligência local e distribuída foram capazes de analisar dados sistêmicos necessários à recomposição de sistemas complexos;
- Foi usada comunicação ponto-a-ponto bidirecional com o protocolo TCP/IP adotado pela norma IEC 61.850;
- Foi usado arquivo com estrutura de dados padrão XML, de fácil leitura e utilizado pela norma IEC 61.850 e pelo padrão CIM (Commom Information Model);
- Os agentes foram desenvolvidos em uma plataforma aberta, interoperável e multiplataforma, em conformidade com o padrão FIPA;
- Modularidade, escalabilidade, extensibilidade e tolerância à falha, características intrínsecas dos SMA, proporcionam ao SMRA maior facilidade de manutenção e flexibilidade para expansão das funcionalidades dos agentes e criação agentes;
- O SMRA possibilita a redução de custos operacionais com deslocamento de equipe para recomposição da rede elétrica;
- O tempo de recomposição da rede elétrica é reduzido, proporcionando melhoria na qualidade de serviço e melhoria da imagem da empresa.

8. Conclusão

Neste artigo foi apresentado um sistema multiagente para recomposição automática (SMRA) para redes elétricas de distribuição de energia, que usa uma abordagem de sistemas distribuídos. Os resultados dos testes em laboratório demonstraram a eficácia da comunicação entre o JADE e o simulador, e capacidade de interação, cooperação, negociação e análise de prioridades e restrições operativas dos agentes durante o processo de recomposição. Estes resultados comprovam o potencial dos SMA para desenvolvimento de sistemas com inteligência distribuída aplicados às redes elétricas inteligentes. Além disso, a pesquisa e desenvolvimento destaca a capacidade do JADE para aplicações de sistema para operação em tempo real.

Está em fase de pesquisa e desenvolvimento no GREI-UFC o desenvolvimento de novas funcionalidades para os agentes e a implementação de agentes inteligentes em hardware embarcados para integração do SMRA aos relés de proteção de sistemas elétricos.

Referências

Ansari, J., Gholami, A., & Kazemi, A. (2015). Holonic structure: a state-of-the-art control

architecture based on multi-agent systems for optimal reactive power dispatch in smart grids. *Iet Generation Transmission & Distribution*, *9*(14), 1922–1934.

Bellifemine, F., Caire, G., & Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. Chichester, UK: John Wiley & Sons, Ltd.

Davidson, E. M., McArthur, S. D. J., McDonald, J. R., Cumming, T., & Watt, I. (2006). Applying multi-agent system technology in practice: Automated management and analysis of SCADA and digital fault recorder data. *IEEE Transactions on Power Systems*, *21*(2), 559–567.

Do Nascimento, L. L., & Rolim, J. G. (2013). Multi-Agent system for adaptive protection in microgrids. *2013 IEEE PES Conference on Innovative Smart Grid Technologies, ISGT LA 2013*.

Dominguez, J. S., Cerqueira, A. J., Dominguez, D. S., Frias, D., & Iglesias, S. M. (2015). Using a multi-agent system for monitoring indicators of quality of service in power distribution networks. *IEEE Latin America Transactions*, *13*(4), 1048–1054.

Elkhatib, M. E., El-Shatshat, R., & Salama, M. M. a. (2011). Novel coordinated voltage control for smart distribution networks with DG. *IEEE Transactions on Smart Grid*, *2*(4), 598–605.

FIPA. (2015). The Foundation for Intelligent Physical Agents standards. Retrieved January 1, 2015, from http://www.fipa.org

Jayasinghe, S. L., & Hemapala, K. T. M. U. (2015). Multi Agent Based Power Distribution System Restoration—A Literature Survey. *Energy and Power Engineering*, *7*(12), 557–569.

Khamphanchai, W., Pipattanasomporn, M., Kuzlu, M., Zhang, J., & Rahman, S. (2015). An Approach for Distribution Transformer Management With a Multiagent System. *IEEE Transactions on Smart Grid*, *6*(3), 1208–1218.

Ma, R., Chen, H. H., Huang, Y. R., & Meng, W. (2013). Smart grid communication: Its challenges and opportunities. *IEEE Transactions on Smart Grid*, *4*(1), 36–46.

Momoh, J. (2012). *Smart Grid: Fundamentals of Design and Analysis*. Wiley-IEEE Press.

Ruta, M., Scioscia, F., Loseto, G., & Di Sciascio, E. (2014). Semantic-based resource discovery and orchestration in Home and Building Automation: a multi-agent approach. *IEEE Transactions on Industrial Informatics*, *10*(1), 730–741.

Sampaio, R. F., Barros, J. V. C., Leão, R. P. S., Barroso, G. C., Araújo, R. M., Leão, R. P. S., … Sá, M. B. (2012). Metodologia para Desenvolvimento de um Sistema de Reposição Automática para um Projeto Piloto de Redes Inteligentes. In *XX Seminário Nacional de Distribuição de Energia Elétrica, XX SENDI, Rio de Janeiro, Brasil*.

Sudhakar, T. D., & Srinivas, K. N. (2011). Restoration of power network - a bibliographic survey. *European Transactions on Electrical Power*, *21*(1), 635–655.

Wooldridge, M. J. (2009). *An Introduction to MultiAgent Systems* (2nd Editio). Jonh Wiley & Sons Ltd.

Zidan, A., El-Saadany, E. F., & El Chaar, L. (2011). A cooperative agent-based architecture for self-healing distributed power systems. *2011 International Conference on Innovations in Information Technology, IIT 2011*, 100–105.

Algoritmo distribuído para autorrecuperação de *smart grids* utilizando um sistema multiagente reativo

Italo R. Campos[1], **Filipe Saraiva**[1]

[1]Instituto de Ciências Exatas e Naturais – Universidade Federal do Pará (UFPA)
Caixa Postal 479 – 66.075-110 – Belém – PA – Brazil

italo.ramon.campos@gmail.com, saraiva@ufpa.br

Abstract. *This paper presents a distributed algorithm for smart grids self-healing in distribution level, implemented by a reactive multiagent system. Through the multiagent system, is possible to coordinate the switches of the power system, so that it adapts to the changes in the grid. To validate the proposed algorithm, is used a test model with 15 nodes, which one of them is a source. The results are obtained via computational simulation and shown in this paper.*

Resumo. *Este artigo apresenta um algoritmo distribuído para autorrecuperação de redes elétricas de distribuição do tipo smart grids, implementado por um sistema multiagente reativo. Por meio do sistema multiagente, é possível coordenar as chaves do sistema elétrico, de modo que ele se adapte às mudanças que ocorrem na rede. Para validar o algoritmo proposto, utiliza-se um modelo de testes com 15 nós, sendo um deles uma fonte. Os resultados da pesquisa são obtidos mediante simulação computacional e apresentados neste artigo.*

1. Introdução

Na área dos Sistemas Elétricos de Potência (SEP), *smart grid* é um conceito que recebe cada vez maior atenção, uma vez que traz importantes funcionalidades sobre os sistemas elétricos tradicionais, tais como gerenciamento remoto, comunicação de via dupla, segurança, controle em tempo real, acesso à informação em tempo real, inclusão em larga escala de fontes distribuídas de energia, entre outros [Saraiva 2015]. Esses aspectos dos *smart grids* configuram o setor elétrico como um relevante campo de estudo, especialmente quando se fala em técnicas da computação.

Smart grid é o termo utilizado para designar o emprego de tecnologias de informação e comunicação aos sistemas elétricos, compreendendo todos os seus subsistemas [Larik and Mustafa 2015]. O resultado dessa combinação é a entrega de uma gama de funcionalidades, como as descritas acima. Além disso, esses sistemas podem ser estratificados em termos de camadas, definidas em camada básica, de comunicação e de aplicação [Jia et al. 2011].

No setor de distribuição, elenca-se abordagens na esfera da computação que giram em torno de técnicas de inteligência artificial e computacional em alto nível. Mais especificamente, essas abordagens atuam sobre a camada de aplicação dos *smart grids*, através da qual objetivam desenvolver métodos para prover funcionalidades específicas ao nível de distribuição dos SEP. A subárea de distribuição é uma das mais importantes dos

SEP devido ser a conexão entre os consumidores e o setor de transmissão [Jia et al. 2011], além de serem expostos à grandes variações de demanda e de ambiente.

No que diz respeito ao nível de distribuição, os *smart grids* assumem um papel de referência para o desenvolvimento do futuro dos *grids*. A expansão dessa área figura como grande aliada do desenvolvimento social, uma vez que, de alguma forma, o desenvolvimento do setor elétrico está vinculado ao desenvolvimento da sociedade. Prover novas funcionalidades aos *grids* tradicionais significa aproximar o consumidor final das informações relativas à rede – como o consumo –, assim como melhorar os canais de comunicação entre as prestadoras de serviço e seus clientes. Além disso, os *smart grids* são apontados como "aquecedores" do mercado elétrico, tornando-o campo de interesse por organizações que atuam na área de compra e venda dinâmica de ativos, no caso, de energia elétrica [Siano 2014]; [Motamedi et al. 2012]; [Rahimi and Ipakchi 2010].

Uma das principais funcionalidades dos *smart grids* é a autorrecuperação, e existem muitos autores que realizam pesquisas para abordar o problema usando um vasto leque de métodos. De fato, muitos métodos computacionais podem ser aplicados para resolver problemas de *smart grids*, especialmente os métodos de aproximação, já que eles podem entregar uma solução aproximada em um tempo computacional satisfatório. Dessa forma, os métodos de inteligência artificial e computacional são largamente utilizados pela comunidade científica.

O trabalho de [Ferreira et al. 2013], por exemplo, modela o problema de autorrecuperação usando uma função objetivo de minimização, a qual é processada em um algoritmo genético. Em uma outra abordagem, [Mahdi and Genc 2019] utilizam um método de ilhamento de diferentes conjuntos de nós da rede elétrica para realizar a autorrecuperação. A escolha das ilhas é realizada por meio de diversos algoritmos, dentre eles o algoritmo de agrupamento *K-means* e um algoritmo de lógica *Fuzzy*. Por outro lado, utilizando abordagens distribuídas, os trabalhos de [Souza 2015], [Sharma et al. 2018] e [Wang et al. 2016] relatam como é possível atingir a autorrecuperação através da coordenação mútua entre agentes inteligentes no sistema.

No campo das abordagens distribuídas, encontra-se o uso da técnica de sistemas multiagente, que consiste na organização de múltiplos agentes inteligentes, envolvidos em um ambiente, os quais se coordenam para atingir seus objetivos individuais e/ou coletivos [Bellifemine et al. 2007]. [Saraiva 2015] utiliza sistemas multiagente para implementar e simular uma série de funcionalidades dos *smart grids*, a partir das quais é possível validar que sistemas multiagente são ferramentas interessantes para o uso na simulação e implementação de *smart grids*.

Neste artigo, é apresentado um algoritmo distribuído para realizar a autorrecuperação de *smart grids* no nível de distribuição. A metodologia deste trabalho inclui a utilização de um sistema multiagente para simular redes elétricas de distribuição, bem como as operações que elas realizam para que a autorrecuperação seja efetuada. Este trabalho é a validação e melhoria do algoritmo apresentado em uma publicação anterior [Campos and Saraiva 2018].

Por meio de simulação computacional, o trabalho objetiva validar o algoritmo de autorrecuperação, inserindo-o na literatura como uma alternativa para implementação dessa funcionalidade em *smart grids*. Para validá-lo, foi desenvolvido um modelo de

testes de rede elétrica de distribuição com 15 nós, utilizando como referência o modelo de testes da IEEE com 33 nós [Baran and Wu 1989]. Os resultados foram coletados mediante *logs* escritos pelos agentes do sistema e, posteriormente, analisados.

O artigo é estruturado da seguinte forma: a presente seção apresentou o contexto, o objetivo, a metodologia e breves conceitos relacionados à *smart grids* e autorrecuperação; a Seção 2 descreve o problema abordado pela pesquisa e esclarece detalhes sobre as restrições do sistema elétrico levadas em conta pela pesquisa; posteriormente, a Seção 3 detalha o algoritmo de autorrecuperação e o sistema multiagente modelado; a seguir a Seção 4 apresenta como se dá o processo de simulação, bem como os resultados obtidos, além das respectivas discussões; por fim, a Seção 5 finaliza o trabalho, mostrando as conclusões da pesquisa.

2. Descrição do problema

A autorrecuperação pode ser definida como o restabelecimento automático de energia elétrica às cargas que foram afetadas por uma interrupção no seu abastecimento [Campos and Saraiva 2018]. O termo "automático" está relacionado à realização ativa de operações pela própria rede elétrica, sem a intervenção de um operador humano.

Quando se fala em redes de distribuição, existe uma série de restrições que precisam ser obedecidas para que não haja problemas de operação. Uma dessas restrições é o *princípio da radialidade*, que diz que as estruturas de redes de distribuição não podem conter ciclos, uma vez que, se ocorridos, podem causar um colapso em grandes proporções em toda a estrutura do *grid*. Nesta etapa do trabalho considera-se apenas o princípio da radialidade durante a execução do algoritmo de autorrecuperação.

Em termos de definição, tem-se que o conjunto de linhas ativas de uma rede elétrica de distribuição pode ser descrito como um grafo G, onde todo passeio p em G é definido por

$$p = \{v_1, v_2, ..., v_k, v_{k+1}\},\ \forall v_1 \neq v_{k+1},\ \text{tal que } k > 0. \tag{1}$$

É pertinente salientar que as linhas ativas de uma rede de distribuição compreendem no conjunto de todas as linhas que estão conduzindo corrente elétrica ao longo da rede em um dado momento, isto é, que possuem uma chave elétrica fechando o circuito. Uma linha que não esteja, em um determinado momento, fazendo esse trabalho, não é uma linha ativa.

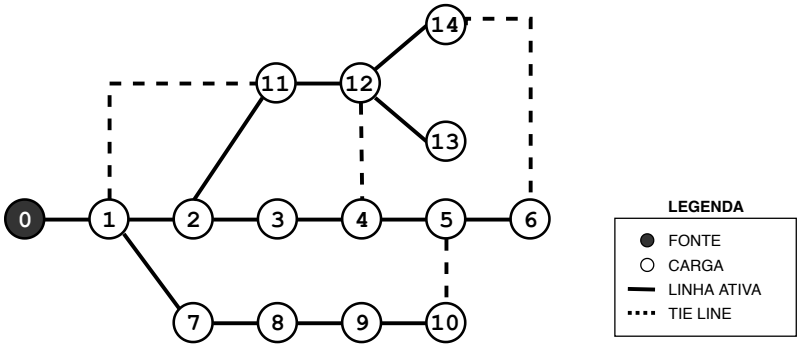


Figura 1. Topologia do modelo de testes de 15 nós.

Para realizar os estudos com o algoritmo desenvolvido por esta pesquisa, foi necessário utilizar um modelo de testes de rede elétrica de distribuição, para servir como suporte para as simulações e para as operações que o sistema multiagente realiza. Esta pesquisa, então, desenvolveu um modelo de testes com 15 nós, sendo um deles uma fonte, a partir da qual inicia o fluxo de corrente elétrica para os demais nós da rede. A topologia e os parâmetros utilizados no modelo foram aferidos com base no modelo de testes de 33 nós da IEEE, bem como nos valores de demanda, tensão, resistências das linhas, entre outros, definidos em [Baran and Wu 1989]. A Figura 1 traz a topologia do modelo de testes de 15 nós, ao passo que a Tabela 1 apresenta os os parâmetros do modelo de testes elaborado.

			Br. Parameters		Sn. Nd. Parameters	
Br. No	Rc. Nd.	Sn. Nd.	r(Ω)	x(Ω)	PL(W)	QL(var)
1	0	1	0.0922	0.047	100000	60000
2	1	2	0.493	0.2511	90000	40000
3	2	3	0.366	0.1864	120000	80000
4	3	4	0.3811	0.1941	60000	30000
5	4	5	0.819	0.707	60000	20000
6	5	6	0.1872	0.6188	200000	100000
7	1	7	0.164	0.1565	90000	40000
8	7	8	0.409	0.479	90000	40000
9	8	9	1.5042	1.3554	90000	40000
10	9	10	0.4095	0.4784	90000	50000
11	2	11	0.4512	0.3083	420000	200000
12	11	12	0.898	0.7091	420000	200000
13	12	13	0.896	0.7011	60000	25000
14	12	14	0.2842	0.1447	60000	25000
			Tie lines			
15	1	11	1.5	1.5		
16	4	12	1	1		
17	5	10	0.5	0.5		
18	6	14	0.5	0.5		

Tabela 1. Parâmetros de rede do modelo de teste de 15 nós.

Observando a Figura 1, conta-se 14 linhas ativas e 4 *tie lines*. *Tie lines* (ou linhas de reserva), são linhas de distribuição presentes na rede com chave aberta, isto é, não conduzem corrente elétrica em determinado momento. O propósito dessas linhas é serem usadas como alternativas para religar nós que estejam afetados por uma eventual interrupção no abastecimento elétrico. Entende-se, portanto, que o objetivo do sistema multiagente é, depois de detectar uma falha no fornecimento de energia, reconfigurar a rede usando as *tie lines*, de forma que seja restabelecido o suprimento de energia elétrica aos nós afetados, obedecendo à restrição anteriormente descrita.

3. O sistema multiagente e o algoritmo de autorrecuperação

A presente abordagem utiliza-se de sistemas multiagente, que é uma técnica distribuída cujo objetivo é modelar um problema em termos de agentes independentes, de modo que eles operem em conjunto para alcançar objetivos específicos [Bellifemine et al. 2007].

Pelo fato de o problema abordado apresentar um "gatilho" – que é a detecção da falta de energia –, preferiu-se empregar um sistema multiagente do tipo reativo.

[Bellifemine et al. 2007] apontam sistemas multiagente desse tipo como mais apropriados para utilizar em casos onde se espera uma obtenção rápida de respostas. Para a implementação dos agentes, utilizou-se o *JADE Framework* [JADE 2017], que proporciona uma gama de ferramentas para comunicação de agentes, além de ser muito usado para abordagens com sistemas multiagente reativo.

Em termos de modelagem, foram desenvolvidos dois tipos de agentes para lidar com o problema em análise nessa pesquisa: o Agente de Carga (ou *Load Agent*) e o Agente de Chave (ou *Switch Agent*).

Os *Agentes de Carga* (AC) são os que possuem maior quantidade de atribuições no sistema. Dentre elas pode-se destacar:

- Verificação periódica dos sensores dos nós da rede elétrica, monitorando o ambiente e detectando eventuais faltas;
- Envio e manutenção de dados atualizados relacionados ao seu próprio estado e aos estados de agentes vizinhos, respectivamente;
- Conexão a novos AC, em tempo de execução, buscando atender ao dinamismo da rede elétrica;
- Eleição e candidatura à posição de agente ativo do conjunto de nós em falta;

Os *Agentes de Chave* (SW), por sua vez, são agentes com atribuições bem específicas. São elas:

- Conexão de um par de AC em tempo de execução;
- Execução de operações de abertura e fechamento de chaves elétricas, de acordo com as mensagens recebidas dos AC;

Para cada nó (ou carga) da rede elétrica, existe um AC alocado, ao passo que para cada linha, incluindo *tie lines*, existe um SW alocado. Portanto, no modelo utilizado são instanciados 15 AC, 14 SW para as linhas ativas e outros 4 SW para *tie lines*.

Há uma característica do sistema multiagente implementado importante de ser enfatizada. Para manter a abordagem descentralizada, os AC são instanciados sem conhecer o endereço dos demais agentes do sistema. Para conectá-los entre si, os SW são instanciados tendo como parâmetros os endereços de um par determinado de AC, mediando a conexão entre eles. Como consequência desse fato, os AC possuem uma visão limitada do sistema, conhecendo apenas os AC vizinhos a eles, com os quais mantém comunicação. Essa característica foi nominada *visão mínima* do sistema.

3.1. O algoritmo distribuído

O algoritmo proposto por essa pesquisa possui quatro etapas básicas: isolamento, eleição de agente ativo, mapeamento e tomada de decisão. Cada uma dessas etapas é descrita nas próximas subseções. A partir deste ponto do trabalho, os termos “AC”, “nó” e “carga” serão usados indiscriminadamente, referindo-se a um determinado nó da rede elétrica que é controlado por um AC.

3.1.1. Isolamento

Ao ser detectada a falta no sistema, os AC afetados por ela iniciam o processo de isolamento. O processo de isolamento nada mais é do que a atualização de dados mútua entre os AC que foram afetados pela falta, com o objetivo de identificar o local afetado.

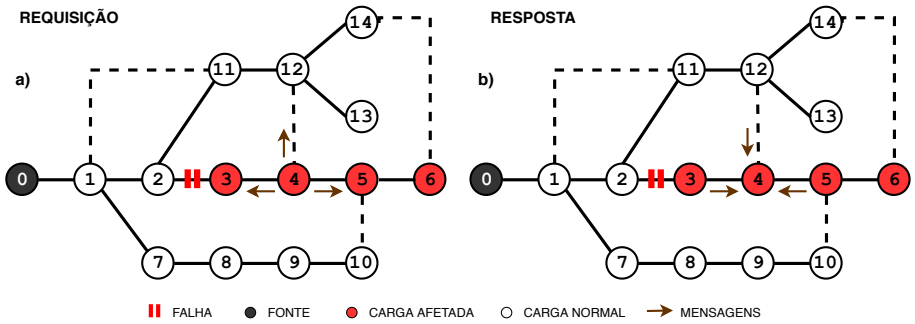


Figura 2. Passo de isolamento seguido pelo AC4.

A atualização dos nós é feita através da troca de mensagens. Os AC enviam mensagens a todos os seus vizinhos, buscando conhecer quais têm o fornecimento de energia normal e quais não. Essas mensagens seguem o protocolo requisição-resposta, isto é, o AC remetente faz uma requisição de atualização e o AC destinatário responde com uma mensagem, cujo conteúdo é um objeto com informações sobre o seu estado atual. Para uma melhor compreensão, será usado um exemplo com o modelo de testes, onde se tem uma falha na linha $(2, 3)$. Nesse contexto, a área afetada é composta pelos nós 3, 4, 5 e 6. Concentrando-se apenas no nó 4 desse conjunto, teremos um comportamento como o descrito na Figura 2.

3.1.2. Eleição de agente ativo

Terminada a etapa de isolamento, é necessário escolher um dos AC para realizar a tomada de decisão em nome da área afetada. Esse processo se torna importante porque caso dois ou mais AC realizem a tomada de decisão, corre-se o risco de entrarem em conflito e haver decisões que violem a restrição de radialidade da rede. Sendo assim, os AC da área afetada participam de um processo de eleição, com o objetivo de escolher um deles para realizar o papel de agente ativo (AA). Ao fim do processo, os demais AC que não forem eleitos como AA, tornam-se agentes passivos (AP), aguardando as orientações do AA eleito e cooperando nos processos posteriores do algoritmo.

Somente AC externos podem se candidatar ao papel de AA. Um AC externo é um AC que tem pelo menos uma *tie line* que o liga a um outro AC que está com o abastecimento energético normal. Por consequência, AC que não sejam externos tornam-se automaticamente AP nesta etapa do algoritmo.

Em termos de sistema, a eleição é feita utilizando um espaço para publicação de serviços de agentes, disponibilizado pelo JADE na forma de *páginas amarelas*. As páginas amarelas são controladas por um agente específico da plataforma, denominado *Directory Facilitator* (DF), que é responsável por receber as mensagens de registro, cancelamento e consulta de serviços disponibilizados na plataforma. Os agentes candidatos, então, se registram no DF e aguardam um *limiar de tempo* para que todos os outros agentes candidatos tenham tempo de se registrar. Esse processo é necessário devido a visão mínima dos AC.

Para esta pesquisa, foi definido um limiar no valor de 370 ms, através do método de tentativa e erro. Esse valor foi aferido depois de uma série de testes com diversos

modelos de rede no ambiente de simulação e nunca foi extrapolado em nenhuma das simulações. Entretanto, sabe-se que em situações mais próximas à realidade, esse tempo pode variar, especialmente devido às variações da rede de comunicação.

Depois do período de espera, cada AC candidato consulta novamente o DF da plataforma em busca dos dados dos possíveis outros AC candidatos. Nesse momento, aplica-se o método de eleição escolhido por esta pesquisa, que consiste no AC candidato com maior ID[1]. O AC que tiver o maior ID dos registrados no DF, será o AA da área afetada. Os demais AC candidatos tornam-se AP nesta etapa do algoritmo.

No exemplo da Figura 2, os agentes candidatos são os AC 4, 5 e 6, deixando o AC3 como AP. Ao final do processo de eleição, os AC 4 e 5 serão AP e o AC6 será o AA da área afetada.

3.1.3. Mapeamento da área afetada

Uma vez eleito o AA da área afetada, é necessário que ele conheça os endereços dos AC afetados, dos SW envolvidos na falta, das *tie line* disponíveis, entre outras informações. Isso é necessário pelo fato de o AA conhecer, num primeiro momento, apenas as informações de seus AC vizinhos. Para lidar com o problema, realiza-se um mapeamento na área afetada, utilizando um método baseado em busca recursiva de grafos.

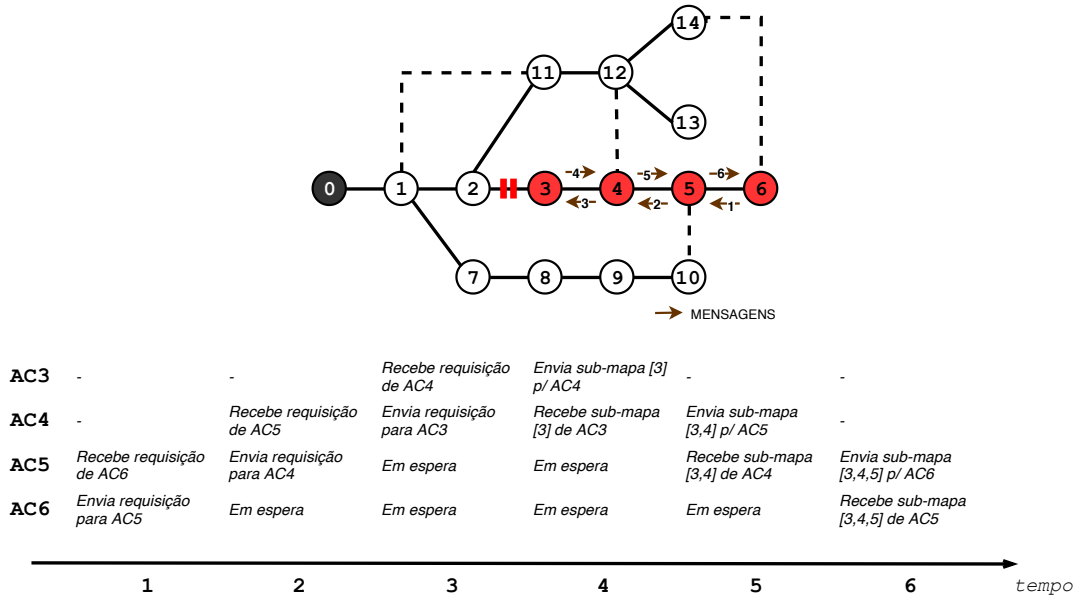


Figura 3. Passo de mapeamento realizado pelo agente ativo AC6.

Primeiramente, o AA envia uma requisição de mapeamento a todos os AC vizinhos afetados pela falta, esperando que eles respondam, por sua vez, com o sub-mapa correspondente a todos os seus vizinhos também afetados. Esse processo é repetido recursivamente em todos os vizinhos do AA até que chegue aos AC que não possuem vizinhos

[1]O ID de um AC é um número inteiro estabelecido para cada carga da rede elétrica, que é único entre todas as cargas do sistema

em falta, tempo quando todo o processo será revertido. Ao fim de todo o processo, o AA terá obtido o mapa com todos os endereços e informações da área afetada.

A etapa de mapeamento é ilustrada na Figura 3, que segue o exemplo tomado anteriormente. Observa-se que, no decorrer do tempo, há requisições recursivas ao longo da área afetada e espera dos AC que realizam a requisição, até que no tempo 4 as requisições começam a ser respondidas e o mapa vai sendo construído. É relevante notar que esse mapeamento é realizado através da troca coordenada de mensagens, que transportam requisições e estruturas complexas de dados da rede elétrica.

3.1.4. Tomada de decisão e religamento

Depois de mapeada a área afetada, o AA agora precisa realizar procedimentos para tomar decisão sobre como religar a área afetada à rede ativa, utilizando as *tie lines* disponíveis. Para isso, é utilizado o cálculo de fluxo de potência pelo do método de varredura de [Kersting 2012]. Esse cálculo permite obter informações cruciais sobre uma rede de distribuição a partir de um ponto por onde se inicia o fluxo de corrente elétrica.

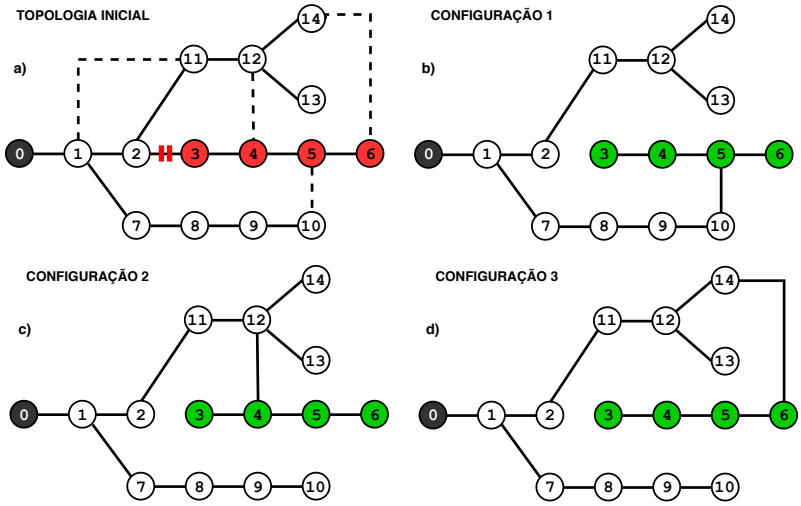


Figura 4. Configurações possíveis para a rede, para as quais o AA aplica o cálculo de fluxo de potência.

O critério de escolha de *tie lines* utilizado por esta pesquisa é o menor valor de perdas elétricas obtidas para determinada configuração de rede. Em outras palavras, entre as configurações possíveis para religar a área afetada, o AA deve escolher a configuração que resultar em menores perdas elétricas para o sistema.

Depois de realizar o cálculo do fluxo de potência para cada uma das configurações possíveis e obter seus respectivos valores de perdas elétricas, o AA escolhe a configuração mais interessante e comunica os SW responsáveis pelas chaves para que as operações de rede sejam executadas. Primeiramente o AA comunica ao SW da linha onde ocorreu a falha, solicitando a abertura da mesma e, posteriormente, solicita ao SW da *tie line* escolhida que feche o circuito, restabelecendo o suprimento de energia na área afetada.

4. Simulações e resultados

As simulações foram realizadas da seguinte maneira: para representar o sensor de leitura das cargas da rede, utilizou-se o sistema de arquivos do Sistema Operacional, onde cada carga possui um arquivo único que representa o estado do fornecimento de energia. Em termos dos AC, a verificação é realizada através de um método que retorna um valor *booleano*, conforme denota o Algoritmo 1.

Algorithm 1 Função de verificação de sensor dos AC.

```
1: procedure LER_SENSOR(file)
2:     if file.exists() then
3:         return True                          ▷ Fornecimento normal
4:     else
5:         return False                         ▷ Fornecimento interrompido
6:     end if
7: end procedure
```

Para analisar as operações que os agentes realizaram na rede e verificar se a energia foi de fato restabelecida à área afetada, analisou-se a criação desses *arquivos de energia* pelos AC, bem como os *logs* escritos por AC e SW. O ambiente computacional utilizado nas simulações tem a configuração conforme segue: Sistema Operacional Linux (*Arch*), RAM de 10,7 GB e CPU Intel Core i5-2310 4 $\times$ 2.9 GHz, com instruções 64 bits.

4.1. Resultados

Foram realizadas 10 simulações com o modelos de testes e os dados delas são apresentados na Tabela 2. Cada simulação parte da topologia inicial do modelo de testes, isto é, a simulação é iniciada, os agentes recuperam a rede, escrevem *logs* e, a seguir, são finalizados.

Simulação	Ch. aberta	Nós afetados	Ch. fechada	Tempo (ms)	Perdas (kW)
1	$(2, 3)$	4	$(6, 14)$	942	1,9493
2	$(1, 7)$	4	$(5, 10)$	862	1,8035
3	$(11, 12)$	3	$(6, 14)$	777	2,4197
4	$(2, 11)$	4	$(6, 14)$	769	9,6396
5	$(1, 2)$	9	$(1, 11)$	856	48,6029
6	$(1, 2)$	9	$(1, 11)$	802	48,6029
7	$(2, 3)$	4	$(5, 10)$	780	2,3636
8	$(2, 3)$	4	$(5, 10)$	858	5,5397
9	$(1, 2)$	9	$(5, 10)$	808	53,2739
10	$(2, 11)$	4	$(4, 12)$	852	19,6604

Tabela 2. Resultados das simulações com o modelo de testes.

Ressalta-se que, para as simulações de 7 a 10, foram utilizados valores de resistência e reatância diferentes dos definidos no modelo de testes original (vide Tabela 1). O objetivo com essas alterações é de verificar como o sistema multiagente se comporta quando submetido à mudanças na estrutura da rede elétrica. As alterações nos valores de resistência e as linhas que sofreram as alterações são especificadas na Tabela 3.

Em todas as simulações os agentes conseguiram executar corretamente as operações de reconfiguração de rede, realizando a recuperação dos nós afetados pela

falta. Pode-se observar que, nos dados da Tabela 2, obteve-se como maior tempo de autorrecuperação o valor de 942 ms, ao passo que para o menor tempo, o valor alcançado é de 769 ms. A média dos valores de tempo de autorrecuperação é calculado em 830,6 ms.

Simulação	Linha	Resistência original (Ω)	Resistência modificada (Ω)
7	(6,14)	0.5	2.0
8	{(6,14), (5,10)}	{0.5, 0.5}	{2.0, 1.5}
9	(1,11)	1.5	2.0
10	(6,14)	0.5	1.5

Tabela 3. Alterações nos valores de resistências de algumas linhas para as simulações de 7 a 10.

É possível observar ainda que, agrupando os dados da Tabela 2 pelo número de nós afetados, há a formação de três grupos: os grupos com 3, 4 e 9 nós afetados. A média de tempo de autorrecuperação para cada grupo é calculada em 777 ms, 843,8 ms e 822 ms, respectivamente.

4.2. Simulações 1 e 7

Nas simulações 1 e 7, observa-se um comportamento do sistema multiagente interessante de ser enfatizado. O cálculo de perdas elétricas considera diversas variáveis da rede, entre elas a resistência e reatância das linhas de distribuição.

Observando os dados da simulação 1 na Tabela 2, percebe-se que o sistema multiagente detecta a falha na linha $(2, 3)$ e utiliza a *tie line* $(6, 14)$ para restaurar o fornecimento de energia elétrica à área afetada, como mostra a Figura 5a.

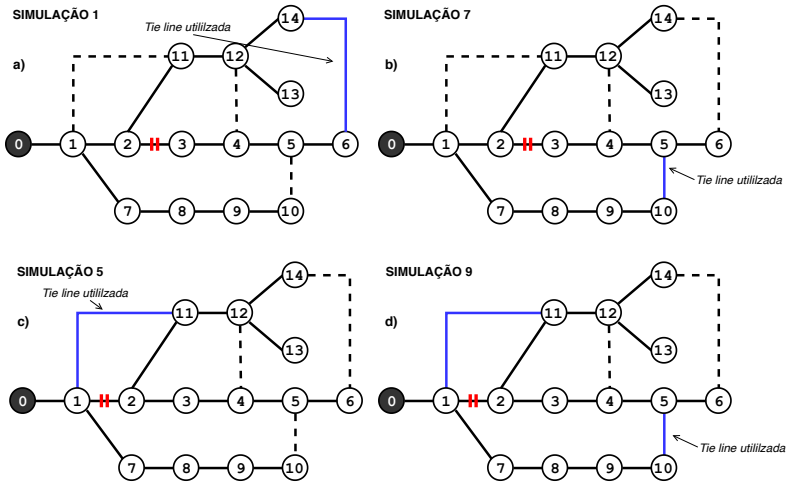


Figura 5. Topologia da rede para as simulações 1, 5, 7 e 9.

No entanto, na simulação 7, o valor de resistência é aumentado em 4 vezes na linha $(6, 14)$. Em face disso, para o mesmo ponto de falha que na simulação 1, o sistema multiagente escolhe dessa vez a linha $(5, 10)$ (Figura 5b). Essa mudança na escolha se justifica pelo fato de o valor de perdas elétricas ter aumentado em função do aumento da resistência na linha $(6, 14)$, efetuado nessa simulação. Observa-se também que o valor de perdas elétricas para a simulação 7 foi maior do que o obtido na simulação 1, razão pela qual a linha $(5, 10)$ não é escolhida na simulação 1.

4.3. Simulações 5 e 9

No caso das simulações 5 e 9, o processo é semelhante ao que ocorre nas simulações 1 e 7, respectivamente. Entretanto, nota-se que os valores de perdas elétricas são muito maiores para as simulações 5 e 9. Isso ocorre porque o número de nós envolvidos na falta, que é 9, é consideravelmente maior que nas simulações 1 e 7. Esse fator, assim como a resistência das linhas, impacta no cálculo de fluxo de potência e, consequentemente, na decisão dos AC do sistema.

5. Conclusões e trabalhos futuros

Este trabalho apresentou um algoritmo distribuído, implementado por um sistema multiagente reativo, que segue quatro etapas básicas para reconfigurar a rede e realizar a autorrecuperação de *smart grids* no nível de distribuição. Por meio de simulação computacional, foi possível validar o algoritmo proposto nesta pesquisa, bem como a sua implementação utilizando sistemas multiagente. A pesquisa também apresentou resultados de 10 simulações, realizadas em ambiente computacional, para um modelo de testes de rede de distribuição com 15 nós, sendo um deles uma fonte.

Uma característica importante do algoritmo implementado é que somente os nós envolvidos na falta, de fato, executam operações para a autorrecuperação da rede. Isso permite que o sistema consiga concentrar a carga de trabalho somente na região do *grid* onde ocorreu a falta, deixando os demais nós livres.

Uma segunda característica, é que o algoritmo realiza, somente duas operações de chaveamento. Isso porque a etapa de tomada de decisão é feita por meio do cálculo de fluxo de potência. Operações de chaveamento são custosas à rede, portanto é de grande interesse minimizá-las durante abordagens de reconfiguração da rede.

Como pontos sensíveis do sistema, é possível concluir que o algoritmo não lida com todo tipo de alteração que pode ocorrer na rede, como, por exemplo, nos casos em que a energia volta à área afetada antes do término do algoritmo. Além disso, nota-se que o método de eleição do agente ativo não é robusto contra atrasos que possam ocorrer na rede de comunicação, já que se baseia em um limiar de tempo determinado por tentativa e erro.

Atualmente, os autores estão trabalhando em melhorar o método de eleição de agentes ativos no sistema, baseando-o em uma outra técnica que não depende de um tempo de espera. Em trabalhos futuros, espera-se também ampliar as restrições consideradas pelo algoritmo para outras variáveis do sistema elétrico de distribuição, como queda de tensão, prioridade de cargas, análise de demanda, entre outras. Para resolver algumas dessas questões, pode-se usar técnicas como balanceamento de carga, corte seletivo de cargas e ilhamento com geração distribuída, métodos já utilizados em outros trabalhos e que podem ser incorporados ao algoritmo proposto.

Referências

Baran, M. E. and Wu, F. F. (1989). Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Transaction on Power Delivery*, 4(2).

Bellifemine, F., Caire, G., and Greenwood, D. (2007). *Developing multiagent systems with JADE*. Jhon Wiley & Sons, Liverpool.

Campos, I. R. C. and Saraiva, F. (2018). Proposta de modelo de autorrecuperação de sistemas de distribuição de energia elétrica utilizando sistemas multiagente. In *12th Workshop-School on agents, environments and applcations*, Fortaleza. 12th Workshop-School on agents, environments and applcations.

Ferreira, L. R., Siebert, L. C., Ayala, H., and A. R. Aoki, L. C. D. (2013). Solução do problema de self-healing para redes de distribuição radiais através de otimização via algoritmo genético. In *Simpósio Brasileiro de Automação Inteligente 2013, UFC*, Fortaleza. Simpósio Brasileiro de Automação Inteligente 2013.

JADE (2017). Java agent development framework (version 4.5.0). `http://jade.tilab.com/`.

Jia, D., Meng, X., and Song, X. (2011). Study on technology system of self-healing control in smart distribution grid. In *China Electric Power Research Institute*, Beijing. The International Conference on Advanced Power System Automation and Protection.

Kersting, W. (2012). *Distribution System Modeling and Analysis*. CRC Press, Boca Raton.

Larik, R. M. and Mustafa, M. W. (2015). Technologies used in smart grid to implement power distribution system. *TELKOMNIKA Indonesian journal of electrical engineering*, 16(2):232–237.

Mahdi, M. and Genc, V. M. I. (2019). A real-time self-healing methodology using model- and measurement-based islanding algorithms. *IEEE Transactions on Smart Grid*, 10(2):1195–1204.

Motamedi, A., Zareipour, H., and Rosehart, W. D. (2012). Electricity price and demand forecasting in smart grids. *IEEE Transactions on Smart Grid*, 3(2):664–674.

Rahimi, F. and Ipakchi, A. (2010). Demand response as a market resource under the smart grid paradigm. *IEEE Transactions on Smart Grid*, 1(1):82–88.

Saraiva, F. O. (2015). *Aplicações híbridas entre sistemas multiagentes e técnicas de inteligência artificial para redes inteligentes de distribuição de energia elétrica*. PhD thesis, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

Sharma, A., Srinivasan, D., and Trivedi, A. (2018). A decentralized multi-agent approach for service restoration in uncertain environment. *IEEE Transactions on Smart Grid*, 9(4):3394–3405.

Siano, P. (2014). Demand response and smart grids—a survey. *Renewable and Sustainable Energy Reviews*, 30:461 – 478.

Souza, F. A. (2015). Modelo baseado em sistema multiagente para autorrecuperação com corte seletivo de carga e ilhamento com geração distribuída para redes elétricas inteligentes. Dissertação, Departamento de Engenharia Elétrica, Universidade Federal do Paraná, Curitiba.

Wang, Z., Chen, B., Wang, J., and Chen, C. (2016). Networked microgrids dor self-healing power systems. *IEEE Transaction on Smart Grids*, 7(1):310–319.

