



Anais do XII Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações

— WESAAC 2018 —

Organizado por

Mariela Inés Cortés

Jerusa Marchi

Eurico Vasconcelos

Fortaleza, 02 – 04 de Maio, 2018.

Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações — XII WESAAC /Cortés M.I.; Marchi, J.; Vasconcelos, E. (Org). ANAIS. — — Fortaleza, 2018.

268p. :il.

ISSN 2177-2096

1. Agentes Inteligentes. 2. Sistemas de Agentes de Software. 3. Ambientes para Agentes. 4. Aplicações de Agentes. I. Cortés M. II. Marchi, J. III. Vasconcelos, E.

CDD

PREFÁCIO

Este documento contém os trabalhos apresentados na Décima Segunda Edição do WESAAC (Workshop Escola de Sistemas de Agentes, seus Ambientes e Aplicações). O WESAAC 2018 foi realizado na cidade de Fortaleza/CE, com o apoio da Universidade de Fortaleza (UNIFOR), entre os dias 02 e 04 de Maio de 2018.

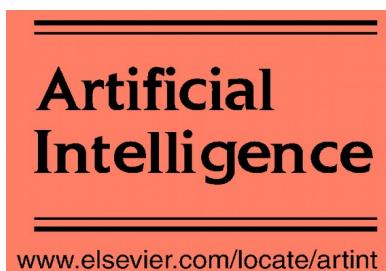
Com o objetivo de integrar pesquisadores e estudantes de todos os níveis na área de Agentes e Sistemas de Agentes e divulgar as atividades dos diversos grupos de pesquisa do Brasil, bem como promover o intercâmbio de conhecimentos e experiências, a 12^a edição do WESAAC contou com Oficinas e Palestras proferidas por professores e pesquisadores com excelência em suas áreas de conhecimento, com apresentações de Trabalhos Completos e Resumos Estendidos e também com apresentação de Pôsteres.

Gostaríamos de agradecer aos pesquisadores convidados, Mario Paolucci e José Vasco Furtado que abrilhantaram o evento com suas palestras. Também agradecemos a Saulo Popov Zambiasi e Jaime Sichman 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 pesquisa. Gostaríamos de agradecer aos membros do comitê de programa, aos revisores adicionais pelo criterioso trabalho desenvolvido. Para esta edição, tivemos o suporte financeiro do Artificial Intelligence Journal (www.elsevier/locate/artint), da Mob Telecom (www.mobtelecom.com.br), da Alura (www.alura.com.br) e da Casa do Código (www.casadocodigo.com.br), que foram fundamentais para a viabilidade do evento, bem como o apoio das instituições organizadoras, Universidade Estadual de Fortaleza, Universidade Federal de Santa Catarina e Universidade de Fortaleza.

Fortaleza, 2018.
Mariela Inés Cortés
Jerusa Marchi
Eurico Vasconcelos

Organização e Patrocínio



Funding Opportunities for Promoting
AI Research



alura



Organização

Organização Geral

Mariela Inés Cortés (UECE)

Coordenação do Comitê de Programa

Jerusa Marchi (UFSC)

Organização Local

Eurico Vasconcelos (UNIFOR)

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)
Rejane Frozza (UNISC)

Comitê de Programa

Adamatti, Diana	FURG (Brasil)
Alencar, Fernanda	UFPE (Brasil)
Araújo, Ricardo Matsumura	UFPel (Brasil)
Bajo Pérez, Javier	Universidad Politécnica de Madrid (Espanha)
Balsa, João	Univ. Lisboa (Portugal)
Barbosa, Raquel	FURG (Brasil)
Batista de Almeida Rego, Joilson	UFAL (Brasil)
Bazzan, Ana L. C.	UFRGS (Brasil)
Billa, Cleo	FURG (Brasil)
Boissier, Olivier	EMSE (França)
Bordini, Rafael	PUCRS (Brasil)
Brandão, Anarosa Alves Franco	USP (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)
Coutinho, Luciano	UFMA (Brasil)
de La Prieta Pintado, Fernando	Universidad de Salamanca (Espanha)
Dimuro, Graçaliz	FURG (Brasil)
Ductor, Sylvain	UECE (Brasil)
Ferreira Jr., Paulo R.	UFPel (Brasil)
Frozza, Rejane	UNISC (Brasil)
Giménez Lugo, Gustavo	UTFPR (Brasil)
Gonçalves, Eder Mateus Nunes	FURG (Brasil)
Hubner, Jomi Fred	UFSC (Brasil)
Leite, João	Nova LINCS (Portugal)
Lemke, Ana Paula	IFRS (Brasil)
Lorenzi, Fabiana	ULBRA (Brasil)
Marchi, Jerusa	UFSC (Brasil)
Moreira, Álvaro	UFRGS (Brasil)
Nardin, Luis Gustavo	University of Idaho (USA)
Nunes, Ingrid	UFRGS (Brasil)
Okuyama, Fabio	IFRS (Brasil)
Porto Fernández, Marcial	UECE (Brasil)
Ricci, Alessandro	University of Bologna (Italia)
Rocha Costa, Antônio Carlos	UFRGS (Brasil)
Rosa, Paulo	IME-RJ (Brasil)
Sanchotene de Aguiar, Marilton	UFPel (Brasil)
Sichman, Jaime Simão	USP (Brasil)
Silva, João Luis	UCS (Brasil)
Silveira, Francisca R. de Vasconcelos	IFCE (Brasil)
Silveira, Ricardo Azambuja	UFSC (Brasil)
Simari, Guillermo Ricardo	Universidad Nacional del Sur (Argentina)
Tacla, Cesar A.	UTFPR (Brasil)
Tedesco, Patrícia	UFPE (Brasil)
Torres da Silva, Viviane	UFF (Brasil)
Trigo, Paulo	ISEL (Portugal)
Vasconcelos, Wamberto	University of Aberdeen (UK)
Vieira, Renata	PUCRS (Brasil)
Villarrubia González, Gabriel	Universidad de Salamanca (Espanha)
Webber, Carine	UCS (Brasil)
Werneck, Vera	UFRJ (Brasil)
Zatelli, Maicon	UFSC (Brasil)

SUMÁRIO

ARTIGOS COMPLETOS

1.	Elementary Economic Systems in Material Agent Societies Antonio Carlos Rocha Costa	12-24
2.	Uma Proposta de Mapeamento entre i* Normativo e o Diagrama NorMAS-ML de Normas Emmanuel Sávio Silva Freire e Thiago Gomes De Souza	25-36
3.	An Agent-Based Model for Normative Hierarchical Organizations considering Goal Decomposition Process Emmanuel Sávio Silva Freire, Robert Marinho Da Rocha Júnior, Mariela Inés Cortés e Gustavo Augusto De Lima Campos	37-48
4.	Um modelo de Multiagentes Normativos para Descrição de Personagens de Jogos Sérios Jonathan Morris Samara, Cesar Augusto Tacla, Sebastião Ribeiro Junior, Rafael Bee, Klaus de Geus, Sergio Scheer e Bruno Soares de Souza	49-60
5.	Composição de Técnicas de Inteligência Artificial em uma Arquitetura Multinível para Emergência de Comportamentos em Agentes Cooperativos João Rogério Vieira Neto e Jerusa Marchi	61-72
6.	Impacto da Confiança em Simulação Baseada em Agentes para Cadeias de Suprimentos André Jalbut e Jaime Sichman	73-84
7.	Agente Inteligente Embarcado baseado em Sistemas Ciberfísicos no contexto da Indústria 4.0 Braian Konzgen Maciel, Eder Mateus Nunes Gonçalves, Gabriel Machado Balota e Mario Ricardo Marques Junior	85-96
8.	Avaliação da testabilidade do modelo organizacional Moise baseado em Redes de Petri Bruno Rodrigues e Eder Mateus Gonçalves	97-108
9.	Modelo de Fusão de Dados com Incerteza para Consciência Situacional Munyque Mittelmann, Jerusa Marchi e Aldo von Wangenheim	109-120
10.	Análise de Dilemas em Natyasastra: Um Sistema de Resolução Dramática para Autorregulação de Processos de Trocas Sociais em Sistemas Multiagentes Nelson de Faria Traversi, Diana Francisca Adamatti, Graçaliz Dimuro e Renata Gomes Wotter	121-130

11.	Segregação Racial e Valorização do Território: uma modelagem baseada em agentes Nelson de Faria Traversi, Gustavo Lima e Diana Francisca Adamatti	131-142
12.	Aplicando árvore de decisão para a localização de padrões no modelo Vágner de Oliveira Gabriel, Diana Adamatti, Cleo Billa e Fabiana Lorenzi	143-153
13.	Using agent-based artificial financial market to analyse market manipulation Luiza Biasoto, Everton Reis e Jaime Sichman	154-166
14.	Simulador de NoCs modelado como um SMA Nelson de Faria Traversi, Gustavo Lima, Diana Francisca Adamatti, Odorico Mendizabal, Cristina Meinhardt e Eduardo Brião	167-178
15.	Transporte de Agentes Cognitivos entre SMA Distintos Inspirado nos Princípios das Relações Ecológicas Vinicius Souza de Jesus, Fabian Cesar Pereira Brandão Manoel, Carlos Pantoja e Jose Viterbo	179-187
16.	Um modelo para simulação de formação de membranas e micelas Nilzair Agostinho e Diana Francisca Adamatti	188-199
17.	Um Agente Autônomo Concorrente para o Manipulador Robótico Jaco Kinova Edi M. M. De Araujo e Augusto Loureiro Costa	200-211

ARTIGOS CURTOS

1.	Especificação de Casos de Uso para a Modelagem de Requisitos de Sistemas Multiagente Normativos Emmanuel Sávio Silva Freire e Patrícia Maria Barbosa Ferreira	212-217
2.	Epidemiologia: Compreensão da Dinâmica das Doenças e Emergência através da Modelagem Bianca Marques, Douglas Félix e Diana Adamatti	218-225
3.	Developing a smart parking solution based on a Holonic Multiagent System using JaCaMo Framework Lucas Fernando Souza de Castro, André Pinz Borges e Gleifer Alves	226-231
4.	Intelligent agent applied to digital games Patrick Rodrigues e João Luís Tavares Da Silva	232-237
5.	Explorando a Comunicação entre Sistemas Multi-Agentes Embarcados em Ambientes Inteligentes para IoT: Uma Proposta de Laboratório Palloma Da Silva Machado Nunes, Vinicius Souza de Jesus, Fabian Cesar Pereira Brandão Manoel, Leandro Marques Samyn, Carlos Eduardo Pantoja, Igor Mendes de Almeida e Thiago Corrêa Picanço	238-243
6.	Proposta de modelo de autorrecuperação de sistemas de distribuição de energia elétrica utilizando sistemas multiagente Italo Ramon Campos e Filipe Saraiva	244-249
7.	Modelo de Raciocínio e Protocolo de Negociação para um Estacionamento Inteligente com Mecanismo de Negociação Descentralizado Felipe Felix Ducheiko e Gleifer Alves	250-256

RESUMOS ESTENDIDOS

1.	Modelagem do Ritmo Circadiano utilizando Sistemas Multiagente: um estudo de caso da influência da dor Angélica Theis Dos Santos, Catia Maria Dos Santos Machado e Diana Francisca Adamatti	257-260
2.	Sistemas Multiagente como Ferramenta para Desenvolvimento do Pensamento Computacional Vinicius Martins e Diana Adamatti	261-264
3.	Interpretação de imagens como tarefa cognitiva: uma abordagem inicial com redes neurais de aprendizagem profunda Cleverson de Souza Carneiro, Nicolas Mansur Beleski e Gustavo Alberto Giménez-Lugo.	265-268

Elementary Economic Systems in Material Agent Societies

Antônio Carlos da Rocha Costa¹

¹Programa 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 the elementary economic systems of material agent societies, on the bases of the notions of (individual and group) elementary economic behavior, elementary economic exchange and elementary economic process. The equilibrium of an elementary economic system is defined in terms of the equilibrium of the set of group elementary economic processes that constitute such system. A case study illustrates the proposed concepts.