Mosaik e PADE: Sistemas Multiagentes e Co-Simulação para Modelagem de Redes Elétricas Inteligentes

Lucas S Melo[1], Filipe Saraiva[2], Ruth P. S. Leão[1], Raimundo F. Sampaio[1], Giovanni C. Barroso[1]

[1]Departamento de Engenharia Elétrica – Universidade Federal do Ceará (UFC)
Caixa Postal 6.001 – Campus do Pici – Fortaleza – CE – Brazil

[2]Faculdade de Computação – Instituto de Ciências Exatas e Naturais
Universidade Federal do Pará (UFPA) – Belém – PA – Brazil

{lucassmelo,rleao,rfurtado}@dee.ufc.br

saraiva@ufpa.br, gcb@fisica.ufc.br

Abstract. *This paper describes the integration process between two tools in order to perform co-simulation for representation and analysis of dynamic environments in the context of smart grids. The integrated tools are Mosaik, a software to co-simulation management, and PADE, a software to multi-agent systems development. As a study case for demonstrate the integration, a scenary was utilized composed of a low voltage electricity distribution grid with 37 load bus, 20 photo-voltaic distributed generations, randomly connected to load bus, as well as, 20 PADE agents associated to distributed generation, modeling the behavior of electricity storage systems. The simulation results show the integration happening and demonstrate how useful is to model the dynamics of distributed electric resources with multi-agent systems.*

Resumo. *Este artigo descreve o processo de integração entre duas ferramentas para realização de co-simulação na representação e análise de ambientes dinâmicos no contexto de redes elétricas inteligentes. As ferramentas integradas são o software para gerenciamento de co-simulação Mosaik e o software para desenvolvimento de sistemas multiagentes PADE. Como estudo de caso para demonstração da integração foi utilizado um cenário composto de uma rede de distribuição de energia elétrica em baixa tensão com 37 barras de carga, 20 gerações fotovoltaicas distribuídas aleatoriamente nas barras de carga, e 20 agentes PADE associados às gerações distribuídas que modelam o comportamento de um sistema de armazenamento de energia. Os resultados obtidos da simulação comprovam a eficácia da integração e demonstram a utilidade de sistemas multiagentes em modelar comportamentos e dinâmicas em redes elétricas inteligentes.*

1. Introdução

O sistema elétrico atual vem passando por uma quebra de paradigma devido à inserção de novas tecnologias na rede elétrica, que têm como objetivo produzir energia limpa e renovável, melhorar a qualidade e a confiabilidade da rede elétrica aumentar sua eficiente. A inserção de recursos energéticos distribuídos (RED) pode ser apontada como

uma das grandes responsáveis pelas mudanças e alterações pelas quais vem passando a rede elétrica [Kok and Widergren 2016]. RED são dispositivos conectados à rede elétrica, principalmente em baixa e média tensão, que têm capacidade de gerar energia e/ou armazená-la, por exemplo, mini e micro gerações e sistemas de armazenamento tais como as baterias.

Em se tratando de micro e mini geração, a tendência atual é que estas sejam do tipo solar fotovoltaica e eólica, ou seja, fontes não despacháveis e que entregam energia à rede de acordo com a disponibilidade de recursos naturais, que por natureza são intermitentes (luz solar e ventos). Esse comportamento de mini e micro gerações conectadas à rede elétrica geram grande imprevisibilidade na produção de energia e pode acarretar sérios desequilíbrios entre geração e consumo, prejudicando a estabilidade do sistema elétrico [Soares et al. 2017].

Assim, entre os principais fatores necessários para que a transição entre uma rede elétrica tradiconal e uma rede elétrica moderna aconteça está a utilização de gerenciamento e operação da rede elétrica baseados em sistemas distribuídos e que possibilitem a integração dos diversos RED sem afetar critérios de qualidade e eficiência.

Para isso, é necessário que testes e simulações possam ser executados antes da implementação em dispositivos reais conectados à rede elétrica. Uma solução bastante utilizada para esse tipo de análise é o esquema de co-simulação que consiste da integração de vários processos de simulação coordenados entre si em uma mesma base de tempo e com possibilidade de envio e recebimento de informações entre as diferentes instâncias de simulação.

Muitos trabalhos adotam esse esquema para integrar os comportamentos de diferentes dispositivos, simulados em plataformas diversas, em um processo de simulação global. Em [Albagli et al. 2016] os autores propõem a utilização da técnica de high-level architecture (HLA) na integração entre simuladores e SMA para simulação de redes elétricas inteligentes. Em [Duan et al. 2017] é mostrada a integração entre agentes desenvolvidos em JADE e um ambiente de co-simulação de redes elétricas inteligentes. Em [Cullen et al. 1987] a integração entre o gerenciador de co-simulação Mosaik e o simulador de ambientes de redes de comunicação OMNet++ é apresentada e apontada como essencial para análise do comportamento dos dispositivos comunicantes mediante diferentes cenários de utilização da rede de comunicação.

Este trabalho descreve a integração entre o framework para desenvolvimento de sistemas multiagentes PADE e o framework para realização de co-simulação Mosaik, ambos software livre e desenvolvidos em linguagem de programação Python. Para demonstrar os resultados desse integração entre softwares um exemplo de aplicação na modelagem da dinâmica de RED em uma rede de distribuição de energia elétrica de baixa tensão é apresentado.

As próximas seções estão organizadas da seguinte forma: a Seção 2 descreve alguns aspectos sobre a dinâmica dos RED, a Seção 3 descreve o processo de co-simulação e uma alternativa para integração de SMA nesse processo, a Seção 4 descreve as APIs do Mosaik, explicando como se dá o processo de integração com a plataforma; a Seção 5 apresenta de que maneira foi realizada a integração com o PADE, sendo a principal contribuição desse artigo; em seguida, a Seção 6 apresenta o estudo de caso proposto,

executando sobre a integração do PADE com o Mosaik. Ao final, a Seção 7 conclui o artigo com alguns comentários sobre o estudo.

2. Gerenciamento de Recursos Energéticos Distribuídos

A dinâmica dos RED altera muitos princípios de operação do sistema elétrico de potência tal como concebido atualmente: um número relativamente pequeno de grandes centrais geradoras são conectadas a centros consumidores por meio de um sistema de transmissão de energia que se estende por vários quilômetros [Kok and Widergren 2016].

Ao contrário, quanto maior for a inserção de RED no sistema elétrico, a operação desse sistema deixa de ser centralizada e passa a ser distribuída entre milhares de unidades que precisam estar coordenadas e serem capazes de reagir às inúmeras flutuações tanto no lado da demanda quanto no lado da geração.

Para que esse cenário se concretize alguns fatores são essenciais:

1. Alterações na legislação e no modelo de estruturação no setor elétrico devem ser implementadas;
2. Sistemas de tecnologia da informação que permita comunicação rápida e bidirecional entre as entidades que participarão do esforço de prover estabilidade e qualidade no fornecimento de energia elétrica;
3. Utilização de esquemas de gerenciamento e operação da rede elétrica baseados em sistemas distribuídos e que possibilitem a integração dos diversos RED sem afetar critérios de qualidade e eficiência.

Em muitos casos, a realização de co-simulação para análise de esquemas de controle e de operação em ambientes de redes elétricas inteligentes (REI) é indispensável uma vez que uma grande quantidade de comportamentos e entidades são simuladas, muitas vez em diferentes plataformas de software.

Coordenar os diferentes processos de simulação é uma tarefa complexa que exige a coordenação entre os simuladores, além do gerenciamento das dependências de dados interrelacionados.

3. PADE e Mosaik: SMA integrado ao ambiente de co-simulação

Considerada uma linguagem de fácil aprendizagem e com muitos recursos para computação numérica, Python vem sendo cada vez mais utilizada pela comunidade acadêmica em diversas áreas de conhecimento [Brown et al. 2017]. Algumas ferramentas voltadas à realização de análises científicas vem sendo desenvolvidas em Python, com destaque, na área de redes elétricas inteligentes, para o *framework* em realização de co-simulação Mosaik [Mosaik 2018], que vem sendo desenvolvido por pesquisadores do instituto alemão de pesquisa e desenvolvimento OFFIS vinculado à universidade Carl von Ossietzky em Oldenburg.

A principal funcionalidade provida pelo Mosaik é a integração de processos de simulação existentes em um contexto comum, com a finalidade de representar a dinâmica de sistemas elétricos [Schütte et al. 2011]. Ou seja, cada um dos simuladores são executados separadamente, com seu próprio loop de eventos. Mosaik sincroniza os processos de simulação e gerencia a troca de dados entre eles, provendo as seguintes funcionalidades:

- API para possibilitar a comunicação Mosaik/Simulador/Mosaik;
- Implementação de *handlers* para diferentes tipo de processos de simulação;
- Composição de cenários de simulação;
- Geração de uma base de dados comum, coordenando a troca de dados entre os simuladores.

Diversas estratégias de controle podem ser utilizadas em simulações envolvendo instâncias que modelam o comportamento de RED. Sistemas multiagentes (SMA) vêm sendo utilizados para prover funções de controle distribuído em sistemas que integram RED em ambientes de REI [Kumar Nunna and Doolla 2013].

O framework Python Agent DEvelopment (PADE) [PADE 2018], foi desenvolvido inicialmente pelo Grupo de Redes Elétricas Inteligentes da Universidade Federal do Ceará e inspirado pelo tradiconal framework para SMA JADE. O PADE é uma alternativa para implementação e execução de sistemas de agentes reativos em Python, sendo construído no topo do framework para desenvolvimento de sistemas assíncronos Twisted [Twisted 2018], provê aos agentes módulos de programação, interface de gerenciamento e comunicação de acordo com os padrões especificados pela Foundation for Interoperable Physical Agents (FIPA).

Utilizado em conjunto com o framework para co-simulação Mosaik, o PADE pode se tornar uma ferramenta indispensável na modelagem dos comportamentos e na dinâmica de controle dos dispositivos simulados no ambiente de redes elétricas inteligentes.

4. API Mosaik para integração de simuladores

Um dos componentes mais importantes disponibilizados pelo Mosaik para prover o ambiente de co-simulação é sua API para permitir a integração de diferentes processos de simulação em uma base de tempo comum. Existem duas opções de utilização da API disponibilizada pelo Mosaik: API de alto nível e API de baixo nível.

Conforme mostrado na Figura 1, tanto na API de alto nível quanto na API de baixo nível, o Mosaik provê comunicação com cada uma das instâncias de simulação enviando mensagens padronizadas no formato JavaScript Object Notation (JSON) por meio de sockets TCP/IP. A API de alto nível é utilizada para integrar simuladores que não possuem loop de execução próprio e que podem ser modelados em linguagem de programação Python. Essa implementação da API de alto nível do Mosaik trata-se de um módulo Python que pode ser importado e utilizado em um arquivo de código fonte. Suas classes implementam a comunicação direta com o núcleo de simulação do Mosaik e deixam transparente ao usuário os processos necessários à integração.

A API de baixo nível deve ser utilizada em simuladores que possuem loop de execução próprio, ou ainda, que não possam ser implementados utilizando Python. Nesse caso, a implementação dos sockets TCP/IP e o tratamento das mensagens JSON enviadas pelo Mosaik devem ser realizadas no simulador. Essa foi a abordagem desenvolvida neste trabalho.

O Mosaik realiza os procedimentos de criação de cenário, controle do fluxo de informações e compartilhamento de base de dados comum, como já mencionado anteriormente, por meio de mensagens padronizadas para cada um dos simuladores gerenciados no processo de co-simulação. Na Figura 2 são apresentas as mensagens mais comuns utilizadas pelo Mosaik, destacando-se:

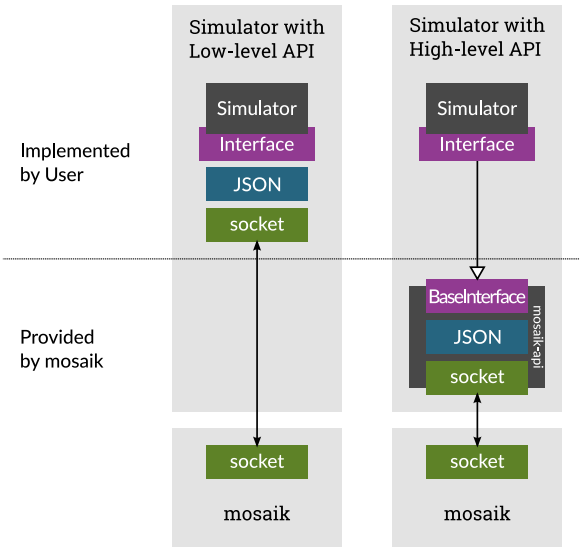


Figura 1. Modelo de API Mosaik de alto nível e de baixo nível.

- *init(sid, sim_params)*: Inicializa o simulador com o identificador sid e recebe parâmetros adicionais passados pela API do Mosaik;
- *create(num, model, model_params)*: Cria num instâncias do modelo especificado com os parâmetros enviados pela API do Mosaik;
- *step(time, inputs)*: Executa um passo de simulação no tempo especificado pelo mecanismo de sincronismo do Mosaik com os valores de entrada enviados por outro simulador previamente conectado à instância de simulação e retornando à API do Mosaik o tempo para a próxima chamada de execução do simulador;
- *get_data(outputs)*: Recebe requisição para envio de dados, previamente especificados, à API do Mosaik que cuidará em direcioná-los ao simulador que os receberá como parâmetro input do método setup.

Na próxima Seção será descrito qual dessas APIs foi selecionada para implementação da integração com o PADE, além de detalhar como foi realizada essa integração.

5. Implementação de API PADE para integração com Mosaik

Para integrar os agentes do ambiente de execução do PADE com o gerenciador de simulação do Mosaik, é necessário utilizar a API de baixo nível disponibilizada pois, apesar do código fonte dos agentes PADE estar em Python, o PADE possui seu próprio loop de execução e portanto não consegue integrar em alto nível com o Mosaik. A API de baixo nível irá enviar uma sequência de mensagens padronizadas em formato JSON para um socket TCP/IP previamente informado nas configurações do Mosaik, que deverá receber, processar e responder as mensagens que o Mosaik enviar.

As mensagens enviadas pelo Mosaik ao simulador integrado via API de baixo nível seguem o padrão estabelecido na Figura 3, em que o campo *Header* indica o número de bytes, codificado em unsigned integer (uint32), contidos no campo *Payload*, que é codificado em UTF-8 e contém uma lista no formato JSON com três campos:

- *Type*: codificação para classificar tipo de mensagem sendo 0 para requisição, 1 para resposta e 2 para falha;

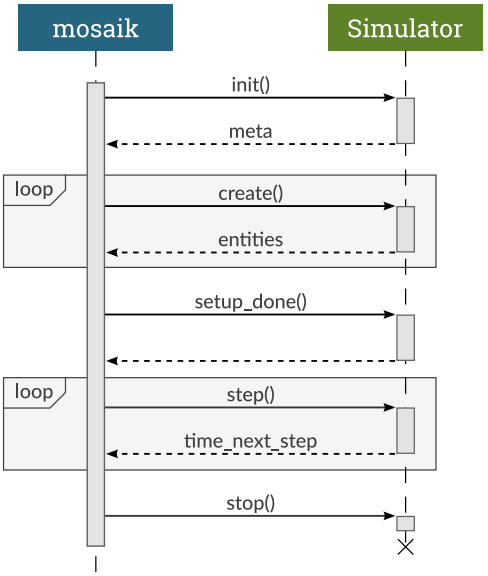


Figura 2. Sequencia de mensagens enviadas entre Mosaik e processo de simulação.

- *ID*: Identificador único para cada requisição enviada pelo Mosaik para os simuladores. As respostas precisam ter o mesmo identificador que a mensagem de requisição;
- *Content*: Conteúdo da mensagem, padronizado como uma lista JSON e que depende do tipo de mensagem.

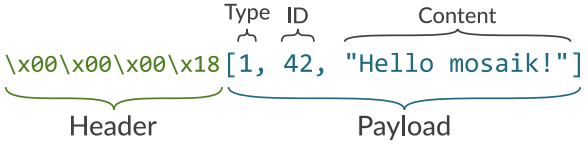


Figura 3. Estrutura das mensagens definidas pelo Mosaik para coordenação e sincronismo no ambiente de co-simulação.

O Mosaik faz extensivo uso da estrutura de dados Python chamada dicionário, que trata-se de um conjunto de pares (chave, valor) para configurar seus modelos de simulação. Dessa forma, o primeiro procedimento para que o PADE possa ser integrado com o Mosaik é a inclusão do nome do modelo que irá representar os agentes PADE, assim como seu endereço na rede (endereço IP e porta de execução do agente) no dicionário que fornece informações a respeito dos simuladores que serão executados durante o ciclo de simulação. Um dicionário contendo essas configurações é apresentado na Figura 4, em que é possível visualizar as declarações de simuladores que utilizam a API de alto nível e o modelo que representa um agente PADE utilizando sua API de baixo nível.

No lado do PADE, o agente precisa implementar uma classe que irá tratar cada uma das requisições enviadas pelo Mosaik. O diagrama de classes UML mostrando a relação da classe `MosaikSim` com a classe `Agent` do PADE é apresentado na Figura 5.

Após estabelecida a conexão entre Mosaik e PADE, o processo de gerenciamento de simulações do Mosaik inicia o envio de mensagens de controle para o agente PADE.

```
    sim_config = {
        "ExampleSim": {"python": "example_sim:ExampleSim"},
        "PadeSim": {"connect": "127.0.0.1:20000"}
    }

```

Figura 4. Configuração de simulador Mosaik e agente PADE integrados em co-simulação.

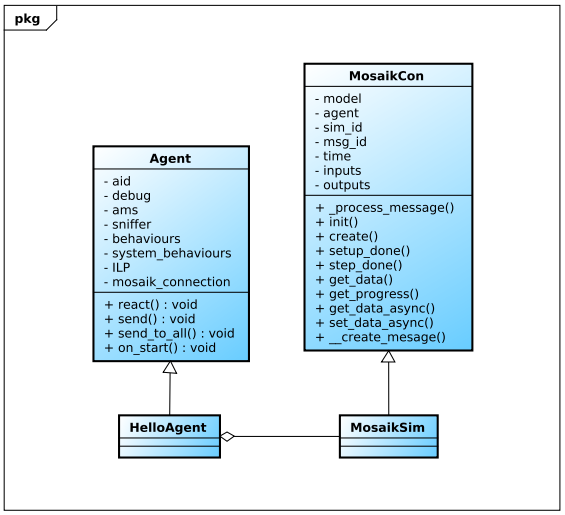


Figura 5. Diagrama de classes UML.

Essas mensagens são identificadas pelo PADE e processadas separadamente pelos métodos desenvolvidos na classe `MosaikCon` do PADE, conforme mostrado na Figura 5.

Um exemplo de classe que herda a classe `MosaikCon` no código fonte de um agente PADE pode ser visualizada no Algoritmo 1, que mostra a classe `MosaikSim` e cada um dos métodos da classe `MosaikCon` que estão sendo sobrescritos de acordo com o comportamento desejado nos procedimentos da simulação.

```
class MosaikSim(MosaikCon):

    def __init__(self, agent):
        super(MosaikSim, self).__init__(MOSAIK_MODELS, agent)
        self.entities = list()
        self.loc_name = self.agent.aid.localname

    def create(self, num, model, init_val, medium_val):
        entities_info = list()
        for i in range(num):
            self.entities.append(init_val)
            display_message(self.loc_name, str(init_val))
            entities_info.append(
                {'eid': self.sim_id + '.' + str(i),
                 'type': model,
                 'rel': []})
        return entities_info

```

```
19      def step(self, time, inputs):
20          if time % 501 == 0 and time != 0:
21              display_message(self.loc_name, '{:4d}'.format(time))
22          return time + self.time_step
23
24      def get_data(self, outputs):
25          response = dict()
26          for model, list_values in outputs.items():
27              response[model] = dict()
28              for value in list_values:
29                  response[model][value] = 1.0
30          return response
```

Algoritmo 1. Código exemplo de implementação da classe MosaikCon.

A próxima seção apresentará o estudo de caso utilizando a integração entre o PADE e o Mosaik.

6. Estudo de Caso

Para demonstrar a potencialidade da integração entre os *frameworks* Mosaik e PADE na implementação de modelos computacionais para simulações de ambientes de redes elétricas inteligentes, será realizado um estudo de caso para simulação da dinâmica dos RED tipicamente presentes em uma rede de distribuição de baixa tensão, ou seja, carga residencial, geração fotovoltaica e sistema de armazenamento de energia. Os principais elementos do estudo de caso são:

- Rede de distribuição secundária (baixa tensão) com um transformador de distribuição e 37 barras associadas à cargas residenciais e RED;
- 20 gerações distribuídas solar-fotovoltaicas ao longo da rede;
- Para cada geração distribuída, está associado um agente PADE, denominado *Device Agent* que simulará o comportamento de um sistema de armazenamento de energia que armazenará 10% da energia gerada pelo sistema fotovoltaico;
- Integração com ferramenta de cálculo de fluxo de carga para verificação do estado da rede, com especial atenção no critério de nível de tensão;
- Integração com ferramenta para visualização de resultados via interface *web*.

A topologia do sistema elétrico pode ser observada na Figura 6, em que é possível identificar cada um dos nós que compõem a rede de energia elétrica em nível secundário de distribuição, assim como suas interconexões. A rede é composta por um transformador e 37 nós que podem ter cargas associadas ou não à geração e sistema de armazenamento de energia elétrica.

Antes de analisar os resultados obtidos da simulação é importante descrever o cenário geral, já que o principal objetivo desta seção é demonstrar a integração entre as ferramentas Mosaik e PADE e não os aspectos de qualidade de energia ou de estabilidade do sistema, parâmetros geralmente analisados neste tipo de aplicação.

O estudo de caso foi estruturado tomando como base um exemplo de utilização do Mosaik disponível publicamente em repositório online [Mosaik 2017] e modificado para a inclusão dos agentes PADE. Um repositório on-line de livre acesso está disponível com o código fonte deste estudo de caso[1].

[1] https://github.com/grei-ufc/mosaik-demo-wesaac-2019

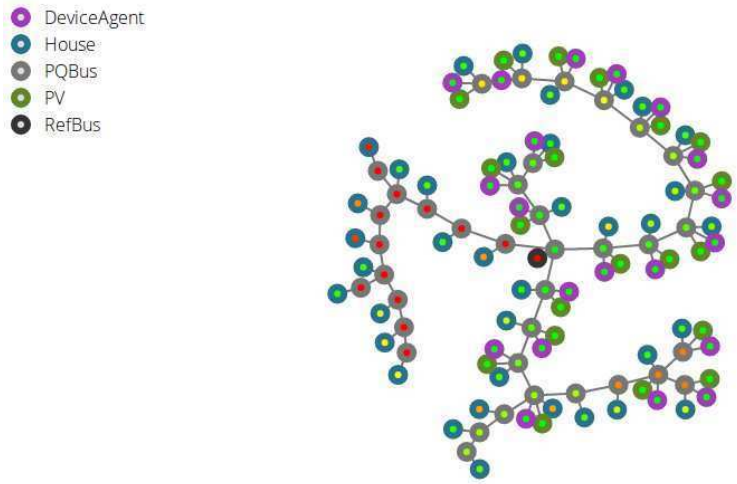


Figura 6. Rede teste.

No exemplo são utilizados plugins para integração do Mosaik com ferramentas de armazenamento de dados, acesso à arquivos CSV, ferramenta de análise de fluxo de carga e visualização via interface *web*. Na Figura 7 é possível visualizar a declaração de cada um dos simuladores utilizados na co-simulação:

```
sim_config = {
    'CSV': {
        'python': 'mosaik_csv:CSV',
    },
    'DB': {
        'cmd': 'mosaik-hdf5 %(addr)s',
    },
    'HouseholdSim': {
        'python': 'householdsim.mosaik:HouseholdSim',
    },
    'PyPower': {
        'python': 'mosaik_pypower.mosaik:PyPower',
    },
    'WebVis': {
        'cmd': 'mosaik-web -s 0.0.0.0:8000 %(addr)s',
    },
}

```

Figura 7. Configuração dos simuladores no Mosaik.

O simulador de nome *CSV* é utilizado para leitura de dados de medição da potência de saída de geradores solar fotovoltaicos e utilizados para integrar o processo de simulação. O simulador *DB* irá receber os resultados de parâmetros elétricos da simulação e armazenar os dados em um arquivo de dados *hdf5*. O simulador *HouseHoldSim* gera uma série temporal que representa a energia consumida em uma residência com número arbitrário de moradores. O simulador *PyPower* integra a ferramenta de fluxo de carga *PyPower* com Mosaik para realizar as análises a cada passo de simulação estabelecido. Por fim o simulador *WebVis* é uma ferramenta para visualização gráfica das dinâmicas de

simulação, gerando a topologia gráfica da rede e gráficos de nível de tensão e potência gerada/consumida em cada um dos nós.

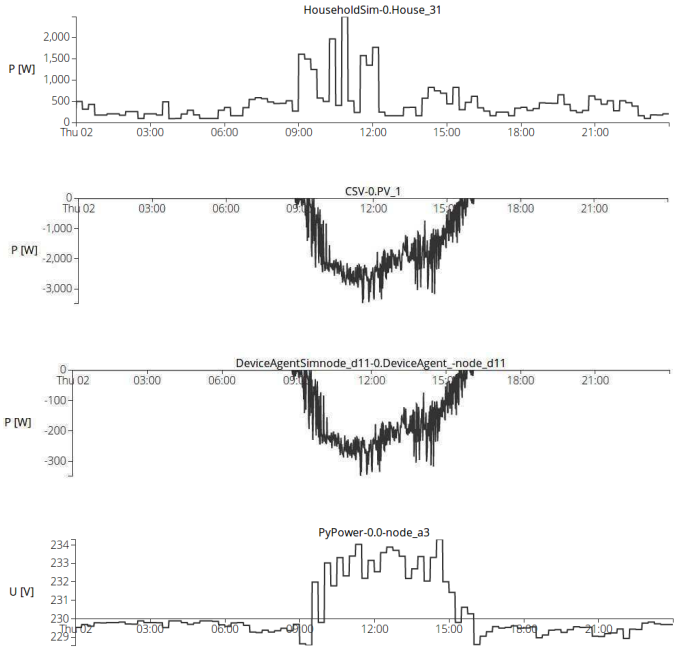
Para integrar os agentes PADE ao processo de simulação é necessário declarar simuladores que irão representar os agentes, da mesma forma que os demais simuladores foram declarados.

Após a definição dos agentes, é necessário associá-los aos geradores e as barras da rede de distribuição. Isso é feito utilizando o método `connect()`, disponibilizado pela API do Mosaik para que os simuladores possam trocar dados entre si.

Realizada a definição dos modelos de simulação e de seus respectivos simuladores, bem como a criação das instâncias de modelo que serão executadas nos simuladores e do estabelecimento das conexões entre estes, o ambiente de co-simulação está pronto para ser executado por meio do comando `run(until=END)` que inicializa o processo de simulação e estabelece o tempo total que ela deve durar, definido em segundos na variável `END`.

Durante o processo de co-simulação é possível ter acesso aos dados gerados utilizando uma interface gráfica *web*, bastando inserir em um navegador de internet o endereço IP e a porta do plugin `WebVis`, que disponibiliza visualmente os dados de cada uma das instâncias de simulação previamente configuradas. Na Figura 8 são mostrados os valores de demanda da residência, a potência de saída do sistema fotovoltaico, a parcela de 10% da potência gerada armazenada na bateria e o nível de tensão na barra considerada.

Figura 8. Resultados da simulação.



Na Figura 8 é possível também observar a semelhança entre as formas de onda que representam a potência gerada pelo sistema fotovoltaico e a energia armazenada pelo sistema de armazenamento, como era de se esperar já que existe uma correspondência direta entre os dois valores, diferindo somente por um fator de escala de 10%.

O agente PADE que modela o comportamento do sistema de armazenamento recebe os dados de geração, aplica o fator de escala de 10% e envia este valor de energia armazenada para o simulador responsável pelo fluxo de carga que irá agregar os valores de potência gerada/consumida e descontar o valor de energia armazenada para o cálculo das tensões em cada uma das barras do sistema.

Assim, esta simulação apresentou o estudo de caso utilizando Python e Mosaik para identificação do estado do sistema elétrico de distribuição, levando-se em conta a presença de 37 cargas, 20 geradores fotovoltaicos distribuídos, e 20 dispositivos de armazenamento que guardam 10% da energia produzida. Mesmo com todos esses elementos presentes e tendo os armazenadores modelados como agentes, foi possível verificar o impacto dos mesmos no sistema elétrico a partir da integração com o Mosaik.

7. Conclusão

Neste artigo foi demonstrado a integração entre duas ferramentas que podem ser de grande ajuda na modelagem e análise de sistemas de energia elétrica em um contexto de redes elétricas inteligentes com presença de recursos energéticos distribuídos.

A primeira ferramenta chama-se Mosaik que é um software implementado na linguagem de programação Python com o objetivo de prover um serviço de integração de simuladores em co-simulação, oferecendo ferramentas para descrição de cenário, gerando uma base de tempo comum entre os simuladores e provendo serviço de troca de mensagens entre estes.

A segunda ferramenta analisada chama-se PADE, um framework em Python para modelagem de sistemas multiagentes baseados nos padrões estabelecidos pela FIPA, que provê bibliotecas para modelagem dos comportamentos dos agentes, ambiente de execução e interface gráfica para gerenciamento.

As duas ferramentas foram integradas com êxito e foi demonstrada por meio de um estudo de caso que apresenta uma rede de distribuição com RED em que os agentes PADE simulam o comportamento de um sistema de armazenamento de energia. Os resultados demonstram a troca de informações dos simuladores com os agentes e a possibilidade de implementação de esquemas de controle mais complexos em que seja necessário a utilização de modelos de inteligência distribuída.

Vale ressaltar que as duas ferramentas são software livre e disponíveis gratuitamente em repositórios *on-line*.

8. Agradecimentos

Os autores do artigo agradecem à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro e à Universidade Federal do Ceará (UFC) e Universidade Federal do Pará (UFPA) pela disponibilização do espaço físico e da utilização de seus laboratórios.

Referências

Albagli, A. N., Falcão, D. M., and De Rezende, J. F. (2016). Smart grid framework co-simulation using HLA architecture. *Electric Power Systems Research*, 130:22–33.

Brown, T., Hörsch, J., and Schlachtberger, D. (2017). PyPSA: Python for Power System Analysis.

Cullen, M. F., Miller, S. F., and Hord, R. M. (1987). A model-based system for object recognition in aerial scenes. *Proceedings of SPIE - The International Society for Optical Engineering*, 726:530–538.

Duan, Y., Luo, L., Li, Y., Cao, Y., Rehtanz, C., and Küch, M. (2017). Co-simulation of distributed control system based on JADE for smart distribution networks with distributed generations. *IET Generation, Transmission & Distribution*, 11(12):3097–3105.

Kok, K. and Widergren, S. (2016). A Society of Devices: Integrating Intelligent Distributed Resources with Transactive Energy. *IEEE Power and Energy Magazine*, 14(3):34–45.

Kumar Nunna, H. S. V. S. and Doolla, S. (2013). Multiagent-based distributed-energy-resource management for intelligent microgrids. *IEEE Transactions on Industrial Electronics*, 60(4):1678–1687.

Mosaik (2017). Mosaik demo. `https://bitbucket.org/mosaik/mosaik-demo`.

Mosaik (2018). Mosaik software. `https://bitbucket.org/mosaik/mosaik`.

PADE (2018). Pade software. `https://github.com/grei-ufc/pade`.

Schütte, S., Scherfke, S., and Tröschel, M. (2011). Mosaik: A framework for modular simulation of active components in Smart Grids. In *2011 IEEE 1st International Workshop on Smart Grid Modeling and Simulation, SGMS 2011*, pages 55–60.

Soares, J., Fotouhi Ghazvini, M. A., Borges, N., and Vale, Z. (2017). A stochastic model for energy resources management considering demand response in smart grids. *Electric Power Systems Research*, 143:599–610.

Twisted (2018). Twisted matrix labs. `https://github.com/twisted/twisted`.

Arisa Nest – Uma Plataforma Baseada na Nuvem para Desenvolvimento de Assistentes Virtuais

Saulo Popov Zambiasi[1], Ricardo J. Rabelo[2]

[1]Centro Tecnológico - Universidade do Sul de Santa Catarina (UNISUL)
Av. Pedra Branca, 25 - Cidade Universitária Pedra Branca - Palhoça - SC - Brasil

[2]Centro Tecnológico - Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina (UFSC) - Florianópolis - SC - Brasil

saulopz@gmail.com, ricardo.rabelo@ufsc.br

Abstract. *The use of virtual assistants has become popularized by their adhesion by large companies. This is because they have brought benefits to a certain extent in people's daily activities or in their tasks in organizations, with reminders, automation of repetitive tasks, proposing solutions for various situations, etc.. To meet this demand, several tools have been developed to create this type of agent. Following the same trends, this paper presents the Arisa Nest platform, under the Platform as a Service model, as a tool to create virtual assistants with resources to manage information about users, create algorithms to give more effective answers, consume web services to the interoperability with other systems and with proactivity behaviors. Finally, some cases are presented using the platform in real scenarios and tests.*

Resumo. O uso dos assistentes virtuais tem se popularizado pela sua adesão por grandes companhias. Isso porque eles têm trazido benefícios, até certo ponto, nas atividades diárias das pessoas ou em suas tarefas nas organizações, com lembretes, automatização de tarefas repetitivas, propondo soluções para várias situações, etc.. Para comportar essa demanda, diversas ferramentas têm sido desenvolvidas para a criação desse tipo de agente. Seguindo o mesmo viés, esse artigo apresenta a plataforma Arisa Nest, sob o modelo de *Platform as a Service*, como uma ferramenta de criação de assistentes virtuais com recursos de gerir informações sobre os usuários, criar algoritmos para dar respostas mais eficazes, consumir serviços web para a interoperabilidade com outros sistemas e com comportamentos de proatividade. Ao final, são apresentados alguns casos utilizando a plataforma em cenários reais e de testes.

1 Introdução

Princípios como interoperabilidade, modularidade, virtualização, informações em tempo real, orientação a serviços, descentralização e autonomia têm sido considerados de grande importância para as empresas. A utilização de robôs de software (*softbots*, *bots* ou assistentes virtuais) é vista com solução para automatizar e auxiliar as pessoas na execução de algumas tarefas, em vários níveis de inteligência e autonomia [Rabelo, Romero e Zambiasi 2018]. Um tipo de *softbot*, na forma de um Software Assistente Pessoal, foi concebido em Zambiasi e Rabelo [2012] para interagir com máquinas, computadores, bancos de dados e outros sistemas de informação para auxiliar as pessoas

na execução de diversas tarefas. O bot interagia por conversa em linguagem natural via *instant messaging* no dispositivo móvel do usuário, podia mandar relatórios por e-mail e executava tarefas de forma autônoma, com ou sem a necessidade da confirmação do usuário.

O uso de assistentes virtuais para uma melhor interação com usuários na execução de variadas tarefas não é um assunto novo. Com os avanços das tecnologias de informação, algumas grandes empresas de software passaram a desenvolver sistemas computacionais assistentes para que empresas pudessem adotar seu uso nos seus processos de negócio [Markoff 2008]. Contudo, apesar desses esforços, ainda existem diversas limitações quanto à complexidade na implementação de chatbots e assistentes virtuais mais sofisticados e inteligentes, na integração com outros sistemas, flexibilidade e escalabilidade em termos de modificação e adição de novas ações, etc.

No intuito de contribuir para as pesquisas nessa linha, este artigo apresenta a plataforma Arisa Nest, baseada na nuvem, modelo *Platform as a Service* (PaaS), que permite a criação de assistentes virtuais com recursos como: editor de scripts em linguagem de programação Lua para responder ao usuário com mensagens mais eficazes; chamada a serviços web externos à plataforma; gerenciamento de crenças com base nos perfis dos usuários; editor e executor de comportamentos para pró-atividade; e interação por voz. Um conjunto de casos foram usados para testar a plataforma.

2 Assistentes Virtuais

Os Assistentes Virtuais provém da tecnologia dos *chatbots*, agentes de conversação para interação com os usuários em linguagem natural utilizando técnicas de inteligência artificial. Hoje, muitos desses artefatos de software não apenas interagem com o usuário via conversa, mas também podem executar tarefas para eles. O primeiro chatbot foi criado por Joseph Weizenbaum em 1966 com o nome de ELIZA e simulava a conversa com uma psicoterapeuta [Weizenbaum 1966]. Para Abu Shawar e Atwell [2015], um esforço bastante relevante hoje na área dos agentes de software de conversação, ou *chatbots*, é o ALICE, uma evolução do ELIZA, que utiliza uma linguagem chamada AIML (*Artificial Intelligence Markup Language*) sob XML (*eXtensible Markup Language*) para a criação da sua base de conversação, atualmente na versão 2.

Além de linguagens específicas e técnicas de programação de *chatbots*, existem também plataformas para facilitar a criação e servir como ambiente de execução, dando suporte ao ciclo de vida do bot e ferramentas de administração, contabilização, etc. Atualmente existem algumas plataformas bastante em ênfase, tais como a **Alexa** da Amazon. Todo o domínio da Alexa envolve um conjunto de dispositivos e ambiente de desenvolvimento. O usuário pode interagir com vários dispositivos. A interação com a Alexa pode se dar via software multiplataforma, além de dispositivos próprios para interação via voz, chamados de Alexa Echo. O Alexa Skills fornece um conjunto de ações e interações com a Alexa. Essas ações são as skills, que executam tarefas como tocar uma música, responder questões, fornecer informações do tempo, controlar as luzes da casa, etc.. Mas também é possível ao usuário criar seus próprios skills personalizados utilizando o Alexa Skills Kit. Ele permite a chamada de Amazon Web Services (AWS), integração com outros serviços desenvolvidos por outros em qualquer outra linguagem e em outros servidores. O Alexa Skills Kit fornece também um conjunto de APIs para integração com diversos dispositivos [ALEXA 2019]. É possível

programar serviços para serem utilizados nos Skills da Alexa através da interface da *Amazon Web Services* (AWS). Quando um skill é criado, dá-se um nome e, ao conversar com a Alexa, o usuário chama diretamente aquele skill. Os skills são responsivos, executando algo requisitado pelo usuário. Há uma plataforma para pesquisa de skills já publicados que o usuário pode agregar ao seu assistente. Em tempo, por meio dos skills, é possível construir uma sequência de diálogos entre o usuário e a Alexa [Tingiris 2018]. No AWS Lambda, o usuário paga apenas pelo tempo de computação consumido, sendo permitido uma quantidade de requisições/execuções gratuitas por mês. É possível criar, nas skills, respostas aos usuários por meio de uma estrutura de fatos em Node.js e em diversas linguagens de programação, como javascript, python, etc. [AWS 2019].

Outro exemplo é o **Watson Conversation**. Ele se caracteriza como "uma API para desenvolvimento de bots, com uma interface simples" que permite que pessoas sem conhecimentos de programação possam alimentar sua base de conhecimento. É preciso ter uma conta na IBM Cloud e, até certo ponto, pode ser utilizado de forma gratuita. A plataforma possui um ambiente de desenvolvimento da base de conversação baseada em intenções, entidades e diálogos. As intenções são ações atreladas às perguntas feitas pelo usuário. É necessário dar exemplos de frases de entrada que, posteriormente, o sistema generaliza para tentar identificar padrões nas intenções do usuário. As entidades são complementos. Por exemplo, se o usuário quer fazer um pedido de pizza, isso é uma intenção, as entidades são informações como sabor, tipo de massa, CEP da entrega, etc. Para as entidades, é possível trabalhar com sinônimos, como no caso de mussarela e queijo, ou adicionar expressões regulares. Os diálogos são utilizados para construir a interação com os usuários. Cada nó do diálogo pode responder algo, e um nó pode possuir nós filhos, ou um complemento de resposta. Os nós possuem uma interação if-then-else e podem armazenar variáveis [Mazon 2018]. As vantagens da utilização dele é a facilidade de criação, sem necessidade de conhecimentos de programação, além da generalização dos exemplos de frases de entrada do usuário nas inteções.

Outra plataforma é o **Dialogflow** da Google, que permite a criação de bots por voz e textos com tecnologias de Inteligência Artificial, com interação via site web, dispositivos móveis, Google Assistente, Amazon Alexa, Facebook e outros. Sua estrutura é baseada em fluxos de diálogos. Os bots podem trabalhar em assuntos diferentes, gerenciam os padrões de entradas de mensagens (*intents*), e geram respostas que podem levar por um caminho específico ou ramificado de perguntas e respostas, lembrando um fluxograma. O Dialogflow usa algoritmos da Google de inteligência artificial e aprendizado de máquina para generalizar os *intents*, aprendendo a encontrar padrões para responder de forma mais eficiente. Para alimentar os *intents*, basta adicionar um conjunto de exemplos de mensagens do usuário. O site do Dialogflow possui boa documentação em texto, imagem e vídeo sobre como trabalhar com a plataforma e boas práticas no processo da criação da base de conhecimento de um agente de conversação, *Conversation Design*. Há uma forma de proatividade por meio de eventos. Eles permitem que o bot invoque *intents* baseados em algo que acontece, ao invés de uma mensagem do usuário. Ele suporta eventos do Google Assistente e outras plataformas com base em ações realizadas pelo usuário [DIALOGFLOW 2019].

O **Pandorabots**[1] é um serviço online para construção e disponibilização de chatbots na web. O usuário adiciona elementos AIML na base de conversação e pode testar na própria plataforma, analisando os logs e identificando o que o bot não consegue responder para, então, adicionar novos elementos AIML. Em tempo, como os chatbots têm se tornado cada vez mais uma ferramenta de relevância no atendimento em grandes empresas, o mercado desse serviço tem atraído diversos investidores, *startups* e empresas de desenvolvimento. Sendo assim, há uma ampla gama de exemplos que ainda poderiam ser citados.

3 Plataforma Arisa Nest

A Plataforma Arisa Nest (Figura 1) é uma evolução da implementação apresentada em [Zambiasi e Rabelo 2012]. Ela é hoje uma ferramenta *PaaS*, onde os usuários podem hospedar assistentes virtuais e gerenciá-los por meio de uma interface web. Ela possui uma estrutura de criação de diálogos orientados a contextos para alimentar a base de conversação, um editor de scripts em linguagem de programação Lua que pode ser usado para complementar as respostas dos diálogos com resultados da execução de algoritmos, serviços web no padrão SOAP ou REST podem ser consumidos pelos scripts para executar coisas de fora da plataforma, gerenciamento de crenças com informações dos usuários, comportamentos em Lua que podem ser agendados e executados conforme mudança do estado do bot fornecendo proatividade. Ela permite a criação de vários bots e cada bot pode ter vários usuários colaboradores.

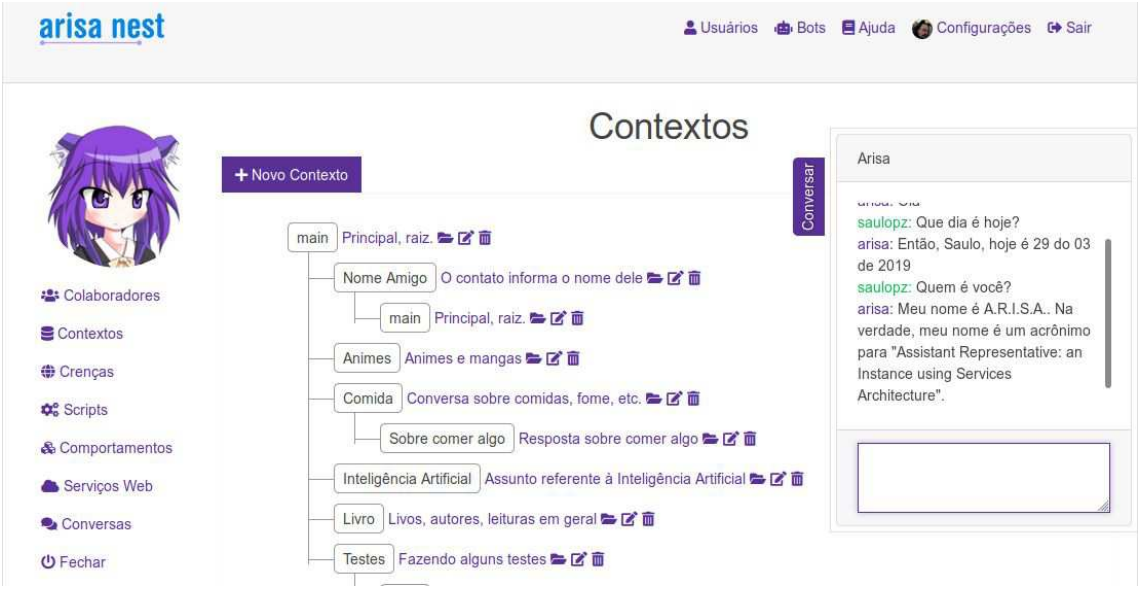


Figura 1. Interface da plataforma Arisa Nest, mostrando o menu superior, o menu do bot à esquerda, a árvore de contextos e a janela de chat para testes de conversas.

A base de conversação pode ser vista como uma árvore de contextos (Figura 1). Um contexto é usado para definir um assunto e pode-se mudar de assunto conforme a conversa com o bot. Por exemplo, a pergunta "o que você prefere?" é respondida de forma diferente para os contextos "comida" e "livro". É necessário haver um contexto inicial/pai chamado main, ou 0, e dentro dos contextos são criados os diálogos.

[1] https://home.pandorabots.com

Os diálogos servem para identificar uma mensagem do usuário e produzir uma resposta. O campo Condições efetua testes lógicos baseados nas suas crenças e nas crenças dos usuários para permitir ao bot usar ou não esse diálogo para responder uma mensagem. O campo Padrões é usado para identificar se o diálogo bate com a mensagem do usuário, no caso positivo, é escolhida uma resposta aleatória. É possível inserir vários padrões e respostas separados por ponto e vírgula. Um link e uma imagem também podem ser utilizados como parte da resposta. O campo "Ir Para" serve para direcionar o motor do chatbot para outro contexto. Há um tipo de diálogo especial, chamado Fuga, em que deixa-se o campo padrões vazio e, caso o bot não encontre nenhum diálogo que fecha naquele contexto, e existir um diálogo tipo fuga, ele escolhe uma resposta aleatória do mesmo. Nos padrões, o símbolo "%" representa um caractere curinga, que pode estar entre palavras, no final e/ou no início, conforme pode ser visto na Figura 2.

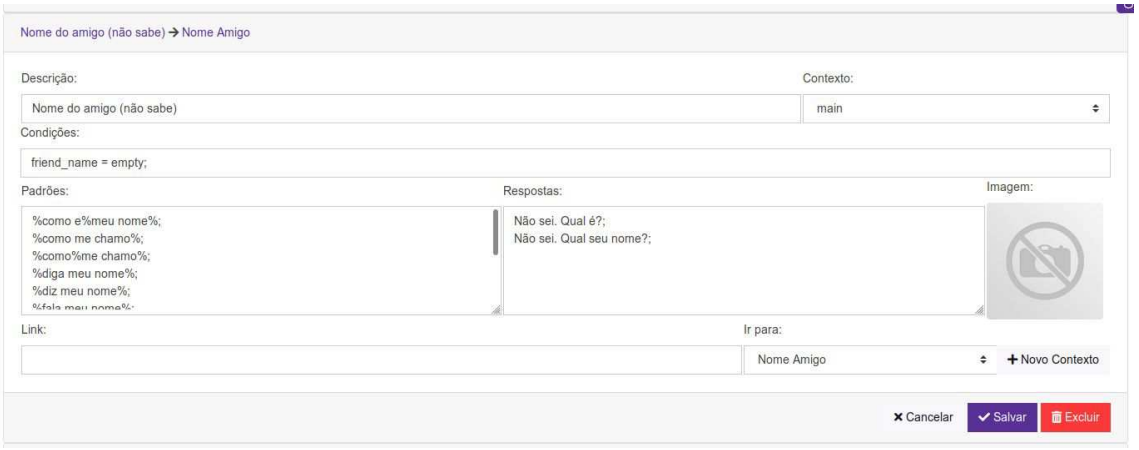


Figura 2. Formulário de diálogos

Os diálogos permitem alguns tipos especiais, em padrões e respostas, para gerir as crenças locais e globais e utilizar um algoritmo para complementar uma resposta. Por exemplo:

Padrão: %meu nome e **{friend_name};**
Resposta: Muito prazer **{friend_name}.** Me chamo **{bot_name};**
Mensagem usuário: Olá, meu nome é **Saulo**
Resposta do bot: Muito prazer **Saulo.** Me chamo **Arisa**

Nesse caso, o bot pega a informação no local de {friend_name} e armazena como uma crença local. {bot_name} é usando na resposta e provêm de uma crença global. Quando a informação não precisa ser armazenada, mas apenas usada para algo e depois descartada, coloca-se um sinal de $ antes do nome:

Padrão: %quanto e **{$1}** mais **{$2};**
Resposta: O resultado é **{@soma $1 $2};**
Mensagem usuário: Quanto é **2** mais **5**?
Resposta do bot: O resultado é **7**

As informações $1 e $2 são utilizadas apenas para o processamento nesse diálogo. Nele também que foi usado um sinal @. Ele indica a chamada ao script `soma` e

passa como parâmetro duas informações, 2 e 5. Os **scripts** são algoritmos desenvolvidos em linguagem de programação Lua na plataforma e podem ser usados de diversas formas, como executar cálculos e operações complexas, além de consumir serviços web externos à plataforma. Por exemplo, o script do diálogo anterior poderia ser apenas:

```
return tonumber(args[1]) + tonumber(args[2])
```

Ou o retorno de uma chamada a um serviço web:

```
return wscall('math', 'sum', { a = args[1], b = args[2] })
```

Em ambos os casos pode-se observar os argumentos de entrada `args`. No segundo caso, é usada a função `wscall`, que consome um serviço web cadastrado na plataforma chamado **math** usando a operação **sum**. No editor de scripts e comportamentos há um botão que, quando clicado, o usuário seleciona o serviço e a operação. O texto da chamada do serviço é adicionado automaticamente no editor, no cursor de edição, facilitando a criação do código.

Os **Comportamentos** são executados regularmente ou conforme agendamento configurável. Por exemplo, o comportamento da Figura 3 é agendado para todo dia 1º de Janeiro, às 0h01min desejando feliz ano novo para todos os seus contatos do telegram.

```
1  friends = json.decode(friendList())
2  for i = 1, #friends do
3      id = friends[i]
4      im = getLocal(id, 'im')
5      if im == 'telegram' then
6          message = getDialog('main', 'feliz_ano_novo', null)
7          sendMessage(id, message)
8      end
9  end
```

Figura 3. Comportamento feliz_ano_novo

A função `friendList` retorna os IDs de todos os contatos do usuário em uma string JSON. `getLocal` retorna uma crença de um contato, conforme seu **ID**. No caso, foi retornado o *instant messaging* (IM) usado pelo usuário. Caso seja 'telegram', utiliza uma resposta aleatória de um diálogo chamado **feliz_ano_novo** do contexto **main**, com a função `getDialog`, e envia como mensagem para o usuário com `sendMessage`. Algumas outras funções da plataforma que podem ser utilizadas tanto nos scripts quanto nos comportamentos são: `globalList` retorna a lista dos nomes das crenças globais; `localList` lista dos nomes das crenças locais de um contato; `getGlobal` retorna o valor de uma crença global; `setGlobal` altera uma crença global; `setLocal` altera uma crença local; `deleteGlobal` exclui uma crença global; `deleteLocal` exclui uma crença local; `last` retorna dia e horário da última mensagem enviada por um usuário ao bot; `sendToBot` envia uma mensagem para outro bot; `callRest` chama um serviço REST.

Há uma área de visualização das conversas, com informações das crenças locais e registros das conversas. Uma mensagem com cor diferente nas conversas indica que o bot usou uma resposta de fuga. Isso facilita para identificar mensagens que o bot não

está conseguindo responder. Do lado direito do plataforma, há um botão que permite abrir uma janela de chat para testar o bot na própria plataforma (Figura 1). A plataforma suporta multi linguagem e está disponível atualmente nos idiomas português e inglês.

3.1 Considerações sobre a Implementação

O motor do chatbot da Arisa Nest foi implementado na linguagem PHP e sua base de conhecimentos é armazenada em banco de dados Postgresql. O sistema de execução baseado em contextos inicia no contexto raiz *main*. A mensagem entrada do usuário é comparada com os padrões cadastrados nos diálogos daquele contexto. Caso algum padrão de um ou mais diálogos fechar com a entrada do usuário, é escolhido um diálogo aleatório, dentre os encontrados, e uma resposta aleatória desse. Antes do bot enviar uma resposta, a mensagem do usuário é quebrada em partes e as crenças são extraídas da mensagem e usadas para as crenças, para compor a resposta ou para serem utilizadas como parâmetros de algum script. Se o diálogo escolhido possui um link para outro contexto, o usuário é direcionado para ele. Quando o usuário envia uma nova mensagem, caso ela não encontre um diálogo no contexto atual, o motor navega para o contexto pai, e assim segue até que chegue ao contexto raiz. Se mesmo no contexto raiz, o motor não encontrar padrões que fecham com a mensagem, ele responde com uma resposta de "fuga". Os scripts são utilizados pelos diálogos, podem ler e escrever crenças, mandar mensagens para outros usuários ou bots, executar operações externas por meio de serviços web SOAP ou REST. Os comportamentos, por sua vez, são a forma dos bots agirem de forma proativa. Para que essa autonomia seja possível, é necessário que um programa executor de comportamentos. Neste caso, foi implementado um programa de execução de comportamentos em linguagem Go[2]. O programa executa em *background* no sistema operacional, verificando quais comportamentos estão configurados para execução e os executa, conforme configuração de agendamento. A interface do usuário foi implementada utilizando PHP, HTML, CSS, JavaScript, Ajax e Bootstrap 4. A plataforma encontra-se instalada e rodando em um servidor na nuvem com Sistema Operacional Ubuntu 18.04, servidor HTTP Apache 2 e banco de dados Postgresql, sob o domínio https://arisa.com.br/nest.

3.2 Arisa Nest como Plataforma para Agentes

A plataforma Arisa Nest permite a criação de múltiplos bots que podem executar algoritmos, consumir serviços web e podem conversar entre si para colaborar na resolução de problemas. Tal como os agentes [Russell e Norvig 2013] eles podem perceber os mais diversos possíveis ambientes, conforme funcionalidades e interoperabilidades fornecidas por meio do acesso à serviços web e mensagens de usuários ou outros bots (habilidade social). Armazenam crenças para compor seu estado interno, executam seu processamento interno por meio de scripts e algoritmos de comportamentos que podem alterar seu estado e agir de forma autônoma e proativa. Por meio desses recursos, é possível construir na plataforma agentes que vão desde os puramente reativos, até os cognitivos e/ou baseados em objetivos e utilidade.

[2] Linguagem de programação criada pela Google, baseada na linguagem Limbo do sistema operacional Inferno, focada em programação concorrente e produtividade. Em 2009 foi lançada como código livre e atualmente com licença BSD. https://golang.org/

Considera-se aqui o seguinte cenário fictício simplificado para uma prova de conceito. O funcionário Saulo é responsável pelo controle de estoques de produtos em uma empresa. Um agente foi criado para auxiliá-lo, verificando regularmente a situação em um sistema legado e, caso o estoque de um produto esteja com a quantidade menor do que uma certa quantidade pré configurada, o agente inicia um processo de compra, com a autorização do funcionário. Caso o processo de compra, seja autorizado, o agente entra em contato com um agente de vendas da empresa fornecedora e faz o pedido.

Algumas considerações sobre as atividades e comunicação entre os agentes na plataforma Arisa Nest: (i) a comunicação entre os agentes pode ser padronizada para facilitar o entendimento entre eles com palavras chaves e parâmetros ou pode-se utilizar mensagens em um formato padrão, como JSON; (ii) o reconhecimento de outro agente se dá pelo seu ID na plataforma e o servidor da plataforma, podendo o outro agente estar localizado em outro servidor que também possua a plataforma instalada; (iii) Um agente entra em contato com o outro por meio de mensagens e ela é analisada pelo motor de chatbot, que encontra o diálogo correspondente e inicia um script; (iv) Um processo automatizado ou conversação entre os agentes ou entre um agente e seu usuário pode ser iniciado por um comportamento ao verificar mudanças nas crenças ou por mudanças em uma informação consumida via um serviço web.

O algoritmo da Figura 4 é um comportamento agendado para executar a cada hora. Ele verifica o estoque e, caso necessário, cria uma ordem de compra, pedindo confirmação do usuário. Em caso afirmativo, efetua a compra.

```
1  produto.id = json.decode(wscall('estoque', 'baixo', null))
2  if produto.id ~= '' then
3      fornecedor = json.decode(wscall('fornecedor', 'get', { produto = produto.id }))
4      if fornecedor.id ~= '' then
5          ordemjson = wscall('ordem', 'novaCompra', { fornecedor = fornecedor.id,
6              produto = produto.id, quantidade = produto.quantidadeCompra })
7          setLocal(getGlobal('funcionario_id'), 'ordem', ordemjson)
8          ordem = json.decode(ordemjson)
9          mensagem = getDialog('estoque', 'envia_ordem_funcionario', { ordem.id, produto.nome,
10             produto.quantidadeCompra, fornecedor.nome, fornecedor.preco })
11         sendMessage(getGlobal('funcionario_id'), mensagem)
12     end
13 end
```

Figura 4. Comportamento verifica_estoque.

Seguindo sua execução, na linha 1 ele consome um serviço web de acesso a um sistema legado, que retorna informações de um produto se o mesmo está com o estoque baixo. Na linha 3 consome um serviço que encontra um fornecedor cadastrado que trabalha com aquele produto e, caso exista mais de um, faz uma escolha conforme critérios daquele serviço. Nas linhas 4 e 5 é criada uma ordem de compra em um sistema legado, e as informações retornam no formato JSON. Na linha 7, as informações dessa ordem são armazenadas como crenças do usuário, pois a mesma ainda necessita de confirmação e o usuário pode perguntar mais informações sobre a mesma antes de confirmar. Nas linhas 9 e 10 é criada uma mensagem para o usuário com base no diálogo de nome `envia_ordem_funcionario` do contexto `ordem`, passando os parâmetros necessários (Figura 5). Na linha 11, a mensagem é enviada para o funcionário responsável (Figura 6).

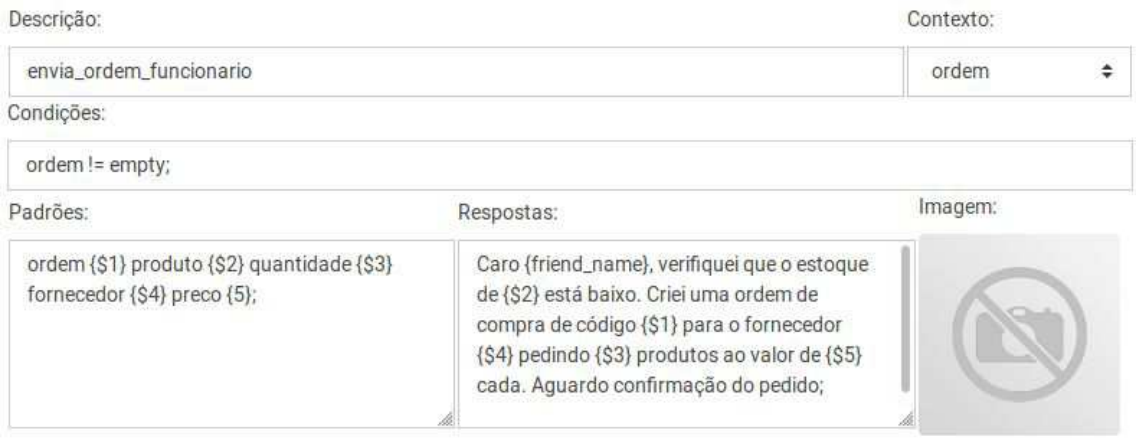


Figura 5. Diálogo envia_ordem_funcionario.

Quando o usuário envia a mensagem confirmando ou cancelando a ordem, é acionado um script que envia uma mensagem ao agente fornecedor. Caso o fornecedor não puder cumprir com a venda, a ordem é cancelada e o usuário é informado, caso contrário a compra é efetivada (Figura 6).

Mensagem Arisa para fornecedor:

```
pedido ordem 2923 produto 213 quantidade 10
```

Três possíveis respostas do fornecedor:

```
venda 2923 ok| venda 2923 em falta| venda 2923 produto inexistente
```

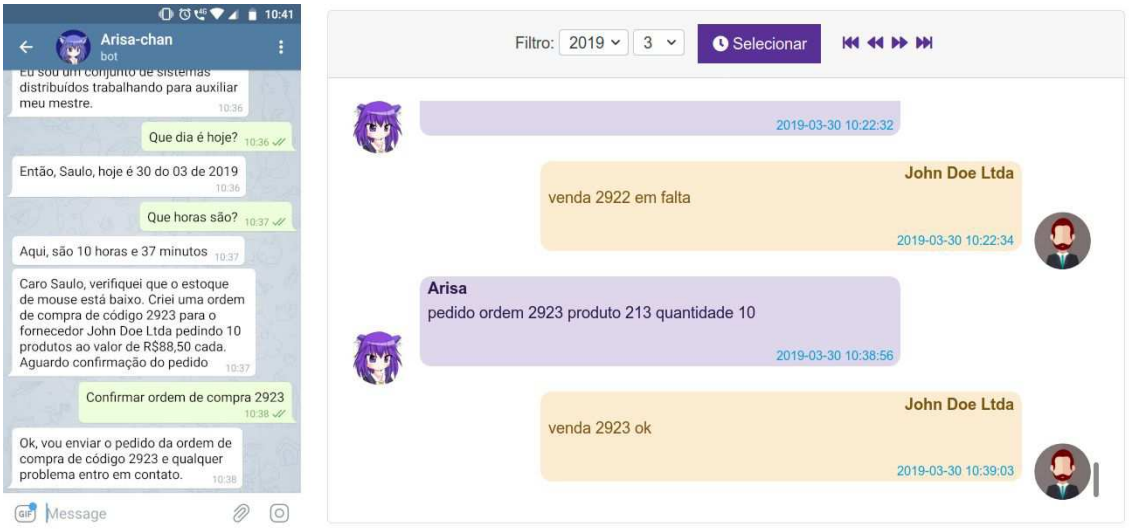


Figura 6. Esquerda: interação com o usuário via telegram; Direita: registro da interação entre os bots na plataforma.

Ao interagirem, os bots alteram seus estados internos por meio das crenças, scripts e comportamentos. Eles podem interagir tanto com seus usuários como com outros bots para alcançar seus objetivos. Mesmo que a plataforma possa ter vários bots executando ao mesmo tempo, cada bot precisa ser cadastrado e configurado individualmente. No momento a plataforma não permite a criação e exclusão dinâmica

de bots como para um sistema multiagentes dinâmico, mas tal recurso já encontra-se em desenvolvimento.

3.3 Considerações da Plataforma

Comparando a plataforma Arisa Nest com as ferramentas citadas na seção 2, a ELIZA, ALICE e Pandorabots são apenas agentes de conversação e não executam tarefas e pouco fazem no caso de adquirir informações dos usuários. No caso da ALICE e Pandorabots, há o uso de uma linguagem AIML, já bastante madura, mas trabalhar com essa linguagem não é tão intuitivo e requer um certo aprendizado e conhecimento, enquanto a alimentação dos diálogos da Arisa Nest é mais simples e requer apenas entender a estrutura dos padrões e crenças. Já no caso do Watson Conversation, ele tem uma interface de fácil aprendizado e de uso relativamente intuitivo, não requer entender padrões dos *intents*, além de utilizar algoritmos de inteligência artificial do Watson para identificar padrões de entrada parecidos com os *intents* criados pelo usuário para poder generalizar. Entretanto, ele não executa algoritmos criados pelo usuário, nem tem proatividade e seu foco é na conversação. O Dialogflow é uma ferramenta bem aprimorada para trabalhar com agentes de conversação e é orientado a fluxos de conversas estilo fluxograma, algo que poderia se comparar em parte com os contextos da Arisa Nest. Ele possui também um recurso de proatividade através de eventos, mas ele serve para iniciar um contato com o usuário por algum fluxo de diálogo. A forma dele trabalhar com execução de algum algoritmo é por meio do recurso de *fulfillment*, sendo necessário a configuração de um *webhook* para executar um script em javascript sob Node.js. Entretanto, mesmo se mantendo nos padrões da grande estrutura da Google, há uma certa complexidade para se trabalhar com os scripts e requer um conhecimento mais aprimorado. Para trabalhar com scripts na Arisa Nest basta conhecer as funções de acesso à plataforma, um pouco da linguagem de programação Lua, largamente utilizada em jogos digitais. Além disso, eles podem ser escritos, testados e executados na própria plataforma. A Alexa é uma plataforma bastante completa e tem a vantagem de utilizar toda a estrutura da Amazon, incluindo o AWS para a criação de scripts, hardwares e dispositivos. Ela já vem com uma grande gama de *skills* e o usuário pode adicionar outros, via ferramenta de compartilhamento ou produzidas pelo próprio usuário. Também há proatividade por meio de eventos. Contudo, seu nível de complexidade está no mesmo nível, ou maior, que o Dialogflow.

4 Casos e Testes

A Arisa Nest foi testada e avaliada em quatro casos. O primeiro foi um projeto de estágio no curso de Engenharia de Controle e Automação da UFSC, em 2018, pelo aluno Ricardo Ventura e aplicado na secretaria acadêmica do curso de Pós-Graduação em Engenharia Mecânica (PosMec) [Ventura 2018]. Para este projeto foi utilizada a versão anterior da plataforma, apenas com os recursos de conversação via árvore de contextos. O aluno passou por um breve treinamento para a utilização do sistema, criação de contextos, diálogos, estrutura dos padrões e criou uma base de conversação sem maiores problemas. A maior dificuldade foi na aquisição, filtragem e estruturação das informações para inserir no sistema. O bot foi avaliado pelas respostas preenchidas em um formulário na interface web do bot e, conforme os resultados, para 70% dos usuários, o bot respondeu corretamente aos questionamentos, 60% responderam que

tornou mais fácil o acesso à informação da PosMec e para 60% o sistema contribuiu para padronizar as informações. Atualmente o bot ainda encontra-se em funcionamento e foi migrado para a versão atual da plataforma Arisa Nest.

O segundo caso foi o bot Riunita, desenvolvido em 2017 e 2018 como projeto de pesquisa do Artigo 170[3] pelo acadêmico Laércio de Sant'Anna Filho para o Repositório Institucional (RIUNI[4]) da Universidade do Sul de Santa Catarina [Sant'Anna Filho 2018]. Foi criado um módulo no canto inferior do site do RIUNI. O projeto focou na resolução de dúvidas dos usuários. Houve um treinamento de 30 minutos para a utilização da plataforma e formas de criação dos contextos e diálogos. Mas, foi na prática que o aluno compreendeu a estrutura de uma base de conversação baseada em padrões de entrada e contextos. Atualmente o bot se encontra em funcionamento no Repositório Institucional e os resultados do projeto foram apresentados na Jornada Unisul de Iniciação Científica (JUNIC) de 2018 [UnisulHoje, 2018].

O terceiro caso foi o CMBOT, desenvolvido pelos acadêmicos Itamar Ghidini e Winicius Mattos [Ghidini e Mattos 2018] como Trabalho de Conclusão de Curso (TCC) no curso de Sistemas de Informação na Universidade do Sul de Santa Catarina (UNISUL) e aplicado no ambiente de desenvolvimento da empresa em que um dos alunos envolvidos no TCC era funcionário. Sua finalidade seria recepcionar o cliente (usuário dos sistemas da empresa) e auxiliá-lo a completar o atendimento até seu fechamento. No site de atendimento foi criado um módulo web no canto inferior direito do site para interagir com o bot. O projeto foi desenvolvido na versão atual da plataforma, mas, além da base de conversação, usaram apenas o recurso de crenças para tratar algumas informações dos clientes. Foi dado um pequeno treinamento aos alunos, mas eles tiveram acesso a um tutorial na web, facilitando o aprendizado. O maior problema encontrado por eles foi a falta de documentação interna e dos atendimentos feitos pelos funcionários, indicando a necessidade de se documentar o processo e as vantagens de haver um sistema que gere registros automatizados, como no caso do chatbot. O sistema foi avaliado por um conjunto de usuários (6 desenvolvedores e 3 atendentes de suporte) e foi aplicado um questionário. Conforme os resultados, 78% responderam que o chatbot deixa a forma de atendimento mais eficiente e assertiva, 66% concordaram que a ferramenta facilita, pois permite a abertura de chamado fora do horário do trabalho, 66% disseram que o bot deve ser proativo para interagir com o cliente no caso dele ficar muito tempo na seção sem atividade, 100% concordaram que a tecnologia é importante para o atendimento, resolvendo pelo menos as dúvidas mais simples do cliente. Por fim, 89% disseram que a base de conhecimento está totalmente relacionada a eficiência, o que demonstra a importância de uma boa construção dessa base. Foi também identificado que para os atendimentos mais diretos e simples foi preferido o uso do chatbot pela agilidade e disponibilidade da ferramenta, sem a necessidade da espera em fila do atendimento.

O quarto caso foi o Projeto de Final de Curso (PFC) do aluno Ricardo Ventura [Ventura 2019], do curso de Engenharia de Controle e Automação da UFSC e se deu por uma evolução do seu estágio acadêmico. A avaliação produzida naquele projeto

[3] Artigo 170: programa de bolsas de pesquisas da Secretaria da Educação do Estado de Santa Catarina para incentivar potenciais talentos entre alunos de cursos de graduação com um grau de carência econômica e financeira.

[4] RIUNI: repositório de produções intelectuais da comunidade acadêmica da Unisul

trouxe questionamentos para melhorias no processo de atendimento aos usuários de chatbots. O aluno usou todos recursos disponíveis na atual versão da plataforma Arisa Nest, indo mais no viés da assistência virtual do que especificamente de um chatbot. A intenção foi criar um bot para executar ações para os usuários, automatizar processos e, inclusive, com integração à serviços web padrão SOAP, desenvolvidos pelo próprio aluno, providos pela UFSC e por outras empresas ou ecossistemas.

Após analisar algumas situações, foram modelados os processos, definindo os diálogos, as crenças, scripts e implementação dos comportamentos. Entre algumas atividades do bot estão: informar o índice de aproveitamento semestral acumulado do aluno, as aulas de um dia na semana, cardápio do dia no restaurante universitário, a nota das disciplinas cursadas, alterar e-mail cadastrado, fazer matrícula, fornecer histórico semestral e, como proatividade, informar diariamente as aulas do usuário do dia seguinte. Outra atividade do bot, mais para um coordenador de estágios e PFC, foi automatizar o processo analisar alguns critérios do termo de compromisso de estágio em que é necessário pesquisar informações em diferentes locais, findando o resultado das análises com o envio de um e-mail ao coordenador e ao aluno analisado.

Também foi criado um outro cenário para automatizar parte do atendimento aos clientes no processo de venda em uma empresa fictícia. Este cenário está mais para testes na plataforma para verificar a aplicabilidade de cenários do Operador 4.0 na mesma. Este se daria pela utilização de assistente virtual para auxiliar um operador de equipamentos industriais em um ambiente inteligente, nos moldes da indústria 4.0, em que o assistente pode tomar algumas decisões e realizar algumas tarefas de forma independente. Esse cenário foi uma adaptação simplificada de um caso mais complexo abordado no artigo "*Softbots Supporting the Operator 4.0 at Smart Factory Environment*" [Rabelo, Romero e Zambiasi 2018]. Segundo Ventura [2019], com a utilização dos recursos da plataforma, a implementação foi simples. O bot recebia uma informação que iniciava o processo por meio da execução de um script. A busca de informações se dava através de chamadas na camada de serviços e, conforme a orquestração dos serviços e atividades implementadas no script Lua, o resultado cumpriu com as diretrizes pré-programadas e concluindo a tarefa. Para ele, "todas as funcionalidades da plataforma Arisa Nest se mostraram bastante robustas e complementares entre si. A possibilidade de consumir serviços web, executar scripts, armazenar crenças" e automatização com os comportamentos possibilita "uma grande variedade de implementações em diferentes cenários".

Atualmente a plataforma está sendo utilizada para quatro outros projetos em andamento: um PFC do curso de Automação da UFSC pelo acadêmico Murilo Rodrigues Padilha Leite, continuação do projeto do Ricardo Ventura na PosMec; um Projeto PUIC do aluno João Ricardo dos Santos Kleine Buckstegge na UNISUL; um projeto de mestrado no Programa de Pós-Graduação em Engenharia de Automação e Sistemas (PGEAS) na UFSC pelo aluno Brunno Abner Machado; e em um teste de utilização de assistentes virtuais para o PJe (Processo Judicial Eletrônico) do Conselho Nacional de Justiça (CNJ).

5 Conclusões e Trabalhos Futuros

Este artigo apresentou uma plataforma de assistentes virtuais chamada Arisa Nest, no modelo arquitetural *Platform as a Service* (PaaS) de disponibilização baseado na

nuvem, que provê recursos de gerenciamento de uma base de conversação, gerenciamento de informações na forma de crenças, scripts para execução de algoritmos, acesso a serviços externos padrão SOAP e REST e comportamentos proativos. Foram apresentados alguns exemplos de plataformas atuais no mercado para identificar alguns recursos desses tipos de ferramenta e para fazer uma análise comparativa com a plataforma Arisa Nest.

Foi feita uma apresentação da plataforma, alguns recursos, maneira de utilizá-los e, como proposta complementar, um caso foi apresentado como forma de utilização da plataforma também como um ambiente de criação de agentes, suportando todo seu ciclo de vida, comportamento e acesso a outros recursos, dispositivos e informações por meio do consumo de serviços web. Em seguida, foram apresentados alguns casos que utilizaram a plataforma como base para projetos de pesquisa e trabalhos de fim de curso de graduação. Cada caso foi avaliado em termos da utilização da plataforma. No contexto do que foi apresentado, testado e avaliado, a Plataforma se mostrou como uma ferramenta com certa maturidade e robustez, tanto para chatbots como para assistentes virtuais e agentes, resolvendo os casos testados de forma satisfatória.

A Arisa Nest é uma ferramenta que se encontra em constante evolução, serve como ambiente de pesquisa, tanto nas teorias e tecnologias utilizadas para seu desenvolvimento com a agregação de algoritmos e novos recursos, como para os projetos que a utilizam como plataforma para assistentes virtuais. Quanto à utilização da plataforma, alguns projetos já citados encontram-se em processo de criação e avaliação. Quanto ao seu desenvolvimento em si, ainda estão sendo feitas várias implementações para que a plataforma fique mais fácil de usar, criação de mais funções de acesso aos recursos da plataforma pelos scripts e comportamentos, funções para o bot criar, excluir e editar contextos, diálogos e crenças da sua própria base de conhecimento, se auto-clonar, criar outros bots e migrar para outros servidores e, ainda em análise, a adição de um recurso/editor de comportamentos baseado em agentes BDI (*Beliefs, Desire, Intentions*).

Referências

Abu Shawar, B.; Atwell E.. (2015) “ALICE Chatbot: Trials and Outputs”. Comp. y Sist., México, v. 19, n. 4, p. 625-632.

ALEXA. (2019) “Developer documentation: browse the technical documentation for Alexa, Amazon Appstore, and devices”, https://developer.amazon.com/documentation/, Março/2019.

AWS. (2019) “AWS Lambda”, https://aws.amazon.com/pt/lambda/, Março/2019.

DIALOGFLOW. (2019) “Dialogflow: build natural and rich conversational experiences”, https://dialogflow.com, Março/2019.

Ghidini, I.; Mattos, W.W.. (2018) “Desenvolvimento e aplicação de um chatbot para auxiliar o atendimento ao cliente”. Universidade do Sul de Santa Catarina, Sistemas de Informação, Trabalho de Conclusão de Curso. https://www.riuni.unisul.br/handle/12345/5986, Março/2019.

Markoff, J.. (2008) “A Software Secretary That Takes Charge”, In: New York Times, http://www.nytimes.com/2008/12/14/business/14stream.html?_r=1&scp=7&sq=personal%20assistant&st=cse, Março/2019.

Mazon, S.. (2018) “Desenvolvendo chatbots com Watson conversation”, https://www.ibm.com/developerworks/br/library/desenvolvendo-chatbots-com-watson-conversation/index.html, Março/2019.

Rabelo, R.J.; Romero, D; Zambiasi, S.P.. (2018) “Softbots Supporting the Operator 4.0 at Smart Factory Environments”. Moon I., Lee G., Park J., Kiritsis D., von Cieminski G. (eds) Advances in Production Management Systems. Smart Manufacturing for Industry 4.0. APMS 2018. IFIP Advances in Information and Communication Technology, vol 536. Springer, Cham.

Russel, S.; Norvig.. (2013) “Inteligência Artificial”. 3ª Ed. Tradução da terceira edição. Rio de Janeiro: Editora Campus / Elsevier.

Sant’Anna Filho, L. (2018) “Estratégia para apoiar o povoamento do repositório institucional de uma universidade de santa catarina utilizando um assistente virtual”, Universidade do Sul de Santa Catarina, Sistemas de Informação, Relatório de Estágio.

Tingiris, S.. (2018) “Amazon Alexa development 101”, https://youtu.be/QkbXjknPoXc, Março/2019.

UnisulHoje. (2018) “Assistente digital que auxilia repositório institucional é apresentada na Junic”, http://hoje.unisul.br/assistente-digital-que-auxilia-repositorio-institucional-e-apresentada-na-junic/, Março/2019.

Weizenbaum, J.. (1966) "Eliza - a computer program for the study of natural language communication between man and machine". Communications of the ACM, ACM, v. 9, n. 1, p. 36–45.

Ventura, R.. (2018) “Desenvolvimento de um chatbot para apoio a secretarias de cursos de universidades: um caso no programa de Pós-Graduação em Engenharia Mecânica”. Universidade Federal de Santa Catarina, Departamento de Automação e Sistemas. Relatório de Estágio.

Ventura, R.. (2019) “Desenvolvimento de um assistente virtual híbrido com propriedades de chatbot e de automação de processos de negócios”, Universidade Federal de Santa Catarina, Departamento de Automação e Sistemas, Projeto de Final de Curso.

Zambiasi, S.P.; Rabelo, R.J.. (2012) A Proposal for Reference Architecture for Personal Assistant Software based on SOA. IEEE Latin America Transactions, 10(1):1227-1234.

RESUMOS ESTENDIDOS

Uma Proposta Preliminar de Síntese Automática de Normas para Sistemas Multiagente

Jhonatan Alves, Jomi Fred Hübner, Jerusa Marchi

[1]Programa de Pós-Graduação em Engenharia de Automação e Sistemas
Universidade Federal de Santa Catarina
Caixa Postal 88040-900 – Florianópolis -– SC – Brasil

jhonatan.alves@posgrad.ufsc.br, {jomi.hubner,jerusa.marchi}@ufsc.br

Resumo. *Normas têm sido usadas em sistemas multiagentes para regular a sociedade dos agentes na forma de regras que, em certos contextos, impõem restrições sobre os seus comportamentos. Sob estas circunstâncias, este trabalho apresenta a proposta preliminar de um framework para a síntese automática de normas. Como resultados esperados pretende-se que as normas obtidas sejam capazes de evitar conflitos no sistema mantendo os seus objetivos alcançáveis e as ações da sociedade coerentes.*

Abstract. *Norms have been used in multiagent systems to regulate the agent society in the form of rules that, in certain contexts, impose restrictions on their behavior. Under these circumstances, this paper presents a preliminary proposal of a framework for the automated synthesis of norms. As expected, it is hoped that the obtained norms will be able to avoid conflicts in the system keeping its objectives achievable and the actions of the society coherent.*

1. Introdução

Em sistemas multiagente (SMA) a autonomia é a principal característica de um agente, pois lhe confere a capacidade de decidir como e quando interagir com os demais membros da sociedade para contemplar os seus objetivos [Wooldridge and Jennings 1995]. Entretanto, as decisões tomadas pelos agentes podem causar conflitos no sistema dirimindo a sua eficiência e estabilidade [Weiss 1999]. Sob a luz desses fatos, uma questão crucial no desenvolvimento dos SMA é *"como garantir a eficiência e preservar propriedades desejáveis do SMA de modo que a autonomia dos agentes seja respeitada?"* [Boella et al. 2006].

Para isso, é necessário que o comportamento dos agentes seja regulado através de um mecanismo de coordenação para que ajam de forma coerente no sistema [Huhns and Stephens 1999]. Desde o trabalho precursor de [Shoham and Tennenholtz 1992] as normas têm sido usadas em SMA para regular a sociedade na forma de regras que, em certos contextos, governam os comportamentos dos agentes. O estudo das normas neste campo é extenso e variado. Entre os tópicos de estudo tem-se a síntese que se refere ao processo da criação de um conjunto de normas para o sistema [Frantz and Pigozz 2018]. Neste sentido, este trabalho apresenta a proposta preliminar de um framework que monitora o comportamento dos agentes e, conforme conflitos são detectados, sintetiza normas para garantir a estabilidade e o alcance dos objetivos do sistema.

Este trabalho está organizado da seguinte forma: na seção 2 a síntese de normas em SMA e trabalhos correlatos são apresentados e discutidos; na seção 3 a proposta deste trabalho é apresentada; e, por fim, na seção 4 as conclusões e direções futuras são apresentadas.

2. Síntese de Normas e Trabalhos Correlatos

A síntese de normas pode ser classificada, basicamente, como *off-line* ou *on-line*. A síntese off-line ocorre antes da execução do sistema e tem como principal vantagem permitir que a sociedade dos agentes seja regulada desde o início das suas interações, uma vez que as normas são obtidas previamente à incialização do sistema. Ainda, a síntese off-line se distingue entre as abordagens manuais e automáticas. A abordagem manual é realizada pelo projetista, ao passo que a abordagem automática é realizada por um mecanismo computacional.

Por outro lado, a síntese on-line é automática e ocorre em tempo de execução do sistema. Este tipo de síntese se distingue entre as abordagens por emergência e observação. Na abordagem por emergência o projetista do sistema fornece, em tempo de projeto, um conjunto de ações a partir do qual os agentes decidem através de diversas interações qual ação será adotada como obrigatória por toda a sociedade. Na abordagem por observação a síntese consiste em determinar em quais circunstâncias as normas devem ser criadas. Para isso, faz uso de um mecanismo computacional que, ao identificar conflitos durante a interação dos agentes, sintetiza normas para evitá-los em novas situações.

Dentro desse contexto, buscou-se na literatura trabalhos de síntese de normas em SMA, onde foram analisados 15 trabalhos, dentre os quais 6 são abordagens off-line e 9 on-line. Dentre os trabalhos off-line [Shoham and Tennenholtz 1992, Shoham and Tennenholtz 1995, van der Hoek et al. 2007, Alrawagfeh and Meneguzzi 2015] são manuais. A síntese manual é propensa a erros e complexa de ser realizada para sistemas com grandes espaços de estados. Os trabalhos [Christelis 2011, Morales et al. 2018] são off-line e automáticos, porém, ambos compartilham do problema de que certas normas têm a chance de nunca se tornarem ativas, pois em tempo de execução do sistema os agentes podem não apresentar os comportamentos para os quais aquelas normas foram sintetizadas, o que pode implicar em um esforço computacional desnecessário para sintetizar algumas normas.

Os trabalhos [Kittock 1995, Walker and Wooldridge 1995, Onn and Tennenholtz 1997, Epstein 2001, Boella and van der Torre 2007, Savarimuthu et al. 2008, Griffiths and Luck 2011, Mahmoud et al. 2015] realizam a síntese sob a perspectiva da emergência. Nesse tipo de abordagem os agentes devem possuir maquinário para sintetizarem a norma além de se disporem a participar do processo de emergência, o que nao pode ser garantido. Além do mais, tais trabalhos se limitam a encontrar apenas uma norma. O trabalho de [Morales et al. 2013] visa obter as normas por observação, deste modo as normas são sintetizadas para o comportamento corrente dos agentes. Entretanto, para que as normas sejam sintetizadas é necessário que ocorram, primeiramente, conflitos no sistema. A cada novo conflito, uma norma é sintetizada. Ao longo da execução do sistema, várias normas são obtidas e inseridas em

uma rede hierarquica de generalização/especialização, onde as normas mais gerais se tornam visíveis aos agentes. Entretanto, para isso, várias normas específicas são geradas.

Visto que a síntese off-line pode gerar normas não úteis ao sistema e que a síntese por emergência exige o engajamento dos agentes no processo de criação, este trabalho tem como objetivo responder a seguinte pergunta: *"Como sintetizar automaticamente normas para SMA que estejam melhor direcionadas ao comportamento corrente dos agentes e que viabilizem o alcance dos objetivos do sistema*?".

3. Proposta

Para responder à pergunta de pesquisa, este trabalho tem como objetivo desenvolver um framework que sintetiza normas em modo on-line para SMA. Como abordado anteriormente, a escolha pela síntese on-line se dá pelo fato de que através deste modo é possível obter um conjunto de normas que regulam o comportamento vigente dos agentes. Em suma, a proposta preliminar do framework funciona em 4 etapas: (i) construção da história corrente dos estados do sistema percorridos pelos agentes conforme são monitorados (onde a história é um subespaço do espaço de estados do sistema); (ii) detecção de conflitos no estado corrente do sistema; (iii) análise da alcançabilidade dos objetivos do sistema antes de uma nova norma ser criada; (v) síntese de uma norma para evitar o conflito detectado. A nova norma é, então, disponibilizada aos agentes. A visão geral do framework está ilustrada na figura 1.

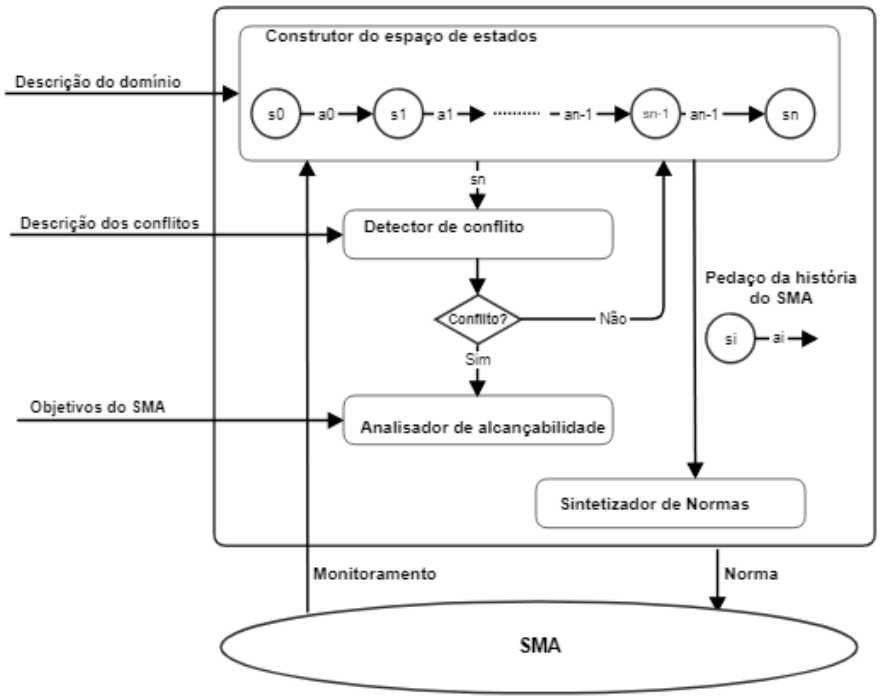


Figura 1. Framework proposto.

3.1. Definição conceitual

A primeira etapa é realizada pelo módulo *construtor de história* que constrói a história corrente do SMA à medida que o monitora. Uma história é uma sequência finita de transições de estados que parte do estado incial do sistema até o estado onde o sistema

se encontra em equilíbrio. Por estado de equilíbrio entende-se o estado a partir do qual uma estimativa indica que não haverá mais conflitos no SMA. Uma transição que parte de um estado s' para um estado s'' denota a ação executada pelo agente em s' e s'' denota o estado do SMA alcançado após a execução de tal ação. Na história, o estado corrente é representado por s_n.