Resumo. Este artigo caracteriza formalmente os sistemas econômicos elementares das sociedades de agentes materiais, tendo por base as noções de comportamento econômico elementar, troca econômica elementar e processo econômico elementar (tanto individuais como grupais). O equilíbrio de um sistema econômico elementar é definido em termos do equilíbrio do conjunto dos processos econômicos elementares grupais que constitui tal sistema. Um estudo de caso ilustra os conceitos propostos.

1. Introduction

The concept of *energy systems* of material agent societies, introduced in [Costa 2017b], assumes that energy is produced by, and distributed to, the material agents of material agent societies in the form of *energy objects*. That formalization, however, does not specify any particular way producers and consumers can interact to do that.

In the present paper, we focus on one particular mode of interactive distribution of energy objects, namely, *elementary economic processes*. We say that the set of elementary economic processes of a material agent society constitutes the *elementary economic system* of that society.

To allow for elementary economic processes to occur in a material agent society, we require that the material agents be capable of producing other types of objects, besides energy objects, so that different kinds of objects can be exchanged for each other, in the *elementary economic exchanges* that constitute those elementary economic processes.

For simplicity, in this paper we consider just one other type of objects, besides energy objects. We call them *chips*. Elementary economic processes are assumed, thus, to involve just the exchange of *energy objects* for *chips*, and vice-versa.

2. Material Agent Societies and their Energy Systems

We first summarize the concepts of *material agent*, *material agent society* and *energy systems* of material agent societies as these concepts were introduced in [Costa 2017b].

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. We call *material agent society* any agent society whose agents are all material agents.

We consider here only material agent societies organized around an *energy system*, i.e., a particular social subsystem capable of *producing* and *distributing* energy objects within the society in a way that guarantees its *energetic autonomy* [Costa 2017b]. We call *energy producer* any material agent that participates in the operation of that energy system. As in [Costa 2017b], we assume that all producers are *energetically self-sufficient*, that is, are capable of producing all the energy they need for their own operation. The other material agents of the society, are said to be *energy consumers*.

3. Elementary Economic Behaviors and Exchanges

We take George Homans' model of social behaviors and exchanges [Homans 1961] as the operational model on the basis of which we define *elementary economic behaviors and exchanges*. This way, our *elementary economical model* builds on the assumption that any material agent *mag* is capable of performing the following two types of actions:

- *deliver an object to another agent* at a time t , denoted $\text{deliver}^t(\textit{mag}, \textit{obj}, \textit{mag}')$, where *mag* is the deliverer, and *mag'* is the receiver, of *obj*;
- *receive an object from another agent* at a time t , denoted $\text{receive}^t(\textit{mag}, \textit{obj}, \textit{mag}')$, where *mag* is the receiver, and *mag'* is the deliverer, of *obj*.

Besides, we assume that each material agent is capable, for each type of existent object, to account for the *sum total of objects* of that type that it has sent or received at each time.

Homans' model is heavily based on Burrhus Skinner's notion of *operant conditioning* [Skinner 1991]. Thus, a central feature of our model is a *reinforcement process* between *sequences* of actions of receiving objects from another material agent (*receive* actions), and *sequences* of actions of delivering objects of other types to that material agent, in return (*deliver* actions). More precisely, reinforcement processes operate between the *temporal rates* of those two sequences, that is, between the *temporal rate* at which the receiving operations are performed, and the *temporal rate* at which the delivering operations are performed.

4. Individual Elementary Economic Behaviors and Exchanges Formally Defined

4.1. Terms for Denoting Individual Elementary Economic Behaviors

The following *variables* range over the following sets (variables may appear in expressions with various types of decorations):

- *mag*, ranging over the set **Mag** of material agents;
- *beh*, ranging over the set **Beh** of behaviors of material agents;
- *afval*, ranging over the set **AffVal** of affective values of material agents;
- *exch*, ranging over the set **Exch** of exchanges between material agents ;
- *obj*, ranging over the set **Obj** of objects that may be exchanged by material agents;
- p, q, \dots , ranging over the set **Sit** of social situations in a material agent society;
- t and τ , ranging over the set $T = \{0, 1, \dots\}$ of time instants.

Behaviors and affective values may be assigned to material agents:

- $mag.beh$: behavior of a definite material agent;
- $mag.afval$: affective value of a definite material agent.

The *rates of performance* of a behavior, or of an exchange process, is denoted by the operator “ $\langle \rangle$ ” applied to that behavior or exchange process: $\langle beh \rangle$ or $\langle exch \rangle$.

4.2. Basic formulas

Basic formulas are of one of the following forms:

- (a) statements about *rates of performances* of behaviors or exchanges, or about *tendencies of variation* in such rates, thus:
 - $\langle beh \rangle \top$: behavior has a *high* rate of performance;
 - $\langle beh \rangle \bowtie$: behavior has a *medium* rate of performance;
 - $\langle beh \rangle \perp$: behavior has a *low* rate of performance;
 - $\langle beh \rangle \uparrow$: behavior has an increasing rate of performance;
 - $\langle beh \rangle \downarrow$: behavior has a decreasing rate of performance;
- (b) statements about affective assessments of rates of performances of actions, namely, $\langle mag_1.beh \rangle_p [mag_2.afval]_q$, whose meaning is that the *rate* of the behavior beh of the material agent mag_1 has a *defined value*, in situation p , and that such rate of behavior is evaluated with affective value $afval$ by the material agent mag_2 , in situation q . If $p = q$, one may write: $\langle mag.beh \rangle [mag.afval]_p$. All the components of the formula are optional, except for the $\langle beh \rangle$ component.

The following illustrate some of the formal variations of such formulas:

- definiteness of the rate of a behavior :
 - any formula in the set $\{\langle beh \rangle, \langle beh_i \rangle\}_{i \in \mathbb{N}}$, meaning that beh (or $\langle beh_i \rangle$) has a defined rate of performance;
- situated definiteness of the rate of a behavior:
 - any formula in the set $\{\langle beh \rangle_p, \langle beh_i \rangle_p\}_{i,p \in \mathbb{N}}$, meaning that beh (or beh_i) has a defined rate of performance in situation p ;
- situated definiteness of the rate of behavior of a particular material agent:
 - any formula in the set $\{\langle mag_i.beh_j \rangle_p\}_{i,n,p \in \mathbb{N}}$, meaning that beh_j of mag_i has a defined rate of performance in situation p ;
- affective evaluation by an implicitly specified material agent:
 - any formula in the set $\{\langle mag.beh \rangle [afval], \langle mag_i.beh_j \rangle [afval_k]\}_{i,j,k \in \mathbb{N}}$, meaning that: (a) $\langle mag.beh \rangle$ (or $\langle mag_i.beh_j \rangle$) has a defined rate of performance, and (b) it is evaluated with affective value $afval$ (or $afval_k$) by an implicitly specified evaluator material agent.

4.3. Compound formulas

- (a) The *propositional composition* of formulas is given by the usual *propositional operators* ($\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, \dots$), assumed to have their usual precedence degrees.

The meaning of $\neg \langle beh \rangle$ is that it is false that variable $\langle beh \rangle$ has a defined value, i.e., it is false that the behavior beh is not in execution. The formula $\neg \langle beh \rangle [afval]$ should be read as $\neg (\langle beh \rangle [afval])$, that is, the operator of affective evaluation has precedence over \neg . The same happens with the other propositional operators, so that to express the simultaneous affective evaluation of two behaviors one should write $(\langle beh_i \rangle \wedge \langle beh_j \rangle) [afval]$.

(b) Functional dependence compound formulas express *monotonic qualitative functional dependences* between two rates of performance of behaviors. The *functional dependence* compound formulas have the basic forms:

$$\langle mag_i.beh_i \rangle \nearrow \langle mag_j.beh_j \rangle \quad \text{and} \quad \langle mag_i.beh_i \rangle \searrow \langle mag_j.beh_j \rangle$$

The meaning of $\langle mag_i.beh_i \rangle \nearrow \langle mag_j.beh_j \rangle$ is that: (a) the rates of performance of beh_i of material agent mag_i , and of beh_j of mag_j , are both defined, and (b) the rate of beh_j *monotonically increases* with the rate of beh_i . Analogously, the meaning of $\langle mag_i.beh_i \rangle \searrow \langle mag_j.beh_j \rangle$ is that the functional dependence between the two rates of behaviors is *monotonically decreasing*.

Possible decorations of the *functional dependence* compound formulas with situation indexes are as follows: $\langle mag_i.beh_i \rangle \nearrow_p \langle mag_j.beh_j \rangle$ and $\langle mag_i.beh_i \rangle \searrow_p \langle mag_j.beh_j \rangle$. Notice that, since the *functional dependence* has to be determined in a single situation, the *situation indexes* of the two behaviors have to refer to the same situation.

There is no notion of *functional dependence* between affective evaluations of defined values of variables, given the assumption of *evaluation autonomy* of the material agents. So the only way *functional dependence* compound formulas may be decorated with affective evaluation operations is by indicating that the *whole functional dependence* is affectively evaluated, in the general forms: $[\langle mag_i.beh_i \rangle \nearrow_p \langle mag_j.beh_j \rangle] [mag_k.afval_k]$ and $[\langle mag_i.beh_i \rangle \searrow_p \langle mag_j.beh_j \rangle] [mag_k.afval_k]$, meaning that the indicated *functional dependences* are evaluated by material agent mag_k with affective value $afval_k$.

(c) The *ordering* of rates of performance of behaviors, and of results of affective evaluations of rates of behaviors, is given by the partial order relation " \leq ". We assume that " \leq " operates uniformly on *affective values* and on *rates of performances of behaviors*, thus: $afval_1 \leq afval_2$ and $\langle beh_1 \rangle \leq \langle beh_2 \rangle$.

For any behavior beh it holds that its *lowest* rate of performance (given by \perp), *highest* rate of performance (given by \top), and *medium* rate of performance (given by \bowtie) are ordered by the partial ordering relation " \leq " as: $\perp \leq \bowtie \leq \top$. For the affective values resulting from the affective evaluation of rates of behavior, we extend the use of " \leq ":

$$\begin{aligned} \langle mag_1.beh_1 \rangle [mag_2.afval_2]_p \leq \langle mag_3.a_3 \rangle [mag_4.afval_4]_p &\Leftrightarrow \\ \langle mag_1.beh_1 \rangle [mag_2.afval_2]_p \wedge \langle mag_3.a_3 \rangle [mag_4.afval_4]_p &\wedge afval_2 \leq afval_4 \end{aligned}$$

4.4. The Individual Behavioral Conditioning Rules

The fundamental relation between behaviors, according to Skinner [Skinner 1991] is the relation of *operant conditioning*, that is, the relation according to which an spontaneous behavior (an *operant*) is led to be performed in the context of, and in accordance to, a certain *conditioner*. The conditioning occurs because the *conditioner* behavior is assumed to represent a combination of events that is relevant for the internal functioning of the *operant agent*, so that it is capable of influencing (positively or negatively) the way that agent performs the *operant* behavior that is being conditioned.