A segunda etapa é iniciada pelo módulo *detector de conflitos* cuja função é averiguar a existência de algum conflito no estado corrente da história do SMA. Se houver, então há a necessidade de que uma norma seja sintetizada e dispobibilizada aos agentes para que eles evitem-no futuramente. Caso contrário o fluxo de execução do framework retorna à primeira etapa.

Uma abordagem ingênua para evitar que o conflito detectado seja evitado pelos agentes em situações futuras é sintetizar uma norma que proíbe a execução da ação que fez com que o sistema transitasse do estado s_{n-1} para o estado s_n quando um certo contexto for satisfeito. Entretanto, há a possibilidade de que a norma criada nessas condições impeça o alcance de algum objetivo do sistema. Logo, para evitar a criação de uma norma obstrutora de objetivos o framework dá início à terceira etapa que é realizada pelo módulo *analisador de alcançabilidade*.

O analisador realiza backtracking sobre a história do sistema a fim de determinar o primeiro estado predecessor a s_n a partir do qual pode-se satisfazer os objetivos. Diferentes técnicas baseadas em planejamento ou grafos podem ser utilizadas pelo verificador em seu processo de backtracking. Uma vez que o estado predecessor foi encontrado dá-se início à quarta e última etapa do framework que é realizada pelo módulo *sintetizador de normas*.

O sintetizador recebe um recorte da história do sistema, mais especificamente o estado predecessor obtido na etapa anterior e a ação executada a partir daquele estado. Com o intuito de que a norma seja genérica (aplicabilidade em diferentes estados), o sintetizador obtém a norma como uma regra abstrata, onde os termos que compõem a norma (isto é, os termos do contexto e da ação) são predicados com variáveis livres. Desta forma, a norma não se aplica a um estado específico e através de um processo de unificação o agente pode verificar a aplicação da norma no estado no qual que se encontra. Em seguida, a norma é disponibilizada em uma base para que os agentes possam acessá-la a fim de raciocinarem sobre restrições comportamentais que elas possam vir aplicar.

3.2. Definição Formal

A seguir apresenta-se um conjunto de definições que formalizam os conceitos envolvidos no framework proposto neste trabalho.

Definição 1 *Seja* $M = \{Ag, Ac, S, P, At, G, T\}$ *um sistema multiagente, onde* $Ag = \{ag_1, ag_2, \ldots, ag_n\}$ *denota um conjunto de agentes,* $Ac = \{ac_1, ac_2, \ldots, ac_m\}$ *um conjunto de ações que os agentes podem executar,* P *um conjunto de predicados,* At *um conjunto de átomos proposicionais definidos sobre* P*,* S *o conjunto de estados do sistema,* $G = \{g_1, g_2, \ldots, g_k\}$ *um conjunto de objetivos do sistema com* $g_i \subset At$ *e* $T : S \times Ac \rightarrow S$ *uma função de transição de estados que indica que a partir do estado* $s \in S$ *pela aplicação de uma ação* $a \in Ac$ *o sistema transita para um estado* $s' \in S$*.*

Ainda, $S \subseteq 2^{At}$ e assume-se sobre a construção dos estados a hipótese do mundo fechado, isto é, átomos proposicionais que não estão contidos em um estado $s \in S$

são considerados como falsos e o mundo evolui de forma determinística. A função de transição T indica que apenas uma ação pode ser realizada a cada estado.

Definição 2 *Seja $C = \{c_1, c_2, \ldots, c_n\}$ um conjunto de conflitos, onde c_i é uma expressão lógica cujos termos pertencem a P e faz uso dos conectivos da lógica de primeira ordem.*

Um estado $s \in S$ é um estado de conflito caso $s \models c_i$ (a condição c_i é verdadeira em s). Nesse caso, diz-se que s detrimenta o alcance dos objetivos do sistema M.

Definição 3 *Uma norma é um par $\langle \varphi, \theta(ac) \rangle$, onde $\varphi \subset P$ denota o contexto no qual a norma se torna ativa, θ é o operador deôntico* proibido *e $ac \in Ac$ é a ação que é proibida de ser executada no contexto φ.*

Sendo os conflitos e as normas definidos sobre o conjunto P faz-se com que ambos sejam genéricos. Através de um processo de unificação é possível verificar a aplicabilidade destes elementos aos estados do SMA.

4. Conclusão

Este trabalho apresentou a proposta preliminar de um novo framework para a síntese automática de normas. Percebe-se que as seguintes limitações necessitam de uma solução prioritária para que a solução apresentada possa avançar: (i) como estipular um método para estimar o estado a partir do qual o SMA entra em equilíbrio; (ii) em caso de conflito, como determinar a ação responsável ao permitir que os agentes executem ações em paralelo; (iii) como avaliar se uma dada norma é eficiente em evitar o conflito para o qual ela foi sintetizada; (v) como determinar de modo eficiente a alcançabilidade dos objetivos do sistema.

Referências

Alrawagfeh, W. and Meneguzzi, F. (2015). Utilizing permission norms in bdi practical normative reasoning. In *Revised Selected Papers of the International Workshops on Coordination, Organizations, Institutions, and Norms in Agent Systems X - Volume 9372*, pages 1–18, New York, NY, USA. Springer-Verlag New York, Inc.

Boella, G. and van der Torre, L. (2007). Norm negotiation in multiagent systems. *International Journal of Cooperative Information Systems (IJCIS) Special Issue: Emergent Agent Societies*, 16(2).

Boella, G., van der Torre, L., and Verhagen, H. (2006). Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, 12(2):71–79.

Christelis, G. D. (2011). *Automated Norm Synthesis in Planning Environments*. PhD thesis, University of Edinburgh.

Epstein, J. M. (2001). Learning to be thoughtless: Social norms and individual computation. *Computational Economics*, 18(1):9–24.

Frantz, C. K. and Pigozz, G. (2018). Modeling norm dynamics in multi-agent systems.

Griffiths, N. and Luck, M. (2011). Norm diversity and emergence in tag-based cooperation. In De Vos, M., Fornara, N., Pitt, J. V., and Vouros, G., editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems VI*, pages 230–249, Berlin, Heidelberg. Springer Berlin Heidelberg.

Huhns, M. N. and Stephens, L. M. (1999). Multiagent systems. chapter Multiagent Systems and Societies of Agents, pages 79–120. MIT Press, Cambridge, MA, USA.

Kittock, J. (1995). Emergent conventions and the structure of multi–agent systems. In *Lectures in Complex systems: the proceedings of the 1993 Complex systems summer school, Santa Fe Institute Studies in the Sciences of Complexity Lecture Volume VI, Santa Fe Institute*, pages 507–521. Addison-Wesley.

Mahmoud, S., Griffiths, N., Keppens, J., Taweel, A., Bench-Capon, T. J. M., and Luck, M. (2015). Establishing norms with metanorms in distributed computational systems. *Artificial Intelligence and Law*, 23(4):367–407.

Morales, J., Lopez-Sanchez, M., Rodriguez-Aguilar, J. A., Wooldridge, M., and Vasconcelos, W. (2013). Automated synthesis of normative systems. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 483–490, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Morales, J., Wooldridge, M., Rodriguez-Aguilar, J. A., and Lopez-Sanchez, M. (2018). Off-line synthesis of evolutionarily stable normative systems. *Autonomous Agents and Multi-Agent Systems*, 32(5):635–671.

Onn, S. and Tennenholtz, M. (1997). Determination of social laws for multi-agent mobilization. *Artificial Intelligence*, 95(1):155 – 167.

Savarimuthu, B. T. R., Cranefield, S., Purvis, M., and Purvis, M. (2008). Role model based mechanism for norm emergence in artificial agent societies. In Sichman, J. S., Padget, J., Ossowski, S., and Noriega, P., editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, pages 203–217, Berlin, Heidelberg. Springer Berlin Heidelberg.

Shoham, Y. and Tennenholtz, M. (1992). On the synthesis of useful social laws for artificial agent societies (preliminary report). In *AAAI*, pages 276–281. AAAI Press / The MIT Press.

Shoham, Y. and Tennenholtz, M. (1995). On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 73(1):231 – 252. Computational Research on Interaction and Agency, Part 2.

van der Hoek, W., Roberts, M., and Wooldridge, M. (2007). Social laws in alternating time: effectiveness, feasibility, and synthesis. *Synthese*, 156(1):1–19.

Walker, A. and Wooldridge, M. (1995). Understanding the emergence of conventions in multi-agent systems. In *International Conference on Multiagent Systems*, ICMAS'95, pages 384–389. MIT Press.

Weiss, G., editor (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152.

Implementação de Diálogos Argumentativos em Sistemas Multiagentes

Adler M. Cachuba, Ayslan T. Possebom

Instituto Federal do Paraná - IFPR
R. José F. Tequinha, 1400, Jardim das Nações – 87703-536 – Paranavaí – PR – Brasil

adlermateuself@gmail.com, ayslan.possebom@ifpr.edu.br

Abstract. *Argumentative dialogues consist in a dynamic process where two or more agents can, by means of an interaction among them, exchange information using arguments. These arguments represent the agents' opinions about certain information (claim), having a justification for those opinions (premise). This work introduces a proposal for computational implementation so that agents can communicate using argumentation. As a result, we have a set containing all the related arguments sent about some issue where different argumentation semantics can be applied.*

Resumo. *Diálogo argumentativo consiste em um processo dinâmico onde dois ou mais agentes podem, por meio de uma interação entre eles, trocar informações por meio de argumentos. Estes argumentos representam as opiniões dos agentes sobre determinada informação (conclusão do argumento), contendo uma justificativa para esta opinião (premissa do argumento). Este trabalho apresenta uma proposta de implementação computacional para que agentes possam se comunicar utilizando argumentação. Como resultado, obtêm-se uma tabela contendo todos os argumentos emitidos onde diferentes semânticas de argumentação podem ser aplicadas.*

1. Introdução

Quando um grupo de agentes pertencem a um determinado ambiente, frequentemente estes agentes se comunicam trocando mensagens entre eles. Estas mensagens são transmitidas do emissor aos seus receptores com o intuito de solicitar (ou fornecer) por alguma informação ou execução de alguma ação [Amgoud, Maudet and Parsons 2002]. Para que agentes se comuniquem e tomem decisões em conjunto (negociação, determinação de ações conjuntas, determinação do grau de consenso do grupo, etc.), as mensagens enviadas devem ter uma sequência que indica o fluxo do diálogo entre os agentes, ou seja, a conversação entre os agentes. Desta forma, podemos saber se as mensagens transmitidas foram aceitas ou rejeitadas pelos receptores, as contrapropostas, entre outros tipos de mensagens [Amgoud, Maudet and Parsons 2002].

O conteúdo das mensagens transmitidas pode representar um argumento, sendo a opinião do agente a respeito do que se está sendo discutido juntamente com a justificativa para esta opinião. De acordo com [Dung 1995], a argumentação consiste na troca de argumentos entre os agentes e na aplicação de semânticas para determinar os argumentos mais aceitos pelo grupo.

Para Greco (2015), um diálogo argumentativo é um processo social de interação entre os agentes que tenta resolver desacordos em um grupo. Na área da Inteligência Artificial, quando a argumentação é aplicada a apenas um agente, ela permite tanto a escolha de um conjunto de argumentos que se sustentam, quanto a escolha do melhor argumento. Já para sistemas multiagentes, a argumentação pode ser utilizada na escolha do melhor argumento a ser enviado ao grupo, ou então na criação de argumentos que sustentem ou que rejeitem um determinado ponto de vista.

Diferentes sistemas de argumentação foram propostos, tais como sistemas para argumentação abstrata [Dung 1995][Caminada 2008] e argumentação estruturada [Besnard and Hunter 2009][Amgoud and Prade 2009]. Apesar da argumentação estar em constante evolução, poucas implementações computacionais podem ser aplicadas em sistemas reais. Desta forma, este trabalho tem o objetivo de desenvolver uma implementação computacional onde um grupo de agentes pode se comunicar, trocando argumentos que representem suas opiniões a respeito dos argumentos transmitidos em um diálogo, de forma que no final do diálogo, o conjunto de argumentos possa ser analisado por alguma semântica de argumentação. Estas semânticas representam a aceitabilidade dos argumentos e podem indicar, por exemplo, uma alternativa de decisão preferida pelo grupo ou um conjunto de argumentos que melhor justificam a preferência ou discordância do grupo sobre determinada informação.

2. Referencial Teórico

Seja AG um conjunto finito de agentes argumentativos onde $AG = \{ag_1, \ldots, ag_n\}$ com $n > 0$, e $ag_i \in AG$ representando um agente. Um agente argumentativo é responsável pela criação de argumentos utilizando uma linguagem base que representa seus conhecimentos. Baseando-se no trabalho de Possebom, Morveli e Tacla (2016), um agente argumentativo pode ser definido como:

Definição 1: Um agente ag_i é uma tripla $< K, A, S >$ onde K representa a base de conhecimentos deste agente, A representa o argumento que está sendo discutido pelo grupo de agentes em um determinado instante e S representa um conjunto de argumentos a serem emitidos ao grupo.

O tipo de argumentação realizada pelos agentes argumentativos usa o conceito de argumento dedutivo [Besnard and Hunter 2009]. Um argumento é formado por um par $< \Phi, \alpha >$ onde Φ representa a premissa do argumento, ou seja, uma justificativa ou explicação para o argumento, e α representa a conclusão deste argumento. Tem-se que (i) a premissa do argumento não pode ser inconsistente, (ii) a premissa deve, por meio de um mecanismo de inferência, levar a conclusão do argumento, e (iii), a premissa deve ser um subconjunto mínimo de K (base de conhecimento do agente argumentativo).

Quando algum agente argumentativo transmite um argumento ao grupo, este argumento deve ser armazenado em suas respectivas bases A. Caso os agentes possuam contra-argumentos, eles devem ser armazenados em suas respectivas bases S. Um contra-argumento representa um conflito com outros argumentos já apresentados no diálogo. Estes conflitos podem gerar duas formas de ataques entre argumentos [Parsons and McBurney, 2003]: *undercut* e *rebuttal*. Em resumo, um argumento $\arg_i$ ataca por *undercut* um argumento $\arg_j$ quando a conclusão de $\arg_i$ é logicamente equivalente à negação de alguma informação presente na premissa de $\arg_j$. Tem-se que um argumento

$\arg_i$ ataca por *rebuttal* um argumento $\arg_j$ quando a conclusão de $\arg_i$ é logicamente equivalente à negação da conclusão de $\arg_j$.

Exemplo 1: considerando o conhecimento do agente representado por fórmulas em lógica proposicional, temos que $\arg_i =< \{a, a \rightarrow b\}, b >$ ataca $\arg_j =< \{\neg b, \neg b \rightarrow c\}, c >$ por *undercut*, enquanto que $\arg_i$ ataca $\arg_k =< \{d, d \rightarrow \neg b\}, \neg b >$ por *rebuttal*.

Para que a troca de mensagens entre os agentes possa ser controlada, utiliza-se um agente chamado mediador. Ele é responsável por controlar a troca de mensagens entre os agentes argumentativos, sincronizando a troca de mensagens e determinando quais argumentos emitidos são válidos para o diálogo atual. Um argumento é válido quando ele ainda não foi discutido pelo grupo até um determinado instante.

Definição 2: Um agente mediador med é uma tripla $< DT, WB, AGENDA, AG >$ onde DT representa uma tabela de diálogo que armazena os argumentos discutidos pelo grupo de agentes argumentativos, WB representa uma fila de agentes argumentativos que desejam emitir argumentos ao grupo, $AGENDA$ representa uma lista de argumentos emitidos por um determinado agente argumentativo e AG consiste em um conjunto de agentes argumentativos que pertencem a um diálogo.

O agente med realiza sequência de tarefas apresentadas no Algoritmo 1:

Algoritmo 1 Processo de diálogo argumentativo
Entrada: Conjunto de agentes argumentativos, Conjunto de assuntos a serem discutidos
Saída: Tabela de diálogo

```
1  procedimento run
2      para cada assunto faça:
3          configurar_diálogo
4          informar_assunto
5          questionar_ataques
6          solicitar_inscrição
7              para cada agente inscrito
8                  solicitar_argumentos
9                  para cada argumento enviado
10                     validar_argumento
11                     informar_argumento
12                     questionar_ataques
13                     solicitar_inscrição
14                 fim_para
15             fim_para
16     fim_para
17 fim_procedimento
```

A configuração do diálogo (linha 3) envolve a inicialização da DT, WB e $AGENDA$ para que o mediador possa receber argumentos e inscrições. Em seguida, o mediador informa o assunto que será discutido ao grupo (linha 4) e os agentes argumentativos buscam por contra-argumentos, armazenando-os em suas respectivas bases S (linha 5). Agentes cujas bases $S \neq \emptyset$ possuem argumentos para serem emitidos ao grupo e informam sua inscrição para falar quando solicitados (linha 6). Agentes que desejam emitir argumentos são inseridos em WB, não podendo ocupar mais de uma posição nesta fila ao mesmo tempo. O mediador solicita os argumentos ao agente argumentativo ocupante da primeira posição em WB (linha 8). Após enviar seus argumentos, sua base S é esvaziada, visto que estes argumentos já foram enviados para discussão. Para cada argumento enviado, este argumento é validado (linha 10) para garantir que ainda não tenha sido discutido pelo grupo (evitando repetições), ele é informado ao grupo (agentes argumentativos armazenam-no em suas bases A) (linha 11),

são questionados por contra-argumentos (agentes argumentativos armazenam seus ataques em suas bases S) (linha 12) e os agentes que possuem argumentos para serem emitidos ao grupo são inscritos novamente em WB (linha 13). O processo de diálogo é encerrado quando não existem mais assuntos para serem discutidos, não existam mais agentes inscritos em WB e não existem mais argumentos para discussão na $AGENDA$.

3. Implementação do diálogo argumentativo

Para implementar o diálogo argumentativo, utilizou-se frameworks já existentes capazes de representar agentes inteligentes e argumentos dedutivos. Observando-se as características reativas dos agentes argumentativos, decidiu-se optar pelo framework JADE[1] (JAVA Agent DEvelopment Framework).

Os agentes argumentativos e o agente mediador são apresentados na Figura 1. A comunicação com o ambiente (e, consequentemente, com os demais agentes) é realizado por meio da troca de mensagens. Em JADE, o envio de mensagens ocorre por meio da classe ACLMessage. De acordo com a performativa estabelecida na mensagem, o seu conteúdo indica uma determinada ação que o agente deve executar

Com relação ao agente argumentativo, ao receber uma mensagem com a performativa CFP, o agente deve enviar todos os seus argumentos presentes na base S e então esvaziá-la, sendo que esta resposta utiliza a performativa PROPOSE. Ao receber mensagem com performativa REQUEST_WHENEVER, o agente argumentativo deve buscar por ataques ao argumento de sua base A e armazená-los em S. Ao receber uma mensagem com performativa INFORM_REF, o agente deverá atualizar o argumento presente na base A. Para mensagens com performativa REQUEST_WHEN, caso a base $S \neq \emptyset$, o agente responde o valor "true" utilizando a performativa SUBSCRIBE, caso contrário, responde com o valor "false". Cada tipo de mensagem recebida é implementada em um comportamento CyclicBehaviour do agente.

Para representar argumentos dedutivos e construção de argumentos, utilizou-se o framework Tweety Project[2] por meio da biblioteca Deductive Argumentation. Como os argumentos criados são do tipo DeductiveArgument e estes objetos não podem ser transmitidos de um agente ao outro utilizando ACLMessage, os argumentos são convertidos para String ao serem enviados. No agente receptor, esta String é convertida novamente para objetos DeductiveArgument para serem manipulados como argumentos.

O agente mediador não possui uma característica reativa, visto que sua atividade consiste em controlar um processo de comunicação que segue uma ordem de execução. Neste sentido, utilizou-se um agente disponível no framework JADE chamado GatewayAgent, implementado pela classe JadeGateway. O uso de GatewayAgent se torna necessário visto que apenas agentes podem enviar e receber mensagens no framework. Com isso, o agente mediador pode se comunicar (por meio do seu GatewayAgent) com os demais agentes argumentativos do sistema e receber respostas quando solicitado.

As ações de informar argumento (performativa INFORM_REF) e assunto (performativa REQUEST_WHENEVER) atual, solicitar inscrições (performativa REQUEST_WHEN), solicitar argumentos (performativa CFP) e solicitar ataques ao

[1] JADE disponível em http://jade.tilab.com
[2] Tweety Project disponível em http://tweetyproject.org

argumento atual (performativa REQUEST_WHENEVER) são implementados utilizando o comportamento OneShotBehaviour. As ações para receber inscrições (performativa SUBSCRIBE) e receber os argumentos de um determinado agente (performativa PROPOSE) são implementados utilizando o comportamento SimpleBehaviour.

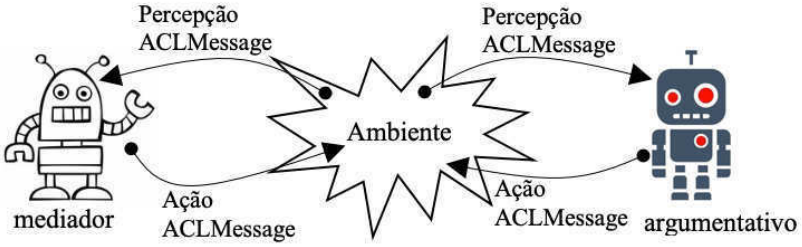


Figura 1. Agente argumentativo e mediador: mensagens recebidas (percepção) e mensagens transmitidas (ação)

4. Resultados Obtidos

A implementação dos diálogos argumentativos usando JADE e TweetyProject pode ser encontrada em https://github.com/apossebom/MASDialogue. O arquivo masdialogue/ Principal.java define as características iniciais para a execução, tais como definir todos os agentes argumentativos e suas respectivas bases de conhecimento e a lista de assuntos a serem discutidos. Em seguida, os agentes argumentativos são criados, seguido pela criação do agente mediador. Dois agentes argumentativos foram fornecidos: Maria e José. A base de conhecimentos do agente Maria é formada por $K_{\{Maria\}} = \{a, a \rightarrow b, \neg b, \neg a, c, c \rightarrow a\}$ e a base de conhecimentos do agente José é formada por $K_{\{José\}} = \{\neg a, b \rightarrow a, \neg c, \neg c \rightarrow d, e\}$. Considere os átomos das fórmulas sendo representados por: a prova fácil; b ir bem na prova; c estudar para a prova; d fazer recuperação; e pedir ajuda ao professor.

O agente med informa o assunto a para discussão (diálogo para saber se uma prova estava fácil) e solicita por ataques. Os agentes José e Maria possuem argumentos para $\neg a$ e se inscrevem para falar. Após José emitir seu argumento $arg_0 = \langle\{\neg a\}, \neg a\rangle$, este argumento é validado e inserido em DT, informado ao grupo e novos ataques são encontrados. O próximo agente a falar, Maria, envia seus argumentos (ataques ao assunto a ou a outros argumentos emitidos no diálogo, no caso, ataques a arg_0). Ao enviar seus argumentos para med, estes argumentos, um a um, serão validados, inseridos em DT, informados ao grupo e novos ataques ao argumento atual sendo discutido são solicitados. O diálogo se encerra quando não existem mais agentes inscritos para falar (a base $S = \emptyset$ para todos os agentes e $WB = \emptyset$) e todos os argumentos já tiverem sido discutidos pelo grupo ($AGENDA = \emptyset$). A execução do diálogo sobre a gerou como resultado a tabela de diálogo apresentada na Tabela 1. Convertendo o diálogo para grafo de argumentos [Dung 1995] e, por meio de alguma semântica de argumentação, podemos observar que uma extensão pode sugerir uma conclusão para o diálogo. Por exemplo, utilizando semântica *Preferred* no conjunto com maior número de argumentos preferidos (ex: $\{arg_0, arg_3, arg_4, arg_5, arg_6, arg_{14}, arg_{15}, arg_{16}, arg_{17}\}$), tem-se que as fórmulas destes argumentos sugerem $\neg a, \neg b$ e $\neg c$, indicando que a prova não estava fácil, não foram bem na prova e não estudaram o suficiente).

Tabela 1. Tabela de diálogo para o assunto a.

Sequência	Emissor	Argumento
0	José	<{ !a },!a>
1	Maria	<{ a },a>
2	Maria	<{ c, !clla },a>
3	Maria	<{ !allb, !b },!a>
4	José	<{ !c },!c>
5	Maria	<{ !a, !clla },!c>
6	Maria	<{ !allb, !b, !clla },!c>
7	Maria	<{ !allb, a },b>
8	Maria	<{ c },c>

Sequência	Emissor	Argumento
9	Maria	<{ !allb, c, !clla },b>
10	Maria	<{ !b, a },!(!allb)>
11	Maria	<{ !a, c },!(all!c)>
12	Maria	<{ !b, c, !clla },!(!allb)>
13	Maria	<{ !allb, !b, c },!(all!c)>
14	Maria	<{ !allb },!allb>
15	Maria	<{ !b },!b>
16	Maria	<{ !clla },all!c>
17	José	<{ !blla, !a },!b>

4. Considerações Finais

Este trabalho apresentou uma proposta de implementação de diálogos argumentativos para ser utilizado em sistemas multiagentes, onde os argumentos emitidos/recebidos pelos agentes possuem uma estrutura definida (premissa e conclusão) e utilizam uma linguagem lógica (lógica proposicional) para representar o conhecimento dos agentes.

Dois tipos de agentes foram implementados: argumentativo e mediador. O agente argumentativo é o responsável por construir argumentos e argumentar. O agente mediador é o responsável por conduzir o processo de diálogo, sincronizando a troca de mensagens e garantindo que os argumentos discutidos sejam válidos. A partir desta implementação, diferentes abordagens podem ser estendidas, tais como calcular o consenso do grupo sobre determinado argumento ou sobre alternativas de decisão, aplicação de semânticas de argumentação ou ponderação dos argumentos.

Referências

Amgoud, L. and Maudet, N. and Parsons, S. (2002).An argumentation-based semantics for agent communication languages. In: Proceedings of the 15th ECAI, p. 38–42.

Amgoud, L. and Prade, H. (2009) Using arguments for making and explaining decisions. Artificial Intelligence, Elsevier B.V., v. 173, n. 3-4, p. 413–436.

Besnard, P. and Hunter, A. (2009). Argumentation based on classical logic. In Argumentation in Artificial Intelligence, pages 133–152. Springer.

Caminada, M. (2008). A gentle introduction to argumentation semantics. Disponível em: <https://users.cs.cf.ac.uk/CaminadaM/publications/Semantics_Introduction.pdf>. Acessado em: 24/03/2019.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial intelligence, 77(2):321–357.

Greco, S. (2015). Argumentative Dialogue. Key Concepts in Intercultural Dialogue, No. 73. Disponível em: < https://centerforinterculturaldialogue.files.wordpress.com/2015/10/kc73-argumentative-dialogue.pdf>. Acessado em: 24/03/2019.

Parsons, S. and McBurney, P. (2003). Argumentation-based dialogues for agent coordination. Group Decision and Negotiation, 12(5):415–439.

Possebom, A. T. and Morveli, M. and Tacla, C. A. (2016) Protocolo para diálogos argumentativos no auxílio da decisão consensual em sistemas multiagentes. WESAAC 2016, pp.169-174.

Organização de lascas de madeira: uma análise utilizando simulação baseada em agentes

Carlos Eduardo Pereira de Quadros, Vágner de Oliveira Gabriel, Alessandro de Lima Bicho, Diana Francisca Adamatti

C3 - Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Av. Itália km 8 S/N - 96203-900 - Campus Carreiros – Rio Grande – RS / Brazil

{carlos.quadros, dmtbicho, dianaadamatti}@furg.br, vagnergabriel@terra.com.br

Abstract. *The Termites model available in the NetLogo tool serves as the basis for this research. This work presents results of the analysis of the organization of wood chips using agent-based simulation, as well as the variation of the number of termites in the environment in relation to the work of moving the splinters arranged in the simulation environment. The proposal of the study shows results not investigated in the original model, through two hypotheses: to investigate if the average of the wood chips formed at the end of the simulation decrease or not with the increase of the number of termites in the environment; analyze if there is a saturation point in the number of termites used in the simulation to complete the task.*

Resumo. *O modelo Termites disponibilizado na ferramenta NetLogo serve como base para essa investigação. Este trabalho apresenta resultados da análise da organização de lascas de madeira utilizando simulação baseada em agentes, assim como a variação do número de cupins no ambiente em relação ao trabalho de movimentação das lascas dispostas no ambiente de simulação. A proposta do trabalho aponta resultados não investigados no modelo original, através de duas hipóteses: investigar se a média das lascas de madeira formadas ao final da simulação diminuem ou não com o aumento do número de cupins no ambiente; analisar se existe um ponto de saturação no número de cupins empregados na simulação para a execução da tarefa por completo.*

1. Introdução

O desenvolvimento de modelos de sistemas, técnicas e resoluções de problemas através da inspiração obtida pela natureza é conhecido como biomimético ou bioinspirado. Ambos termos tem como premissa simular comportamentos existentes na fauna desde microorganismos até coletivos de animais, insetos e também a flora [BJÖRN, 2003]. Através da simulação desses comportamentos, sejam eles individuais ou coletivos, podemos entender de qual maneira funciona o sistema como um todo e qual seria a reação a partir de intervenções hipotéticas impostas ao mesmo. A simulação feita através de sistemas multiagente possibilita reinterpretar o ambiente real no virtual. No caso da simulação especificamente bioinspirada, podemos extrair da natureza as respostas para problemas cotidianos e investigar determinadas situações através de emergências no sistema, por exemplo.

O modelo Termites disponibilizado na ferramenta NetLogo [WILENSKY, 1997] simula a organização de lascas de madeiras feitas por cupins que, dependendo do tempo de simulação, movimentam as mesmas formando apenas uma pilha de lascas de madeira. Os *patches* são pequenos quadrados programáveis que juntos formam o espaço de simulação, ou seja, servem para discretizar o ambiente e representam o lugar por onde os agentes percorrem. As lascas de madeira são representadas pelos *patches* do NetLogo que possuem a cor amarela e os que não possuem lascas de madeira são representados pela cor preta (espaços vazios). Os cupins seguem um conjunto de regras simples. Após andar no ambiente de simulação aleatoriamente, quando um cupim encontra uma lasca de madeira ele pega essa lasca, segue andando de forma aleatória até encontrar outra lasca e, ao encontrar outra lasca de madeira, ele procura um local vazio próximo para largar a lasca que havia pegado anteriormente [RESNICK, 1997]. No momento que o cupim está procurando lascas ele é representado pela cor branca e quando ele está com alguma lasca ele é representado pela cor laranja.

Este trabalho tem como objetivo principal investigar questões não detalhadas no modelo original, através da implementação de funções que lidam com a parte de mensuração de resultados apresentados ao longo da simulação. A primeira hipótese considerada para ser analisada é se a média de pilha de lascas de madeira formadas ao final das simulações diminuem com o aumento de força de trabalho (número de cupins). A segunda hipótese a ser testada, caso a primeira se concretize em relação à diminuição da média de pilhas ao final da simulação, é descobrir qual o ponto de saturação. Ou seja, descobrir qual o número de cupins necessários para deixar ao final da simulação apenas uma pilha de lascas de madeira.

O artigo está organizado da seguinte maneira: na seção 2 apresentamos a ferramenta utilizada e a relação da bioinspiração com esta pesquisa, na seção 3 aprofundamos a explicação sobre o modelo desenvolvido. Na seção 4 mostramos todos os experimentos realizados na pesquisa e na seção 5 sintetizamos o trabalho de maneira a explicar a proposta, os resultados obtidos, e por fim, é explanada a possibilidade de outros trabalhos em relação à metodologia apresentada no atual trabalho.

2. Referencial teórico

No contexto dos sistemas multiagentes, um agente é uma unidade capaz de cumprir determinadas tarefas interagindo tanto com o meio quanto com as demais unidades inseridas no ambiente, sempre respeitando as regras propostas no cenário.

O comportamento emergente do modelo é a coleção de aparas de madeira em uma única pilha. O comportamento não é expressamente concebido em cada programação do cupim, mas sim o resultado de todo sistema [BJÖRN, 2003]. Exemplos de padrão biológico incluem um cardume de peixes, uma coluna de invasão de formigas, a intermitência síncrona de vaga-lumes, e a arquitetura complexa de um cupinzeiro [CAMAZINE *et al.*, 2001]. Formigas, abelhas, cupins - todos os insetos sociais mostram capacidades de resolução de problemas coletivos impressionantes. Propriedades associadas com o seu comportamento em grupo como organização, robustez e flexibilidade são vistos como características que, sistemas artificiais para otimização, controle ou execução de tarefa devem exibir [BONABEAU; DORIGO; THERAULAZ, 1999; BONABEAU, 2001]. Segundo Detanico, Teixeira e Silva (2010), o conceito de Biônica ou Biomimética consiste em analisar sistemas naturais e reproduzir seus princípios de solução. As soluções da natureza podem contribuir para o processo criativo de projeto, tanto na forma de analogia como através de seus padrões geométricos/matemáticos.

A partir do momento em que buscamos soluções de problemas na natureza, percebemos que inúmeras invenções ditas feitas pelo homem já existem. De acordo com Benyus (2007), nossas vigas e escoras já estão nas folhas do nenúfar e nas hastes do bambu, nossos sistemas de aquecimento central e ar- condicionado são superados pelos 30º centígrados do cupinzeiro, nosso radar mais sofisticado é surdo se comparado ao sistema de captação de frequências do morcego.

3. Materiais e métodos

O NetLogo [WILENSKY, 1999] é um ambiente de modelagem programável para simular fenômenos naturais e sociais. É adequado para modelar sistemas complexos o observar o seu desenvolvimento ao longo do tempo.

O modelo Termites apresentado na ferramenta de simulação NetLogo é simples. Este modelo apresenta apenas o controle sobre o número de agentes que irá efetuar a simulação no ambiente e a densidade de lascas de madeira que será distribuída de forma regular e randômica no espaço. Dentro da biblioteca do NetLogo, a simulação de cupins não mensura nada em relação ao comportamento dos agentes ou investiga respostas sobre o que resultou da movimentação das lascas.

Partindo desse princípio, existem algumas questões que não foram completamente esclarecidas no modelo principal. Utilizando o modelo original disposto na ferramenta e aplicando algumas implementações, foi possível responder alguns questionamentos que não haviam sido respondidos, como exemplo:

01 - Como fica a distribuição, ou como são aglomeradas as pilhas de lascas de madeira com apenas um cupim no ambiente? E com muitos cupins?

02 - Quando há duas pilhas iguais de lascas de madeira em lados opostos, como é formada a movimentação após um certo tempo de simulação? E quando há uma pilha maior que a outra?

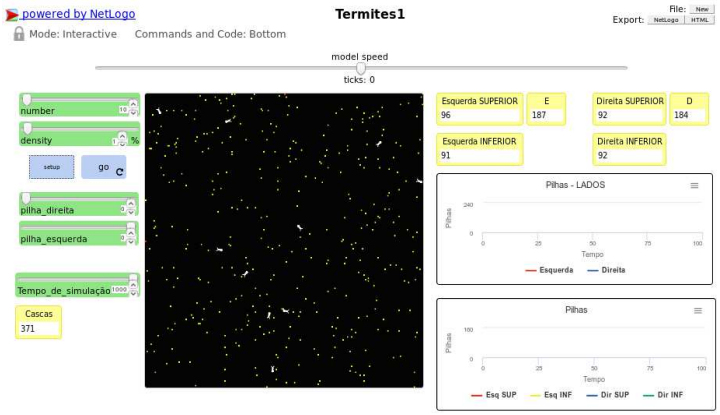


Figura 1. Modelo *Termites* implementado

Nesta pesquisa em específico será abordada a análise da variação do número de cupins no ambiente de simulação em relação à organização das lascas de madeira, questão 01 supracitada.

A Figura 1 apresenta o modelo estendido implementado, além dos controles básicos de densidade de lascas e número da cupins, foram criadas a implementações do tempo de simulação e a contagem do número de pilhas que eram formadas no final da simulação. O comportamento emergente da simulação de movimento de lascas de madeira, sem tempo definido, é a formação de apenas uma pilha. Para isso foi colocado

um tempo limite de cem mil *ticks* (unidade de medida do NetLogo) e avaliado quantas pilhas eram formadas ao final da simulação. Também foram investigadas, através da divisão do ambiente em quatro espaços de igual proporção, o número de lascas que permaneciam nos mesmos.

Através dessa divisão do ambiente foi possível identificar quantos e quais espaços ficavam sem lascas no final da simulação. Para identificar estes espaços, imaginemos um plano cartesiano em que a divisão é apresentada em quatro quadrantes. No caso da simulação, estes foram nomeados da seguinte forma: Canto Superior Esquerdo, Canto Inferior Esquerdo, Canto Superior Direito e Canto Inferior Direito. Ainda neste ponto, foram analisados em que momento um determinado "Canto" do espaço era zerado, ou seja, não possuía mais nenhuma lasca de madeira. Esta medida é anotada na unidade de *tick em* que o cupim carrega a última lasca existente no espaço.

A primeira simulação contou apenas com 1 cupim, e para cada simulação foram executadas 5 rodadas para coletar seus resultados. Posteriormente foi acrescentado mais 1 cupim no cenário de simulação até um total de 10 cupins existentes na simulação (todas simulações contaram com o mesmo número de rodadas). Após serem efetuadas todas as simulações foi realizada a coleta dos seus resultados para a análise da organização das lascas de madeira no ambiente de simulação. O número de lascas de madeira existente no ambiente era de 4 mil lascas aproximadamente, pois varia por conta da densidade escolhida para as simulações que era de 10%. A densidade de 10% permanece para todas as simulações propostas neste trabalho.

No primeiro experimento, conjunto de 10 simulações com tempo limitado, a medida em que aumentamos o número de cupins no ambiente a média do número de pilhas de lascas de madeira ao final das simulações diminui. Conforme o Figura 2, o conjunto de 5 simulações com apenas 1 cupim teve uma média final de número de pilhas de 6,4 e o conjunto de 5 simulações com 10 cupins teve uma média de 1,4 pilhas ao final. De acordo com o gráfico, podemos notar que o número de pilhas diminui efetivamente à medida que aumentamos a força de trabalho, ou seja, número de cupins no ambiente.

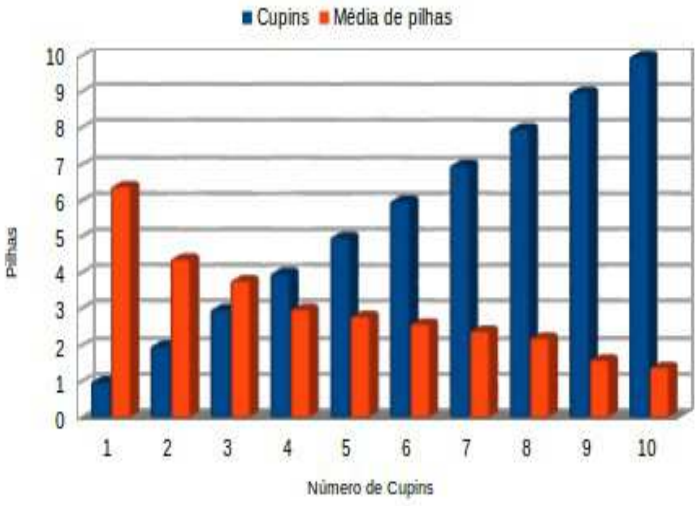


Figura 2. Simulações de 1 até 10 cupins

Considerando a primeira hipótese, o número médio de pilhas diminui à medida que aumentamos o número de cupins no ambiente de simulação. Feita a simulação e análise da primeira hipótese, passamos para a identificação do ponto de saturação, ou seja, segunda hipótese.

Como a primeira fase de simulações consistiu em simulações de 1 até 10 cupins e este número quase alcançou o número desejado de apenas uma pilha no final das simulações. Então, optou-se por seguir pelo incremento de 1 cupim em cada simulação subsequente.

Conforme Figura 3, logo após a simulação contendo 10 cupins, podemos perceber que o ponto de saturação foi encontrado na simulação com 11 cupins, pois tanto esta quanto a simulação com 12 cupins alcançaram a média de uma pilha ao final da simulação. Portanto, para que tenhamos o trabalho feito por completo, ou seja, para que tenhamos apenas uma pilha de lascas de madeira ao final de uma simulação com o tempo de cem mil *ticks* previamente configurado, precisamos de 11 cupins no ambiente.

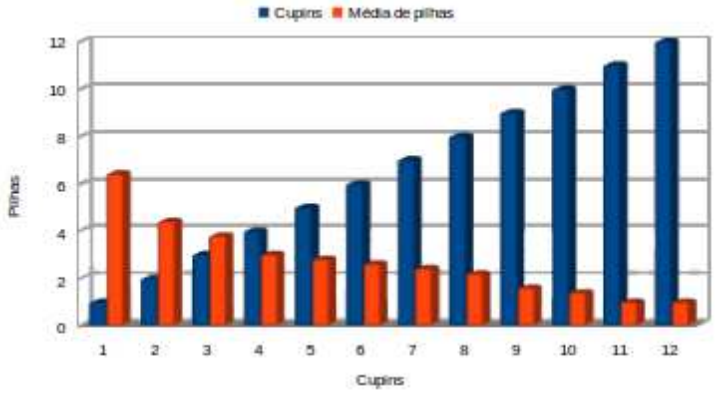


Figura 3. Simulações para comprovar saturação

4. Considerações Finais

Este trabalho apresentou a análise da organização de lascas de madeira no modelo de simulação NetLogo. Neste foram propostas duas hipóteses: 1) A primeira hipótese consistia em investigar se a média das lascas de madeira formadas ao final da simulação diminuíam ou não com o aumento do número de cupins no ambiente e esta se confirmou, pois houve a diminuição da média ao passo que era aumentado o número de cupins no ambiente; 2) A segunda hipótese a ser provada dependia diretamente da primeira e também foi comprovada, pois há um ponto de saturação no número de cupins empregados na simulação para a execução da tarefa por completo, ou seja, formar apenas uma pilha ao final da simulação.

A forma que os cupins trabalham é descentralizada, ou seja, não possuem uma ordenação direta ou regra para carregar as lascas de madeira. Portanto, comparar o trabalho apresentado com outro centralizado, seria uma possibilidade para trabalhos futuros. Uma hipótese interessante, tratando-se de regramento no ambiente, seria categorizar determinados tipos de cupins e dividi-los em grupos com funções diferentes, formando um sistema hierárquico, o que é comumente visto na natureza.

Referências

BENYUS, J. M. (2007) Biomimética: inovação inspirada pela natureza. São Paulo: Cultrix.

BJÖRN, A. (2003) Design of simulated human behaviour. Umeå: Umeå University, 2003.

BONABEAU, E.; DORIGO, M.; THERAULAZ, G. (1999) Swarm intelligence: from natural to artificial systems. Oxford: Oxford University Press.

BONABEAU, E. (2001) Self-organization in biological systems. Princeton University Press.

CAMAZINE, S. et al. (2003) Self-organization in biological systems. Princeton University Press.

DETANICO, F. B.; TEIXEIRA, F. G.; SILVA, T. K. (2010) A Biomimética como método criativo para o projeto de produto. Design & Tecnologia, n. 2. Disponível em: http://migre.me/uCYTH. Acesso em: 01 ago. 2016.

RESNICK, M. (1997) Turtles, termites and traffic jams: explorations in massively parallel microworlds. Cambridge: MIT Press.

WILENSKY, U. (1997) NetLogo termites model. Evanston: Northwestern University.

WILENSKY, U. (1999) NetLogo. Evanston: Northwestern University.

Um estudo voltado a Modelos Ambientais envolvendo Sistemas Multiagentes e/ou Jogos de Papéis

Bruna S. Leitzke[1], **Diana F. Adamatti**[1]

[1]Programa de Pós Graduação em Modelagem Computacional (PPGMC)
Universidade Federal do Rio Grande (FURG)
Rio Grande – RS – Brazil

brunaleitzke@hotmail.com, dianaada@gmail.com

Abstract. *For the management of natural resources to be carried out in a responsible way, one must think of tools that help in decision making and analyze the interations among the agents involved in the problem. Based on Artificial Intelligence, as a way to simulate future situations related to the management of natural resources, the Multiagent Systems and the Role-Playing Games, are techniques that can be used. In this paper, some recent papers, which address these techniques in the context of Ambiental Models, are explored in order to better understand their use together for future applications in the area.*

Resumo. *Para que a gestão dos recursos naturais seja feita de maneira responsável, deve-se pensar em ferramentas que auxiliem na tomada de decisão e analisem as interações entre os agentes envolvidos no problema. Baseados na Inteligencia Artificial, como uma forma de simular situações futuras relacionadas à gestão de recursos naturais, os Sistemas Multiagentes e os Jogos de Papéis são técnicas que podem ser utilizadas. Neste trabalho, alguns artigos recentes, que abordam essas técnicas no contexto de Modelos Ambientais, são explorados, com o intuito de um melhor entendimento sobre seu uso em conjunto, visando futuras aplicações na área.*

1. Introdução

Uma das preocupações em pesquisas relacionadas à gestão dos ecossistemas é a forma de promover o crescimento social e econômico sem afetar o equilíbrio ambiental. Alternativas para resolver esse problema são dadas por métodos relacionados a Inteligência Artificial (IA), que pode produz resultados que respondem as questões percebidas com base no conhecimento armazenado [Ponte et al. 2016].

Algumas técnicas vem sendo estudadas para resolver os problemas que envolvem Modelos Ambientais (MA), como os Sistemas Multiagente (SMA) e os Jogos de Papéis (*Role-Playing Games* - RPGs). De forma geral, essas ferramentas podem ser aplicadas em situações com vários agentes envolvidos que participam de um espaço de gerenciamento de negociação – que é um espaço onde grupos discutem formas de organizar e planejar ações, levando em conta a tomada de decisão conjunta.

Um dos objetivos de estudar as técnicas de SMA, ou ainda, a Simulação Baseada em Multiagente (*Multi-Agent-Based Simulation* (MABS)), é dado pela possibilidade de criar novas ferramentas, como RPGs. Com isso, se torna viável construir sociedades virtuais sem trazer consequências efetivas para a vida real [Bousquet et al. 2002]. Nesse

sentido, este artigo visa apresentar trabalhos relacionados ao tema, de forma a explorar pesquisas atuais que envolvem Modelos Ambientais e as técnicas de SMA e/ou RPGs. Particularmente, alguns artigos voltados para a gestão de recursos naturais serão mencionados, para um melhor entendimento sobre o assunto, dando suporte para estudos futuros.

2. Trabalhos Relacionados

A gestão dos ecossistemas visa o cumprimento de funções sociais e ambientais, necessitando assim da participação cidadã no processo de planejamento e transformação. Para isso, deve-se considerar a colaboração de diferentes organizações envolvidas, formando uma rede estruturada e apoiada por mecanismos e ferramentas que possibilitam o desenvolvimento de ações mais justas, incluindo questões ambientais produtivas e econômicas [Dos Santos et al. 2016]. Além disso, a preservação dos ecossistemas também deve ser estudada com o intuito de analisar possíveis cenários, determinando a melhor forma de gerir esses espaços para gerações futuras [Dupont et al. 2016].

Algumas pesquisas recentes utilizam técnicas como forma de auxiliar em MA. Na Tabela 1 estão alguns desses artigos e as técnicas utilizadas.

Tabela 1. Alguns artigos que relacionam MA com técnicas de SMA e/ou RPG.

Identificação	Referência	Método(s) utilizado(s)
1	[Ghazi et al. 2018]	SMA
2	[Dos Santos et al. 2016]	SMA
3	[Dupont et al. 2016]	SMA
4	[Li et al. 2017]	SMA
5	[Ponte et al. 2016]	SMA
6	[Yang et al. 2018]	SMA
7	[Han et al. 2018]	SMA
8	[Shelton et al. 2018]	SMA e RPG
9	[Perrotton et al. 2017]	SMA e RPG
10	[Page et al. 2016]	SMA e RPG

As ideias principais dos trabalhos que abordam apenas as técnicas de SMA serão resumidas a seguir. SMA servem como uma ferramenta importante para simular e analisar sistemas complexos, onde o principal objetivo é dividir o sistema em agentes, simulando a tomada de decisão no nível micro para chegar em soluções comuns no nível macro [Li et al. 2017]. E eles podem ser classificados de acordo com [Ghazi et al. 2018]:

- O mecanismo de tomada de decisão;
- O uso ou não de dados reais;
- O objetivo da simulação e;
- A representação do espaço e do tempo.

Utilizando um modelo de dispersão e um modelo de previsão, [Ghazi et al. 2018] criaram uma ferramenta de MABS que modelava a população de controladores de fonte de emissão como uma rede de agentes, a partir de um jogo. E, como estudo de caso, escolheram a região de Annaba (nordeste da Argélia), por possuir siderúrgicas. Assim, chegaram em uma ferramenta de tomada de decisão que pode auxiliar as agências de controle de poluição do ar.

O estudo de [Dos Santos et al. 2016] apresentou uma análise sobre a realidade atual do experimento SJVG (*San Jerónimo Vegetable Garden*), que é um exemplo de um ecossistema urbano localizado em Sevilha, Espanha. Para isso, o SMA foi concebido como um sistema de agente multi-dimensional tipo BDI, analisando crenças, desejos e intenções dos agentes do problema. Para isso, foi utilizada a estrutura JaCaMo (Jason, CArtAgo e MOISE +), que cobre alguns dos níveis de abstrações que são necessários para o desenvolvimento de SMA sofisticados.

No trabalho de [Dupont et al. 2016], foi utilizada uma modelagem baseada em SMA para resolver o problema da gestão de conservação de habitats e espécies do projeto Natura 2000[1], onde o local escolhido foi na península de Crozon, no oeste de Britany (França). Para a modelagem foi necessário realizar três etapas principais. A primeira foi a modelagem conceitual, utilizando a bordagem ComMod (*The Companion Modelling approach*), que visa compreender os sistemas complexos através de SMA. Depois foi desenvolvido e validado um protótipo do *software*, implementado na plataforma Cormas. E por último, o modelo foi validado a partir de simulações.

[Li et al. 2017] analisaram os sistemas de alocação e fluxo do nexo[2] de água, energia e alimento (*Water-Energy-Food* - WEF). Os autores utilizaram três agentes para modelar esse problema: agente doméstico, agente da empresa e agente governamental, e foram estabelecidas as regras e padrões de consumo para esses agentes. Para simular o Sistema Multiagente, foi utilizado o modelo do consumo humano do NetLogo, e então foi explorada a relação entre o comportamento de agentes no nível micro e o modelo macro derivado da interação entre centenas de indivíduos independentes, que consequentemente gerou um padrão de consumo entrelaçado.

A água é um elemento indispensável para a população em geral. Mas, infelizmente, o gerenciamento desse recurso ainda é precário em muitos lugares. Para gerir adequadamente a distribuição da água, diferentes organizações devem se reunir e pensar em planos de gestão que satisfaçam as necessidades de todos os envolvidos. Quando esse gerenciamento não ocorre de forma justa, ou simplesmente não existe, podem ocorrer situações graves, como desperdício em excesso ou escassez de água.

Com o uso de um Sistema de Apoio à Decisão Inteligente, a pesquisa de [Ponte et al. 2016] foi realizada para auxiliar um problema de Gerenciamento de Demanda de Água. Um sistema foi projetado a partir da estrutura da rede de abastecimento de água de Gijón, Espanha, e de um conjunto de seis diferentes agentes. A simulação foi realizada a partir de dados da cidade de Gijón, um modelo de distribuição de demanda de água, e parâmetros aleatórios para introduzir diferentes fontes de incerteza. Com isso, os autores observaram a redução de custos, relacionado ao armazenamento e bombeamento de água de emergência, a partir do sistema implementado.

O gerenciamento de recursos hídricos pode ser voltado aos efeitos causados por inundações. No trabalho de [Yang et al. 2018] foi possível observar o comportamento de indivíduos em diferentes cenários através de um SMA. As famílias individuais foram

[1] http://ec.europa.eu/environment/nature/natura2000/index_en.htm

[2] A palavra nexo foi traduzida da palavra *nexus* em inglês, a qual é apresentada no texto original *Water-energy-food nexus in urban sustainable development an agent-based model*. Ela pode ser entendida como a união de dois ou mais elementos.

consideradas como os agentes do problema, e um estudo de caso foi realizado na bacia hidrográfica urbanizada no rio Ng Tung, no norte de Hong Kong. Um modelo de escoamento superficial foi utilizado, simulando as inundações, e foi reamostrado e importado para a plataforma de modelagem NetLogo. Assim, com as respostas geradas, os autores constataram que a falta de recurso econômico, o tempo de espera, e as informações de aviso mostraram ser fatores importantes no risco de danos causados por enchentes.

Ainda neste contexto, para propor um esquema ideal de alocação de recursos hídricos, [Han et al. 2018] desenvolveram um modelo de otimização em dois níveis. E a teoria dos jogos cooperativos pôde ser utilizada como uma alternativa para auxiliar na organização e classificação das tomadas de decisão dos atores dentro do sistema. Uma forma de validar o modelo foi utilizar, como um estudo de caso, a bacia do rio Hanjiang, localizado na China, onde os resultados mostraram melhorias no sistema.

O RPG é uma ferramenta que permite que jogadores e observadores aprendam e reflitam sobre respostas de um sistema [Page et al. 2016]. Dessa forma, os participantes são submetidos a problemas semelhantes aos da vida real, onde devem chegar em soluções por meio da tomada de decisão [Adamatti et al. 2009]. Essa técnica pode ser utilizada para auxiliar na modelagem de sistemas socioambientais, assim eles são construídos a partir das respostas dos atores a situações apresentadas no jogo. O intuito é que ao longo de várias sessões, as partes interessadas analisem os problemas apresentados e tomem decisões de forma cooperativa, o que consequentemente gera respostas mais coerentes a todos os envolvidos [Étienne 2010].

[Shelton et al. 2018] relataram os resultados de uma experiência do uso de um RPG como ferramenta de validação de um SMA, na cidade do México. O projeto de modelagem chamado A Dinâmica da Adaptação Multi-Escalar em Megacidades foi desenvolvido para facilitar a modelagem participativa em problemas de vulnerabilidade crônica à escassez de água e inundações. Os agentes considerados foram as autoridades de abastecimento de água da cidade e bairros vulneráveis. Os autores criaram um modelo de decisão inicial para os residentes, em forma de jogo de tabuleiro. E criaram um modelo usando o *software* de análise de decisão multicritério *Super Decisions* v. 1.6.0[3]. Então, o jogo foi apresentado aos residentes, gerando discussões que induziram ele a ser uma ferramenta autônoma para aprendizagem e pesquisa.

Já no trabalho de [Perrotton et al. 2017] foi relatada a experiência inicial de um projeto que visava criar uma arena, onde comunidades locais e gerentes de áreas protegidas conseguissem se reunir, discutir, negociar e produzir planos de manejo eficazes em aldeias no oeste do Zimbábue, África. Essas aldeias são vizinhas ao Parque Nacional de Hwange e a Floresta de Sikumi, onde ambas são áreas protegidas. Para o estudo, foi adotada a abordagem ComMod. Assim, os autores conseguiram projetar uma versão protótipo em forma de um jogo de RPG. Esse modelo simulava interações entre atividades agrícolas, pastoreio de animais e vida selvagem em uma paisagem virtual. Com isso, os atores puderam validar a ferramenta, propondo ideias e discutindo sobre melhorias, chegando a uma versão do jogo mais realista, e que foi chamada de Kulayinjana.

O problema de gerenciamento de ecossistemas também pôde ser abordado no trabalho de [Page et al. 2016], com o intuito de desenvolver um RPG, chamado ReHab, e

[3]https://www.superdecisions.com/

que foi implementado na plataforma Cormas e no NetLogo. Utilizando a abordagem ComMod, o jogo tratava da harmonização entre regeneração de biomassa, e o habitat de reprodução de aves migratórias protegidas. Assim, foram organizadas sessões de RPG para explorar a eficácia e os desafios da comunicação na melhoria da gestão. E foi constatado que o cenário de não comunicação entre os agentes permitiu a identificação das estratégias individuais. Enquanto que, com a comunicação entre eles, houve a possibilidade do uso de estratégias coletivas ou o surgimento de conflitos, onde os acordos foram discutidos e propostos pelos jogadores.

3. Considerações finais

Os artigos abordaram técnicas utilizadas para resolver problemas envolvendo o gerenciamento de recursos naturais, apresentando a metodologia e resultados. E ainda, alguns relataram as experiências obtidas com a aplicação dessas ferramentas.

Para resolver os MA os autores utilizaram SMA, empregando os devidos agentes dos problemas e simulando os sistemas. Para validar os métodos, na maioria dos trabalhos, houve o estudo de caso, gerando resultados mais próximos da realidade. Além disso, em muitos casos, cenários futuros foram projetados e analisados a partir das simulações. O que mostra que com o uso de SMA é viável construir um planejamento da gestão de recursos de maneira mais organizada, levando em conta os problemas e a tomada de decisão participativa.

Nos trabalhos que abordaram o desenvolvimento e aplicação de jogos como forma de auxiliar na construção e modelagem dos problemas, os resultados obtiveram um acréscimo significativo na análise dos modelos. As ferramentas apresentaram uma representação próxima da realidade, que permitia aos participantes reproduzir as práticas reais e melhorar a compreensão dos sistemas.

As pesquisas apresentadas neste artigo demonstraram que o gerenciamento de recursos deve ser organizado e planejado de forma participativa, com o intuito de gerar simulações mais reais para os problemas. Com base nisso, pode-se concluir que as técnicas apresentadas de SMA e RPG são ferramentas que facilitam a tomada de decisão entre os atores envolvidos no gerenciamento dessas áreas. Os MAs podem ser simulados com base na tomada de decisão conjunta, pensando em todas as partes que estão inseridas em um determinada situação. Assim, em muitos casos, pode-se projetar situações futuras, onde os agentes são induzidos a agir coletivamente para garantir boas soluções para os sistemas.

4. Agradecimentos

As autoras agradecem ao Programa de apoio ao Ensino e à Pesquisa Científica e Tecnológica em Regulação e Gestão de Recursos Hídricos – Pró-Recursos Hídricos Chamada N°16/2017, pelo auxílio financeiro no desenvolvimento desta pesquisa.

Referências

Adamatti, D. F., Sichman, J. S., and Coelho, H. (2009). An analysis of the insertion of virtual players in gmabs methodology using the vip-jogoman prototype. *Journal of Artificial Societies and Social Simulation*, 12(3):7.

Bousquet, F., Barreteau, O., d'Aquino, P., Etienne, M., Boissau, S., Aubert, S., Le Page, C., Babin, D., and Castella, J.-C. (2002). Multi-agent systems and role games: Collective learning processes for ecosystem management. *Complexity and Ecosystem Management: The Theory and Practice of Multi-agent Approaches*, pages 248–285.

Dos Santos, F. P., Adamatti, D., Rodrigues, H., Dimuro, G., Jerez, E. D. M., Dimuro, G., et al. (2016). A multiagent-based tool for the simulation of social production and management of urban ecosystems: a case study on san jerónimo vegetable garden-seville, spain. *Journal of Artificial Societies and Social Simulation*, 19(3):1–12.

Dupont, H., Gourmelon, F., Rouan, M., Le Viol, I., and Kerbiriou, C. (2016). The contribution of agent-based simulations to conservation management on a natura 2000 site. *Journal of environmental management*, 168:27–35.

Ghazi, S., Dugdale, J., and Khadir, T. (2018). A multi-agent based approach for simulating the impact of human behaviours on air pollution. *Informatica*, 42(2).

Han, Q., Tan, G., Fu, X., Mei, Y., and Yang, Z. (2018). Water resource optimal allocation based on multi-agent game theory of hanjiang river basin. *Water*, 10(9):1184.

Li, G., Wang, Y., Huang, D., and Yang, H. (2017). Water-energy-food nexus in urban sustainable development: an agent-based model. *International Journal of Crowd Science*, 1(2):121–132.

Page, C. L., Dray, A., Perez, P., and Garcia, C. (2016). Exploring how knowledge and communication influence natural resources management with rehab. *Simulation & Gaming*, 47(2):257–284.

Perrotton, A., de Garine-Wichatitsky, M., Valls-Fox, H., and Le Page, C. (2017). My cattle and your park: codesigning a role-playing game with rural communities to promote multistakeholder dialogue at the edge of protected areas. *Ecology and Society*, 22(1).

Ponte, B., De la Fuente, D., Parreño, J., and Pino, R. (2016). Intelligent decision support system for real-time water demand management. *International Journal of Computational Intelligence Systems*, 9(1):168–183.

Shelton, R. E., Baeza, A., Janssen, M. A., and Eakin, H. (2018). Managing household socio-hydrological risk in mexico city: A game to communicate and validate computational modeling with stakeholders. *Journal of environmental management*, 227:200–208.

Yang, L. E., Scheffran, J., Süsser, D., Dawson, R., and Chen, Y. D. (2018). Assessment of flood losses with household responses: Agent-based simulation in an urban catchment area. *Environmental Modeling & Assessment*, pages 1–20.

Étienne, M. (2010). *La modélisation d'accompagnement: une démarche participative en appui au développement durable*. Éditions QUAE.

Sistemas Multiagente e Jogos de Papéis para Gestão de Recursos Naturais

Bruna Leitzke[1], Giovani Farias[1], Marla Melo[1], Matheus Gonçalves[1], Míriam Born[2], Paulo Rodrigues[1], Vinícius Martins[1], Raquel Barbosa[3], Marilton Aguiar[2], Diana F. Adamatti[1]

[1]Centro de Ciências Computacionais (C3)
Universidade Federal do Rio Grande (FURG) – Rio Grande – RS – Brasil

[2]Programa de Pós-Graduação em Computação
Universidade Federal de Pelotas (UFPEL) – Pelotas – RS – Brasil

[3]Instituto Federal do Rio Grande do Sul (IFRS) – Campus Rio Grande
Rio Grande – RS – Brasil

{marilton,mbborn}@inf.ufpel.edu.br, brunaleitzke@hotmail.com,

{dianaada,giovanifarias,marlamelo.sinfo,m2gonsalvez,paulofglr,

raq.mbarbosa,vimiciusbormar27}@gmail.com

Abstract. *This paper aims to present the general view of the modeling of a computer game based on MultiAgent Systems (MAS) and Role-Playing Game (RPG) for the management of natural resources. In this way, the ComMod methodology was used considering the São Gonçalo and Mirim Lagoon basin. The initial problem modeling was performed using the integration diagram, representing the problem overview, and the UML diagrams (class), which expose the formalization of actions and the interactions among the agents of the system.*

Resumo. *O presente artigo tem como objetivo apresentar a visão geral da modelagem de um jogo computacional baseado em Sistemas Multiagente (SMA) e jogos de papéis (RPG) para o gerenciamento de recursos naturais. Desta forma, utilizou-se a metodologia ComMod, considerando a bacia hidrográfica do São Gonçalo e da Lagoa Mirim. A modelagem inicial do problema foi realizada utilizando o diagrama de integração, representando a visão geral do problema, e o diagrama UML (classe), que expõe a formalização das ações e as interações entre os agentes do sistema.*

1. Introdução

O gerenciamento de recursos naturais é uma área que busca melhores formas de gerenciar terras, águas, plantas e animais, baseado em qualidade de vida das pessoas no presente e para gerações futuras. Essa área ganhou visibilidade com a noção de desenvolvimento sustentável, fazendo com que os governos vejam e compreendam o mundo de outra forma. O gerenciamento dos recursos naturais foca especificamente no entendimento técnico científico de recursos e ecologia e como esses recursos podem dar suporte à vida animal [Holzman 2009].

De acordo com [Fuller et al. 2007], existem três desafios computacionais ligados ao gerenciamento de recursos naturais: gerenciamento e comunicação de dados, análise de dados, e controle e otimização. Para resolver tais desafios, a utilização de ferramentas computacionais com técnicas de inteligência artificial (IA), entre elas sistemas multiagente, pode ser uma solução, visto que elas têm a flexibilidade necessária para tratar a dinâmica existente em recursos naturais.

Os sistemas multiagente estudam o comportamento de um conjunto independente de agentes com diferentes características, evoluindo em um ambiente comum. Esses agentes interagem uns com os outros e tentam executar suas tarefas de forma cooperativa, compartilhando informações, evitando conflitos e coordenando a execução das atividades [Gilbert and Troitzsch 2005]. Adicionalmente, o uso de simulação como ferramenta de apoio à tomada de decisão é eficiente, porque é possível verificar detalhes com grande precisão [Frozza 1997]. Simulação Multiagente (*Multi-Agent-Based Simulation* - MABS) é a união de Sistemas Multiagente e Simulação e é utilizado por procurar unir perspectivas interdisciplinares de estudo [Le Page et al. 2015] [Page et al. 2016].

A utilização integrada de MABS e RPG *(Role-Playing Game)*, a qual consiste em uma técnica onde os jogadores "interpretam" uma personagem, criada dentro de um determinado cenário (ambiente) [Adamatti 2007], iniciou-se com pesquisas desenvolvidas pelo CIRAD[1] (*Centre de coopération internationale en recherche agronomique pour le développement*), França, no início dos anos 2000. Esse grupo desenvolveu uma metodologia denominada "*ComMod: The Companion Modelling* approach"[2], onde os participantes têm um importante papel no processo de tomada de decisão e entendimento dos problemas socioambientais a serem resolvidos, sendo que trabalhos já foram realizados em diferentes países do mundo, incluindo o Brasil.

Este estudo tem como objetivo a utilização de simulação multiagente (MABS) e jogos de papéis (RPG) com a finalidade de obter-se uma gestão participativa dos recursos hídricos. A pesquisa tem como foco a base de dados das bacias hidrográficas da Lagoa Mirim e do Canal São Gonçalo, no Rio Grande do Sul, das quais envolvem, entre outras, as cidades de Rio Grande e Pelotas. O projeto visa a aplicação-piloto do trabalho no Comitê de Gerenciamento das Bacias Hidrográficas da Lagoa e do Canal.

O artigo está organizado da seguinte forma. Na Seção 2 é apresentada a modelagem inicial do problema, bem como os diagramas de integração e classe com as especificações destes e, na Seção 3 são apresentadas as conclusões e as próximas etapas deste estudo.

2. Modelagem Inicial do Problema

A modelagem inicial do problema consiste em representar as interações básicas entre os agentes do sistema, os papéis e a atuação destes no ambiente, conforme Figura 1. Neste estudo, os agentes são classificados de acordo com os papéis que assumem e divididos em três grupos (reguladores, fiscalizadores ou produtores).

Agentes reguladores são responsáveis por administrar os recursos financeiros, oriundos de impostos e taxações atrelados à sociedade, com o objetivo de contro-

[1] www.cirad.fr

[2] https://www.commod.org/en

lar/mitigar a poluição (através da criação de leis, incentivos fiscais, obras para diminuir a poluição, etc.) sem prejudicar os mecanismos de produção. Neste ambiente, os agentes que assumem papéis de prefeito ou vereador pertencem ao grupo dos agentes reguladores, os quais podem negociar entre si (por exemplo, através de troca de mensagens) para decidir quais ações realizar no ambiente.

Agentes fiscalizadores têm como objetivo fiscalizar ou informar irregularidades atreladas à produção e exploração do ambiente. Conforme apresentado na Figura 1, agentes fiscalizadores são aqueles que assumem os papéis de fiscal da agência ambiental (por exemplo, FEPAM no RS) ou ONG (Organização Não-Governamental).

Neste caso, o fiscal é responsável por fiscalizar os agentes que pertencem ao grupo dos produtores, isto é, o fiscal pode, por exemplo, aplicar multas aos agentes produtores que forem pegos infringindo alguma lei/regra imposta pelos agentes reguladores. A ONG é responsável por informar aos agentes reguladores o estado atual dos níveis de poluição do ambiente, com o objetivo de conscientizar/pressionar os outros agentes a realizarem ações que diminuam os níveis de poluição.

Agentes produtores são responsáveis por explorar o ambiente com o objetivo principal de obter recursos financeiros. Estes agentes são os maiores geradores de poluição e, consequentemente, de recursos financeiros no ambiente, podendo assumir os papéis de empresário ou agricultor. O empresário é responsável por disponibilizar equipamentos e insumos necessários para a produção. No entanto, o agricultor é responsável por utilizar os equipamentos e insumos que julgar mais adequado para a sua produção. Deste modo, a interação entre os agentes produtores ocorre através da compra/aluguel e venda de equipamentos e insumos, onde um agente agricultor pode comprar de um agente empresário e consequentemente um empresário pode vender/alugar a um agricultor.

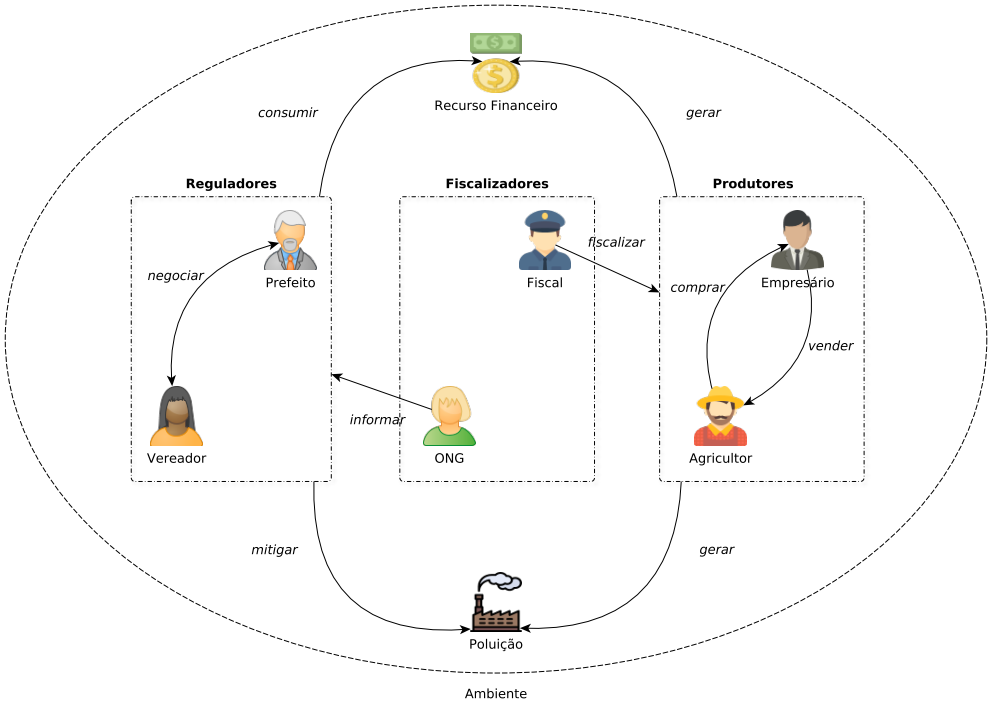


Figura 1. Diagrama de Integração

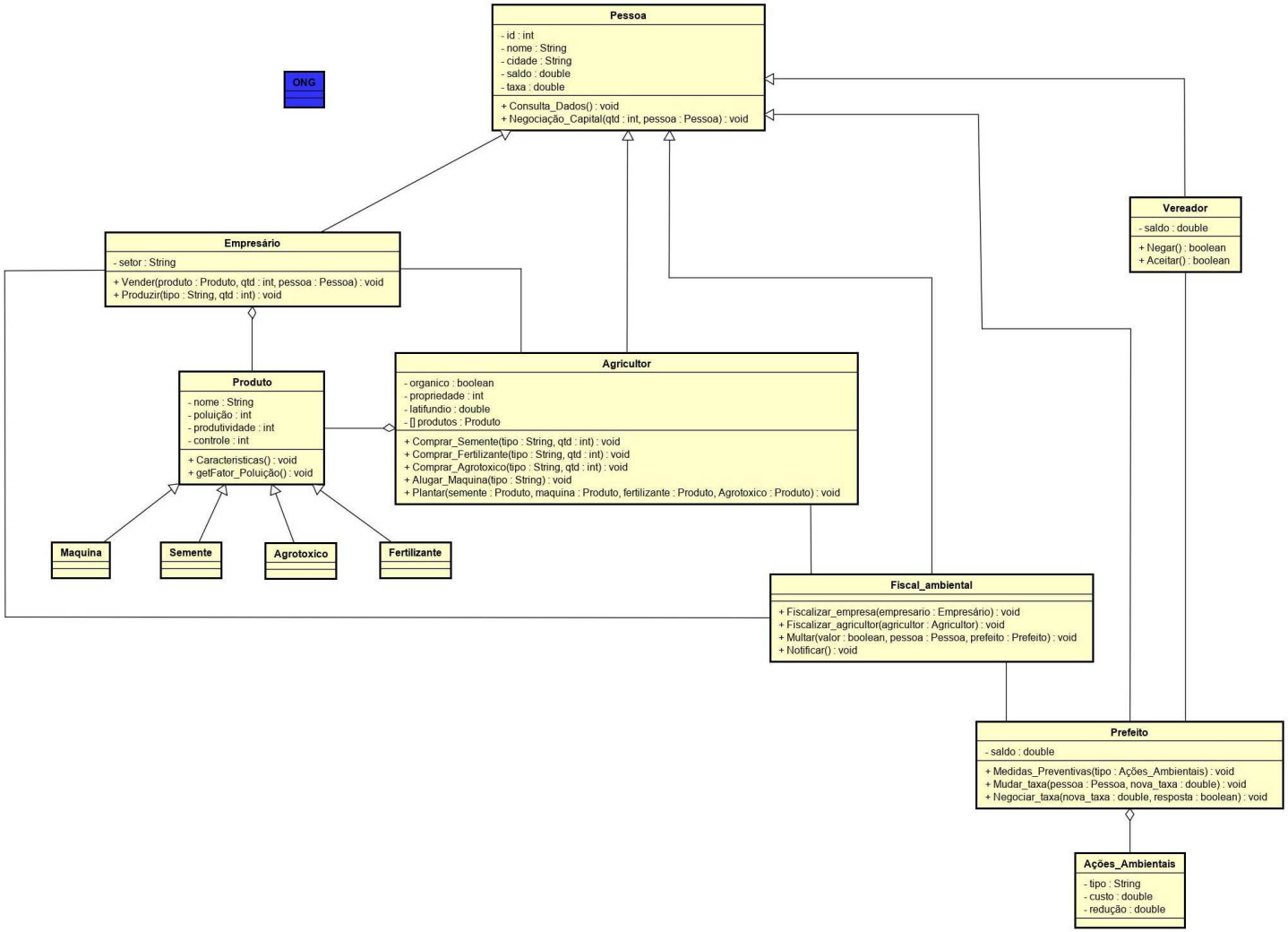


Figura 2. Diagrama de Classes Geral

No contexto deste estudo, a utilização da *Unified Modelling Language* (UML) para a modelagem do sistema torna-se uma abordagem adequada, visto que, tanto os papéis quanto as ações destes podem ser visualizadas de forma objetiva. Para a modelagem inicial foi utilizado o diagrama de classe, modelado na ferramenta Astah UML[3].

O objetivo desta forma de representação é a definição das classes a serem implementadas no sistema, assim como apresentar seus atributos, operações e relações. Neste diagrama (vide Figura 2) é representada a estrutura de classes de cada um dos agentes, mostrando as interações com as respectivas classes agregadas e a comunicação entre eles.

A superclasse da qual todas as outras derivam e compartilham os atributos e métodos é a classe Pessoa, que generaliza as operações mais básicas exercidas pelos agentes. Na subclasse Empresário é feita a agregação da classe Produto, que definirá a função empresarial que este terá. A comunicação com a classe Agricultor representa os trâmites de compra e venda, e com o Fiscal_Ambiental, representa a fiscalização e os procedimentos de multa devido à excedência de poluição.

Na subclasse Agricultor também é feita a agregação da classe produto, porém com função similar a de um estoque, separando os tipos e subtipos de produtos. A comunicação entre essa classe e a Empresário é necessária por causa da relação compra e venda que apresentam (como citado). E, também como na classe Empresário, a comunicação com a classe Fiscal_Ambiental representa a fiscalização. Na subclasse Fiscal_Ambiental a comunicação com as subclasses Empresário e Agricultor representa a fiscalização exercida nas duas funções, e com o Prefeito representa a reportação das multas aplicadas e a situação ambiental.

Na subclasse Prefeito é feita a comunicação com o Fiscal_Ambiental, comentado anteriormente. A comunicação com a classe Vereador é dada para fins políticos como alterações de taxas impostos, criação de novas taxas, etc. O papel da subclasse Vereador é dada em conjunto com a subclasse Prefeito, já comentada. A classe ONG é uma classe que representa uma interface da situação atual do jogo que notifica e interage com as outras classes, porém, no papel de um NPC (do inglês, *Non Player Character*).

3. Conclusões e Próximos Passos

O presente artigo apresentou a modelagem, em fase inicial de desenvolvimento, de um jogo computacional baseado em sistema multiagente e jogos de papéis para a gestão de recursos naturais.

No processo de desenvolvimento desta modelagem, aplicou-se a abordagem ComMod (que utiliza sistemas baseados em agentes e a vantagem dos jogos de papéis, visto que estes possibilitam a simulação de inúmeros cenários sociais) para apoiar os processos de tomada de decisão e entendimento dos problemas socio-ambientais, na gestão participativa que procura incluir todos os indivíduos que estão envolvidos em uma questão. E também, o diagrama de integração, sendo possível uma visão geral dos atores e suas possíveis ações e, o diagrama de classe que propiciou a definição e formalização destas ações (métodos da classe) e as interações (relacionamentos) dos agentes.

Os próximos passos consistirão na: (i) avaliação de ferramentas para desenvolvimento de sistemas baseados em agentes; a partir da análise realizada, (ii) selecionar a fer-

[3]http://astah.net/

ramenta mais apropriada para a concepção de um modelo computacional para o problema apresentado; e, finalmente, (iii) a implementação dos agentes, com suas características e comportamentos, bem como o processo de interação entre eles.

Um forte candidato para desenvolvimento do sistema proposto é o ambiente de desenvolvimento GAMA (https://gama-platform.github.io/), que permite a modelagem e a simulação de agentes espacialmente explícitos. Outra ferramenta em análise é o *framework* JaCaMo (http://jacamo.sourceforge.net/), que combina três tecnologias separadas (Jason, CArtAgO e $\mathcal{M}$OISE) para o desenvolvimento de sistemas multiagente.

Agradecimentos

Os autores deste artigo agradecem ao Programa de apoio ao Ensino e à Pesquisa Científica e Tecnológica em Regulação e Gestão de Recursos Hídricos – Pró-Recursos Hídricos Chamada N° 16/2017, pelo auxílio financeiro no desenvolvimento desta pesquisa.

Referências

Adamatti, D. F. (2007). *Inserção de jogadores virtuais em jogos de papéis para uso em sistemas de apoio à decisão em grupo: um experimento no domínio da gestão de recursos naturais*. PhD thesis, Escola Politécnica – Universidade de São Paulo, São Paulo, Brasil.

Frozza, R. (1997). Simula: Ambiente para desenvolvimento de sistemas multiagentes reativos. Master's thesis, Universidade Federal do Rio Grande do Sul (UFRGS).

Fuller, M. M., Wang, D., Gross, L. J., and Berry, M. W. (2007). Computational science for natural resource management. *Computing in Science & Engineering*, 9(4):40.

Gilbert, N. and Troitzsch, K. (2005). *Simulation for the social scientist*. McGraw-Hill Education (UK).

Holzman, B. (2009). Natural resource management. [Online; accessed 30 apr. 2019] http://online.sfsu.edu/bholzman//courses/GEOG 20657/.

Le Page, C., Bobo Kadiri, S., Towa, K., William, O., Ngahane Bobo, F., and Waltert, M. (2015). Interactive simulations with a stylized scale model to codesign with villagers an agent-based model of bushmeat hunting in the periphery of korup national park (cameroon). *Journal of Artificial Societies and Social Simulation*, 18(1).

Page, C. L., Dray, A., Perez, P., and Garcia, C. (2016). Exploring how knowledge and communication influence natural resources management with rehab. *Simulation & Gaming*, 47(2):257–284.

Estudo das emoções em multidões

Michael O. Trujillo[1], Marla Melo[1], Diana Adamatti[1], Leonardo Emmendorfer[1]

[1]Centro de Ciências Computacionais
Universidade Federal do Rio Grande (FURG)
Caixa Postal 474 - 96201-900 – Rio Grande – RS – Brazil

molmost@unal.edu.co, marlamelo.sinfo@gmail.com

dianaada@gmail.com, leonardo.emmendorfer@gmail.com

Abstract. *It is explore the problem of the mathematical modeling of emotions in a crowd using the definition of emotions as infections. To do this, it is used an implementation in Netlogo where shows the propagation of an infection in a population. So, this implementation is modified to be used in the study of emotions in a crowd. It is study a particular case of the implementation and finally it is do it a mathematical modeling of it.*

Resumo. *Neste trabalho é explorado o problema da modelagem matemática das emoções em uma multidão usando a definição de emoções como infecções. Para fazer isso, é usada uma implementação no Netlogo que mostra a propagação de uma infecção em uma população. Assim, a implementação é modificada para ser usada no estudo das emoções em uma multidão e finalmente é realizada uma modelagem matemática deste caso particular.*

1. Introdução

O estudo das emoções é tão complexo que tem sido desenvolvidas diferentes teorias das emoções (Teorias Appraisals, Teorias Dimensionais, Racionais, anatômicas [Scherer et al. 2001] [Marsella et al. 2010]) com o propósito de explicá-las. Assim, estudar as emoções em uma multidão deveria ser uma tarefa ainda mais complexa. O problema motivador deste estudo é *Como estudar as emoções em uma multidão tendo em conta a complexidade de uma multidão e das emoções de cada indivíduo?*

Assim, o objetivo deste trabalho é tentar estabelecer um caminho para responder a pergunta. Para abordar este problema vai se observar as emoções como infecções e vai se adaptar uma implementação do Software Netlogo[1], onde é estudado o problema das infecções em multidões. É feita uma reinterpretação da simulação para o caso das emoções em uma multidão e finalmente é estudada uma modelagem matemática da propagação das infecções em uma multidão e é reinterpretada para o caso das emoções. Devido à complexidade do problema é estudada somente uma emoção discreta (*ficar feliz)* (o conceito desta emoção está embasado na teoria OCC [Ortony et al. 1990]) para um caso particular da simulação no Netlogo.

Assim, a estrutura do artigo é feita da seguinte maneira: A seção 2 apresenta a metodologia. A seção 3 apresenta uma revisão bibliográfica. Na seção 4 estão os trabalhos relacionados ao tema. A seção 5 apresenta a simulação realizada e o modelo matemático. A seção 6 as conclusões do trabalho.

[1]NetLogo [Wilensky et al. 1999] é uma linguagem de programação e um ambiente de modelagem multiagente para simular fenômenos naturais e sociais [Tisue and Wilensky 2004]. A ferramenta disponibiliza uma biblioteca de modelos para aprendizagem.

2. Metodologia

A metodologia do trabalho é baseada na teoria de contágio das emoções [Hatfield et al. 1994] e no comportamento das multidões estudado em diferentes pesquisas. Primeiramente, estuda-se a teoria OCC das emoções, que mostra as emoções no caso de um indivíduo. A seguir, é preciso estudar o comportamento de uma multidão para propor um modelo um pouco mais realista com respeito as emoções. O problema de estudar as emoções tendo em conta o comportamento das multidões é que o comportamento é bastante complexo, por isso, somente é estudada uma emoção (estado emocional) *ficar feliz*.

Após, é proposta uma simulação para avaliar as interações entre emoções na multidão, tendo em conta que cada indivíduo tem seu próprio comportamento e suas próprias emoções. Isto quer dizer que a emoção na simulação tem um caráter aleatório para cada agente, mas cada agente pode afetar o comportamento dos agentes de seu ambiente.

3. Referencial Teórico

3.1. Multidão

Uma multidão pode ser definida como um grande grupo de indivíduos no mesmo ambiente físico, onde a multidão não precisa ter uma estrutura de comportamento [Adamatzky 2005]. Muitas vezes, em uma multidão, o comportamento das pessoas é semelhante mas, individualmente, na mesma situação, este indivíduo não atuaria da mesma maneira [Banarjee et al. 2005]. Por exemplo, caminhar nu pela rua não é uma ação que as pessoas fazem de maneira individual, mas em certo tipo de multidões é uma ação com certo tipo de objetivo real. Isso porque, segundo Graumann (1985) [Adamatzky 2005], quando um indivíduo se une a uma multidão, atua de uma maneira irracional, de acordo com os seguintes conceitos [Adamatzky 2005]:

- desindividualização: o anonimato dos membros da multidão e o senso de poder invencível produzido por estar na multidão levam a uma difusão de seus sentimentos de responsabilidade pessoal, uma perda de identidade pessoal;
- contágio: ações e emoções se espalham através da multidão através de uma forma de imitação mútua, levando à uniformidade e homogeneidade em que as diferenças pessoais desaparecem;
- sugestionabilidade: aceitação de influência em bases irracionais por algum tipo de ligação emocional e atitude submissa a uma pessoa ou grupo.

3.2. Emoções em uma multidão

Em uma multidão, as emoções podem ser consideradas como moléculas, no sentido da quantidade muito grande de interações entre as pessoas [Adamatzky 2005]. As interações em uma multidão podem ser imprevisíveis, e, no geral, não tem um padrão racional, pelo contrário, um padrão emocional e irracional [Adamatzky 2005]. Neste trabalho as interações são consideradas como transmissão de emoções, isto é, emoções pensadas como infecções. Por exemplo, em [Hill et al. 2010] o estado emocional "contente" é tratado como uma infecção, onde a sua transmissão é feita pelo contato das pessoas na sua rede social. O contágio das emoções, é descrito em [Hatfield et al. 1994] como transmissão de informação de uma maneira consciente ou inconsciente.
Tendo em conta o anterior, neste trabalho a emoção *ficar contente* é definida como *uma reação a um evento externo (situação externa)* [Ortony et al. 1990].

4. Trabalhos Relacionados

Dentro do escopo do trabalho alguns artigos relacionados foram encontrados com respeito ao contágio das emoções. Foram classificados dessa maneira com o propósito de ter uma revisão bibliográfica para trabalhos futuros nessas áreas.
Artigos no grupo *SIR - SISa* são pesquisas que usam os modelos (SIR - SISa) de como é propagada uma infecção. Trabalhos no grupo *Modelos Computacionais* são artigos que tem alguma implementação computacional relacionada com a ideia de emoções como infecções. Da mesma maneira, no grupo *Redes Sociais* estão trabalhos que estudam a propagação das emoções olhadas como infecções nas redes sociais.