The *operant conditioning rules* that we introduce below, attempt to formally capture some of the fundamental operant conditioning propositions that Homans took from

Skinner's behavioral psychology [Skinner 1991], and that he adopted as the behavioristic foundation of his social exchange theory [Homans 1961].

In Homans' conceptualization, the *operant* is a behavior that some agent directs toward another agent, and the *conditioner* is some reaction that the latter directs toward the former. In our *economic interpretation* of Homans' concept of social behavior, we limit both the *operant* and the *conditioner* behaviors to be sequences of actions of *delivering* and *receiving* objects (more precisely, *energy objects* and *chips*).

Skinner's basic rule, fully adopted by Homans, is: *an operant is conditioned by a conditioner whenever it happens that an increase or decrease in the rate of performance of the conditioner impacts the rate of performance of the operant behavior.*

The usual interpretation of this proposition is that the operant agent evaluates (positively or negatively) the variation (increase or decrease) in the rate of performance of the conditioner behavior, and reacts by varying accordingly the rate of performance of the operant behavior. We formally present this interpretation by the set of formulas:

$$\begin{aligned} \langle mag_1.receive(mag_1, obj_1, mag_2) \uparrow \rangle [mag_1.+] &\rightsquigarrow \langle mag_1.deliver(mag_1, obj_2, mag_2) \uparrow \rangle \\ \langle mag_1.receive(mag_1, obj_1, mag_2) \downarrow \rangle [mag_1.+] &\rightsquigarrow \langle mag_1.deliver(mag_1, obj_2, mag_2) \uparrow \rangle \\ \langle mag_1.receive(mag_1, obj_1, mag_2) \uparrow \rangle [mag_1.-] &\rightsquigarrow \langle mag_1.deliver(mag_1, obj_2, mag_2) \downarrow \rangle \\ \langle mag_1.receive(mag_1, obj_1, mag_2) \uparrow \rangle [mag_1.-] &\rightsquigarrow \langle mag_1.deliver(mag_1, obj_2, mag_2) \downarrow \rangle \end{aligned}$$

where the " \rightsquigarrow " symbol denotes the impact relation between the (positive or negative) evaluation, by mag_1 , of the variation (increase or decrease) in the rate of reception of the *conditioner object* obj_1 from mag_2 , and the consequent variation (increase or decrease) in the rate of performance of the delivery of the object obj_1 to mag_2 , by mag_1 .

We may use the following abbreviations for the increase or decrease in the rate of the conditioned operant:

$$\begin{aligned} \langle mag_1.receive(mag_1, obj_2, mag_2) \rangle \rightarrow\!\!\!\rightarrow^+ \langle mag_1.deliver(mag_1, obj_1, mag_2) \rangle \\ \langle mag_1.receive(mag_1, obj_2, mag_2) \rangle \rightarrow\!\!\!\rightarrow^- \langle mag_1.deliver(mag_1, obj_1, mag_2) \rangle \end{aligned}$$

The following are the *conditioning rules for individual elementary economic behaviors* that we establish on the basis of Homans' interpretation of Skinner's basic rule. They are characterized by the type of the conditioning acting on the operant behavior ("+" or "-") and by the level of activity of the conditioner (" \top ", " \bowtie " or " \perp "):

1. Rule $BC_{(+,\bowtie)}$: If mag_1 evaluates *positively* (+) the reception of obj_2 from mag_2 , in return to mag_1 delivering obj_1 to mag_2 , and mag_2 delivers obj_2 to mag_1 at a *regular temporal rate* (\bowtie) then: the more frequently mag_2 delivers obj_2 to mag_1 , the more frequently will mag_1 deliver obj_1 to mag_2 . Formally:

$$\frac{\langle mag_1.receive(mag_1, obj_2, mag_2) \rangle \rightarrow\!\!\!\rightarrow^+ \langle mag_1.deliver(mag_1, obj_1, mag_2) \rangle \\ \quad \langle mag_2.deliver(mag_2, obj_2, mag_1) \rangle \bowtie \\ \hline \langle mag_2.deliver(mag_2, obj_2, mag_1) \rangle \nearrow \langle mag_1.deliver(mag_1, obj_1, mag_2) \rangle}{\langle mag_2.deliver(mag_2, obj_2, mag_1) \rangle \nearrow \langle mag_1.deliver(mag_1, obj_1, mag_2) \rangle} BC_{(+,\bowtie)}$$

2. Rule $BC_{(+,\top)}$: If mag_1 evaluates *positively* (+) the reception of obj_2 from mag_2 , in return to mag_1 delivering obj_1 to mag_2 , and mag_2 delivers obj_2 to mag_1 at a *very high temporal rate* (\top) then: the more frequently mag_2 delivers obj_2 to mag_1 , the less frequently will mag_1 deliver obj_1 to mag_2 . Formally:

$$\frac{\langle mag_1.receive(mag_1, obj_2, mag_2) \rangle \rightarrow^+ \langle mag_1.deliver(mag_1, obj_1, mag_2) \rangle \\ \quad \langle mag_2.deliver(mag_2, obj_2, mag_1) \rangle \top}{\langle mag_2.deliver(mag_2, obj, mag_1) \rangle \downarrow \langle mag_1.deliver(mag_1, beh, mag_2) \rangle} BC_{(+,\top)}$$

3. Rule $BC_{(+,\perp)}$: If mag_1 evaluates *positively* (+) the reception of obj_2 from mag_2 , in return to mag_1 delivering obj_1 to mag_2 , and mag_2 delivers obj_2 to mag_1 at a *very low temporal rate* (\perp) then: the more frequently mag_2 delivers obj_2 to mag_1 , the more frequently will mag_1 deliver obj_1 to mag_2 . Formally:

$$\frac{\langle mag_1.receive(mag_1, obj_2, mag_2) \rangle \rightarrow^+ \langle mag_1.deliver(mag_1, obj_1, mag_2) \rangle \\ \quad \langle mag_2.deliver(mag_2, obj, mag_1) \rangle \perp}{\langle mag_2.deliver(mag_2, obj, mag_1) \rangle \nearrow \langle mag_1.deliver(mag_1, beh, mag_2) \rangle} BC_{(+,\perp)}$$

4. Rule BC_- : If mag_1 evaluates *negatively* (-) the reception of obj_2 from mag_2 , in return to mag_1 delivering obj_1 to mag_2 then: the more frequently mag_2 delivers obj_2 to mag_1 , the less frequently will mag_1 deliver obj_1 to mag_2 . Formally:

$$\frac{\langle mag_1.receive(mag_1, obj_2, mag_2) \rangle \rightarrow^- \langle mag_1.deliver(mag_1, obj_1, mag_2) \rangle}{\langle mag_2.deliver(mag_2, obj, mag_1) \rangle \downarrow \langle mag_1.deliver(mag_1, beh, mag_2) \rangle} BC_-$$

5. Rule BC_\vee : Any *increase* in frequency of a particular behavior beh_1 by mag entails by that very fact a *decrease* in the frequency of any alternative behavior beh_2 that mag can perform. Formally:

$$\frac{beh_i, beh_j \in beh[mag]}{\langle mag.beh_i \rangle \downarrow_{i \neq j} \langle mag.beh_j \rangle} BC_\vee$$

where $beh[mag]$ denotes the set of behaviors that mag is capable of performing.

4.5. Terms for Denoting Individual Elementary Economic Exchanges

An *individual elementary economic exchange* is a pair of operant conditionings acting between two material agents, mag_1 and mag_2 , so that the delivery of the object obj_1 , by mag_1 to mag_2 acts as a *conditioner* to the delivery of the object obj_2 by mag_2 to mag_1 , and vice-versa.

We say that an *individual elementary economic exchange* between mag_1 and mag_2 , involving the exchange of the objects obj_1 and obj_2 between them, is performed in the *doubly positive mode of exchange* if and only if:

1. obj_1 is an *energy object* and obj_2 is a *chip*, or vice-versa;
2. $\langle mag_1.receive(mag_1, obj_2, mag_2) \rangle \rightarrow^+ \langle mag_1.deliver(mag_1, obj_1, mag_2) \rangle$
3. $\langle mag_2.receive(mag_2, obj_1, mag_1) \rangle \rightarrow^+ \langle mag_2.deliver(mag_2, obj_2, mag_1) \rangle$

We denote such a doubly positively reinforced *individual elementary economic exchange* by: $\langle mag_1/obj_1 \rangle \rightarrow^+ \langle mag_2/obj_2 \rangle$.

The *temporal evolution* of any individual elementary economic exchange may lead the material agents involved in it to change the way they evaluate the conditioners

they receive from their partners. Variations in the rate of reception of conditioners may also lead the operant agent to change such evaluations. As a result, the following are also *modes of exchange* that may occur during the performance of an individual elementary economic exchange:

- *mixed modes of exchange*:
 $\langle mag_1/obj_1 \rangle \dashv\ddash^+ \langle mag_2/obj_2 \rangle$ and $\langle mag_1/obj_1 \rangle \dashv\ddash^- \langle mag_2/obj_2 \rangle$;
- *doubly negative mode of exchange*:
 $\langle mag_1/obj_1 \rangle \dashv\ddash^- \langle mag_2/obj_2 \rangle$.

4.6. Equilibrium of Individual Elementary Economic Exchanges

We say that the individual elementary economic exchange:

$$ie2exch = \langle mag_1/obj_1 \rangle \dashv\ddash^+ \langle mag_2/obj_2 \rangle$$

which is performed at a time τ , is *equilibrated* at that time if and only if it holds, for each of the behaviors of mag_1 and mag_2 , that they are performed at a *medium rate* at that time, that is: $\langle mag_1/obj_1 \rangle^\tau \bowtie$ and $\langle mag_2/obj_2 \rangle^\tau \bowtie$.

We denote by $equil[ie2exch]^\tau$ the fact that the individual elementary economic exchange $ie2exch$ is equilibrated at the time τ .

5. Individual Elementary Economic Processes

5.1. Sequential Composition of Operant Conditionings of Individual Elementary Economic Behaviors

Operant conditionings between individual elementary economic behaviors can be *sequentially composed*, in the sense that a behavior beh_2 that is conditioned by a behavior beh_1 can, itself, condition a behavior beh_3 . In such situation, one can say that behavior beh_1 also conditions behavior beh_3 .

The following are the rules defining such *sequential compositions*, considering the possible positive or negative conditionings that the behaviors may have on each other.

$$\begin{array}{c} \frac{\langle beh_1 \rangle \rightarrow\rightarrow^+ \langle beh_2 \rangle \quad \langle beh_2 \rangle \rightarrow\rightarrow^+ \langle beh_3 \rangle}{\langle beh_1 \rangle \rightarrow\rightarrow^+ \langle beh_3 \rangle} BehSC_1 \\ \hline \frac{\langle beh_1 \rangle \rightarrow\rightarrow^- \langle beh_2 \rangle \quad \langle beh_2 \rangle \rightarrow\rightarrow^- \langle beh_3 \rangle}{\langle beh_1 \rangle \rightarrow\rightarrow^+ \langle beh_3 \rangle} BehSC_2 \\ \hline \frac{\langle beh_1 \rangle \rightarrow\rightarrow^+ \langle beh_2 \rangle \quad \langle beh_2 \rangle \rightarrow\rightarrow^- \langle beh_3 \rangle}{\langle beh_1 \rangle \rightarrow\rightarrow^- \langle beh_3 \rangle} BehSC_3 \\ \hline \frac{\langle beh_1 \rangle \rightarrow\rightarrow^- \langle beh_2 \rangle \quad \langle beh_2 \rangle \rightarrow\rightarrow^+ \langle beh_3 \rangle}{\langle beh_1 \rangle \rightarrow\rightarrow^- \langle beh_3 \rangle} BehSC_4 \end{array}$$

5.2. Sequential Composition of Individual Elementary Economic Exchanges

Individual elementary economic exchanges can also be sequentially composed. The following is a *sample* of the rules defining such sequential compositions:

$$\begin{array}{c}
 \frac{\langle mag_1/obj_1 \rangle + \nrightarrow^+ \langle mag_2/obj_2 \rangle \quad \langle mag_2/obj_2 \rangle + \nrightarrow^+ \langle mag_3/obj_3 \rangle}{\langle mag_1/obj_1 \rangle + \nrightarrow^+ \langle mag_3/obj_3 \rangle} \\
 \\
 \frac{\langle mag_1/obj_1 \rangle - \nrightarrow^- \langle mag_2/obj_2 \rangle \quad \langle mag_2/obj_2 \rangle - \nrightarrow^- \langle mag_3/obj_3 \rangle}{\langle mag_1/obj_1 \rangle + \nrightarrow^+ \langle mag_3/obj_3 \rangle} \\
 \\
 \frac{\langle mag_1/obj_1 \rangle - \nrightarrow^+ \langle mag_2/obj_2 \rangle \quad \langle mag_2/obj_2 \rangle - \nrightarrow^+ \langle mag_3/obj_3 \rangle}{\langle mag_1/obj_1 \rangle + \nrightarrow^+ \langle mag_3/obj_3 \rangle} \\
 \\
 \frac{\langle mag_1/obj_1 \rangle - \nrightarrow^+ \langle mag_2/obj_2 \rangle \quad \langle mag_2/obj_2 \rangle + \nrightarrow^- \langle mag_3/obj_3 \rangle}{\langle mag_1/obj_1 \rangle - \nrightarrow^- \langle mag_3/obj_3 \rangle}
 \end{array}$$

5.3. Individual Elementary Economic Processes Formally Defined

We call *individual elementary economic process* any finite, non-null, time-indexed sequence of compositions of individual elementary economic exchanges, of the form:

$$ie2iproc^t = \langle mag_1/obj_1 \rangle^t \underset{c_{2,1}}{\nrightarrow^{c_{1,2}}} \langle mag_2/obj_2 \rangle^t \underset{c_{3,2}}{\nrightarrow^{c_{2,3}}} \dots \underset{c_{n,n-1}}{\nrightarrow^{c_{n-1,n}}} \langle mag_n/obj_n \rangle^t$$

where each $\langle mag_i/obj_i \rangle^t \underset{c_{j,i}}{\nrightarrow^{c_{i,j}}} \langle mag_j/obj_j \rangle^t$ is an individual elementary economic exchange that may occur at the time t , the *conditioning signs* are $c_{i,j} \in \{+, -\}$, there is *no cycle* in the process (that is, $mag_i/obj_i \neq mag_j/obj_j$, for every i and j), and $n \geq 2$ is the *length* of the individual elementary economic process.

5.4. Equilibrium of Individual Elementary Economic Processes

The equilibrium of an *individual elementary economic process*, at a given time, is given by the fact that each of the exchanges that compose it are equilibrated at that time. That is, the elementary economic process:

$$ie2iproc^t = \langle mag_1/obj_1 \rangle^t \underset{c_{2,1}}{\nrightarrow^{c_{1,2}}} \langle mag_2/obj_2 \rangle^t \underset{c_{3,2}}{\nrightarrow^{c_{2,3}}} \dots \underset{c_{n,n-1}}{\nrightarrow^{c_{n-1,n}}} \langle mag_n/obj_n \rangle^t$$

is equilibrated at the time τ (with $0 \leq \tau \leq t$) if and only if each of the individual elementary economic exchanges that constitute it is equilibrated, that is:

$$\forall \langle mag_i/obj_i \rangle^t \in ie2iproc^t (\text{equil}[\langle mag_i/obj_i \rangle]^{\tau})$$

We denote by $\text{equil}[ie2iproc]^{\tau}$ the fact that $ie2iproc^t$ is equilibrated at the time τ .

6. Group Elementary Economic Behaviors and Exchanges

6.1. Group Elementary Economic Behaviors and Exchanges Formally Defined

The general form of individual elementary economic exchange introduced in Sect. 5, namely, $\langle mag_1/obj_1 \rangle \underset{c_{2,1}}{\nrightarrow^{c_{1,2}}} \langle mag_2/obj_2 \rangle$, is constituted by a combination of elementary economic behaviors performed by two *individual* material agents (mag_1 and mag_2).

We can naturally extend the concept of elementary economic behavior to *groups* of material agents, considering that each such group operates as a *unity*, collectively delivering objects to another group, and collectively being reinforced by the reception of objects from that other group.

Let Mag and Mag' be two such *groups*, and Obj the *set of objects* that they may exchange between them. The operation $deliver^t(Mag, Obj, Mag')$ indicates, then, the delivery of that set of objects by one group to the other, and similarly for the operation $receive^t(Mag, Obj, Mag')$.

A *group elementary economic exchange* between Mag_1 and Mag_2 , exchanging sets of objects Obj_1 and Obj_2 , is denoted by: $\langle Mag_1/Obj_1 \rangle_{c_{2,1}} \rightleftarrows^{c_{1,2}} \langle Mag_2/Obj_2 \rangle$.

6.2. Equilibrium of Group Elementary Economic Exchanges

We say that the group elementary economic exchange:

$$ge2exch = \langle Mag_1/Obj_1 \rangle \rightleftarrows^+ \langle Mag_2/Obj_2 \rangle$$

which is performed at a time τ , is *equilibrated* at that time if and only if it holds, for each of the behaviors of Mag_1 and Mag_2 , that they are performed at a *medium rate* at that time, that is, $\langle Mag_1/obj_1 \rangle^\tau \bowtie$ and $\langle Mag_2/obj_2 \rangle^\tau \bowtie$.

We denote by $equil[ge2exch]^\tau$ the fact that the group elementary economic exchange $ge2exch^t$ is equilibrated at the time τ .

7. Group Elementary Economic Processes

7.1. Group Elementary Economic Processes Formally Defined

A *group elementary economic process* is a time-indexed sequence of group elementary economic exchanges, of the form:

$$ge2proc^t = \langle Mag_1/Obj_1 \rangle^t \rightleftarrows^{c_{1,2}} \langle Mag_2/Obj_2 \rangle^t \rightleftarrows^{c_{2,3}} \dots \rightleftarrows^{c_{n-1,n}} \langle Mag_n/Obj_n \rangle^t$$

7.2. Equilibrium of Group Elementary Economic Processes

We say that the *group elementary economic process*:

$$ge2proc^t = \langle Mag_1/Obj_1 \rangle^t \rightleftarrows^{c_{1,2}} \langle Mag_2/Obj_2 \rangle^t \rightleftarrows^{c_{2,3}} \dots \rightleftarrows^{c_{n-1,n}} \langle Mag_n/Obj_n \rangle^t$$

is *equilibrated* at the time τ (with $0 \leq \tau \leq t$), if and only if each of the group elementary economic exchanges that constitute it is equilibrated, that is:

$$\forall \langle Mag_i/Obj_i \rangle^t \in ge2iproc^t (equil[\langle Mag_i/Obj_i \rangle]^\tau)$$

We denote by $equil[ge2proc]^\tau$ the fact that $ge2proc^t$ is equilibrated at the time τ .

8. Elementary Economic Systems

8.1. Elementary Economic Systems Formally Defined

Formally, we define:

Definition 8.1 *The elementary economic system EES of a material agent society MAgSoc whose population of material agents is Pop, is a time-indexed structure:*

$$EES_{MAgSoc}^t = (Groups^t, Obs^t, GE2Beh^t, GE2Exch^t, GE2Proc^t)$$

where¹:

¹ $\wp(X)$ denotes the powerset of the set X .

- $Groups^t \subseteq \wp(Pop)$ is the family of groups of material agents of Pop that can participate in the elementary economic group processes of $GE2Proc^t$;
- $Objs^t \subseteq \wp(Obj)$ is the family of sets of objects that the groups of material agents of $Groups^t$ can exchange between them during the performance of the group elementary economic processes of $GE2Proc^t$;
- $GE2Beh^t$ is the set of group elementary economic behaviors that the groups of material agents of $Groups^t$ can perform during the performance of the group elementary economic processes of $GE2Proc^t$;
- $E2GExch^t$ is the set of group elementary economic exchanges that the groups of material agents of $Groups^t$ can perform during the performance of the group elementary economic processes of $GE2Proc^t$;
- $GE2Proc^t$ is the set of group elementary economic processes that the groups of material agents of $Groups^t$ can perform in $MAgSoc$ at the time t .

8.2. Equilibrium of Elementary Economic Systems

We say that the elementary economic system:

$$EES_{MAgSoc}^t = (Groups^t, Objs^t, GE2Beh^t, GE2Exch^t, GE2Proc^t)$$

is *equilibrated* at the time τ (with $0 \leq \tau \leq t$) if and only if each of the group elementary economic process that constitute it is equilibrated at that time, that is:

$$\forall ge2proc \in GE2Proc^t (\text{equil}[ge2proc]^\tau)$$

We denote by $\text{equil}[EES_{MAgSoc}]^\tau$ the fact that the elementary economic system EES_{MAgSoc}^τ is equilibrated at the time τ .

9. Ecosystems: A Case Study in Elementary Economical Analysis

We provide here elements for the *elementary economical analysis of ecosystems*. More precisely, we consider: (a) *ecosystems* as material agent societies; and (b) the *interactional structure* that constitute the operational part of the organizational structure of an ecosystem as an *elementary economical system*.

The casting of ecosystems as agent societies was introduced in [Costa 2017a]. The proposal for considering the *interactional structure* of an ecosystem as an *elementary economical system* is presented here for the first time.

Figure 1 pictures a general model for the interactional structure of ecosystems. We base on it the elementary economical analysis that follows.

As in [Costa 2017a], we state that an *ecosystem* is a material agent society $EcoSys^t = (Pop^t, Org^t, MEnv^t)$ where, for each time t :

- Pop^t is the *populational structure* of the society, with:
- Org^t is the *organizational structure* of the society;
- $MEnv^t$ is the society's *material environment*.

so that, regarding Fig. 1, we have:

- $Pop^t = (Plants^t, Herbivores^t, Carnivores^t, Detrivores^t)$;
- $Org^t = (O_2^t, CO_2^t, Water^t, Nutrients^t, EatenBy^t, SolarEnergy^t)$

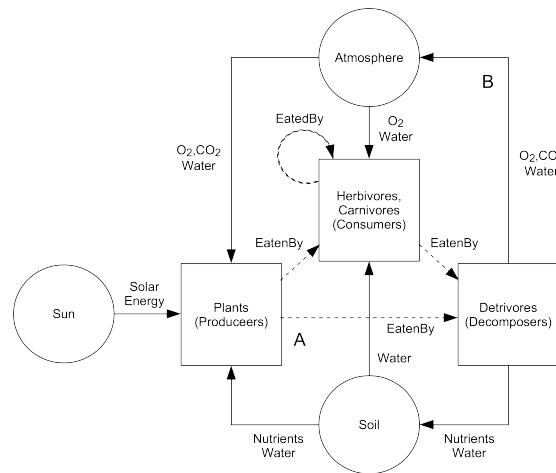


Figure 1. The general interactional structure of ecosystems, based on [Pidwirny 2009].