SIR - SISa	Modelos computacionais	Redes Sociais
[Fu et al. 2014]	[Tsai et al. 2011]	[Baht 2017]
[Hill et al. 2010]	[Bispo and Paiva 2009]	[Coviello et al. 2014]

Tabela 1

5. Simulação Realizada

Neste trabalho, foi utilizado o modelo *vírus* [Wilensky 1998], disponível na ferramenta NetLogo, para realizar a simulação. Nesta simulação, o agente passa por três estágios: *neutro, infected, immune*. O estado *neutro* quer dizer que o agente é suscetível a ficar infectado; o estado *infected* quer dizer que o agente tem o vírus durante um intervalo de tempo; e o estado *immune* quer dizer que o agente é saudável e não tem risco de pegar a infecção durante um intervalo de tempo.Neste modelo, os agentes podem morrer e também podem se reproduzir. Também o tempo de imunidade, com respeito ao vírus, é fixa com um valor de 52 semanas. Este modelo é baseado no modelo epidemiologico SIR (*Susceptible, Infected, Recovered*). O modelo usa uma população fixa quando é iniciado, podendo ter, no máximo, 300 agentes.
Para o caso do contágio das emoções, foram feitas algumas mudanças no código: os agente não morrem e não se reproduzem; e o estado *infected* foi mudado pelo estado *happy*. Além disso, foi adicionado um botão *neutral* que permite mudar a quantidade de agentes no estado neutral. Também foi adicionada uma barra para fazer a variação do tempo de imunidade dos agentes, onde o estado de imunidade quer dizer uma resistência para ficar contente (*happy*).

5.1. Modelo Matemático

O cenário particular da simulação mostra uma relação entre as variáveis *happy, neutral e immune*, conforme Figura 1.

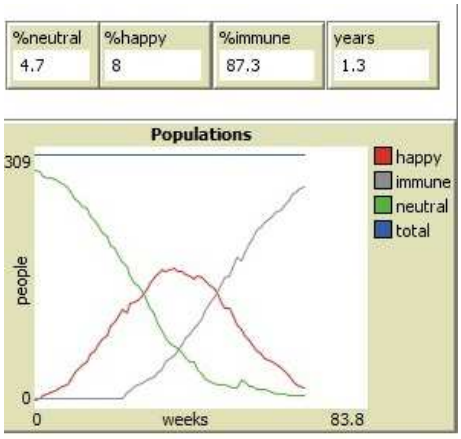


Figura 1. Relacionamento entre as variáveis no modelo modificado no NetLogo.

Para tentar explicar a relação entre as variáveis é estudado um modelo matemático que está baseado em [Keeling and Rohani 2011].
Esta relação é modelada matematicamente, via modelo SIR, com as seguintes equações:

$$\frac{dS}{dt} = -\beta SI, \tag{1}$$
$$\frac{dI}{dt} = \beta SI - \gamma I, \tag{2}$$
$$\frac{dR}{dt} = \gamma I, \tag{3}$$
$$S + I + R = 300, \tag{4}$$
$$S(0) = 280, \tag{5}$$
$$I(0) = 10, \tag{6}$$
$$R(0) = 10, \tag{7}$$
$$R_0 = \frac{\beta}{\gamma}, \tag{8}$$

Onde:

- *S* apresenta a população suscetível a ficar contente;
- *I* apresenta a população contente (infectada), e;
- *R* apresenta a população imune (resistentes a ficar contente);
- γ é a razão de mudança da recuperação, isto é, a razão de mudança de um agente de passar de estado contente ao estado imune (resistente a ficar contente).
- [Keeling and Rohani 2011] explica que o fator $\frac{1}{\gamma}$ apresenta a duração da infecção, isto é, o estado emocional *ficar contente*;
- O fator $\frac{\gamma}{\beta}$ apresenta um limite, onde se $S(0) < \frac{\gamma}{\beta}$, a infecção não vai se propagar, isto é , o estado emocional *ficar contente* não vai se propagar;
- [Keeling and Rohani 2011] também explica que o fator $R_0 = \frac{\beta}{\gamma}$ apresenta o potencial máximo de reprodução da infecção, isto é, o máximo valor que a transmissão da infecção pode atingir.

As soluções do modelo matemático foram as seguintes:

$$S(t) = S(0)e^{-R(t)R_0}, \tag{9}$$
$$= 280e^{-R(t)R_0}, \tag{10}$$

$$R(t) = \frac{1}{R_0^2 S(0)}(S(0)R_0 - 1 + \alpha tanh(\frac{1}{2}\alpha\gamma t - \phi)), \tag{11}$$

$$\alpha = ((S(0)R_0 - 1)^2 + 2S(0)I(0)R_0^2)^{\frac{1}{2}}, \tag{12}$$

$$\phi = tanh^{-1}(\frac{1}{\alpha}(S(0)R_0 - 1)), \tag{13}$$

Tendo resolvido $R(t)$ pelas equações (11), (12) e (13) e também tendo resolvido $S(t)$ pelas equações (9) e (10), é possível resolver também $I(t)$, substituindo as equações (9) e (11), via equação (4).

A solução do sistema expressa a seguinte relação entre as variáveis segundo o estudo de [Keeling and Rohani 2011] e apresentado na figura 2

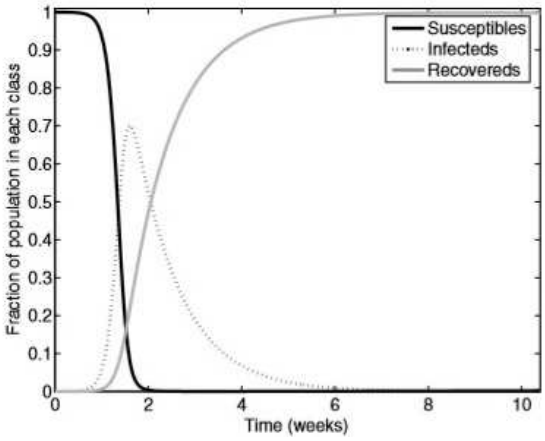


Figura 2. Modelo de infecção para o modelo SIR [Keeling and Rohani 2011].

6. Conclusões

Este trabalho mostrou que é possível usar uma simulação do Netlogo (feita para a difusão de um vírus em uma multidão) para ter uma simulação do estado emocional *ficar contente*, baseado na ideia de emoções como infecções.

Graças a essa simulação, foi possível estudar um caso particular e chegar a uma modelagem matemática. É preciso esclarecer que a semelhança dos comportamentos é somente visual, sendo necessário uma comparação dos dados da simulação modificada e das funções, para ter-se uma validação completa entre modelo matemático e modelo de simulação.

7. Discussões

- A semelhança das figuras 1 e 2 não é uma validação do modelo matemático.
- As modificações feitas na implementação não são uma adaptação no sentido próprio do termo.
- O trabalho feito oferece uma novidade na pesquisa?

Referências

Adamatzky, A. (2005). *Dynamics of crowd-minds: Patterns of irrationality in emotions, beliefs and actions*, volume 54. World Scientific.

Baht, S. A. (2017). Social networking sites and mental health: A review. *International Journal of Advanced Educational Research*, 2:357–360.

Banarjee, S., Grosan, C., and Abraham, A. (2005). Emotional ant based modeling of crowd dynamics. In *Symbolic and Numeric Algorithms for Scientific Computing, 2005. SYNASC 2005. Seventh International Symposium on*, pages 8–pp. IEEE.

Bispo, J. and Paiva, A. (2009). A model for emotional contagion based on the emotional contagion scale. In *2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*, pages 1–6. IEEE.

Coviello, L., Sohn, Y., Kramer, A., and Baht, S. A. (2014). Detecting emotional contagion in massive social networks. *Plos One*.

Fu, L., Song, W., Lv, W., and Lo, S. (2014). Simulation of emotional contagion using modified sir model: a cellular automaton approach. *Physica A: Statistical Mechanics and its Applications*, 405:380–391.

Hatfield, E., Cacioppo, J. T., and Rapson, R. L. (1994). *Emotional Contagion*. Cambridge University Press.

Hill, A. L., Rand, D. G., Nowak, M. A., and Christakis, N. A. (2010). Emotions as infectious diseases in a large social network: the sisa model. *Proceedings of the Royal Society B: Biological Sciences*, 277(1701):3827–3835.

Keeling, M. J. and Rohani, P. (2011). *Modeling infectious diseases in humans and animals*. Princeton University Press.

Marsella, S., Gratch, J., Petta, P., et al. (2010). Computational models of emotion. *A Blueprint for Affective Computing-A sourcebook and manual*, 11(1):21–46.

Ortony, A., Clore, G. L., and Collins, A. (1990). *The cognitive structure of emotions*. Cambridge university press.

Scherer, K. R., Schorr, A., and Johnstone, T. (2001). *Appraisal processes in emotion: Theory, methods, research*. Oxford University Press.

Tisue, S. and Wilensky, U. (2004). Netlogo: A simple environment for modeling complexity. In *International conference on complex systems*, volume 21, pages 16–21. Boston, MA.

Tsai, J., Bowring, E., Marsella, S., and Tambe, M. (2011). Empirical evaluation of computational emotional contagion models. In *International Workshop on Intelligent Virtual Agents*, pages 384–397. Springer.

Wilensky, U. (1998). Netlogo virus - alternative visualization model. `http://ccl.northwestern.edu/netlogo/models/Virus-AlternativeVisualization`. Center for Connected Learning and Computer-Based Modeling.

Wilensky, U. et al. (1999). Center for connected learning and computer-based modeling. In *Netlogo*. Northwestern University.

Tolerância a Faltas em Sistemas Multiagentes Multidimensionais

Luis P. Lampert[1], **Jomi F. Hübner**[1], **Maicon R. Zatelli** [1]

[1]Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

lp.lampert@gmail.com, jomi.hubner@ufsc.br, maicon.zatelli@ufsc.br

Resumo. *A autonomia presente nos Sistemas Multiagentes (SMA) é uma característica que contribui para que esse tipo de sistema seja suscetível a faltas. Com a finalidade de aumentar a confiabilidade, a tolerância a faltas aplicada a SMA é um tópico que vem ganhando destaque. Este trabalho tem como objetivo propor um modelo de tolerância a faltas para SMA que utiliza as abstrações fornecidas pela programação multidimensional (agentes, ambiente e organização). O modelo proposto adiciona a capacidade de monitoramento do estado dos agentes através da instrumentação do ambiente. Além disso, propõe a adição de agentes especializados, que têm a capacidade de monitorar e atuar nos casos de detecção de faltas no sistema.*

1. Introdução

A utilização de computadores é tão presente no cotidiano das pessoas que poucas vezes é questionada qual a confiabilidade dos serviços que esses sistemas oferecem. A ocorrência de falhas nesses sistemas pode ter consequências que variam de simples incomodações até eventuais catástrofes, com sérios prejuízos econômicos e até perda de vidas humanas. Nesse cenário, a *tolerância a faltas* (TF) pode ser descrita como sendo um meio para garantir que o sistema entregue o serviço que originalmente foi projetado, mesmo na presença de erros originados por faltas [Laprie 1992].

Segundo [Potiron et al. 2013], quando se fala de sistemas computacionais tradicionais, a TF é um tópico estudado a bastante tempo e essa base comum de conhecimento é amplamente compartilhada, o que facilita as especificações de sistemas, implementações de técnicas de TF e comparações entre diferentes abordagens. Entretanto, no caso dos SMA, não existem muitos trabalhos que analisam a TF e levam em consideração as particularidades da *autonomia*, que é a principal característica que distingue um SMA de um sistema tradicional. Assim, o estudo de abordagens que sejam adaptadas às especificidades dos SMAs mostra-se importante para o aprimoramento da confiabilidade dos SMAs, permitindo que esses sejam utilizados cada vez mais em aplicações que necessitam de uma maior robustez.

A fim de contribuir para o desenvolvimento da tecnologia de SMA, o presente trabalho apresenta uma proposta inicial de modelo de TF adaptado aos SMA que utilizam as abstrações de multidimensões (agente, ambiente e organização), apresentadas no trabalho de [Boissier et al. 2013]. Até onde se investigou, não foi encontrado trabalho que utilizasse um modelo de TF em SMA e considerasse outras dimensões além da dos agentes. O trabalho faz parte de uma dissertação em andamento e apresenta um estudo

das abordagens de tolerância a faltas existentes na Seção 2, propõe um modelo adaptado à programação multidimencional na Seção 3 e, por último, na Seção 4 faz algumas observações finais.

2. Tolerância a Faltas em Sistemas Multiagentes

Em sistemas computacionais o termo dependabilidade aparece com frequência, trata-se de uma tradução literal do termo em inglês *dependability*, e indica a qualidade do serviço oferecido pelo sistema. Quando a dependabilidade é deficiente, três efeitos podem surgir: a *falha*, o *erro* e a *falta*. Por definição, uma falha ocorre quando o sistema não produz a mesma entrega para o qual este foi especificado inicialmente. O erro é uma parte dos estados do sistema que podem levar a uma falha subsequente. O surgimento de um erro é uma indicação de que uma falta ocorreu no sistema [Laprie 1992]. Quando adicionamos métodos de tolerância a faltas ao sistema, estamos tentando prevenir a ocorrência das falhas através de ações que devem ser tomadas quando os erros são detectados, ou seja, os erros podem ser detectados e, nesse momento, as técnicas de tolerância a faltas devem entrar em ação para tentar evitar que falhas aconteçam.

Para o projeto de um sistema tolerante a faltas é imprescindível que o projetista conheça quais são as faltas que o sistema está suscetível. Entretanto, quando se trata de um SMA, esse tipo de previsão não pode ser feita pois os agentes do sistema possuem leis e objetivos próprios e, em muitos casos, o projetista não tem acesso ou conhecimento dessas informações [Potiron et al. 2013]. Essa *autonomia* é a característica fundamental dos SMA e deve ser levada em conta em qualquer projeto de tolerância a faltas.

Muitos trabalhos que abordam a TF em SMA acabam fazendo abordagens bastante específicas, normalmente lidando com apenas alguns modos de falta, que são os mais comuns para aquele sistema específico [Isong and Bekele 2013]. Os trabalhos de [Stanković et al. 2017] e [Isong and Bekele 2013] investigaram diversas abordagens existentes e listaram três principais características que as classificam: *tipo de detecção de faltas*, *protocolo de tolerância a faltas* e *nível de manipulação de falta*.

2.1. Detecção de Faltas

A detecção de faltas é uma característica chave para as abordagens de TF pois, sem uma detecção adequada, mesmo o melhor tipo de tratamento será ineficaz. As técnicas de detecção podem ser divididas em duas categorias: detecção por *batimento cardíaco* e através da *interação*. Na detecção por batimento cardíaco, as faltas em agentes são identificadas através da troca de mensagens periódicas de confirmação (*acknowledge*) entre agentes comuns e agentes monitores. Essa técnica possui uma implementação simples e permite uma arquitetura modular, porém acaba inundando o sistema com uma série de mensagens. Na detecção por interação, aproveita-se a relação já existente entre agentes, identificando possíveis erros através de ferramentas como *timeout* ou erros em retorno de chamadas (*callbacks*).

2.2. Protocolo de Tolerância a Faltas

O protocolo de TF define como será o procedimento padrão de ação na ocorrência de faltas no sistema. Um dos protocolos mais utilizado é chamado de *não bloqueante*, que permite o reinício de processos que falharam. Dentro dessa categoria, as técnicas que frequentemente aparecem são baseadas em *recuperação de estados* anteriores e na *replicação* de

agentes. A recuperação tenta restaurar o sistema a um estado anterior à ocorrência da falta, evitando assim que processos devam ser reiniciados de pontos iniciais muito distantes dos atuais. Essa técnica necessita que sejam feitos processos periódicos de gravação dos estados do sistema, conhecidos como *checkpoints*, o que acaba demandando custo computacional. A abordagem por replicação utiliza a ideia de manter copias de agentes no sistema, que podem substituir eventuais agentes em estado de erro.

2.3. Nível de Manipulação de Faltas

A última característica indica em qual nível deve ocorrer o gerenciamento das exceções e a recuperação das faltas: no nível dos agentes ou do SMA. Esse tipo do classificação também é abordada no trabalho de [Klein et al. 2003], que utiliza as denominações de abordagem "sobrevivente" e "cidadã".

Na abordagem sobrevivente, adota-se a ideia de que cada agente deve ser responsável por manter a sua própria existência, ou seja, a programação das técnicas de TF devem ser adicionadas juntamente ao código de cada agente. A vantagem dessa abordagem é a independência de plataforma, que permite uma melhor interoperabilidade entre sistemas. Entretanto, uma maior responsabilidade recai sobre os programadores, que devem coordenar o comportamento dos agentes na ocorrência de faltas, que são, muitas vezes, difíceis de prever. Essa adição de código defensivo nos agentes resulta em uma maior dificuldade para manutenção, prejudica o desempenho do sistema e aumenta as chances de ocorrência de novas faltas inseridas pelos programadores.

A abordagem cidadã estabelece que os processos TF devem ser gerenciados por uma entidade especializada e externa ao agente, habitualmente chamada de *sentinela*. Utilizando uma analogia similar a sociedade humana, a abordagem cidadã adota a ideia da criação de instituições que são reconhecidas pelos agentes (os cidadãos). Por exemplo a polícia em uma sociedade, essa organização oferece serviço de segurança aos cidadãos e, em troca, é reconhecida por eles como uma autoridade que tem permissão para atuar no estado de cada um deles, podendo, por exemplo, prender indivíduos que estejam causando problemas aos demais. Segundo [Klein et al. 2003], a principal vantagem neste tipo de abordagem é a noção de expertise para tratamento de faltas, simplificando assim a programação dos agentes "comuns", que ficam focados nos seus objetivos mais básicos. A principal desvantagem dessa abordagem é a redução da interoperabilidade entre sistemas devido à necessidade de utilizar plataformas específicas.

2.4. Trabalhos relacionados

Em seu trabalho, [Hägg 1997] introduziu o termo *sentinela*, uma classe especial de agente que tem a função de proteger algumas funcionalidades do sistema. Esse tipo de agente não pertence ao domínio da solução de problemas, ficando responsável por monitorar os agentes comuns do sistema e, se for detectado a ocorrência de alguma falta, o sentinela pode tomar ações corretivas, como escolher planos alternativos, excluir agentes, alterar parâmetros do sistema e reportar aos operadores humanos. O sentinela pode monitorar a comunicação dos agentes e interagir com eles, além de usar *timers* para detectar agentes mortos ou faltas em links de comunicação.

Em sua tese, [Díaz 2018] apresentou o eJason, um interpretador da linguagem de programação Jason em Erlang. Sua principal contribuição foi o projeto e implementação

deste ambiente de desenvolvimento para SMA que combina as sintaxes e semânticas do Jason, que é uma linguagem de programação para SMA que possui alto nível de abstração para programação de agentes BDI (*belief-desire-intention*), com as ferramentas de tolerância a faltas do Erlang, uma linguagem de programação com características que favorecem o desenvolvimento de aplicações concorrentes, distribuídas e com alta robustez. Entre as ferramentas de TF fornecidas pelo eJason, destacam-se dois tipos de relações que podem ser estabelecidas entre os agentes do sistema: a relação de *monitor* e de *supervisor*. A primeira relação permite que agentes monitores recebam informações sobre acontecimentos nos agentes monitorados. Já a segunda relação permite atuações especiais de controle do agente supervisor sobre os seus supervisionados.

3. Proposta de um Modelo de Tolerância a Faltas para Sistemas Multiagentes Multidimensionais

O modelo proposto neste trabalho faz uma analogia com um sistema de monitoramento de saúde remoto, onde existem equipamentos móveis que acompanham cada indivíduo, monitorando a sua saúde e enviando informações para equipes especializadas que podem agir em eventuais casos de emergência. Esses monitores possuem informações disponíveis em uma tela visível para quem tiver interesse em observa-lá, mas apenas integrantes das equipes especializadas têm acesso a informações sigilosas, habilidade de fazer diagnósticos a respeito da saúde do indivíduo e tomar ações corretivas.

Neste cenário, a proposta inicial de modelo é ilustrada na Figura 1, que representa o SMA com a adição das duas entidades básicas do serviço de TF: o artefato *monitor_de_saude*, instrumentado na dimensão do ambiente e os *agentes_de_saude*, que estão inseridos na dimensão dos agentes.

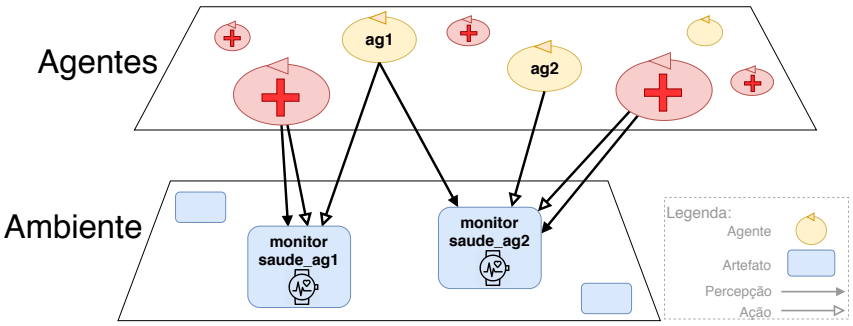


Figura 1. Modelo de TF proposto

Nesta configuração, cada agente do sistema pode ter um artefato monitor de saúde atrelado a si. O monitor mantém uma relação exclusiva com o agente, verificando periodicamente o seu estado de saúde. A saúde do agente é indicada no monitor em forma de batimento cardíaco e a ocorrência de faltas (morte do agente) aparece como a ausência de sinais vitais. Além do monitor cardíaco, outras informações podem ser verificadas nas propriedades observáveis do monitor de saúde, tais como os desejos e intenções dos agentes.

3.1. Exemplo ilustrativo

Para ilustrar o funcionamento do modelo proposto, propõem-se um SMA com três agentes principais: o agente `consumidor`, o agente `pizzaiolo` e o agente `entregador`.

O agente consumidor tem apenas um objetivo, que é `comer_pizza`, e para concretizar esse objetivo ele põe em prática o plano de `ligar_pizzaria` para fazer o seu pedido e fechar um contrato com o agente pizzaiolo. Na sequência, o agente pizzaiolo põe em prática o plano `fazer_pizza` e também `contratar_entrega` juntamente ao agente entregador que, ao ser contratado, executa as funções de `pegar_pizza` e depois `realizar_entrega`.

Durante a entrega da pizza, o agente entregador pode sofrer um acidente e falhar no seu objetivo de entregar o produto. Num sistema não tolerante a faltas o serviço contratado inicialmente pelo consumidor não será entregue e, nem ele, nem o pizzaiolo saberiam informar o que ocorreu com o sistema. Ao adaptar esse sistema ao modelo de TF, propõem-se que o agente entregador seja tolerante a faltas e, com isso, um artefato monitor de saúde é criado para monitorar o estado de saúde do entregador. Além disso, considera-se que, no momento em que o pizzaiolo contrata o entregador, ele passa a monitorar o artefato de saúde atrelado ao entregador, pois ele é diretamente interessado no serviço que este agente contratado está prestando. A Figura 2 ilustra o cenário proposto.

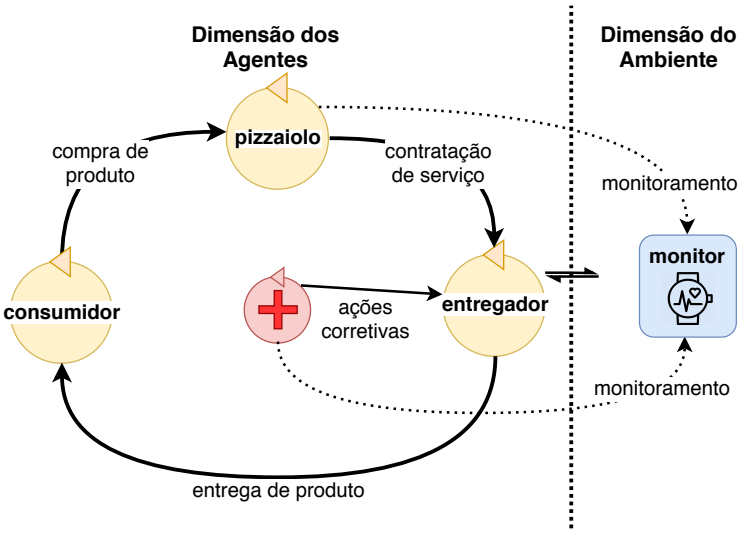


Figura 2. Cenário ilustrativo do SMA

Quando o modelo é implementado, alguns cenários se modificam. O monitor do entregador detecta problemas no batimento cardíaco e emite um aviso para um agente de saúde, que entra em ação e realiza um procedimento de reanimação. Nesse processo, uma cópia do estado mental do entregador é feita, elimina-se o agente em estado de erro e uma replica do agente é inserida no sistema. Caso o estado mental do novo agente não tenha as informações necessárias para executar o plano da entrega (i.e. o agente perdeu a pizza), pode ser executado um plano de contingência, onde o entregador combina de retornar à pizzaria para buscar novamente o pedido.

O agente pizzaiolo, por estar focado no artefato relacionado ao entregador, recebe sinais e mensagens com diversas informações e pode executar planos de contingência para o acidente como, por exemplo: entrar em contato com o cliente e avisar que a entrega do produto vai atrasar; se antecipar, produzindo uma nova pizza e contratando um novo serviço de entrega; aguardar a informação da reinserção do entregador no sistema e

solicitar que ele volte até a pizzaria para buscar o novo produto.

No final de todo processo, o entregador estará apto a finalizar o plano de entrega da pizza e, pela perspectiva do cliente, o serviço foi contratado e entregue corretamente, mesmo com ocorrência de uma falta durante a sua execução. Em muitos casos, o cliente nem ficará sabendo do ocorrido, pois sua preocupação está apenas com o produto final.

4. Conclusões e Trabalhos Futuros

Por se tratar de um trabalho de dissertação em andamento, essa primeira etapa é focada no estudo das abordagens de TF existentes e na elaboração de uma proposta inicial de modelo adaptado à programação multidimencional, que utiliza abstrações da dimensão dos agentes e do ambiente.

Há vários pontos que merecem discussão, um exemplo é como deve ser tratada a questão de informações sigilosas contidas nos artefatos monitores de saúde, nesse ponto é razoável pensar que devem ser criados níveis diferentes de acesso para monitoramento do próprio agente, dos demais agentes do sistema e dos agentes de saúde. Outro ponto de discussão diz respeito a como implementar as ações corretivas de maneira que o modelo fique o mais independe possível da plataforma de programação do SMA.

A sequência do trabalho prevê uma melhor discussão sobre esse e outros pontos do modelo, bem como a sua implementação e integração em uma plataforma de SMA multidimensional e a avaliação do modelo através de testes e análise resultados.

Referências

Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., and Santi, A. (2013). Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, 78:747–761.

Díaz, Á. F. (2018). *eJason : a Framework for Distributed and Fault-tolerant Multi-Agent Systems*. PhD thesis.

Hägg, S. (1997). A sentinel approach to fault handling in multi-agent systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1286:181–195.

Isong, B. E. and Bekele, E. (2013). A Systematic Review of Fault Tolerance in Mobile Agents. *American Journal of Software Engineering and Applications*, 2(5):111–124.

Klein, M., Rodriguez-Aguilar, J.-A., and Dellarocas, C. (2003). Using Domain-Independent Exception Handling Services to Enable Robust Open Multi-Agent Systems: The Case of Agent Death. *Autonomous Agents and Multi-Agent Systems*, 7(1/2):179–189.

Laprie, J. C. (1992). Dependability: Basic Concepts and Terminology. pages 3–245. Springer, Vienna.

Potiron, K., El Fallah Seghrouchni, A., and Taillibert, P. (2013). *From Fault Classification to Fault Tolerance for Multi-Agent Systems*. Number 9781447150459.

Stanković, R., Štula, M., and Maras, J. (2017). Evaluating fault tolerance approaches in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 31(1):151–177.

Algoritmos de coordenação para veículos autônomos em cidades inteligentes

Vilson de Deus Corrêa Júnior[1], **Tiago Luiz Schmitz**[1]

[1]Departamento de Engenharia de Software – Universidade do Estado de Santa Catarina (UDESC)
89140-000 – Ibirama – SC – Brazil

vilson.junior@edu.udesc.br, tiago.schmitz@udesc.br

Abstract. *The current paper presents multi-agents systems coordination algorithms. The introduced solution is a variation of the system developed and used by Akuanduba-UDESC, on the annual Multi-Agent Programming Contest - 2018. Throughout the paper it is going to be shown the utilized strategies, features and the preliminary results.*

Resumo. *O presente trabalho apresenta algoritmos para coordenação de sistemas multiagentes. A solução aqui apresentada é uma variação da solução desenvolvida e utilizada pela equipe Akuaduba-UDESC, na competição anual Multi-agent Programming Contest - 2018. Com isso, será apresentado ao longo do trabalho as estratégias adotadas, ferramentas e resultados preliminares.*

1. Introdução

Com a globalização o conceito de cidades inteligentes é utilizado em diversos países. Esse conceito se apresenta de modo disruptivo, com objetivo de solucionar problemas do dia-dia com o uso de sistemas computacionais. Tais sistemas podem analisar o ambiente por meio de sensores e com isso realizar deliberações sob o mesmo, com intuito de melhorar ou solucionar problemas cotidianos [FAPESP 2018].

Além disso, veículos autônomos podem ser empregados nesses ambientes, como carros, caminhões, drones e motos. Ao adotar tal prática, de modo parcial ou total pode-se retirar a responsabilidade humana da atividade, com isso reduzir o número de acidentes de trânsito [de Sousa Pissardini et al. 2013]. Dessa forma, ao utilizar veículos autônomos, como agentes atuadores em uma cidade inteligente seria um exemplo de sistema multiagentes (SMA).

Para programar esse tipo de sistema em que o objetivo é atuar em um ambiente colaborativo ou competitivo é necessário desenvolver agentes com noções mentais de crenças, desejos, intenções e objetivos [Wooldridge 2001]. Entretanto a coordenação de um SMA tende a ser uma tarefa complexa, para isso é necessário desenvolver algoritmos de coordenação. Ao desenvolver tal sistema é possível reduzir problemas de mobilidade urbana, sustentabilidade e habitabilidade [Taniguchi 2012].

No presente trabalho, os agentes terão responsabilidade de atuar em uma cadeia logística, realizando tarefas com uma abordagem distribuída na busca, montagem, entrega de itens, sem desprezar o gerenciamento da bateria. Para isso foi utilizado o cenário de 2018 da *Multi-Agent Programming Contest* [1] (seção 2). O objetivo principal

[1]*Multi-Agent Programming Contest - https://multiagentcontest.org/*

da competição é que equipes desenvolvam algoritmos de coordenação para SMA, com o intuito atuar em uma cidade inteligente.

A solução apresentada nesse artigo é uma variação da solução apresentada pela equipe Akuanduba-UDESC na competição, ao qual a equipe adotou uma estratégia centralizada no gerenciamento do estoque. Suspeita-se que essa abordagem é ineficiente, tendo em vista que os itens ficarão em somente um estoque, reduzindo assim a eficiência no atendimento de demandas de trabalho. Esse artigo propõe uma abordagem distribuída dos estoques para melhorar o acesso aos itens e consequentemente melhorar a eficiência no atendimento dos trabalhos. As seções 3 e 4 apresentam a estrutura geral e as estratégias adotadas.

2. Cenário da *Multi-Agent Programming Contest*

O ambiente utilizado para simular cidades inteligentes será o da competição anual *Multi-Agent Programming Contest* (MAPC) do ano de 2018. O cenário se passa no ano de 2045 dC após a colonização de Marte. O mapa é retirado do *OpenStreetMap* contém instalações como: estações de recarga, lixões, lojas, depósitos, *workshops*, nós de recurso e poços de água. Todas as instalações, detalhes, rotas e tempo são abstraídos e fornecidos pelo servidor da competição [Ahlbrecht 2018]. A unidade de medida de tempo do ambiente é medida em *steps*.

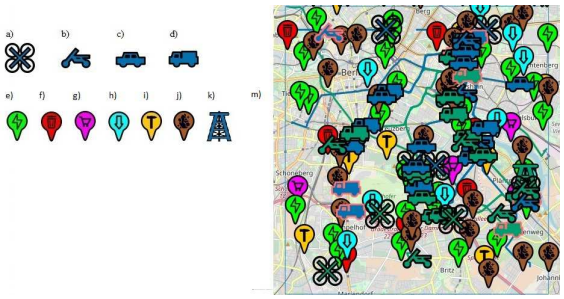


Figura 1. Cenário da competição

Nas estações de recarga é possível o agente recarregar sua bateria. Nos lixões, o agente poderá descartar itens que não seja mais do seu interesse. Nas lojas é possível o agente realizar melhorias em suas características e/ou adquirir itens. Nos depósitos, os agentes poderão guardar os itens obtidos nas lojas e/ou nós de recursos. Enquanto no *workshop* é possível confeccionar itens secundários compostos de itens primários. Em nós de recursos é possível obter itens do tipo primário. Cada equipe pode construir poços de água com objetivo de ganhar pontos na partida.

Os agentes que atuam sob o ambiente necessitam ser coordenados com objetivo de cumprir tarefas como a construção de poços, recarga, confecção de itens e a realização de demandas de trabalho. Eles podem ser do tipo carro, caminhão, drone e moto. Cada um deles contém características particulares como bateria, capacidade de carga, visão, velocidade e habilidade. Além disso, a partida conta com recursos randômicos no tipo de itens, trabalhos, volumes e recompensas [Ahlbrecht 2018].

3. Arquitetura Geral

Para programação do SMA foi utilizado o *framework* JaCaMo, sendo ele composto pelas ferramentas *Jason, CArtAgO* e *Moise* [Boissier et al. 2013]. Dos quais foram utilizado apenas o *Jason* e o *CArtAgO*. O *Jason* implementa o modelo BDI - *Belief, Desires and Intention*, dessa forma é possível desenvolver os agentes com crenças, desejos e intenções [Rao and Georgeff 1991]. Para gerenciar as diferentes tarefas dos agentes foi desenvolvido uma fila de prioridades por meio de crenças, desejos e planos. O desejo *addtask* responsável por adicionar uma nova tarefa ao agente. Os parâmetros do addtask são: o rótulo da tarefa a ser cumprida, a prioridade, a lista de passos para atingir o objetivo e uma lista com o histórico dos passos executados.

O rótulo das tarefas que podem ser desempenhadas pelos agentes ao longo da partida, bem como a prioridade são descritos na tabela 1. Sendo a tarefa de recarga a mais prioritária de todas, com objetivo de garantir a autonomia do agentes em toda partida. Quando existir uma tarefa mais prioritária que a tarefa em execução, haverá uma preempção de tarefas, caso isso ocorra o agente, a tarefa será retomada que não houverem tarefas de maior prioridade.

Tabela 1. Tarefas e Prioridades

Tarefa	Prioridade	Tarefa	Prioridade
Recarga	10	Ajudar outro agente	8,2
Missões	9	Montagem de item	8
Exploração	9	Obter item	8
Devolver itens	8,9	Realizar Trabalho	5
Atualizar Capacidade	8,5	Obter itens primários	4

O CArtAgO foi utilizado para implementar o ambiente, encapsular serviços, funcionalidades e prever situações em tempo de execução [Boissier et al. 2013]. Para estabelecer e manter a conexão ao servidor da competição *MASSim*, foi utilizado o *EISMASSim* do padrão *Environment Interface Standard - EIS*. Por meio dessa solução é possível mapear as chamadas de métodos Java e assim realizar troca de mensagens em XML [Ahlbrecht 2018].

A estrutura geral é representada pela figura 2. O artefato *EISAccess*, filtra todas as propriedades e percepções recebidas do servidor da competição (*EISMASSim*). A coordenação é feita pelo artefato *CoordinationArtifact*, ele delega responsabilidades aos agentes ao longo da partida, como gerenciamento de estoques e assegura que determinada tarefa não tenha mais de um agente comprometido em sua realização. Além disso, os agentes poderão se comunicar diretamente com outros agentes, por meio do sistema de mensagens do *Jason*. Sendo assim, exercer a comunicação e o gerenciamento é fundamental na descentralização de tarefas [Christie et al. 2003].

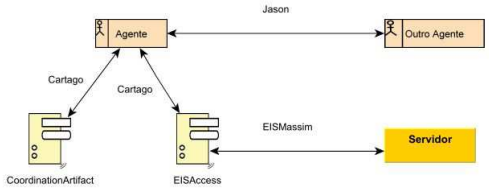


Figura 2. Estrutura Geral

4. Estratégias Adotadas

As estratégias apresentadas nesse artigo são uma variação da solução desenvolvida pela equipe Akuanduba-UDESC, participante da competição MAPC de 2018. No inicio da simulação os agentes exploram o ambiente com o objetivo de descobrir instalações ao longo do mapa. A exploração é realizada com uso de *drones* e o mapa é dividido em quatro partes, para isso, eles informam sua latitude e longitude para o artefato de coordenação.

Que por sua vez, soluciona um problema de programação linear binária, com objetivo de minimizar a distância a ser percorrida pelos agentes [Longaray and Beuren 2001]. Ao longo da exploração, conforme os agentes descobrem as instalações, os agentes do tipo carro, moto e caminhão são alocados por meio de troca de mensagens para buscar o máximo de itens nos nós de recursos e leva-los até os estoques. Na solução apresentada pela equipe Akuanduba-UDESC, a estratégia era centralizar todos os itens em um estoque central, nesse artigo propomos a distribuição deles.

A distribuição dos estoques é feita com base em um fecho convexo cujos vértices são as coordenadas dos estoques, que interligados formam uma estrutura que incorpora todas as outras localizações. Para que não haja recálculo, somente um agente informa ao artefato as posições dos estoques e por sua vez ele retorna a melhor distribuição deles. Com isso o agente informa aos outros agentes os estoques selecionados (figura 3).

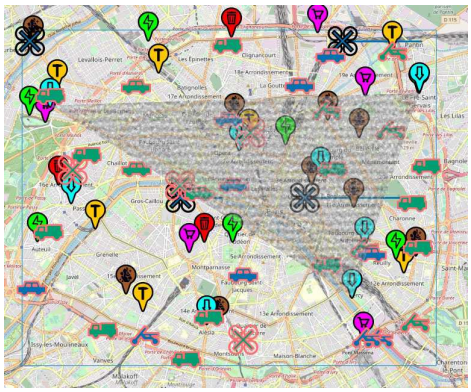


Figura 3. Fecho convexo de estoques

O algoritmo para seleção de estoques (algoritmo 1), pode ser divido em três partes, o cálculo do fecho convexo, cálculo do ponto médio e seleção dos estoques por abrangência.

Algoritmo 1: Seleção de estoques

Entrada: Estoques do Mapa
Saída: Estoques Selecionados
início
    PONTOS = Cálculo do Fecho Convexo(estoques);
    M = Cálculo do Ponto Médio(pontos);
    ES = Seleção de Estoques Por Abrangência(pontos, m);
fim

4.1. Cálculo do Fecho Convexo

Com objetivo de calcular o fecho convexo, uma solução inicial é selecionar o estoques mais ao norte, sul, leste e oeste. Sendo assim, o ponto mais ao norte, é aquele

estoque ao qual possui a maior latitude. A mesma seleção ocorre para os pontos mais ao leste, mais ao sul e mais ao oeste respectivamente. Após obter aos quatro pontos mais extremos, eles são interligados por segmentos de retas de modo a obter o fecho.

O cálculo do ponto médio são considerados os pontos norte, sul, leste e oeste selecionados no passo anterior. Desses pontos é calculado a média das latitudes e longitudes para obter um ponto central do fecho.

Para cada estoque que não faz parte do fecho deverá ser feito uma comparação com cada uma das retas do fecho atual, para verificar se o estoque está no lado interno ou externo do fecho. Para isso são calculadas duas determinantes: *d1* - posição do estoque em análise em relação aresta e *d2* - ponto médio em relação a aresta.

Caso para uma mesma aresta do fecho, os determinantes *d1* e *d2* tenham sinais opostos, significa que o estoque analisado está fora do fecho convexo. Com isso, ocorre um relaxamento da solução e o ponto passa a fazer parte do fecho. Após analisar todos os pontos e não houverem mais possibilidades de relaxamento o menor fecho convexo que abrange todos os estoque é encontrado.

4.2. Seleção de Estoques Por Abrangência

Ao obter a lista de estoques selecionados, podem haver estoques próximos um ao outro. Com objetivo de reduzir a sobreposição é realizado uma rotina que busca eliminar pontos redundantes do fecho. Dessa forma é calculado o ponto médio m e, caso um estoque p possua um raio de abrangência r em relação a m que não alcance algumas das extremidades do mapa, p é desconsiderado e dado como um estoque pouco eficiente.

4.3. Comportamento do Agente

Ao surgir uma demanda de trabalho o agente (independente do tipo) irá analisar qual a sua latitude e longitude atual e consultar a crença do fecho calculado. Ao obter o ponto do fecho, mais próximo dado a sua localização atual ele irá se locomover até o estoque para buscar ou entregar itens, conforme apresenta o algoritmo 2.

Algoritmo 2: Qual estoque utilizar

```
Entrada: Lat, Lon
Saída: Estoque Selecionado
início
    Pontos = Obter pontos do fecho;
    Estoque Selecionado = Obter ponto mais próximo(Lat,Lon,Pontos);
fim
```

4.4. Testes Preliminares

Com o objetivo de validar as estratégias aqui apresentadas foi proposto um cenário de teste contra a equipe Akuanduba-UDESC. Os cenários propostos foram as cidades de Berlim, Copenhague e Paris. Os testes preliminares foram executadas em sementes aleatórias, em partidas com mil *steps* e com trinta e quatro agentes em cada equipe, entre eles quatro drones, dez carros, doze caminhões e oito motos. Os critérios para analise de desempenho foram quem arrecadou mais *massium* (moeda da partida) e realizou mais entregas de trabalhos.

O *massium*, é obtido como recompensa de trabalhos entregues pelas equipes. A probabilidade de surgir uma demanda de trabalho em dado *step* da partida é de 20%, desse modo, em média uma partida de mil *steps* surgem duzentas demandas de trabalho. Porém dado a complexidade dos mesmos, a proporção dos que surgem e dos que são atendidos, ficam em torno de 4%. Sendo assim, ao executar a solução proposta nesse artigo contra a solução da equipe Akuanduba-UDESC, foi observado que a solução distribuída tem desempenho melhor que a solução centralizada, conforme a tabela 2. Sendo a equipe *A* o desempenho da solução apresentada nesse artigo e a equipe *B* a solução da equipe Akuanduba-UDESC.

Tabela 2. Resultados preliminares

Cenário	Dinheiro		Quantidade Trabalhos da Partida	Trabalhos Atendido (%)	
	A	B		A	B
Paris	458	257	195	1,03	0
Copenhague	1068	860	189	2,11	1,59
Berlim	2036	2757	203	4,43	4,43

5. Considerações Finais

Nesse artigo pode-se apresentar como os agentes podem ser coordenados com abordagem distribuída dentro de uma cidade inteligente, com uma arquitetura modular e com o uso de uma fila de prioridades. A arquitetura pode ser adaptada e utilizada para problemas recorrentes do mundo real, pois, ela proporciona criar novas tarefas e novas funcionalidades sem muitas linhas de código. Ao propor uma abordagem distribuída no gerenciamento de estoque, para melhorar a eficiência no atendimento de demandas de trabalho, obteve resultados preliminares satisfatórios. De modo que a solução atual apresentou resultados melhores, que os obtidos anteriormente pela equipe Akuanduba-UDESC na competição MAPC - 2018.

Referências

Ahlbrecht, T. (2018). Simulation platform for the multi-agent programming contest.

Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., and Santi, A. (2013). Multi-agent oriented programming with jacamo. *Science of Computer Programming*, 78(6):747–761.

Christie, A. A., Joye, M. P., and Watts, R. L. (2003). Decentralization of the firm: theory and evidence. *Journal of Corporate Finance*, 9(1):3–13.

de Sousa Pissardini, R., Wei, D. C. M., and da Fonseca Júnior, E. S. (2013). Veículos autônomos: conceitos, histórico e estado-da-arte. In *Anais do XXVII Congresso de Pesquisa e Ensino em Transportes–ANPET*, page 2.