- $MEnv^t = (Atmosphere^t, Soil^t, Sun^t)$

with:

- $O_2^t \subseteq (Detrivores^t \times Atmosphere^t) \cup (Atmosphere^t \times Plants^t) \cup (Atmosphere^t \times Herbivores^t) \cup (Atmosphere^t \times Carnivores^t))$
- $CO_2 \subseteq (Detrivores^t \times Atmosphere^t) \cup (Atmosphere^t \times Plants^t)$
- $Water^t \subseteq (Detrivores^t \times Atmosphere^t) \cup (Atmosphere^t \times Plants^t) \cup (Atmosphere^t \times Herbivores^t) \cup (Atmosphere^t \times Carnivores^t)) \cup (Detrivores^t \times Soil^t) \cup (Soil^t \times Plants^t)) \cup (Soil^t \times Herbivores^t) \cup (Soil^t \times Carnivores^t))$
- $Nutrients^t \subseteq (Detrivores^t \times Soil^t) \cup (Soil^t \times Plants^t)$
- $EatenBy^t \subseteq (Herbivores^t \times Detrivores^t) \cup (Carnivores^t \times Detrivores^t) \cup (Plants^t \times Detrivores^t) \cup (Plants^t \times Herbivores^t) \cup (Plants^t \times Carnivores^t) \cup (Carnivores^t \times Carnivores^t)$
- $SolarEnergy^t \subseteq Sun^t \times Plant^t$

Clearly:

- *Detrivores, Carnivores, Herbivores and Plants* constitute the *populational groups* of the ecosystems;
- the *Sun* and the *Soil* are the source of all *energy* consumed by the populational groups of ecosystems;
- the *Atmosphere* and the *Soil* operate as *transportation means* for the O_2 , CO_2 , *Water* and *Nutrients* that the populational groups exchange between them;
- individuals of a populational group eating individuals of another populational group is also a means for the former group to receive energy from the latter one.

Also: (a) the set of exchanges operating in the ecosystem constitute a complex network, and (b) there are more than two types of objects being exchanged in the system.

So, from the economical point of view, such system surpasses the conditions stipulated by the concepts introduced in the present paper, and an elementary economical analysis of ecosystems can only partially account for it. The following are two of the *elementary economical processes* present in Fig. 1::

$$1. \text{ } ge2proc_A = \langle \text{Detrivores/Soil/Water} \rangle \xrightarrow{+} \langle \text{Plants/Plants} \rangle$$

where $\langle \text{Detrivores/Soil/Water} \rangle$ denotes that *Detrivores* deliver objects of type *Water* through the *Soil* to *Plants*, and $\langle \text{Plants/Plants} \rangle$ denotes that *Plants* deliver objects of type *Plants* directly to (be eaten by) *Detrivores*;

$$2. \text{ } ge2proc_B = \langle \text{Detrivores/Atmosphere/O}_2 \rangle \xrightarrow{+} \langle \text{Carnivores/Carnivores} \rangle$$

which has a reading analogous to the previous one.

So, we can tentatively construe the interactional structure of ecosystems as elementary economic systems as:

$$EES_{M\text{AgSoc}}^t = (Groups^t, Objs^t, GE2Beh^t, GE2Exch^t, GE2Proc^t)$$

with:

- $Groups^t = Pop^t$, that is, all of the *populational groups* of the ecosystem participate in the elementary economic system;
- $O_2, Water \in Objs^t$, since they are *objects* exchanged by the populational groups;
- $deliver(\text{Detrivores}, \text{Water}, \text{Carnivores})$,
 $deliver(\text{Carnivores}, \text{Carnivores}, \text{Detrivores}) \in GE2Beh^t$
since they are *group elementary economic behaviors*;
- $\langle \text{Detrivores}^t.deliver(\text{Detrivores}, \text{Water}, \text{Carnivores}) \rangle$,
 $\xrightarrow{+} \langle \text{Carnivores}^t.deliver(\text{Carnivores}, \text{Carnivores}, \text{Detrivores}) \rangle \in GE2Exch^t$
since it is a *group elementary economic exchange*;
- $ge2proc_A, ge2proc_B \in GE2Proc^t$, since they are *group elementary economic processes*.

10. Conclusion

The concepts introduced in the present paper seem to be the most elementary economical concepts that can fit the basic features of energy systems of material agent societies, as they were defined in [Costa 2017b].

The definition of *full-fledged economic systems* for material agent societies is reserved for future work. It will require the lifting of many of the constraints that are intrinsic to the elementary systems. For instance: (a) that operant conditioning be *direct* (i.e., due to the interacting partner), allowing for *indirect* operant conditionings, through chains of indirect partners; (b) the *absence of cycles* in economic processes, leading to complex networks of interlinked economic processes; (c) admission of *only two types of objects* in economic exchanges.

We remark that we have no knowledge of other reports dealing with the concepts introduced here, analogously to what happened in [Costa 2017b].

References

- 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.
- Homans, G. (1961). *Social Behavior – Its Elementary Forms*. Harcourt, Brace & World, New York.
- Pidwirny, M. (2009). *Fundamentals of Physical Geography* (2nd Ed.). PhysicalGeography.net. Available online at <http://www.physicalgeography.net>.
- Skinner, B. F. (1991). *The Behavior of Organisms: an experimental analysis*. Copley Publishing Group, Acton.

Uma Proposta de Mapeamento entre i* Normativo e o Diagrama NorMAS-ML de Normas*

Emmanuel Sávio Silva Freire¹, Thiago Gomes de Souza²

¹Departamento de Ensino – Instituto Federal do Ceará (IFCE/Campus Morada Nova)
Av. Prefeito Raimundo José Rabelo, 2717 – 62.940-000 – Morada Nova – CE – Brasil

²Departamento de Ensino – Instituto Federal do Ceará (IFCE/Campus Iguatu)
Rua Deoclécio Lima Verde, s/n – Areias II – 63.500-000 – Iguatu – CE – Brasil
savio.freire@ifce.edu.br, {savio.essf, thiagogsouza91}@gmail.com

Abstract. Normative multi-agent systems have been used to develop complex systems. Thus, several modeling languages, such as NorMAS-ML, were defined in order to support the modeling phase, but a few number of works have been addressed the gathering and analysis of requirements phase. In this sense, normative i* framework was defined in order to make possible that requirement analysts can model stakeholders as actors and theirs intentions as goals. Therefore, this paper has the objective the integration between gathering and analysis of requirements and modeling phases, proposing a mapping between normative i* and NorMAS-ML. In addition, a modeling example was done in order to illustrate this mapping.

Resumo. Os sistemas multiagente normativos são utilizados para o desenvolvimento de sistemas complexos. Logo, várias linguagens de modelagem, dentre elas NorMAS-ML, foram definidas para dar suporte à fase de modelagem, porém poucos trabalhos têm abordado a fase de levantamento e análise de requisitos. Neste sentido, o framework i* normativo foi definido para possibilitar que analistas de requisitos possam modelar os stakeholders como atores e suas intenções como objetivos. Assim, esse artigo tem como objetivo a integração entre as fases de levantamento e análise de requisitos e a fase de modelagem, propondo um mapeamento entre i* normativo e NorMAS-ML. Além disso, um exemplo de modelagem foi realizado para ilustrar este mapeamento.

1. Introdução

Os sistemas multiagente normativos (SMANs) são formados por um conjunto de agentes inteligentes que interagem entre si para alcançar os seus objetivos, verificando as normas definidas nesses sistemas para identificar quais as ações que são permitidas, obrigadas ou proibidas de serem executadas [Boella, van der Torre e Verhagen 2006] [Silva, Braga e Figueiredo 2010]. Assim, linguagens de modelagem e de implementação foram propostas para auxiliar o desenvolvimento desses sistemas. Dentre elas, destaca-se a linguagem *Normative Multiagent System Modeling Language* (NorMAS-ML) [Freire et al. 2012] que possibilita a modelagem das entidades encontradas nesses sistemas junto com as normas. Contudo, essa linguagem não dá suporte a fase de levantamento e análise de requisitos.

No processo de desenvolvimento de software, o levantamento de requisitos comprehensíveis por todas as partes envolvidas no desenvolvimento do sistema (clientes,

* Este trabalho faz parte do Projeto número 5925 intitulado “Integração de Requisitos Organizacionais à Modelagem de Sistemas Multiagente Normativos” do Edital IFCE/PIBIC Jr/2017. O autor Thiago Souza agradece pelo suporte financeiro recebido pelo IFCE para a realização deste trabalho.

analistas, desenvolvedores etc.) é um fator básico e extremamente importante, evitando falhas no entendimento do problema a ser solucionado [Franceto 2005]. Assim, a fase de levantamento e análise de requisitos propicia um melhor entendimento das reais necessidades dos usuários do sistema [Sommerville 2011]. Logo, é possível detalhar e priorizar os requisitos permitindo que aqueles essenciais ao sistema possam ser implementados inicialmente. Consequentemente, o usuário tem uma maior probabilidade de receber um sistema que supra as suas necessidades.

Neste sentido, o *framework i** (iStar) [Yu 2002] foi proposto para auxiliar a fase de requisitos por meio de um modelo conceitual, permitindo que os analistas de requisitos possam modelar os *stakeholders* como atores e suas intenções como objetivos. Assim, consegue-se obter uma compreensão sobre os relacionamentos da organização (que define os requisitos, ou seja, uma empresa) e sobre as razões envolvidas nos processos de decisão [Yu 2002]. Entretanto, *i** não permitia a modelagem de requisitos organizacionais para os SMAN, pois não abordava os elementos normativos nem a interação deles com as entidades de um SMAN. Por isso, Siena et al. (2008) propuseram a extensão de *i**, chamada de *i** normativo, para permitir a modelagem de aspectos normativos, indicando as entidades que são reguladas pelas normas.

Considerando que *i** normativo permite a modelagem de requisitos organizacionais para SMAN e NorMAS-ML permite a modelagem das entidades típicas encontradas nesses sistemas, este artigo propõe um mapeamento entre *i** normativo e NorMAS-ML com o intuito de identificar quais as semelhanças e as diferenças conceituais existentes entre eles. Assim, espera-se possibilitar a integração entre a fase de levantamento e análise de requisitos e a fase de modelagem para o desenvolvimento de SMANs, proporcionando uma visão mais completa do sistema. Vale ressaltar que a relevância dessa integração foi discutida em um trabalho inicial [Freire e Cortés, 2016].

Este artigo está organizado como segue. A seção 2 apresenta a linguagem NorMAS-ML, enquanto a seção 3 discorre sobre *i** normativo. O mapeamento entre *i** normativo e NorMAS-ML é detalhado na seção 4. Na seção 5, um exemplo de modelagem é apresentado. As discussões acerca do mapeamento e a sua comparação com os trabalhos relacionados são descritos na seção 6. Finalmente, as conclusões e os trabalhos futuros são abordados na seção 7.

2. NorMAS-ML

A *Normative Multiagent System Modeling Language* (NorMAS-ML) [Freire et al. 2012] é uma linguagem de modelagem que permite o *design* de SMANs. Ela surgiu por meio da extensão da MAS-ML para possibilitar a modelagem das entidades típicas encontradas nos SMANs (ambiente, organização, agente, papel de agente, objeto e papel de objeto) em conjunto com os seguintes elementos que compõem uma norma [Silva, Braga e Figueiredo 2010]: (i) **Conceitos deônticos**, descrevem as restrições de comportamento para os agentes na forma de obrigações, permissões e proibições; (ii) **Entidades Envolvidas**, que são as entidades que possuem o seu comportamento restrinido por uma norma; (iii) **Ações**, que são as ações que estão sendo reguladas pela norma; (iv) **Restrições de Ativação**, que indicam a restrição ou um conjunto de restrições que ativam uma norma; (v) **Sanções**, que são punições/recompensas que uma entidade recebe quando uma norma é violada/seguida, respectivamente; e (vi) **Contexto**, indica a área de atuação (um ambiente ou uma organização) de uma norma.

Adicionalmente, Freire et al. (2012) definiram um novo diagrama estático que permite a modelagem de normas. Um exemplo do diagrama de normas pode ser encontrado

na seção 5. As entidades que podem participar do diagrama são: (i) classe de norma, (ii) classe, (iii) classe de papel de objeto, (iv) classe de agente, (v) classe de papel de agente, (vi) classe de organização e (vii) classe de ambiente. Além disso, dos relacionamentos definidos na UML, também podem ser utilizados os seguintes relacionamentos no diagrama: (i) *ownership*, (ii) *play*, (iii) *inhabit*, (iv) *context*, (v) *restrict*, e (vi) *sanction*.

3. Framework i* Normativo

O framework *iStar* normativo ou *i** normativo [Siena et al. 2008] é uma extensão do framework *i** [Yu 2002] que possibilita a representação de leis e normas utilizando o modelo de ator, objetivo e dependências existente em *i**. Para tanto, os autores consideram as seguintes propriedades das normas relevantes para a aquisição de requisitos: (i) **a relação de compromisso normativo** (*the normative commitment relation*), essa relação é formada por quem criou a norma (*source*), por aquele que é endereçado pela norma (*addressee*) e pelo artefato que será restringido pela norma (*legal artefact*); (ii) **o esquema de uma norma** (*the schema of the norm*), que está relacionado com o padrão de comportamento que a norma impõe para o *addressee*, como por exemplo, formas de atuação, objetivos e princípios a serem adotados; e (iii) **as intenções de conformidade** (*the compliance intentions*), que estão associadas ao impacto real da norma sobre os *stakeholders*, identificando o que eles colocaram em ação para cumprir a norma.

Por meio da análise das três propriedades incluídas na extensão, a Figura 1 apresenta o metamodelo¹ de *i** normativo. Adicionalmente, os modelos que permitem que os engenheiros de requisitos possam modelar os *stakeholders* como atores e suas intenções como objetivos também foram alterados por conta da extensão. Esses modelos são: (i) o modelo de dependência estratégica (*Strategic Dependency – SD*), que foca na relação entre os agentes organizacionais, e (ii) o modelo estratégico de razão (*Strategic Rational – SR*), que possibilita a modelagem detalhada das relações intencionais internas de cada agente organizacional. Um exemplo de modelagem utilizando o framework *i** normativo pode ser encontrado na seção 5.

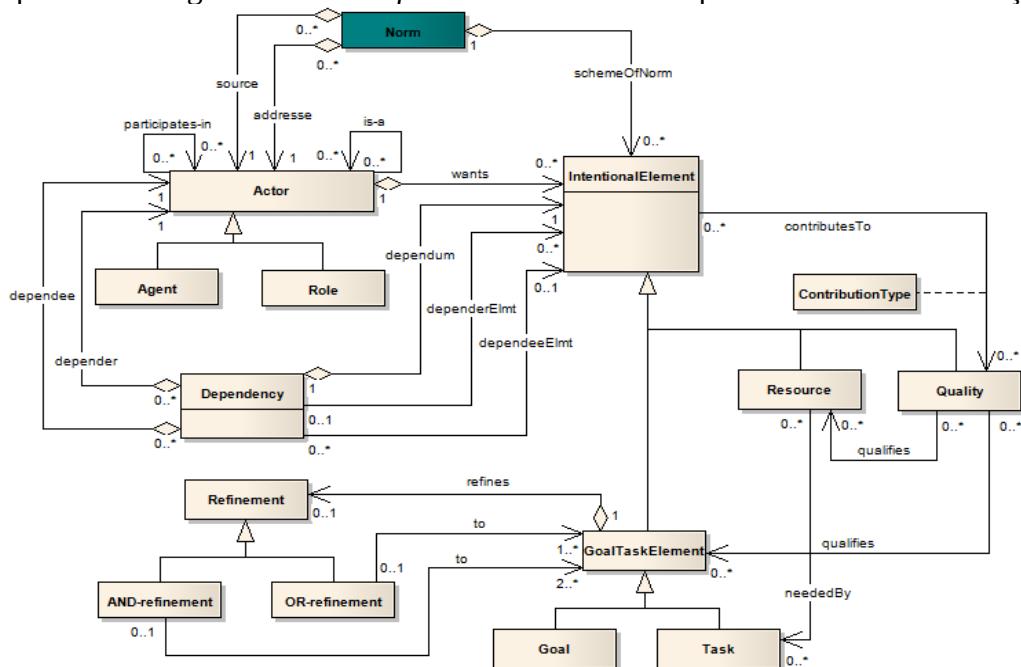


Figura 1. Metamodelo do framework *i** normativo (adaptado pelos autores)

1 Vale ressaltar que esse metamodelo foi proposto pelos autores do presente artigo por meio da comparação entre o metamodelo original de *i** e as propriedades das normas apresentadas na seção 3.

4. Mapeamento entre i* Normativo e o Diagrama NorMAS-ML de Normas

Nesta seção, são apresentadas as diretrizes de mapeamento entre i* normativo e o diagrama NorMAS-ML de normas associado a exemplos para cada diretriz.

4.1. Diretrizes de Mapeamento

As diretrizes de mapeamento determinam as regras pelas quais a associação entre o modelo i* Normativo e o diagrama NorMAS-ML de normas é realizada corretamente. Para tanto, foram modificadas as diretrizes propostas por Alencar et al. (2003) e Melo et al. (2015) que possibilitavam o mapeamento entre i* e o diagrama UML de classes.

O objetivo das diretrizes é manter a consistência e o rastreamento entre o sistema a ser desenvolvido e os objetivos da organização, considerando as normas que regulam o comportamento das entidades que compõem os SMANs. Desta forma, aumenta-se as chances do sistema que está sendo desenvolvido atender os objetivos da organização de uma forma mais precisa. Além disso, pode-se relacionar os artefatos gerados na fase de análise (modelo i* normativo) com os gerados na fase de projeto (diagrama NorMAS-ML de normas).

As diretrizes foram subdivididas nos seguintes sete grupos: (i) D1, mapeamento de atores; (ii) D2, mapeamento de tarefas; (iii) D3, mapeamento dos recursos; (iv) D4, mapeamento de objetivos e objetivos soft; (v) D5, mapeamento de relação entre meios-fim (*means-end*); (vi) D6, mapeamento de decomposição de tarefas; e (vii) D7, mapeamento de normas. O Quadro 1 apresenta as diretrizes de mapeamento entre i* Normativo e o diagrama NorMAS-ML de normas.

Quadro 1. Diretrizes de mapeamento entre i* normativo e o diagrama NorMAS-ML de normas

Tipo	Número	i* Normativo	Diagrama NorMAS-ML de Normas
D1	D1.1	Papel	Classe de Papel de Agente
	D1.2	Agente	Classe de Agente
	D1.3	Posição	Classes de Organização
	D1.4	Relacionamento IS-PART-OF entre papel e posição	Relacionamento ownership entre Classes de Papel de Agente e Organização
	D1.5	Relacionamento IS-PART-OF entre posições	Relacionamento aggregation entre Classes de Posição
	D1.6	Relacionamento IS-PART-OF entre agente e posição	Não apresenta correspondente
	D1.7	Relacionamento IS-A	Generalização/Especialização de classes de Papel de Agente ou entre Classes de Agentes ou de Organizações
	D1.8	Relacionamento i* OCCUPIES entre um agente e uma posição	Não apresenta correspondente
	D1.9	Relacionamento i* COVERS entre uma posição e um papel	Relacionamento ownership entre Classes de Papel de Agente e Organização
	D1.10	Relacionamento i* PLAYS entre um agente e um papel	Relacionamento play entre Classes de Agente e de Papel de Agente
D2	D2.1	Tarefa no modelo SD	Atributo agent action com visibilidade pública ou resource, caso seja uma norma.
	D2.2	Tarefas no modelo SR	Atributo agent action com visibilidade privada ou resource, caso seja uma norma.
D3	D3.1	Recursos definidos no modelo SD	Classe de Objeto, se essa dependência tem a característica de um objeto

	D3.1	Recursos definidos no modelo SD	Atributo agent action com visibilidade privada
	D3.2	Recursos (sub-recursos) definidos no modelo SR	Atributo agent action com visibilidade privada na classe que representa o ator em que o sub-recurso pertence (se esse sub-recurso não pode ser entendido como um objeto).
	D3.2	Recursos (sub-recursos) definidos no modelo SR	Classe de Objeto, se esse sub-recurso tem a característica de um objeto
D4	D4.1	Objetivo (ou Objetivo soft) definido no modelo SD	Atributo goal com visibilidade pública na classe da entidade correspondente.
	D4.2	Objetivo (ou Objetivo soft) definido no modelo SR	Atributo goal com visibilidade pública na classe da entidade correspondente
D5	D5.1	Objetivos (ou Objetivos soft) - Objetivos (ou Objetivos soft)	Representada por regras (podem ser descritas em OCL) da entidade correspondente
	D5.2	Objetivos (ou Objetivos soft)-Tarefa, Recurso-Tarefa	Representada por regras (podem ser descritas em OCL) da entidade correspondente
	D5.3	Tarefa-Tarefa	Representada por regras (podem ser descritas em OCL) da entidade correspondente
D6	D6.1	Decomposição de tarefa no modelo SR	Representada pelas pré e pós-condições (podem ser expressas em OCL) da agent action correspondente
D7	D7.1	Fonte da Norma	Relacionamento context entre Classes de Norma e de Papel de Agente
	D7.2	Norma	Classe de Norma de Obrigação
	D7.3	Endereçado pela Norma	Relacionamento restrict entre Classes de Norma e de Agente ou entre Classes de Norma e de Papel de Agente ou entre Classes de Norma e de Organização.
	D7.4	Prescrições da Norma	Segue as diretrizes D2, D3, D4, D5 e D6.

4.2. Detalhamento das Diretrizes

Nesta seção, as diretrizes de mapeamento serão detalhadas por meio da discussão conceitual dos elementos mapeados e da apresentação da transformação das entidades de i* normativo para seu correspondente no diagrama NorMAS-ML de normas. De acordo com o Quadro 1, a **diretriz D1.1** menciona que um papel deve ser transformado em uma classe de Papel de Agente. Em i* normativo, um papel é uma caracterização abstrata do comportamento de um ator dentro de algum contexto. Esse conceito é equivalente em NorMAS-ML, pois um papel de agente é responsável por restringir e orientar o comportamento de agentes quando executam o papel. A Figura 2 (a) apresenta essa transformação.