FAPESP (2018). Chamada em cidades inteligentes tem resultado de etapa de enquadramento.

Longaray, A. A. and Beuren, I. M. (2001). Cálculo de minimização dos custos de produção por meio da programação linear.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a bdi-architecture. *KR*, 91:473–484.

Taniguchi, E. (2012). The future of city logistics.

Wooldridge, M. (2001). Intelligent agents: The key concepts. In *ECCAI Advanced Course on Artificial Intelligence*, pages 3–43. Springer.

Representação de Objetos do Código de Trânsito Através de Uma Ontologia Para Aplicação em um Veículo Autônomo

Vithor Tozetto Ferreira[1], **Gleifer Vaz Alves**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Ponta Grossa– PR – Brazil

Abstract. *There are many aspects about the development of an autonomous vehicles (AV), among them the interaction of those vehicles with traffic laws. An AV, acting in traffic, will need to follow a set of rules, and its controller will need to be able to utilize these rules in the planning of its actions. The controller of an AV could be modelled as an intelligent agent, still being necessary a representation of the traffic rules, representation which could be done through an ontology. In this paper is presented the proposal of an ontology that will be utilized by an intelligent agent to ensure that the behaviour of the agent (AV) will be in agreement with a subset from the urban traffic rules.*

Resumo. *Existem vários aspectos sobre o desenvolvimento de um veículos autônomos (VA), entre eles a interação desses veículos com as leis de trânsito. Um VA, agindo no trânsito, deverá seguir um conjunto de regras, e seu controlador deverá ser capaz de utilizar essas regras no planejamento de suas ações. O controlador de um VA pode ser modelado como um agente inteligente, sendo ainda necessária uma representação das regras de trânsito, representação que pode ser feita através de uma ontologia. Neste trabalho é apresentada a proposta de uma ontologia que será utilizada por um agente inteligente para assegurar que o comportamento do agente (VA) se dará em acordo com um fragmento das regras de trânsito urbano.*

1. Introdução

Nos últimos anos, ocorreu um crescimento no desenvolvimento de novas tecnologias automatizadas. Para a indústria automotiva, o lançamento de veículos autônomos (VA's) será um grande salto tecnológico. Um veículo cria muitos custos na vida urbana, desde o custo financeiro do valor de compra e manutenção do veículo, até o custo de tempo referente aos engarrafamentos das grandes cidades. Com o advento dos VA's, o motorista não precisaria controlar um veículo, e poderia utilizar o tempo de locomoção para realizar outras tarefas [Silberg et al. 2012].

Um dos grandes problemas que ainda impedem a utilização cotidiana de um VA é o de garantir a segurança do funcionamento desses veículos, e neste artigo, focando especificamente na segurança em relação ao comportamento do VA em acordo com leis de trânsito. Através de um conjunto de tecnologias para captura e interpretação de dados, os VA's atuais conseguem perceber e interagir com segurança e autonomia a certos obstáculos em seus ambientes [Gomes 2014]. Entretanto, para garantir realmente a segurança do funcionamento de um VA é importante que o veículo possua o conhecimento das regras de trânsito do local onde está trafegando, e que estas regras influenciem em seu comportamento [Vellinga 2017].

Conforme mencionado por Prakken [Prakken 2017], a interação de um VA com as regras de trânsito em geral não é tratada nas etapas de desenvolvimento de um VA, mesmo sabendo que nos ambientes comuns de trânsito o VA interagirá com outros veículos, e poderá surpreender um motorista ao tomar uma ação que não condiz com o código de trânsito em vigor onde está trafegando. O VA precisa ser capaz de perceber seu ambiente, considerar suas tarefas e planejar ações seguras para atingir seus objetivos de forma autônoma, e dessa forma, deve ser controlado por um sistema computacional que consiga realizar esta tarefa, e tal sistema pode ser representado por um agente.

De maneira geral, diz-se que um agente é uma entidade situada em certo ambiente, e que é capaz de realizar ações autônomas para atingir seus objetivos [Wooldridge and Jennings 1995]. A um agente inteligente, são também atribuídas as propriedades de autonomia, habilidade social, reatividade e proatividade. Tais propriedades são interessantes para um VA, já que são úteis no contexto de situações reais de trânsito. Mesmo com a utilização de um agente, uma grande dificuldade do desenvolvimento de um VA capaz de agir de forma segura no trânsito ainda permanece: como fazer com que um sistema computacional autônomo utilize as regras de trânsito em seu planejamento de ações.

A adaptação de um conjunto de regras de trânsito para o contexto de um mecanismo autônomo e inteligente é afetada pelo fato das leis estarem descritas em linguagem natural, com a eventual presença de ambiguidades, redundâncias e incoerências, e também por casos onde o comportamento no trânsito é puramente social, condicionado a partir do senso comum [Prakken 2017]. Se faz necessário utilizar uma representação destas leis, que poderá ser utilizada no controle do VA, e tal representação pode ser feita com o auxílio de uma ontologia. Uma ontologia é uma especificação de uma conceitualização, um modelo de dados que possui um domínio, elementos e a relação entre esses elementos e o domínio [Cimiano et al. 2014]. Com uma ontologia, é possível criar uma conexão entre o domínio da linguagem natural e da linguagem artificial, auxiliando no processo de interpretação do ambiente e das leis por parte do VA.

O objetivo específico deste trabalho é criar uma representação dos objetos e leis do Código de Trânsito Brasileiro [BRASIL 1997], e então incorporar esta representação em um agente inteligente modelado como um VA. Este trabalho está diretamente relacionado com o trabalho desenvolvido por Alves et. al [Alves et al. 2018], que tem como objetivo representar as regras de trânsito do Reino Unido por meio de uma formalização (usando operadores LTL), e incorporar tais regras em um agente inteligente, o qual representa um VA. Neste artigo, é apresentada uma representação de objetos através de uma ontologia para descrever elementos específicos de um subconjunto das regras do código de trânsito brasileiro. Além disso, é discutido como tal ontologia poderia ser encapsulada em um agente no controle de um VA, e também como seria possível explorar a verificação formal do comportamento do agente em relação as regras da ontologia.

2. Veículos Autônomos e código de trânsito

Na literatura da área, é muito comum encontrar em publicações de montadoras de veículos o termo direção automatizada, e raramente o termo direção autônoma. O primeiro termo envolve um conjunto de ferramentas que auxiliam na direção, enquanto o segundo se refere ao estado final da automação, onde o sistema teria controle total sobre todas as

funções de controle do veículo [Herrmann et al. 2018]. Em resumo, um veículo autônomo (VA) é um veículo controlado completamente por um sistema, sem o auxílio de um motorista.

Para a autonomia total é importante que, além desses veículos compreenderem e interpretarem seus ambientes de atuação, os VA's tenham a capacidade de atuarem corretamente em seus ambientes. No controle de um veículo, uma ação não pode ser definida somente pela habilidade de compreender obstáculos, sinalizações e usuários das vias terrestres. O VA precisará considerar o que precisa ser feito em determinada situação, e então planejar como realizar determinada ação de forma segura. Para garantir a segurança de suas ações no quesito de tráfego urbano, é necessário que o VA leve em consideração as regras referentes ao seu ambiente, ou seja, um VA situado no trânsito deve utilizar as Regras de Trânsito do local onde está situado para trafegar em seu ambiente.

Conforme mencionado por Prakken [Prakken 2017] e reforçado por Alves et. al [Alves et al. 2018], existe uma lacuna no desenvolvimento de VA's no que diz respeito aos seguintes aspectos: i.) A implantação de um VA considera adequadamente as regras de trânsito? ii.) O comportamento de um VA no tráfego urbano dá-se em acordo com as regras de trânsito? iii.) É necessário em alguma instância alterar e adaptar as regras de trânsito para o adequado comportamento de um VA? Esses aspectos reforçam a necessidade de representar o conhecimento das regras de trânsito para que sejam utilizadas por um agente representando um VA.

2.1. Código de Trânsito Brasileiro

De acordo com o Código de Trânsito Brasileiro [BRASIL 1997], se configura como trânsito a movimentação e imobilização de veículos, pessoas e animais nas vias terrestres. Através do trânsito, as pessoas se movimentam pelas vias, urbanas e rurais, todos os dias. Cada país, dentro de seus territórios, delimitam um conjunto de regras destinadas a controlar o trânsito, e garantir a eficiência e segurança do tráfego.

Essas regras englobam todos os aspectos do trânsito, como o comportamento esperado dos usuários das vias terrestres, a infraestrutura das vias, as sinalizações de trânsito e as punições dadas aos infratores. É esperado que todo cidadão presente no trânsito esteja ciente destas regras, e que as cumpra visando manter a harmonia do trânsito. Os VA's, irão circular no ambiente de trânsito, e precisarão se adequar as regras de circulação dos locais onde trafegam. Para o desenvolvimento deste trabalho, é considerado um fragmento (conjunto de 7 artigos) das regras de cruzamento em vias urbanas do Código de Trânsito Brasileiro. Dentre as regras selecionadas, aqui destaca-se a seguinte:

> Art. 44. Ao aproximar-se de qualquer tipo de cruzamento, o condutor do veículo deve demonstrar prudência especial, transitando em velocidade moderada, de forma que possa deter seu veículo com segurança para dar passagem a pedestre e a veículos que tenham o direito de preferência.

A partir do fragmento de regras do Código Brasileiro de Trânsito, foram extraídos os objetos presentes nestas regras, e estes objetos foram utilizados para a criação da ontologia demonstrada neste artigo. O objetivo desta ontologia é servir como uma base de conhecimento referente aos objetos do trânsito que poderá ser utilizada por um agente. Por exemplo, a partir do artigo 44, apresentado anteriormente, podem ser extraídos os objetos "cruzamento", "veículo" e "pedestre".

3. Definição dos Objetos de uma Ontologia para o Código de Trânsito Brasileiro

O termo ontologia pode ser definido como uma especificação de uma conceitualização. A definição de ontologia pode mudar de acordo com o autor, porém, é comum na grande maioria das definições o termo conceitualização, que se refere a uma visão de mundo, uma forma de descrever um domínio, seus objetos e as relações existentes entre tais objetos [Cimiano et al. 2014].

As ontologias possuem algumas propriedades essenciais: ontologias descrevem um domínio específico; a utilização dos termos deve ser consistente; os conceitos e relações devem ser definidos sem ambiguidades em uma linguagem formal; as relações entre os conceitos determinam a estrutura da ontologia; ontologias podem ser compreendidas e processadas por computadores [Freitas 2017]. A partir destes conceitos foi desenvolvida a *Road Junction Objects Ontology*, demonstrada na figura 1, que visa representar os objetos presentes em um ambiente de trânsito, extraídos a partir das regras do Código de Trânsito Brasileiro [BRASIL 1997].

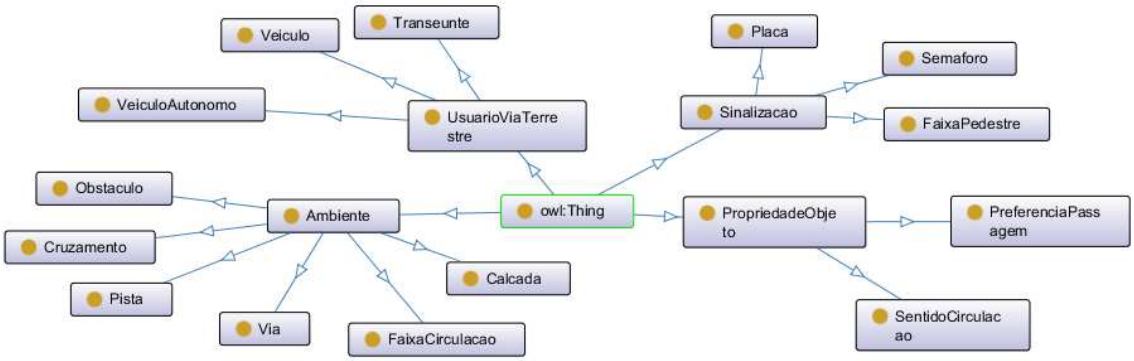


Figura 1. Road Junction Objects Ontology

Road Junction Objects Ontology: Ontologia que representa os objetos presentes no ambiente de trânsito de acordo com um fragmento de regras do Código de Trânsito Brasileiro.

UsuarioViaTerrestre: Os usuários presentes no ambiente.

Transeunte: Usuário das vias terrestres que não utiliza um veículo, e deve circular prioritariamente nas calçadas.

Veiculo: Um meio de transporte destinado à locomoção de passageiros ou cargas, com alguma forma de propulsão, controlado por um ser humano e que circula nas vias terrestres.

VeiculoAutonomo: Um meio de transporte destinado à locomoção de passageiros ou cargas, com alguma forma de propulsão, controlado por um sistema autônomo e inteligente, que pode eventualmente contar com a intervenção de um humano e que circula nas vias terrestres.

Ambiente: O local onde os usuários do trânsito estão presentes.

Via: Superfície por onde transitam todos os usuários das vias terrestres, compreendendo a pista, a calçada e outros espaços de circulação.

Cruzamento: Interseção entre duas vias.

FaixaCirculacao: As subdivisões longitudinais de uma pista que delimitam o espaço de circulação dos veículos.

Calcada: Parte da via reservada ao trânsito de pedestres. Obstaculo: Objeto que interfere na circulação do trânsito, como um pedestre atravessando a via.

Obstaculo: Objeto que interfere na circulação do trânsito, como um pedestre atravessando a via.

Pista: Parte da via destinada a circulação de veículos.

Sinalizacao: Objetos que ordenam ou dirigem como proceder no ambiente.

FaixaPedestre: Espaço destinado ao cruzamento da via por parte dos pedestres, demarcado por faixas no solo.

Placa: Objeto que informa os usuários sobre normas de circulação ou condições do ambiente.

Semaforo: Objeto de sinalização luminosa, que indica a prioridade de passagem no local.

PropriedadeObjeto: Características de um objeto que influenciam na interação do VA com o ambiente.

SentidoCirculacao: Propriedade que indica a direção que os usuários devem circular em uma via ou faixa de circulação.

PreferenciaPassagem: Propriedade referente a preferência de passagem em determinado ambiente.

Ao utilizar representações das regras, é possível desenvolver outras ontologias, similares a apresentada neste trabalho, que englobem as regras de trânsito de outros locais, como por exemplo, as regras de trânsito do Reino Unido. Desta forma, um VA (modelado como um agente inteligente) poderia ser capaz de se movimentar em diferentes jurisdições, alterando o conjunto de regras de trânsito que considera em seu planejamento de ações, visando a mobilidade da utilização desta tecnologia.

4. Considerações Finais

Este trabalho visa obter conclusões referentes a questões presentes na literatura de VA's, como aquelas levantadas por Prakken [Prakken 2017] e Vellinga [Vellinga 2017]: quais meios podem ser utilizados para a representação das regras de trânsito para o contexto de um VA; quais são as limitações de um VA no que se refere a compreensão e execução das regras de trânsito; e quais mudanças podem ser necessárias na legislação de trânsito para a adequação do uso de VA's.

Este artigo se propôs a demonstrar uma ontologia que representa os objetos presentes em um ambiente de trânsito, extraídos do Código de Trânsito Brasileiro, para o contexto de um agente modelado como um veículo autônomo. Como foi apresentado anteriormente, os veículos autônomos serão uma grande revolução tecnológica, mas ainda existem algumas barreiras que impedem a implantação dessa tecnologia, entre elas a dificuldade de garantir que um VA compreenda e obedeça à legislação de trânsito do local que trafega.

A ontologia apresentada neste trabalho ainda é uma fração do trabalho planejado, que visa desenvolver uma representação de conhecimento, que englobe objetos e regras de trânsito, e possa ser utilizada por um VA. A proposta de utilizar representações das regras de trânsito para adaptar estas leis a um contexto utilizável para um VA também é demonstrada em Alves et. al [Alves et al. 2018], onde um conjunto de regras de trânsito do Reino Unido é representado utilizando uma linguagem formal com operadores LTL

(*Linear Temporal Logic*), sendo ainda discutida a seguinte questão: como fazer para que um VA possa transitar entre diferentes países, de forma que possa facilmente reconhecer as diferentes legislações de trânsito em vigor nestes países?

Como trabalho futuro, pretende-se justamente agregar os resultados deste artigo e do trabalho supracitado, para assim expandir a Road Junction Objects Ontology. Nesta expansão a ontologia terá as regras de trânsito do Brasil e do Reino Unido. Além disso, a formalização das regras em LTL poderá beneficiar-se da representação dos objetos. Será então implementado um agente racional modelado como um VA, utilizando a linguagem de programação de agentes *Gwendolen* [Dennis 2017]. Após o desenvolvimento das representações de conhecimento e do agente, será possível realizar a verificação formal do comportamento do VA em relação às regras de trânsito, através da ferramenta AJPF (*Agent Java Pathfinder*) [Dennis et al. 2012].

Referências

Alves, G. V., Dennis, L., and Fisher, M. (2018). Formalisation of the rules of the road for embedding into an autonomous vehicle agent. *International Workshop on Verification and Validation of Autonomous Systems*.

BRASIL, D. (1997). Código brasileiro de trânsito. `http://www.planalto.gov.br/ccivil_03/LEIS/L9503.htm`.

Cimiano, P., Unger, C., and McCrae, J. (2014). Ontology-based interpretation of natural language. *Synthesis Lectures on Human Language Technologies*, 7(2):1–178.

Dennis, L. A. (2017). Gwendolen Semantics: 2017. Technical Report ULCS-17-001, University of Liverpool, Department of Computer Science.

Dennis, L. A., Fisher, M., Webster, M. P., and Bordini, R. H. (2012). Model checking agent programming languages. *Automated Software Engineering*, 19(1):5–63.

Freitas, A. L. S. d. C. (2017). Model-driven engineering of multi-agent systems based on ontology.

Gomes, L. (2014). Hidden obstacles for google's self-driving cars. *MIT Technology Review*.

Herrmann, A., Brenner, W., and Stadler, R. (2018). *Autonomous driving: how the driverless revolution will change the world*. Emerald Publishing, Bingley North America Japan India Malaysia China, first edition. OCLC: 1031123857.

Prakken, H. (2017). On the problem of making autonomous vehicles conform to traffic law. *Artificial Intelligence and Law*, 25(3):341–363.

Silberg, G., Wallace, R., Matuszak, G., Plessers, J., Brower, C., and Subramanian, D. (2012). Self-driving cars: The next revolution. *White paper, KPMG LLP & Center of Automotive Research*, page 36.

Vellinga, N. E. (2017). From the testing to the deployment of self-driving cars: legal challenges to policymakers on the road ahead. *Computer Law & Security Review*, 33(6):847–863.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152.

Classificação das Abordagens de Integração de Agentes com Aplicações Heterogêneas

Otávio A. Matoso[1], **Jomi F. Hübner**[1], **Maicon R. Zatelli**[1]

[1]Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil

otaviomatoso@yahoo.com.br, {jomi.hubner,maicon.zatelli}@ufsc.br

Resumo. *A complexidade dos sistemas atuais demanda soluções computacionais distribuídas e heterogêneas. Em alguns casos, a inclusão de Sistemas Multiagentes (SMA) na solução pode ser promissora. No entanto, a falta de abordagens para integrar agentes com uma grande variedade de aplicações ainda representa um obstáculo à exploração do potencial de SMA em um ambiente distribuído e heterogêneo. Este trabalho investiga o estado da arte relacionado à integração de agentes com aplicações heterogêneas e tem como principal objetivo elaborar uma classificação das abordagens estudadas.*

1. Introdução

Um sistema de informação é normalmente composto por inúmeras aplicações, que podem ser oriundas de desenvolvimento interno da própria empresa, adquiridas de terceiros, parte de sistemas legados ou uma combinação dessas opções [Hohpe and Woolf 2004]. Esse sistema deve ter suas aplicações trabalhando em conjunto, na direção de um mesmo objetivo, levando a um fluxo de informações mais dinâmico e fluído. Para que essas aplicações possam interagir entre si, trocando informações de forma eficiente e coordenada, é preciso que elas estejam bem integradas. Porém, um cenário comum é que essas aplicações sejam heterogêneas, ou seja, atuem em diferentes camadas de diferentes sistemas operacionais e possuam suas próprias estruturas de dados e protocolos de comunicação. Como consequência dessa diversidade, a interoperabilidade se mostra bastante prejudicada.

SMA é uma sub-área da inteligência artificial que facilita a resolução de problemas complexos por meio de uma estratégia onde o problema é decomposto em problemas menores e mais simples, que são tratados por entidades autônomas, os agentes. Um agente pode ser denotado como um sistema computacional baseado em software que possui como principais características autonomia, habilidade social, reatividade e proatividade [Wooldridge and Jennings 1995]. Devido às suas particulares características, SMA vem se consolidando como uma importante abordagem para o desenvolvimento de aplicações, se mostrando adequada para ambientes dinâmicos e abertos.

No entanto, a falta de abordagens para integrar agentes com uma grande variedade de aplicações ainda representa um obstáculo à exploração do potencial de SMA em um ambiente distribuído e heterogêneo. O presente trabalho, que é parte de uma pesquisa em andamento, investiga o estado da arte no que se refere à integração de SMA com entidades heterogêneas e apresenta uma classificação, na forma de tabela, das abordagens estudadas, exibindo as principais características encontradas em cada trabalho.

2. Trabalhos relacionados

A revisão bibliográfica apresentada nesta seção foi realizada com o intuito de investigar abordagens que geram algum nível de interoperabilidade entre agentes e entidades heterogêneas.

Alguns autores abordaram a integração entre SMA e *Serviços Web* (SW). Contreras e Sheremetov [Contreras and Sheremetov 2008] apresentaram um conjunto de extensões que devem ser feitas em uma plataforma de agentes compatível com as especificações FIPA para gerar interoperabilidade entre agentes e SW, permitindo a implementação de aplicações orientadas a serviços usando agentes e SW já existentes. Seguindo a mesma linha, Coria et al. [Coria et al. 2014] propõem um modelo para a composição dinâmica de SW em um ambiente de computação em nuvem. Neste modelo, um SMA analisa a semântica de um conjunto de especificações dadas pelo usuário, realiza descoberta de SW na nuvem a partir da semântica obtida e compõe o sistema usando os SW disponíveis. Em [Tapia et al. 2009] é descrita uma arquitetura multiagentes onde serviços são gerenciados e controlados por agentes deliberativos. Nessa arquitetura, as funcionalidades dos sistemas são modeladas como serviços distribuídos que podem ser invocados pelos agentes.

A integração de SMA com *sistemas legados* também foi investigada. Estratégias para encapsular tais sistemas de modo que eles possam ser tratados como agentes em um SMA são apresentadas em [Li 2010, Zhao et al. 2008]. Li propõe uma estratégia de integração de dados baseada em SOAP/XML para que sistemas legados sejam encapsulados como agentes e possam cooperar entre si. Já Zhao et al. apresentam um mecanismo de encapsulamento baseado no sistema operacional Windows para gerar interface entre sistemas legados e SMA.

Outros trabalhos focaram na integração de agentes com padrões que geram certo nível de interoperabilidade em domínios específicos. Em [Saleem et al. 2010], as funções de entidades compatíveis com o padrão *IEC 61850* de automação de subestação são implementadas diretamente nos agentes, permitindo que um SMA seja modelado para controle e proteção de sistemas de energia elétrica. Já Miranda et al. [Miranda et al. 2012] utilizam SMA para facilitar a troca de informações entre sistemas e equipamentos de saúde. Na arquitetura proposta, entidades que seguem as normas *HL7*[1] podem ser implementadas sob o paradigma de agentes, facilitando a troca de mensagens entre as entidades.

A integração de agentes com uma variedade de recursos e serviços externos foi investigada em outras pesquisas. Em [Vrba et al. 2014], os autores apresentaram um framework para encapsular um SMA como um serviço a ser usado em um *Enterprise Service Bus* (ESB). Um *gateway* é criado para traduzir mensagens FIPA-ACL em mensagens ESB e vice-versa, permitindo comunicação entre agentes e demais entidades presentes no ESB. Cranefield e Ranathunga [Cranefield and Ranathunga 2013] criaram uma interface entre agentes e o Apache Camel, um framework de integração baseado nos *Padrões de Integração Corporativa* (EIPs) apresentados por Hohpe e Woolf [Hohpe and Woolf 2004]. Com isso, tais agentes podem trocar mensagens com outras entidades que possuem interface com esse framework.

[1]Padrão utilizado no setor da saúde para a troca de informações entre aplicações e equipamentos médicos.

3. Classificação dos trabalhos

Considerando a visão geral de cada trabalho, foi possível identificar algumas propriedades em comum no que se refere à integração, tornando possível uma certa aproximação entre as propostas. Foram consideradas as seguintes propriedades:

- *Entidades integradas*: o tipo da entidade que foi integrada com os agentes, sendo SW e sistema legado exemplos;
- *Arquitetura dos agentes*: a arquitetura dos agentes utilizada no respectivo trabalho, por exemplo arquitetura BDI;
- *Padrão de comunicação*: o protocolo adotado para comunicação entre agentes e demais entidades;
- *Descrição das entidades*: forma de acesso dos agentes às funcionalidades das entidades que estão sendo integradas;
- *Registro das entidades*: o local onde são registradas as entidades e/ou suas funcionalidades integradas;
- *Encapsulamento das entidades pelo SMA*: indica se a entidade ou funcionalidade foi encapsulada ou não pelo SMA.

A Figura 1 compara as propostas analisadas levando em consideração as propriedades citadas anteriormente. Em alguns trabalhos, não foi possível identificar determinadas propriedades. Em tal situação, utilizou-se o caractere (-) para indicar tal indefinição.

Trabalhos	Entidades integradas	Arquitetura dos agentes	Padrão de comunicação	Descrição das entidades	Registro das entidades	Encapsulamento das entidades pelo SMA
Contreras e Sheremetov 2008	Serviço Web	*Behavioral*	SOAP	WSDL	*Directory Facilitator*	☑
Coria et al. 2014	Serviço Web	-	SOAP	WSDL	Nuvem	☐
Tapia et al. 2009	Serviços local, Web ou *Stand Alone*	BDI	SOAP	-	Diretório de Serviços	☑
Vrba et al. 2014	ESB JBoss	*Behavioral*	FIPA-ACL	WSDL	Diretório de Serviços	☐
Cranefield e Ranathunga 2013	Componentes Camel	BDI	EIPs	-	URI	☐
Xiangyu Li 2010	Sistema legado	-	SOAP	XML	agente	☑
Zhao et al. 2008	Sistema legado	BDIAIP (BDI Agent Initiative Perception)	COM	DLL/EXE - Componentes COM	*Capability Database*	☑
Saleem et al. 2010	Sistemas de automação de subestação	BDI	IEC 61850	Plano de Controle	Papeis	☑
Miranda et al. 2012	Sistemas de informação em Saúde	*Behavioral*	HL-7	Ontologia	*Directory Facilitator*	☑

Figura 1. Comparação entre trabalhos de integração

Com base na revisão do estado da arte e na classificação apresentada, nota-se que grande parte dos trabalhos apresentam soluções mais específicas, integrando agentes com um conjunto de entidades específicas. Contreras e Sheremetov, Coria et al. e Tapia et al. investigaram a interoperabilidade entre agentes e serviços, majoritariamente SW. Miranda

et al. integraram agentes com equipamentos e sistemas de informação que seguem o HL7, um conjunto de normas internacionais na área da saúde. Seguindo a mesma ideia, Saleem et al. discutiram um mapeamento entre uma arquitetura baseada em agentes e o padrão IEC 61850, adotado em projetos de automação de subestação. Já Zhao et al. e Li utilizaram uma estratégia de encapsulamento para que conjuntos específicos de sistemas legados sejam tratados como agentes e compartilhem informações entre si.

Em caso de um cenário onde um SMA necessite trocar informações com uma variedade de entidades heterogêneas, nenhuma das soluções citadas no parágrafo anterior seriam completamente adequadas. Tendo isso em vista, as pesquisas de Cranefield e Ranathunga e Vrba et al. merecem destaque, uma vez que a estratégia adotada nesses trabalhos permite que os agentes troquem mensagens com uma diversidade de aplicações heterogêneas de forma simultânea. Cranefield e Ranathunga desenvolveram uma interface entre uma plataforma de agentes e o framework de integração Apache Camel, de modo a gerar interoperabilidade entre os agentes e qualquer entidade que também tenha interface com o Camel. Enquanto que Vrba et al. encapsularam um SMA em um ESB, permitindo que os agentes troquem mensagens com qualquer outra entidade presente no ESB.

4. Considerações finais

Este trabalho apresentou uma revisão do estado da arte no que se refere à integração de agentes com uma variedade de entidades heterogêneas, com algumas propriedades sendo analisadas. Verificou-se que grande parte das soluções são direcionadas à integração de agentes com conjuntos específicos de entidades. Nessa ótica, duas pesquisas se destacaram por abrir a possibilidade de integração de agentes com uma gama de tecnologias, em especial a pesquisa de Cranefield e Ranathunga. A escolha desses autores por desenvolver uma interface entre agentes e o Apache Camel se mostrou interessante, pois o Camel é um projeto de código-fonte aberto bastante maduro e fornece conectividade com uma grande variedade de transportes e APIs. Por exemplo, ao integrar agentes com o Camel é possível que esses agentes recebam mensagens de uma fila JMS, realizem requisições HTTP ou consultas SQL.

Uma primeira proposta de trabalho futuro é considerar a dimensão do ambiente como uma abstração de primeira classe em um SMA, como proposto em [Weyns et al. 2007], e investigar seu papel na integração de SMA com entidades heterogêneas, uma vez que este trabalho considerou apenas a dimensão dos agentes. Em [Omicini et al. 2008], ao considerar a integração de SMA com aplicações externas, algumas dessas aplicações podem ser consideradas autônomas, sendo modeladas como agentes, enquanto que outras podem ser não-autônomas e são melhor modeladas como artefatos do ambiente. Seguindo essa ideia e tendo o trabalho de Cranefield e Ranathunga como inspiração, pretende-se propor uma interface entre o Apache Camel e a dimensão do ambiente, acrescentando aos agentes a capacidade de interagir com entidades externas representadas na abstração de artefato do ambiente.

Referências

Contreras, M. and Sheremetov, L. (2008). Industrial application integration using the unification approach to agent-enabled semantic soa. *Robotics and Computer-Integrated Manufacturing*, 24(5):680–695.

Coria, J. A. G., Castellanos-Garzón, J. A., and Corchado, J. M. (2014). Intelligent business processes composition based on multi-agent systems. *Expert Systems with Applications*, 41(4):1189–1205.

Cranefield, S. and Ranathunga, S. (2013). Embedding agents in business processes using enterprise integration patterns. In *International Workshop on Engineering Multi-Agent Systems*, pages 97–116. Springer.

Hohpe, G. and Woolf, B. (2004). *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional.

Li, X. (2010). A multi-agent based legacy information system integration strategy. In *2010 International Conference on Networking and Digital Society*, volume 2, pages 72–75. IEEE.

Miranda, M., Salazar, M., Portela, F., Santos, M., Abelha, A., Neves, J., and Machado, J. (2012). Multi-agent systems for hl7 interoperability services. *Procedia Technology*, 5:725–733.

Omicini, A., Ricci, A., and Viroli, M. (2008). Artifacts in the a&a meta-model for multi-agent systems. *Autonomous agents and multi-agent systems*, 17(3):432–456.

Saleem, A., Honeth, N., and Nordström, L. (2010). A case study of multi-agent interoperability in iec 61850 environments. In *2010 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT Europe)*, pages 1–8. IEEE.

Tapia, D. I., Rodríguez, S., Bajo, J., and Corchado, J. M. (2009). Fusion@, a soa-based multi-agent architecture. In *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)*, pages 99–107. Springer.

Vrba, P., Fuksa, M., and Klíma, M. (2014). Jade-jbossesb gateway: Integration of multi-agent system with enterprise service bus. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3663–3668. IEEE.

Weyns, D., Omicini, A., and Odell, J. (2007). Environment as a first class abstraction in multiagent systems. *Autonomous agents and multi-agent systems*, 14(1):5–30.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The knowledge engineering review*, 10:115–152.

Zhao, C., Li, Q., Wang, M., Wang, Y., and Li, Y. (2008). An agent based wrapper mechanism used in system integration. In *2008 IEEE International Conference on e-Business Engineering*, pages 637–640. IEEE.

Uma comparação entre soluções de smart parkings baseados em agentes inteligentes

Alexandre Lizieri Leite Mellado, Gleifer Vaz Alves, André Pinz Borges

[1]Departamento Acadêmico de Informática
(UTFPR) Universidade Federal Tecnológica do Paraná

mellado@alunos.utfpr.edu.br, {gleifer, apborges}@utfpr.edu.br

Abstract. *The traffic of vehicles in urban areas increases when drivers are searching for a parking space. Intelligent parking lots help in this search by optimizing and managing the available vacancies with the help of multi-agent systems (MASs). In this paper we analyze the characteristics of MASs in the field of Smart Parkings, such as: action policies, MAS hierarchy and communication between agents. The objective is to compare these characteristics to identify similarities and differences between the developed MAS. The first results help us to identify some features related to the development of smart parking solutions, which will be useful in the forthcoming stages of our work.*

Resumo. *O trafego de veículos em áreas urbanas aumenta quando motoristas trafegam em busca de uma vaga de estacionamento. Estacionamentos inteligentes auxiliam nesta busca otimizando e gerenciando as vagas disponíveis com auxílio de Sistemas Multi-Agentes (SMAs). Neste artigo são analisadas características de SMAs no domínio de Smart Parkings, tais como: politicas de ações, hierarquia do SMA e comunicação entre agentes. O objetivo é comparar tais características para identificar semelhanças e diferenças entre os SMAs desenvolvidos. Os resultados preliminares ajudam a identificar algumas características no desenvolvimento de soluções para smart parkings, as quais serão úteis nas proximas etapas deste trabalho.*

1. Introdução

Um dos problemas mais desafiadores a serem resolvidos é o do estacionamento em áreas urbanas. Estudos [Polycarpou et al. 2013] mostram que motoristas, ao procurarem por um espaço de estacionamento, desperdiçam tempo e combustível, aumentando o congestionamento e a poluição do ar. Nem sempre é possível resolver o problema criando mais vagas de estacionamento, o necessário é o desenvolvimento de instalações de estacionamento inteligentes.

Smart City (do inglês, Cidade Inteligente) é um conceito que propõe o desenvolvimento de soluções para essas dificuldades encontradas no ambiente urbano [Di Napoli et al. 2014]. Parte do Smart City, o Smart Parking (do inglês, Estacionamento Inteligente) faz uso de dispositivos e novas tecnologias para otimizar o uso e gerenciamento de um estacionamento [Di Napoli et al. 2014].

Existem maneiras de desenvolver uma solução que segue as ideias do Smart Parking. Este trabalho mostra uma comparação entre algumas soluções que utilizam de agentes racionais. Um agente inteligente racional é uma entidade computacional que percebe

o seu ambiente por meio de sensores, compreende o ambiente e realiza ações neste ambiente por meio de atuadores [Wooldridge 2009].

Um abordagem adotada de atitude mental de um agente é a *Belief-Desire-Intention* (do inglês, crença-desejo- intenção, BDI). A suposição essencial do modelo BDI é que as ações são derivadas de um processo de raciocínio prático, que é composto de duas etapas [Wooldridge 2003]. Na primeira etapa, a definição de objetivos, um conjunto de desejos é selecionado para ser alcançado, de acordo com a situação atual dos conhecimentos do agente. A segunda etapa é responsável pela determinação de como esses objetivos podem ser alcançadas por meio das opções disponíveis para o agente.

Este artigo está organizado da seguinte maneira. A seção 2 apresenta trabalhos de pesquisa que são relacionadas a soluções de Smart Parking utilizando agentes. A seção 3 mostra uma comparação entre estes trabalhos analisados. Por fim a seção 4 descreve as considerações finais do trabalho.

2. Análise de Smart Parking baseados em agentes

Os trabalhos analisados foram escolhidos a partir da busca "Agents Smart Parking" no Portal Capes. O objetivo é analisar os principais aspectos no desenvolvimento de sistemas que utilizam SMAs para smart parking. Existem outros artigos como [Mahmud et al. 2013] e [Lin et al. 2017] que objetivam fazer uma comparação de soluções de smart parking, neste artigo procura-se delinear uma comparação daqueles sistemas que utilizam soluções baseadas em agentes.

2.1. Modelo PARKAGENT

Em [Benenson et al. 2008] é apresentado PARKAGENT, um modelo espacial que constrói em SIG (Sistema de Informação Geográfica) um modelo de ambiente urbano com camadas que representam cada elemento da infraestrutura de tráfego pertinente ao processo de estacionamento. O modelo também é baseado em agentes que funcionam como os motoristas que dirigem até o destino, procuram por um estacionamento, estacionam e deixam o local quando suas atividades no destino terminam.

O modelo contém regras que descrevem o comportamento desses agentes drivers (motoristas) e incluem um descrição da reação de cada agente driver à falta de estacionamentos, diferenças de preços e comportamentos de outros agentes. As ações de direção, busca de estacionamento, estacionar e saída do agente, também estão contidas nestas regras.

A cada iteração (passagem de tempo), cada agente driver pode realizar um movimento. O tamanho desse movimento depende da velocidade do veículo deste agente. Quando um agente driver se aproxima de uma junção ele deve decidir qual direção tomar para alcançar seu destino. No modelo, essa decisão é baseada na comparação da distância até o destino da junção atual e de todos os cruzamentos seguintes. O modelo assume que o agente possui algum conhecimento da cidade e, assim, seleciona uma trajetória mais curta possível. Os agentes são criados no modelo a uma distância determinada do destino (250 metros), e então se tornam "cientes"da necessidade de começar a procurar estacionamento.

Durante a busca de estacionamento, a velocidade de cada carro permanece entre 20 e 25 km/h. Nesse momento cada agente passa a verificar estacionamentos em sua

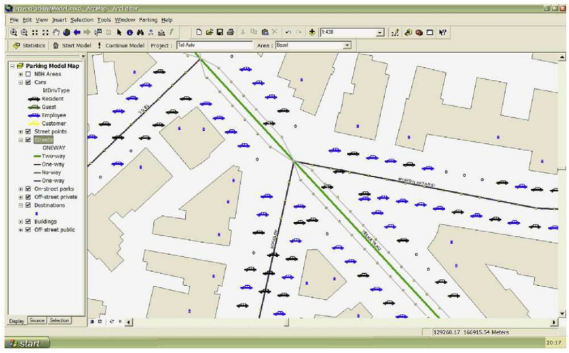


Figura 1. As camadas básicas e derivadas do modelo PARKAGENT na janela do modelo do ArcGIS. Fonte: [Benenson et al. 2008]

área. A velocidade é reduzida para 10 a 12 km/h quando o agente passa a observar qual vaga estacionar. Os agentes driver conseguem verifica se o espaço a frente está livre ou não. Caso ocupado o seu avanço é interrompido. A ordem em que os carros avançam é estabelecida aleatoriamente a cada iteração. Agentes são separados em grupos que implicam em seu destino, tempo de chegada e tempo estacionado.

2.2. Sistema ASPIRE

No artigo [Rizvi et al. 2018] é utilizado um agente de software em nuvem que determina a vaga de estacionamento mais apropriada para um motorista em uma rede de estacionamentos. Cada veículo no sistema contém um agente local com as preferências de seu motorista. Um agente parker na nuvem gerencia a alocação de vagas de cada agente local. Este agente é capaz de interagir com serviços de trafego, clima e emergência relacionados para encontrar a vaga que confere com as preferências do agente local: categoria da vaga, o preço máximo aceitável do estacionamento e tempo tolerado de caminhada até o seu destino a partir dessa vaga.

ASPIRE necessita de 4 blocos para controlar os estacionamentos. O primeiro e segundo bloco são esquematizados na figura 2. O primeiro é a Park Unit (PU) que utiliza uma etiqueta RFID mantida no veículo escaneada nos portões de entrada e saída dos estacionamentos para autenticação e rastreamento do motorista e agente local.

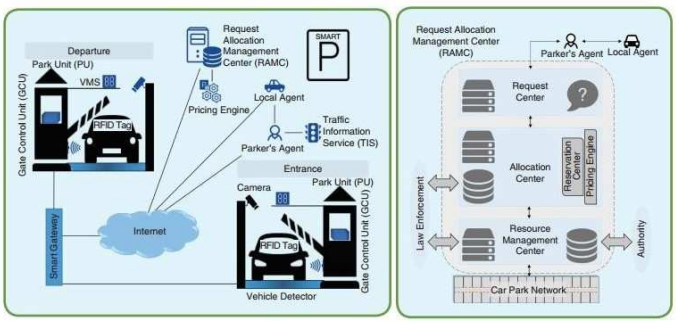


Figura 2. Principais blocos do sistema ASPIRE. Fonte: [Rizvi et al. 2018]

O segundo bloco é o Request Allocation Management Center (RAMC). Este bloco registra, salva e processa informações fornecidas pelas PUs. A transferência de dados ocorre entre a UCP e o RAMC, o RAMC possui três subsistemas (centros) principais:

de requisição, de alocação e de gerenciamento de recursos. O terceiro bloco é o Agente Parker que tem a função de definir a vaga adequada a partir de informações de serviços relacionados e dos dados recebidos do bloco CGAS. O quarto bloco é o Agente Local, que gerencia as preferências do motorista e se comunica com o agente parker realizar pedidos.

2.3. Cooperative Car Parking

Em [Aliedani and Loke 2018] é foi desenvolvido de um Sistema multiagente aberto no qual os veículos (agentes) entram e saem frequentemente do estacionamento e comunicam entre si oportunamente quando estão em uma distância DSRC um do outro. O objetivo dessa comunicação entre agentes é compartilhar informações e conselhos entre si com o intuito de ajudar uns aos outros a encontrar uma vaga o mais próxima possível do destino.

Todo veículo contém um software de agente responsável por cooperar com outros veículos e um aparelho DSRC (Comunicação Dedicada de Curto Alcance) para comunicação wireless entre agentes. O agente inicia a busca por vaga com uma crença de que existe menos vagas perto do destino e elas aumentam ao se distanciar do mesmo. A área do estacionamento é dividida em setores baseados na distância até o destino e os veículos são equipados com sensores para informar o agente ao detectar vagas vazias e ocupadas.

A vaga mais próxima do destino é escolhida pelo agente, o qual então se move em direção a ela. O veículo transmite periodicamente mensagens compartilhando suas intenções com outros carros. Os veículos reúnem conhecimento sobre a demanda (por vagas de estacionamento) de setores. Um veículo pode mudar sua intenção de vaga se perceber que ela está ocupado ou foi selecionada por outros agentes.

Caso o agente verifique que no setor pretendido há outra vaga livre, com base no conhecimento do agente, este seleciona a vaga como sua próxima intenção. Caso contrário, uma nova avaliação da utilidade dos setores disponíveis restantes é calculada. O agente seleciona a área de maior utilidade para se mover, ou seja, uma vaga dessa área de serviço mais alta seria selecionado como nova intenção de vaga. Um veículo pode decidir alterar seu setor de pesquisa de destino em diferentes intervalos de tempo com base em quando reconhece a falta de chance de estacionar na área de destino atual.

2.4. Smart Parking Guidance

O sistema, proposto em [Shin and Jun 2014], destina o carro à uma vaga usando uma regra de despacho baseada na avaliação dos valores de dois tipos de funções de utilidade de estacionamento. A atribuição do carro solicitante à um determinado estacionamento deve ser feita considerando-se as características dinâmicas do estacionamento, como seu uso, a condição do tráfego, a conveniência do usuário e assim por diante. São considerados vários fatores dinâmicos para a alocação do veiculo: tempo de viagem do local atual para o estacionamento, distância do estacionamento até o destino, custo do estacionamento, probabilidade de disponibilidade de vagas e congestionamento de trafego até o estacionamento.