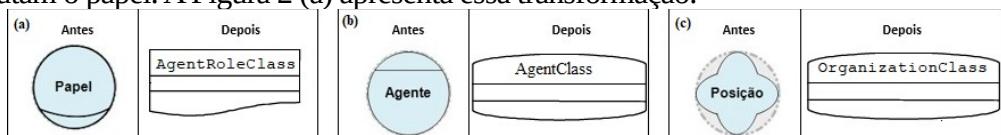


Figura 2. Transformação de atores de i* normativo para entidades de NorMAS-ML

Considerando o conceito de agente em i* normativo é equivalente em NorMAS-ML, a **diretriz D1.2** indica que um agente em i* normativo deve ser transformado em uma classe de agente no diagrama de normas (ver Figura 2 (b)). Em relação à entidade posição, no framework i* normativo, representa um conjunto de papéis normalmente executado por agentes. Esse conceito é encontrado na entidade Organização de NorMAS-ML. Uma organização é capaz de definir papéis que devem ser exercidos pelos agentes.

Logo, possui um conjunto de papéis. Consequentemente, a **diretriz D1.3** determina que uma posição em i* normativo seja transformado em uma classe de organização no diagrama de normas. A Figura 2 (c) ilustra essa transformação.

A **diretriz D1.4** determina que o relacionamento IS-PART-OF entre uma posição e um papel seja mapeado para o relacionamento *ownership* de NorMAS-ML. Ele indica que um proprietário (organização) conheça todos os seus membros (papéis de agente) (Ver Figura 3). Além disso, a **diretriz D1.5** indica que o relacionamento IS-PART-OF entre posições seja transformado no relacionamento aggregation de NorMAS-ML. Vale ressaltar que, em NorMAS-ML, não há correspondente para o mapeamento do relacionamento IS-PART-OF entre uma posição e um agente (**diretriz D1.6**), pois esse relacionamento é feito de forma implícita quando um agente executa um papel de agente pertencente a uma organização.

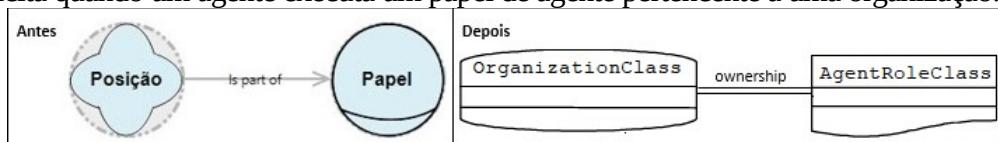


Figura 3. Transformação do relacionamento IS-PART-OF em ownership de NorMAS-ML

A **diretriz D1.7** indica que o relacionamento IS-A deve ser transformado em um relacionamento *specialization* em NorMAS-ML, pois indica relação de herança entre as entidades participantes. Esses relacionamentos possuem a mesma semântica. A Figura 4 apresenta essa transformação. Em relação à **diretriz D1.8**, não é possível representar o relacionamento OCCUPIES em NorMAS-ML pelo mesmo motivo apresentado para a diretriz D1.5.

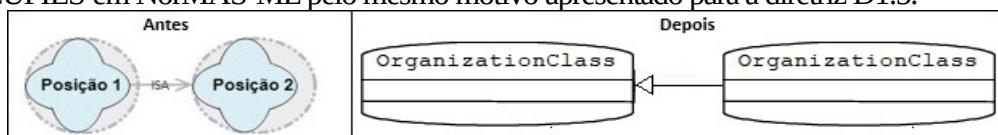


Figura 4. Transformação do relacionamento IS-A em specialization de NorMAS-ML

Considerando que o relacionamento COVERS de i* normativo indica a relação entre uma posição e seus papéis, a **diretriz D1.9** estabelece que esse relacionamento seja mapeado para o relacionamento *ownership* entre classes de papel de agente e de organização. A figura 5 apresenta esse mapeamento. Em relação ao relacionamento PLAYS de i* normativo, a **diretriz D1.10** determina que ele seja transformado no relacionamento *play* de NorMAS-ML, pois apresentam a mesma semântica. A figura 6 ilustra essa transformação.

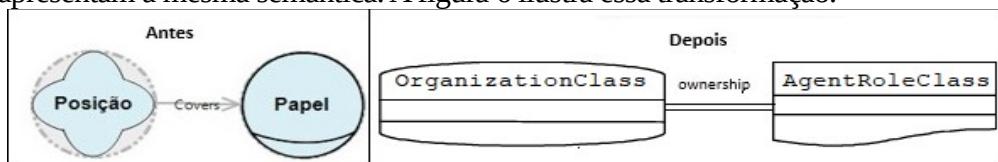


Figura 5. Transformação do relacionamento COVERS em ownership de NorMAS-ML

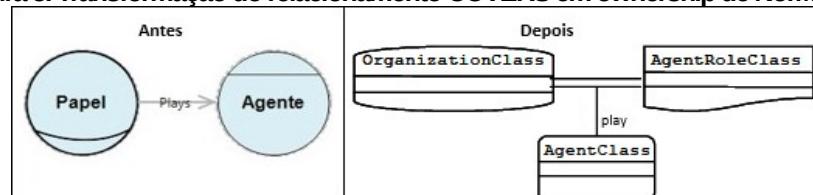


Figura 6. Transformação do relacionamento PLAYS em play de NorMAS-ML

É importante ressaltar que os relacionamentos IS-PART-OF e COVERS de i* normativo foram mapeados para o relacionamento *ownership* de NorMAS-ML, conforme as diretrizes D1.4 e D1.9, respectivamente. Assim, por mais que IS-PART-OF e COVERS tenham semânticas diferentes no modelo i*, o relacionamento *ownership* garante tanto a

associação entre atores (organização e papéis de agente) demonstrando que um ator faz parte de outro ator e a associação entre uma posição (organização) e seus papéis de agente.

A **diretriz D2.1** indica que as tarefas definidas no modelo SD devem ser convertidas em agent actions com visibilidade pública na classe do ator que relaciona com essas tarefas. Além disso, a **diretriz D2.2** estabelece que as tarefas definidas no modelo SR também devem se tornar agent actions, porém com visibilidade privada (Ver Figura 7). É importante ressaltar que um agent action de NorMAS-ML corresponde a uma ação que pode ser executada por um agente, por uma organização ou por um papel de agente.

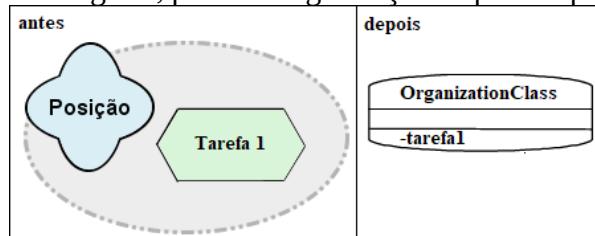


Figura 7. Transformação de tarefas em agent actions de NorMAS-ML

A **diretriz D3.1** afirma que os recursos feitos no modelo SD devem ser transformados em um objeto em NorMAS-ML, caso essa dependência tenha característica de objeto. Entretanto, se a dependência não tiver essa característica, o recurso deve ser transformado em um agent action com visibilidade privada na classe do ator. O mesmo ocorre para os recursos do modelo SR conforme a **diretriz D3.2**. A Figura 8 apresenta esses mapeamentos.

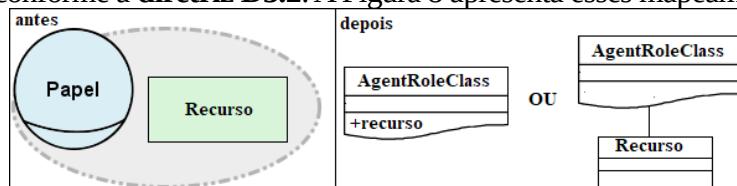


Figura 8. Transformação de recursos em objetos ou agent actions de NorMAS-ML

Em relação aos objetivos no modelo SD e SR, as **diretrizes D4.1 e D4.2** indicam que devem ser transformados em *goals* em NorMAS-ML com visibilidade pública na classe do ator correspondente. Um atributo *goal* representa o objetivo de um agente, de uma organização ou de um papel de agente em NorMAS-ML. A Figura 9 ilustra essa transformação.



Figura 9. Transformação de recursos em objetos ou agent actions de NorMAS-ML

A **diretriz D5** refere-se aos meios-fim (*means-end*), que sugere que pode-se ter formas alternativas para alcançar um objetivo. Assim, as **diretrizes D5.1, D5.2 e D5.3** indicam que deve ser criada uma nota contendo a instrução correspondente, vinculada à classe do ator. A **diretriz D6** está relacionada com a decomposição de tarefas. Essa decomposição indica o que deve ser feito para a realização de uma determinada tarefa. Assim, por meio da **diretriz D6.1**, a decomposição deve também ter uma nota vinculada à classe do ator. A Figura 10 apresenta essas transformações de *means-end* e de decomposição de tarefas.

A **diretriz D7.1** indica que a fonte da norma deve ser transformada para o relacionamento *context* de NorMAS-ML entre classes de norma e de papel de agente. A Figura 11 apresenta esse mapeamento. Além disso, a **diretriz D7.2** estabelece que a norma no *framework i** normativo deve ser transformada na entidade norma de NorMAS-ML com conceito deôntico de obrigação

(Ver Figura 11). Foi utilizado esse conceito deôntico pois em i* normativo não é possível a diferenciação entre normas de obrigação, permissão e proibição no seu metamodelo.

A **diretriz D7.3** afirma que a entidade endereça pela norma deve ser representada pelo relacionamento *restrict* de NorMAS-ML (Ver Figura 11). Assim, pode-se vincular classes de agente, de papel de agente ou de organização a uma classe de norma. Já a **diretriz D7.4** estabelece que as prescrições da norma devem seguir as diretrizes D2, D3, D4, D5 e D6 (Ver Figura 10).

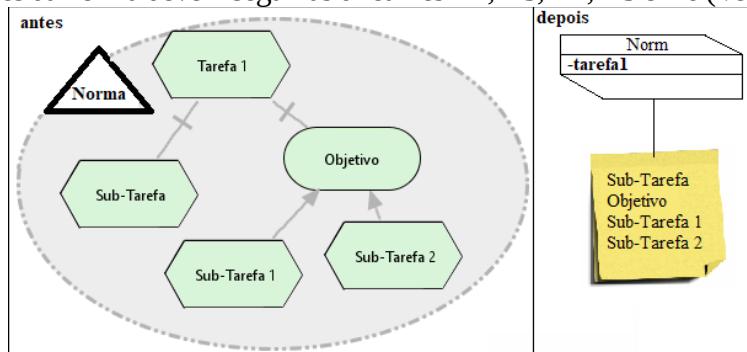


Figura 10. Transformação de means-end e de decomposição de tarefas em NorMAS-ML

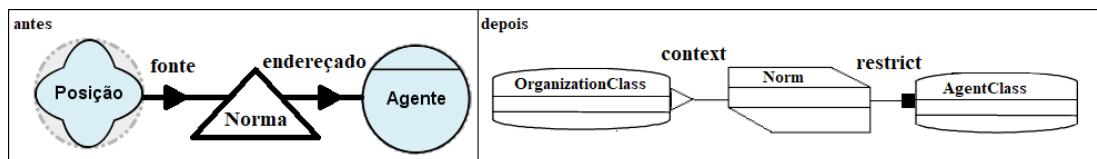


Figura 11. Transformação da norma e seus elementos em NorMAS-ML

5. Exemplo de Modelagem

Para ilustrar o uso das diretrizes propostas na seção anterior, o exemplo de modelagem escolhido foi um sistema *SmartCD* definido por Alencar et al. (2003). Vale ressaltar que esse sistema foi previamente modelado por Melo et al. (2015) para ilustrar as diretrizes de mapeamento entre i* e o diagrama UML de classes. Consequentemente, esse sistema não possui normas associadas as suas entidades. Logo, foram definidas e incluídas normas no sistema *SmartCD* para possibilitar a utilização das diretrizes de mapeamento entre i* normativo e o diagrama NorMAS-ML de normas.