Em [Shin and Jun 2014] são considerandos cinco objetos: estacionamento, sistema de gestão de estacionamentos, servidor central, dispositivo de navegação e o motorista. Cada estacionamento dispõe de um sensor que atualiza a sua disponibilidade. Essa

informações são enviadas ao sistema de gestão de estacionamentos que coleta o status de todos os estacionamentos conectados. O servidor central recebe esses dados do sistema de gestão e os armazena em seu banco de dados.

O processo de alocação começa no momento em que motorista requisita uma vaga pelo dispositivo de navegação em seu carro. O motorista insere o seu destino e a requisição é enviada ao servidor central junto com suas preferências e informações como: local atual do veículo, distância até estacionamentos perto do destino, e distancia de caminhada dos estacionamentos até o destino. O servidor central, um agente, escolhe uma vaga que mais confere com situação atual do motorista, suas preferências e status dos estacionamentos. A vaga é sugerida ao motorista o qual pode escolher reservar a vaga ou dirigir até o local sem essa reserva. O custo do estacionamento começa a ser calculado a partir do momento dessa reserva.

3. Resultados e Discussão

A partir da análise dos artigos mostrados, observa-se que as soluções de Smart Parking são bastante variadas. A tabela 1 mostra algumas características semelhantes encontradas nestes quatro artigos. É possível perceber que podem existir inúmeras combinações de configurações de Smart Parking. Logo, sem um maior número de sistemas analisados, é difícil chegar a uma conclusão abrangente.

	Parkagent	**Aspire**	**CCP**	**SPG**
Hierarquia	Descentralizado	Centralizado	Descentralizado	Centralizado
Qtde. de Estacionamentos	Vários	Vários	Um	Vários
Comunicação	Nenhuma	Agentes com Agente Gerente	Agentes com outros Agentes	Agentes com Servidor Central
Preferências Satisfeitas	Custo e Distância	Tipo de Vaga, Custo e Tempo	Nenhuma Explícita	Disponibilidade, Custo, Trafego, Tempo e Distância
Politica de Ações	Utilização de Regras Pré-Definidas	Determinado pelo Agente Gerente	Utilização de Crenças e Cooperação entre Agentes	Determinado pelo Servidor Central

Tabela 1. Características dos sistemas analisados

Dois dos sistemas possuem uma hierarquia descentralizada. Em [Benenson et al. 2008] os agentes realizam ações para buscar por uma vaga a partir regras definidas no modelo. Em [Aliedani and Loke 2018] os agentes cooperam entre si para encontrar uma vaga o mais próximo do destino. Os outros dois sistema utilizam de uma hierarquia centralizada. Em [Rizvi et al. 2018] o agente parker controle as ações e escolhe as vagas dos agentes clientes. Em [Shin and Jun 2014] um servidor central defini os passos dos agentes que requisitaram uma vaga.

O sistema [Aliedani and Loke 2018], diferentemente dos outros demais implementa apenas um estacionamento. Apenas em [Benenson et al. 2008] não é realizado nenhuma comunicação entre agentes. Nos outros sistemas é feito ou uma interação com um agente gerente [Rizvi et al. 2018], com outros agentes [Aliedani and Loke 2018] ou com um servidor central [Shin and Jun 2014].

Os sistemas desenvolvidos em [Benenson et al. 2008], [Rizvi et al. 2018] e [Shin and Jun 2014] consideram preferências do motorista ao encontrar uma vaga. Os três consideram o preço do estacionamento. O segundo e quarto sistema consideram o tempo gasto de maneiras diferentes. O segundo considera o tempo de caminhada até seu

destino e o quarto consideram tempo de viagem até o estacionamento. Apenas o segundo considera o tipo de vaga desejada. O primeiro e o quarto consideram a distância até o destinho. Apenas o quarto considera disponibilidade de vagas e o trafego até o estacionamento como preferências do motorista. O sistema [Aliedani and Loke 2018] não utiliza explicitamente nenhuma preferência do motorista, mas é utilizado uma crença que vagas mais próxima do destino são mais desejáveis.

4. Conclusão

Neste artigo foi comparados quatro SMAs que gerenciam a busca por vagas em estacionamentos inteligentes. Foram detalhadas as características de como cada sistema funciona e quais são suas prioridades, para identificar semelhanças e diferenças entre os SMAs. Foi identificado que ao existir inúmeras maneiras de desenvolver um Smart Parking é difícil chegar a uma conclusão abrangente ao comparar trabalhos sobre o mesmo. Deve-se considerar os resultados adquiridos neste artigo, principalmente do sistema ASPIRE, como base de desenvolvimento para o futuro. Como próximo trabalho será desenvolvido um sistema multiagente capaz de alocar vagas considerando variáveis de ambiente. Como, por exemplo, aplicar uma reorganização dinâmica das vagas do estacionamento, em termos de preço e tipo, quando ocorre um evento diferenciado (e.g., uma partida esportiva). Ou seja, muda o ambiente, mudam as demandas dos motoristas, por conseguinte altera-se o mecanismo de gerência e alocação das vagas.

Referências

Aliedani, A. and Loke, S. W. (2018). Cooperative car parking using vehicle-to-vehicle communication: An agent-based analysis. *Computers, Environment and Urban Systems*.

Benenson, I., Martens, K., and Birfir, S. (2008). Parkagent: An agent-based model of parking in the city. *Computers, Environment and Urban Systems*, 32(6):431–439.

Di Napoli, C., Di Nocera, D., and Rossi, S. (2014). Negotiating parking spaces in smart cities. In *Proceeding of the 8th International Workshop on Agents in Traffic and Transportation, in conjunction with AAMAS*.

Lin, T., Rivano, H., and Le Mouël, F. (2017). A survey of smart parking solutions. *IEEE Transactions on Intelligent Transportation Systems*, 18(12):3229–3253.

Mahmud, S., Khan, G., Rahman, M., Zafar, H., et al. (2013). A survey of intelligent car parking system. *Journal of applied research and technology*, 11(5):714–726.

Polycarpou, E., Lambrinos, L., and Protopapadakis, E. (2013). Smart parking solutions for urban areas. In *2013 IEEE 14th International Symposium on"A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–6. IEEE.

Rizvi, S. R., Zehra, S., and Olariu, S. (2018). Aspire: An agent-oriented smart parking recommendation system for smart cities. *IEEE Intelligent Transportation Systems Magazine*.

Shin, J.-H. and Jun, H.-B. (2014). A study on smart parking guidance algorithm. *Transportation Research Part C: Emerging Technologies*, 44:299–317.

Wooldridge, M. (2003). *Reasoning about rational agents*. MIT press.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Proposta de implantação de um sistema ciber-físico para um Smart Parking baseado em agentes inteligentes

Pedro W. Botelho[1], André P. Borges[1], Gleifer V. Alves[1]

[1]Departamento Acadêmico de Informática – Universidade Tecnológica Federal do Paraná (UTFPR) – Ponta Grossa – PR – Brasil

pbotelho@alunos.utfpr.edu.br, {apborges,gleifer}@utfpr.edu.br

Abstract. *Smart Parking systems are being used to manage and allocate parking lots in several cities. These systems aim to reduce urban traffic by helping drivers to find a place to park. To assist them with this task it is recommended to use intelligent agents, and in order to act and communicate with a cyberphisical system, these agents find it necessary to use boards and sensors to process information. This paper proposes an architecture that will use Raspberry Pi to embed agents in a Smart Parking and ESP-12e to capture information. As a result, the integration between agents developed in JADE framework and the studied cyberphisical system is expected.*

Resumo. *Sistemas para estacionamentos inteligentes estão sendo utilizados no gerenciamento e alocação de vagas em diversas cidades. Estes sistemas visam a redução do tráfego urbano auxiliando os motoristas na busca por vagas. Para auxilia-los nesta tarefa é recomendado fazer o uso de agentes inteligentes e para que tais agentes possam atuar e se comunicar no sistema ciber-físico é necessário utilizar placas e sensores para processar informações. Neste trabalho é proposto uma arquitetura que utilizará Raspberry Pi na implantação dos agentes e ESP-12e para capturar as informações. Espera-se, como resultado a integração dos agentes desenvolvidos no framework JADE com o sistema ciber-físico estudado.*

1. Introdução

Um *smart parking* (em português, estacionamento inteligente) é um sistema de estacionamento que auxilia os motoristas a encontrar vagas para estacionar. Para isso, é necessário utilizar sensores que detectam se há ou não um veículo estacionado e, então, o sistema direciona o motorista para a vaga (Di Napoli et. al., 2015). Esse conceito vem de *smart cities* (em português, cidades inteligentes) que é uma cidade onde tecnologia da informação e comunicação são associadas com infraestruturas usando novas tecnologias (Batty et. al. 2012).

No projeto Smart Parking desenvolvido em parceria pelo IPB (Instituto Politécnico de Bragança) com a UTFPR-PG (Universidade Tecnológica Federal do Paraná campus Ponta Grossa) estão sendo elaborados protocolos de comunicação e negociação entre agentes (Castro et. al., 2017) (Ducheiko et. al., 2018). Agentes são sistemas computacionais capazes de ações autônomas no ambiente que estão situados visando atingir seus objetivos propostos (Wooldridge, 2002).

Além disso, o projeto também considera a implantação de agentes em plataformas de hardware. O trabalho aqui apresentado está em fase inicial e tem como principal objetivo a implantação de agentes em um CPS (*Cyber-physical system*), em português sistemas ciber-físicos. CPS são elementos computacionais que oferecem comunicação com entidades físicas utilizando plataformas de hardware (Kaithan et. al. 2015). Permitindo assim, que o conceito de *smart parking* seja implantado num ambiente ciber-físico e, portanto, similar a um estacionamento urbano.

Avanços e inovações no campo da tecnologia, como a comunicação sem fio e os dispositivos móveis, permitiram possibilidades de transferir serviços de computadores convencionais para ambientes próximos ao ser humano, utilizando sistemas embarcados (Manogna et. al., 2016). Este trabalho propõe uma arquitetura embarcada que utiliza placas Raspberry Pi, contendo um agente em cada uma delas e componentes ESP-12e, os quais estarão conectados a sensores ultrassônicos de distância, distribuídos uniformemente conforme a quantidade de vagas.

Um *smart parking* utiliza sensores que detectam se há ou não um veículo estacionado, neste caso, um sensor ultrassônico de distância conectado ao ESP-12e. Após a leitura do sensor, o sistema deve informar o motorista qual vaga ele deve estacionar, ou seja, um agente será o sistema embarcado no Raspberry, que retornará ao motorista uma vaga requisitada previamente. Assim, esta arquitetura torna-se viável para o problema dos estacionamentos inteligentes pois é justamente de uma integração hardware e software que o sistema precisa.

Para que isso seja possível, os agentes devem ser implementados em uma determinada linguagem de agentes e então embarcados no hardware. O JADE é um framework para o desenvolvimento de agentes inteligentes totalmente implementado em Java (Greenwood et. al. 2004). Portanto, será utilizado como linguagem de implementação do agente nesta arquitetura pois, como o Raspberry é um microprocessador, é possível executar uma aplicação Java nele.

O restante deste artigo está organizado da seguinte forma. Na seção 2 são descritos alguns trabalhos relacionados ao tema de agentes embarcados em plataformas de hardware e soluções para *smart parking*. Na seção 3, tem-se a descrição da arquitetura proposta no artigo, como ela funcionará e uma descrição das tecnologias envolvidas. Por fim, a seção 4 apresenta as considerações finais a respeito do desenvolvimento do trabalho.

2. Trabalhos Relacionados

Nesta seção descreve-se brevemente alguns trabalhos que apresentam soluções de estacionamentos inteligentes presentes na literatura e trabalhos que utilizam as plataformas de hardware escolhidas para este trabalho, o Raspberry Pi e o ESP-12e.

Em (Khanna, 2016) é proposto um estacionamento inteligente baseado em Internet das Coisas que utiliza o Raspberry como uma unidade processadora, agindo como um intermediário entre os sensores e a *cloud*. Essa arquitetura tem algumas similaridades com o trabalho apresentado aqui, porém não faz uso de agentes inteligentes. O Raspberry é conectado aos sensores e apenas envia quais vagas estão disponíveis, não passando por nenhum processo de negociação de vagas.

Em (Mann, et. al. 2017) é feito um comparativo de dispositivos a serem usados pelo lado do cliente e de baixo custo, como o ESP-12e, Arduino com um módulo de Wi-Fi e um Raspberry Pi. Foi concluído que o uso de um sistema baseado em ESP-12e é a melhor opção para captar informações e enviar para um servidor.

Em (Bensag et. al., 2015) é proposto um agente JADE embarcado em um Raspberry Pi especialmente programado para segmentação por ressonância magnética cardíaca. No artigo, o autor cita que “o Raspberry fornece alta velocidade, melhor precisão, boa flexibilidade e ainda possui um baixo custo para o desenvolvimento de sistemas embarcados”.

Em (Thangam, et. al. 2018) é proposto um sistema de reservas para um *smart parking* utilizando o reconhecimento óptico de caracteres e o reconhecimento facial para fornecer segurança com um Raspberry Pi. O trabalho tem algumas similaridades com a arquitetura proposta neste artigo, pois utiliza tecnologias de Internet das Coisas (IoT), um servidor e um aplicativo, mas também não faz uso de agentes inteligentes.

Conforme observou-se nos trabalhos destacados nesta seção, a utilização dos componentes, Raspberry Pi e ESP-12e são uma alternativa adequada para um ambiente ciber-físico. A partir disso visa-se a criação de um CPS para um estacionamento inteligente.

3. Arquitetura proposta

No estado atual do projeto de pesquisa ao qual este trabalho está associado, estão sendo desenvolvidos paralelamente a implementação de protocolos de negociação descentralizados (atividade A3 da Figura 1) e a análise de requisitos e arquitetura do sistema (atividade A1 da Figura 1) já foi concluída por outro membro do projeto.

Durante o desenvolvimento dos protocolos, os testes estão sendo realizados em um ambiente de simulação e, como a finalidade do projeto Smart Parking é a integração de agentes no hardware, tais protocolos precisam ser testados em um ambiente ciber-físico (atividade A4 da Figura 1).

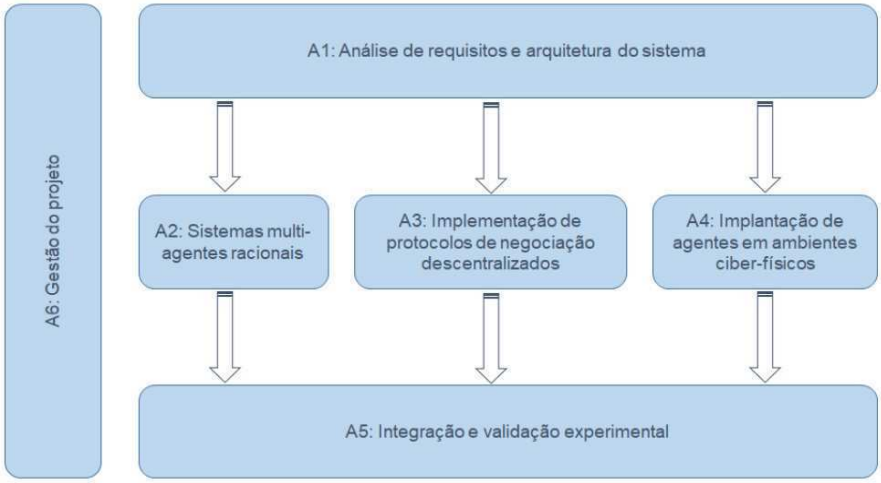


Figura 1: Relações de precedência entre as atividades previstas no projeto.

Considerando este cenário das atividades do projeto de pesquisa, é notória a relevância da arquitetura proposta neste artigo, para que o projeto atinja o último estágio, que é a integração e validação experimental (atividade A5 da Figura 1), para então ser implantada em um estacionamento de fato.

A implantação e validação das atividades dar-se-á em uma arquitetura proposta conforme a Figura 2, sendo composta de:

(a): os motoristas, os quais possuem a intenção de estacionar os seus veículos;

(b): o aplicativo (móvel ou web) que fará a comunicação entre os motoristas e o SMA (Sistema Multi-Agente) presente no servidor;

(c): o servidor *cloud* do aplicativo que possui o SMA e será comunicado com o agente no Raspberry Pi sob demanda;

(d): o Raspberry Pi, contendo um agente desenvolvido em JADE que gerenciará as vagas do estacionamento. Neste agente serão integrados os resultados das atividades A2 e A3 da Figura 1, como por exemplo os protocolos de negociação das vagas;

(e): o protocolo MQTT (Mqtt.org, 2019) que comunica via WiFi o Raspberry Pi com o ESP-12e para que o agente no Raspberry Pi (d) tenha conhecimento das vagas livres e ocupadas;

(f): o ESP-12e conectado a um sensor ultrassônico de distância monitorando se uma vaga está disponível ou não;

(g): as vagas de estacionamento contendo um ESP-12e conectado a um sensor (f) em cada vaga;

(h): um setor do estacionamento.

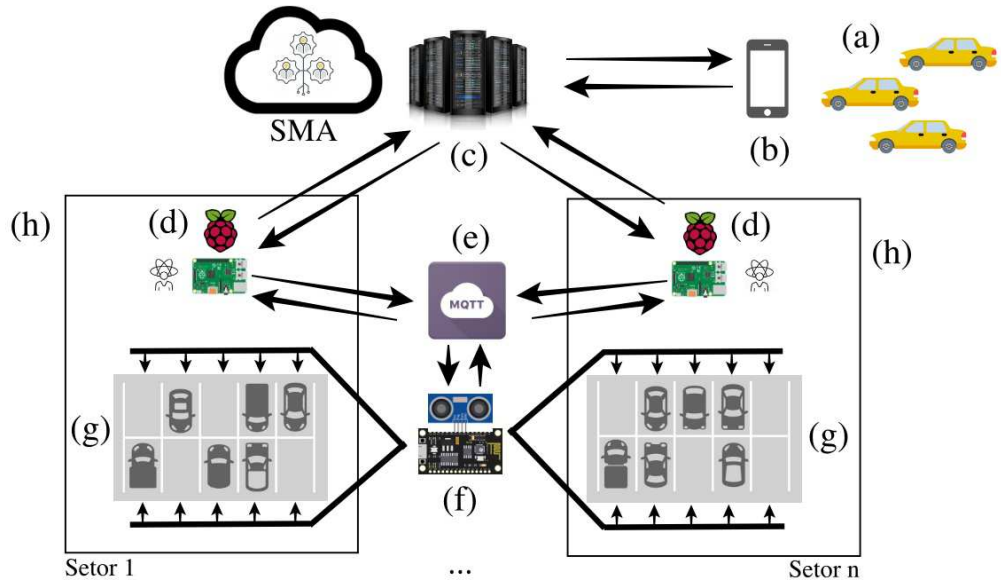


Figura 2: Arquitetura ciber-física proposta.

A estrutura (h) proposta pode ser vista como apenas uma parte de um estacionamento, denominado setor. Contudo, espera-se ser possível replicar tal estrutura, escalando a quantidade dos elementos (d), (f) e (g) de acordo com a necessidade do local. Neste caso, a comunicação entre tais estruturas será de responsabilidade dos Raspberry Pi (d) com os ESP-12e (f).

O Rapsberry Pi é um computador de baixo tamanho e custo com grande capacidade de processamento e comunicação que pode ser utilizado por qualquer pessoa para aprender a programar e interagir com o mundo físico (Raspberrypi.org, 2019). Como dito anteriormente, o Raspberry Pi é um microprocessador e portando, pode-se rodar uma aplicação Java. Deste modo, o agente embarcado nele recebe as requisições do motorista e, através dos protocolos de negociação de vagas contidas no agente, busca a melhor para aquele motorista.

O ESP-12e é um módulo Wi-Fi de baixo custo e com um baixo consumo de energia, feito especialmente para dispositivos e aplicações IoT (Zhang, et. al 2015). Os módulos ESP podem ser programados na IDE do Arduino, por isso serão conectados a um sensor ultrassônico modelo HC-SR04, que é um módulo muito usado em projetos Arduino, facilitando a programação.

Por ser um módulo Wi-Fi, a placa ESP pode ser conectada a internet e pode se comunicar com o Raspberry Pi via protocolo MQTT (*Message Queue Telemetry Transport*). Este protocolo foi desenvolvido pela empresa IBM e é um protocolo de troca de mensagens projetado para ser leve e ideal para conexões máquina-máquina (Mqtt.org, 2019).

Faz-se necessário destacar que o trabalho aqui apresentado tem como escopo a camada ciber-física, portanto, a implementação dos protocolos de negociação de vagas é

desenvolvida em outra camada do projeto Smart Parking, conforme ilustrado previamente na Figura 1.

Por meio da arquitetura proposta, pretende-se que o sistema funcione, da seguinte forma:

1. O motorista (a), a partir de um aplicativo (b), solicitará uma vaga de estacionamento;
2. A requisição será atendida pelo servidor *cloud* (c) e a partir dessa requisição, o SMA irá executar as respectivas negociações para atende-la;
3. O agente SMA, que possui as requisições do motorista, se comunica com o agente no Raspberry Pi (d) enviando seus critérios, por exemplo, o preço da vaga desejada, o local, a distância do motorista até ela, etc.;
4. Neste passo, o agente no Raspberry Pi (d) terá a informação da vaga solicitada pelo motorista e buscará uma vaga exatamente igual. Se não houver nenhuma vaga que atenda o critério, o agente (d) buscará encontrar a vaga mais similar;
5. Cada placa ESP-12e (f) estará conectada a um sensor ultrassônico de distância situado em cada vaga do estacionado, que faz a leitura da disponibilidade dela. Para que o Raspberry Pi saiba se a vaga está disponível, é feita a comunicação dele via MQTT (e) com a placa (f);
6. Em seguida, o agente no Raspberry Pi (d) enviará ao servidor (c) o local da vaga obtida através dos critérios e o mesmo enviará a vaga para o aplicativo (b);
7. De posse desta, o aplicativo (b) repassará esta vaga ao motorista (a) para que possa estacionar.

Uma possível vantagem desta arquitetura é o baixo custo de instalação, visto que as placas de maior custo são utilizadas apenas sob demanda do estacionamento, ou seja, quanto maior o estacionamento, mais placas será necessário. Outra possível vantagem é a autonomia que os agentes possuem, ou seja, uma vez embarcados, eles farão o trabalho de maneira autônoma, sem a necessidade de intervenção humana. Com isso, a manutenibilidade também terá um baixo custo, visto que será feita apenas em casos de falha técnica em uma placa ou um sensor.

Além da mobilidade que os agentes possuem, podendo ir de um setor do estacionamento para o outro, a rapidez de resposta do sistema também é uma possível vantagem, pois o Raspberry Pi possui um poder de processamento satisfatório para o estudo de caso aqui descrito, dando assim uma rápida resposta ao motorista.

4. Considerações finais

Nos últimos anos, os avanços nos estudos em sistemas ciber-físicos e em IoT tornaram o conceito de cidades inteligentes cada vez mais próximo da realidade. Este trabalho apresentou uma proposta de arquitetura ciber-física que conecta Agentes Inteligentes com IoT visando a construção de um estacionamento inteligente. Este sistema fornece um SMA capaz de utilizar protocolos de negociação para gerenciamento e alocação de vagas no *smart parking*.

Este trabalho está em fase inicial, portanto ainda serão avaliados alguns critérios, como a percepção de falhas das placas, a diferença entre sensores falhos e vagas ocupadas, se o tempo de resposta dos sensores e das placas é o desejado, assim como um gargalo que pode ser provocado por muitos agentes trocando mensagens ao mesmo tempo, etc.

Como continuidade imediata deste trabalho, pretende-se implantar a arquitetura proposta neste artigo usando as ferramentas já mencionadas. A partir disso, será executada uma quantidade significativa de diferentes experimentos, os quais possam evidenciar não apenas a robustez, a eficiência, e a aplicabilidade da arquitetura, mas também a evidência de eventuais falhas, e como as mesmas podem ser contornadas no sistema.

Referências

Batty, M, Axhausen, K., et al. “Smart Cities of the Futures”, UCL Working Papers Series, Londres - Inglaterra, v. 188, out. 2012.

Bellifemini, F., Caire, G., Greenwood, D. (2004). Developing Multi-Agent Systems with JADE. John Wiley and Sons, London.

Bensag, H., Youssfi, M., Bouattane, O. (2015). “Embbeded agent for medical image segmentation”. In Proceedings of 27th International Conference on Microeletronics: ICM’15, Casablanca, Morocco.

Castro, L.F., Alves, G.V., Borges, A.P. Using trust degree for agents in order to assign spots in a Smart Parking. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal 6, 45–55. https://doi.org/10.14201/ADCAIJ207624555. 2017.

Di Napoli, C., Di Nocera, D., Rossi, S. (2014). “A Social-Aware Smart Parking Application”. In Proceedings of 15th Workshop “From Objects to Agents”, Catania, Italy.

Ducheiko, F.F., André, P.B., Gleifer, V.A. Implementação de Modelo de Raciocínio e Protocolo de Negociação para um Estacionamento Inteligente com Mecanismo de Negociação Descentralizado. REVISTA JUNIOR - ICCEEg 1, 25–32. 2018.

Kaithan, S.K., McKalley, J.D. (2014). “Design Techniques and Applications of Cyberphisical Systems: A Survey”. In IEEE Systems Journal, p. 1-16, jun. 2015.

Khanna, A. “IoT based Smart Parking System”. In IOTA: International Conference on Internet of Things and Applications, p. 266-270, Jan, 2016.

Mann, G., Iqbal, T., Jayasinghe, S. “Low-Cost and Open Source SCADA Options for Remote Control and Monitoring of Inverters”. In CCECE: Canadian Conference on Electrical and Computer Engineering, p. 1-4, 2017.

Manogna, S., Dakannagari, H.R. (2016). “Internet of Things”. In International Journal of Computer Science and Information Technologies, p. 1567-1570. 2016.

Mqtt.org. “Frequently Asked Questions”, http://mqtt.org, Abril 2019.

RaspberryPi.org. “What is Raspberry Pi?”, https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi, Março 2019.

Thangam, E. C., Mohan, M., Ganesh, J. Sukesh, C.V. “Internet of Things (IoT) based Smart Parking Reservation System using Raspberry-Pi”. In ISSN: International Journal of Applied Engineering Research, p. 5759-5765. 2018.

Wooldridge, M. “An Introduction to Multi-Agent Systems”, 2nd. Ed. [S.1.]: Wiley Publishing, 2009.

Zhang, Y. P., Liu, T. Yang, Z. X. Mou, Y., Wei Y. H., Chen, D. “Design of Remote Control Plug”. In International Conference on Applied Superconductivity and Electromagnetic Devices, p. 29-30, nov. 2015.

Ritmo Circadiano e a Variável Dor: Revisões Sistemáticas com a utilização de Simulação Multiagente

Angélica T. Santos[1], **Andre A. Longaray**[1], **Catia M. Machado**[1], **Diana F. Adamatti**[1]

[1]Programa de Pós Graduação em Modelagem Computacional (PPGMC)
Universidade Federal do Rio Grande (FURG)
Caixa Postal 474 – 96.203.900 – Rio Grande – RS – Brasil

{theisangelica, andrelongaray, catiamachado, dianaadamatti}@furg.br

Abstract. *The circadian rhythm is responsible for the daily variations in metabolism and its disorders has direct implications with many diseases, such as obesity and mental disorders. In this way, the proposed work aims to systematic review of mathematical and computational models based on multiagent simulation to synchronization and desynchronization of the circadian rhythm in relation to the pain variable. A systematic review aims to show the novelty of one work, providing the basis for a complete and impartial bibliographic review, in such a way that it produces results of scientific value.*

Resumo. *O ritmo circadiano é responsável pelas variações diárias no metabolismo e seus distúrbios tem implicações diretas com muitas doenças, como, a obesidade e transtornos mentais. O trabalho proposto tem como meta a revisão sistemática de um modelo matemático e computacional baseado em simulação multiagente, para sincronização e dessincronização do ritmo circadiano em relação a variável dor. Nesse sentido, uma revisão sistemática objetiva auxiliar no embasamento para a revisão bibliográfica completa e imparcial, de tal forma que produz resultados de valor científico.*

1. Introdução

As mudanças cíclicas, que se repetem ao longo de um determinado período, estão relacionadas ao ritmo biológico, mais especificamente ao ritmo circadiano. A regulação destas mudanças é realizada por ritmos circadianos e homeostáticos [Borbély and Achermann 1999], que é caracterizada pela redução significativa da atividade motora e da percepção de estímulos sensoriais.

A simulação computacional é uma técnica que envolve a construção de um modelo para representar uma situação real em posterior experimentação. A simulação necessita de modelo matemático e computacional, mais especificamente, neste trabalho, dos sistemas multiagente, para validar a pesquisa.

Os modelos matemáticos são úteis para representar situações reais, fazer previsões e auxiliar no apoio de decisões. Já os sistemas multiagente, área pertencente a inteligência artificial, permitem, através de suas ferramentas, simular regras de comportamento de um determinado sistema. Nesse contexto, [Ferber 1991] dá uma possível definição pra agentes, que é uma entidade real ou abstrata, sendo capaz de agir sobre ela mesma ou em um ambiente multiagente.

Neste estudo, apresenta-se a revisão sistemática de sistema multiagente, ritmo circadiano e dor, tendo como base estudos já realizados por [Borbély 1982], que desenvolveu um modelo matemático que descreve as curvas do ritmo circadiano, sendo que [Borbély and Achermann 1999] aprimoraram este modelo, na qual [Skeldon 2014] implementou o modelo aprimorado utilizando sistema multiagente. O objetivo deste trabalho é identificar as lacunas e obter o embasamento teórico para o ritmo circadiano e a variável da dor em um sistema multiagente.

2. Fundamentação Teórica

Contextualizando, os ritmos biológicos constituem-se do sistema de temporização endógeno. Esse sistema compreende a rede de osciladores, os quais, medem os ciclos ambientais claro-escuro de 24 horas. Um dos mecanismos importante da regulação do vigília-sono, é mostrado pelo processo $\tilde{S}$ (união do ritmo circadiano e homeostático) . Este por sua vez, é dependente da duração e qualidade do sono [Borbély and Achermann 1999]. A duração da vigília incrementa o processo $\tilde{S}$, aumentando assim o tempo de dormir.

O ritmo circadiano regula os ritmos materiais e psicológicos do ser humano, sendo controlado por um marca-passo localizado no cérebro, que é independente da vigília e do sono. O ritmo homeostático é decorrente da vigília-sono que procede do modelo $\tilde{S}$, na qual é a pressão decorrente do sono acumulado durante o dia e que diminui durante a noite. O ritmo homeostático tem um aumento sinusoidal, desde o início da vigília até o início do sono, na qual sofre uma queda até o seu final [Borbély and Achermann 1999].

A falta de sono ou alteração do sono, devido a dor, provoca prejuízos substanciais no desempenho físico e cognitivo. As tarefas cognitivas sofrem redução da eficiência do processamento cognitivo devido à privação de sono [Ellenbogen 2005]. Segundo a Associação Internacional de Estudos da Dor (*International Association for the Study of Pain* - IASP), a dor é conceituada como "uma experiência sensorial, emocional e desagradável, associada à um dano causado no corpo"[Chapman et al. 1985].

3. Metodologia

Ao desenvolver a pesquisa sobre um determinado tema, constantemente encontra-se resultados não proveitosos. Utilizando-se das principais bases de trabalhos, a revisão sistemática da literatura trata-se de um tipo de investigação focada em uma questão de pesquisa, que visa identificar, selecionar e avaliar trabalhos para o embasamento teórico. Nesse texto são seguidos os passos propostos por [Galvão and Pereira 2014, Keele et al. 2007], que preveem:

- elaboração da pergunta de pesquisa;
- busca na literatura;
- seleção dos artigos;
- extração dos dados;
- avaliação da qualidade metodológica;
- síntese dos dados (metanálise);
- avaliação da qualidade das evidências; e
- redação e publicação dos resultados.

A pergunta de pesquisa que guia este trabalho é: “A modelagem matemática e simulação computacional é capaz de mostrar a influencia da dor no ritmo circadiano"?

Com a delimitação da pergunta de pesquisa, define-se os sub-passos: Avaliação do título, avaliação do resumo, leitura parcial e leitura do texto completo.

No primeiro momento, foram utilizadas palavras-chaves como "circadian AND multiagent", gerando como resultado uma grande quantidade de artigos contendo essas palavras em seus títulos ou definição de palavras-chaves do artigo. Por exemplo, na base de dados do Google Acadêmico, foi encontrado 28.400 artigos, indicando que seria necessário refinar as palavras-chave e as também as bases de dados.

As palavras-chave foram definidas conforme a pergunta de pesquisa, sempre utilizando "AND" ou "OR" para a pesquisa: *circadian; multiagent system (simulation) - com suas variações no singular e plural, com hífen, sem hífen (ver Tabela 1); pain; mathematical model and biological system*.

As bases utilizadas para a pesquisa foram de escolha aleatória, sendo todas inclusas no Acesso ao Portal de Periódicos da CAPES, via Cafe, sendo *Scopus (Elsevier); ScienceDirect; Springerlink e PMC (Pubmed)*. A base da PubMed, desenvolvida pelo Centro Nacional de Informações sobre Biotecnologia (NCBI), está indexada na PMC Central.

Toda a revisão sistemática está sendo realizada com o software gratuito e livre "Mendeley", que auxilia na organização de arquivos eletrônicos (formato PDF ou Bibtex), além de ajudar na normalização de citações e referências geradas automaticamente.

4. Discussões e Resultados

A revisão sistemática está sendo desenvolvida com as áreas que envolvem a pesquisa: **Multiagent - Circadian - Pain**. Na figura 1, é possível analisar a união dos conjuntos das áreas. Cada cor na figura refere-se à uma revisão sistemática.

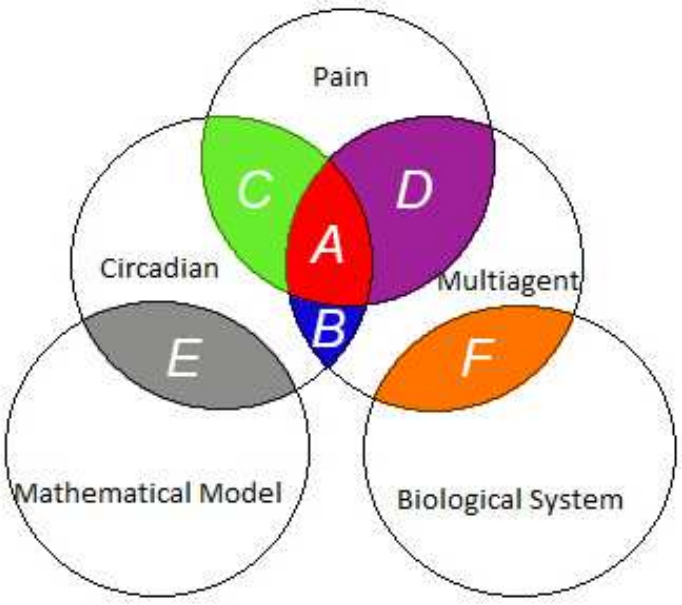


Figura 1. Áreas de Pesquisa envolvidas e suas inter-relações.

- A - Cor vermelho - União das três grandes áreas de pesquisa: **Multiagent - Circadian - Pain**;

- B - Cor azul - União **Circadian - Multiagent**;
- C - Cor verde - União **Circadian - Pain**;
- D - Cor roxo - União **Multiagent - Pain**;
- E - Cor cinza - União **Circadian - Mathematical Model**;
- F - Cor laranja - União **Multiagent - Biological System**.

A primeira revisão sistemática **A - Cor vermelho - União das três grandes áreas de pesquisa: Multiagent - Circadian - Pain**, não foi encontrando nenhum artigo que mostre a união destas três grandes áreas, exceto os publicados pelos autores. Para ter maior garantia desta, fez-se a pesquisa em mais bases de dados, como: Scopus, PMC, SpringerLink, ScienceDirect, Scielo, Science, Web Of Science, Nature e PlosOne.

A segunda revisão sistemática **B - Cor azul - União Circadian - Multiagent**, considera as variações de "multiagent". A Tabela 1 mostra as bases e palavras-chave. Foram encontrados diversos resultados, conforme mostra a Tabela 2, onde inicialmente obteve-se 492 artigos completos. Ao retirar os artigos duplicados, reduziu-se para 299 artigos. Na primeira leitura, avaliando o título, reduziu-se para 69 artigos. Na segunda fase, leitura de título e resumo, restaram 21 artigos, onde deveriam ter no desenvolvimento do trabalho a proposta do "circadiano" e "multiagente". Na terceira fase, leitura parcial (introdução, metodologia e conclusão) restaram 5 artigos. Estes foram avaliados e consideram-se apenas 2 como relacionados ao trabalho para embasamento teórico. Os trabalhos resultantes são: [Andreychenko et al. 2016, Baptista and Costa 2008].

Tabela 1. Bases e Palavras Chave: B - Cor azul - União Circadian - Multiagent

BASE	PALAVRA CHAVE
SCIENCEDIRECT SPRINGERLINK SCOPUS	(ALL ("circadian") AND ALL ("multiagent") OR ALL ("multi-agent") OR ALL ("multiagent simulation") OR ALL ("multi-agent simulation") OR ALL ("multiagent system") OR ALL ("multi-agent system") OR ALL ("multiagents") OR ALL ("multi-agents") OR ALL ("multiagents simulation") OR ALL ("multi-agents simulation") OR ALL ("multiagents simulations") OR ALL ("multi-agents simulations") OR ALL ("multiagents system") OR ALL ("multiagent systems") OR ALL ("multiagents systems") OR ALL ("multi-agents system") OR ALL ("multi-agents systems") OR ALL ("multi-agent system"))
PMC	"(circadian) AND multiagent"; "(circadian) AND multi-agent"; "(circadian) AND multiagent system"; "(circadian) AND multi-agent system"; "(circadian) AND multiagent simulation"; "(circadian) AND multi-agent simulation"; "(circadian) AND multiagents"; "(circadian) AND multi-agents"; "(circadian) AND multiagents simulation"; "(circadian) AND multiagents simulations"; "(circadian) AND multi-agents simulation"; "(circadian) AND multi-agents simulations"; "(circadian) AND multiagents system"; "(circadian) AND multiagents systems"; "(circadian) AND multiagent systems"; "(circadian) AND multi-agents system"; "(circadian) AND multi-agents systems"; "(circadian) AND multi-agent systems".

Na terceira revisão sistemática **C - Cor verde - União Circadian - Pain**, que está

Tabela 2. Revisão Sistemática: B - Cor azul - União Circadian - Multiagent

	Artigos
Artigos encontrados nas bases	492
Artigos não duplicados (retirar os duplicados)	299
Após 1a leitura - título	69
Após 2a leitura - título e resumo	21
Após 3º leitura - introdução, metodologia e conclusão	5
Após 4º leitura completa	2

em andamento nas fases de leitura de título e título e resumo, tem-se um total de 499 artigos completos. Após retirar os duplicados, reduziu-se para 485 artigos.

Na quarta revisão sistemática **D - Cor roxo - União Multiagent - Pain** em andamento na fase de leitura de título e resumo, tem-se 110 artigos completos. Após retirar os artigos duplicados, reduziu-se para 103 artigos, e após a leitura do título, para 44 artigos.

Na quinta revisão sistemática **E - Cor cinza - União Circadian - Mathematical Model** considera o modelo matemático de [Borbély 1982], que rege toda a pesquisa e o modelo de implementação de [Skeldon 2014]. Na Tabela 3 visualiza-se o desenvolvimento da pesquisa. Os trabalhos resultantes são: [Borbély 1982, Daan et al. 1984, Achermann et al. 1993, Borbély and Achermann 1999, Achermann and Borbély 2003, Borbely and Achermann 2011, Borbély et al. 2016].

Tabela 3. Revisão Sistemática: E - Cor cinza - União Circadian - Mathematical Model [Borbély 1982]

	Artigos
Artigos encontrados nas bases	143
Artigos não duplicados (retirar os duplicados)	141
Após 1a leitura - título	89
Após 2a leitura - título e resumo	48
Após 3º leitura - introdução, metodologia e conclusão	27
Após 4º leitura completa	7

Na sexta revisão sistemática sobre **F - Cor laranja - União Multiagent - Biological System** esta em fase de andamento, com 168 artigos para serem analisados.

A revisão sistemática está sendo realizada entre todas as áreas que enquadram-se neste trabalho para servir de embasamento teórico. Após concluída a revisão sistemática, utilizaremos seus resultados para realizar a modelagem do ritmo circadiano e variável dor, sob a perspectiva de sistema multiagente, bem como os parâmetros, agentes e interações que ocorrem.

5. Conclusões

Como resultados preliminares deste trabalho, existem os resultados obtidos das revisões sistemáticas já concluídas e os resultados que serão obtidos nas revisões em andamento. Com as revisões já concluídas, afirma-se que existem poucos estudos que englobam as áreas de pesquisa.

Sendo como objetivo deste trabalho identificar as lacunas e obter o embasamento teórico, vislumbra-se que é necessário desenvolver um ambiente multiagente para ritmo circadiano, considerando a variável dor. O processo de revisão sistemática será um guia referencial no estudo proposto.

Referências

Achermann, P. and Borbély, A. A. (2003). Mathematical models of sleep regulation. *Front Biosci*, 8(Suppl.):S683–S693.

Achermann, P., Dijk, D.-J., Brunner, D. P., and Borbély, A. A. (1993). A model of human sleep homeostasis based on eeg slow-wave activity: quantitative comparison of data and simulations. *Brain research bulletin*, 31(1-2):97–113.

Andreychenko, A. et al. (2016). Analyzing resilience properties in oscillatory biological systems using parametric model checking. *Biosystems*, 149:50–58.

Baptista, T. and Costa, E. (2008). Evolution of a multi-agent system in a cyclical environment. *Theory in Biosciences*, 127(2):141–148.

Borbély, A. A. (1982). A two process model of sleep regulation. *Hum neurobiol*, 1(3):195–204.

Borbely, A. A. and Achermann, P. (2011). Sleep homeostasis and models of sleep regulation. *Principles and Practice of Sleep Medicine*, page 431–444.

Borbély, A. A., Daan, S., Wirz-Justice, A., and Deboer, T. (2016). The two-process model of sleep regulation: a reappraisal. *Journal of sleep research*, 25(2):131–143.

Borbély, A. A. and Achermann, P. (1999). Sleep homeostasis and models of sleep regulation. *Journal of biological rhythms*, 14(6):559–570.

Chapman, C. R., Casey, K., Dubner, R., Foley, K., Gracely, R., and Reading, A. (1985). Pain measurement: an overview. *Pain*, 22(1):1–31.

Daan, S., Beersma, D., and Borbély, A. A. (1984). Timing of human sleep: recovery process gated by a circadian pacemaker. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 246(2):R161–R183.

Ellenbogen, J. M. (2005). Cognitive benefits of sleep and their loss due to sleep deprivation. *Neurology*, 64(7).

Ferber, J. (1991). L'intelligence artificielle distribuée. In *International Workshop on Expert Systems Their Applications*, number 09. Cours.

Galvão, T. F. and Pereira, M. G. (2014). Revisões sistemáticas da literatura: passos para sua elaboração. *Epidemiologia e Serviços de Saúde*, 23:183–184.

Keele, S. et al. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical report, Technical Report. EBSE.

Skeldon, A. (2014). Are you listening to your body clock? `http://personal.maths.surrey.ac.uk/st/A.Skeldon/sleep.html`. [Online; accessed 25-March -2019].

XIII Workshop Escola de Sistemas de Agentes, Seus Ambientes e apliCações