O sistema *SmartCD* permite a realização de pedidos por meio de um catálogo online. Inicialmente, Melo et al. (2015) definiram três posições: *Client*, *Store Management* e *SmartCD*. A última posição corresponde ao sistema a ser desenvolvido que deve tratar as encomendas, notificar as chegadas de CD e fornecer o catálogo. A Figura 12 apresenta o modelo *Strategic Dependence* (SD) definido por esses autores. Adicionalmente, para a posição *SmartCD*, foram definidas as seguintes normas: (i) N1: A posição *SmartCD* definiu que a posição *Internet Sales* deve registrar o cliente antes de efetuar a venda de CDs; (ii) N2: A posição *SmartCD* definiu que a posição *Inventory* deve atualizar o estoque sempre que houver uma nova venda; e (iii) N3: A posição *Internet Sales* definiu que o ator *Office Boy* possui um prazo de duas horas, obrigatoriamente, para a entrega de CDs na cidade que este ator se encontra.

A Figura 13 apresenta o modelo *Strategic Rational* (SR) considerando a posição *SmartCD* em conjunto com as normas apresentadas acima. Esse diagrama foi baseado no exemplo de modelagem apresentado em Melo et al. (2015). Após a transformação de modelos, foi gerado o diagrama NorMAS-ML de normas apresentado na Figura 14, considerado as diretrizes de mapeamento definidas no Quadro 1. No diagrama, os papéis *CDReservation* e *CDDelivery*, o agente *OfficeBoy*, e as posições *Inventory*, *SmartCD*, *Financial*, *InternetSales* e *Client* foram transformados em classes de papel de agente, de agente e de organização, respectivamente,

segundo as **diretrizes D1.1, D1.2 e D1.3**. Além disso, o relacionamento IS-PART-OF foram transformados em agregações entre a organização *SmartCD* e as organizações *Inventory* e *Financial*, conforme a **diretriz D1.5**. Para o relacionamento COVERS foi transformado no relacionamento ownership (**diretriz D1.9**) entre a organização *InternetSales* e os papéis de agente *CDReservation* e *CDDelivery*. O relacionamento PLAYS foi transformado no relacionamento play de NorMAS-ML (**diretriz D1.10**) entre o papel de agente *CDDelivery* e o agente *OfficeBoy*. Vale ressaltar que o relacionamento OCCUPIES não foi mapeado para NorMAS-ML, conforme a **diretriz D1.8**. Entretanto, as entidades ligadas por esse relacionamento foram mapeadas para o diagrama de normas.

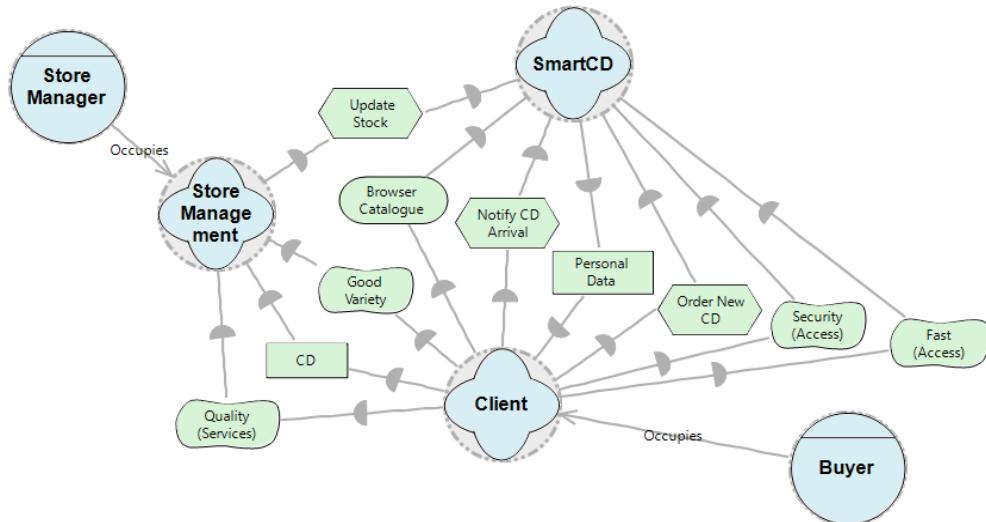


Figura 12. Modelo Strategic Dependence (SD) para o SmartCD [Melo et al., 2015]

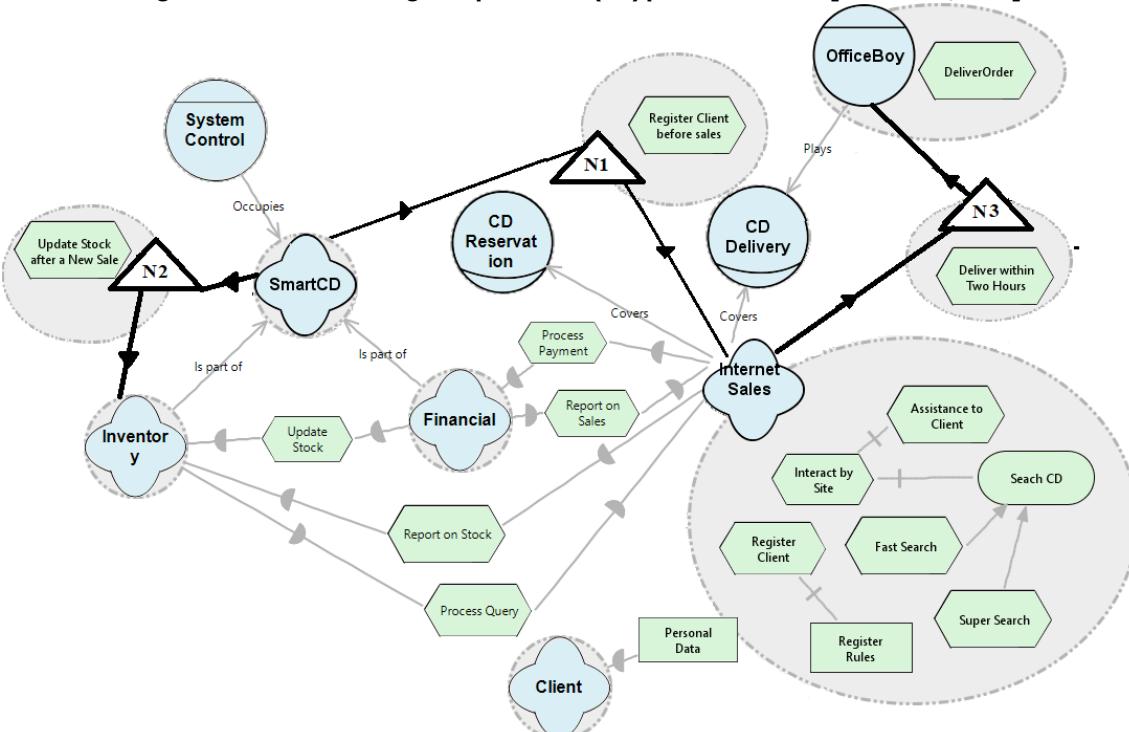


Figura 13. Modelo Strategic Rational (SR) para o SmartCD (Fonte: Próprios Autores)

As tarefas foram mapeadas para *agentActions* nas classes das entidades responsáveis por cada tarefa (**diretriz D2**). Além disso, o recurso *PersonalData*, foi mapeado como uma classe no

diagrama, pois apresentava característica de objeto (**diretriz D3.2**). Conforme as **diretrizes D5** e **D6**, os links de decomposição de tarefas e de *means-end*, foram mapeados para um comentário associado à organização *InternetSales*. Em relação as normas, N1, N2 e N3 foram mapeadas para classes de norma (**diretriz D7.2**), suas fontes foram representadas pelo relacionamento *context* (**diretriz D7.1**) e seus endereçamentos pelo relacionamento *restrict* (**diretriz D7.3**). Finalmente, as prescrições de cada norma foram mapeados para *resource* seguindo a **diretriz D7.4**.

6. Discussão

Por meio das diretrizes de mapeamento apresentado na seção 4, é possível transformar modelos *i** normativo no diagrama NorMAS-ML de normas. Essa transformação de modelos foi exemplificada na seção 5 através da modelagem do sistema *SmartCD*. Como vantagens, pode-se citar: (i) *O analista de requisitos pode concentrar-se no entendimento das necessidades dos usuários do sistema a ser desenvolvido e mapeá-las para os modelos SD e SR de *i** normativo. Além disso, as normas que regulam o comportamento das entidades desse sistema também podem ser modelados;* (ii) *O projetista pode utilizar as diretrizes de mapeamento propostas no presente trabalho para auxiliar na transformação dos modelos *i** normativo no diagrama NorMAS-ML de normas. Com isso, o projetista se concentrará na identificação de possíveis melhorias para o diagrama em conjunto com os analistas de requisitos;* e (iii) *A integração entre as fases de requisitos e de modelagem permite que os erros associados à transformação de modelos entre essas fases seja diminuído, possibilitando uma maior consistência entre os modelos gerados.*

Entretanto, essa pesquisa apresenta as seguintes limitações: (i) *O diagrama *i** normativo apresentado na seção 5 apresenta uma modelagem parcial do sistema SmartCD. Visto que, a modelagem poderia ter abordados os demais atores e novas normas poderiam ter sido propostas;* (ii) *Não foi proposta uma ferramenta para automatizar a transformação de modelos *i** normativo para o diagrama de normas. Por consequência, deve ser feita de forma manual;* e (iii) *Ao transformar os modelos *i** normativo para o diagrama NorMAS-ML de normas, o projetista precisaria seguir as diretrizes e realizar a transformação de forma manual. Logo, possíveis inconsistências poderiam ser encontradas nesse processo.*

Em relação aos trabalhos relacionados, vários autores [Richard 2010] [Sun et al. 2010] propuseram métodos que permitem a transformação entre modelos. Mais especificamente, Lucena et al. (2011) e Aguilar et al. (2010) propuseram a transformação de modelos *i** para modelos arquitetônicos Acme e para a linguagem QVT, respectivamente. Em relação ao escopo do presente trabalho, as abordagens propostas por Melo et al. (2015), Alencar, Giachetti e Pastor (2009), Filho, Zisman e Spanoudakis (2003) e Alencar et al. (2003) se aproximam ao objetivo deste trabalho pelo fato de proporem o mapeamento de *i** para diagramas da UML. No entanto, nenhum dos trabalhos descritos acima consideram o contexto dos SMANs, pois não abordam as normas que regulam as entidades desses sistemas. Com isso, o presente trabalho se diferencia dos demais por utilizar o *framework* *i** normativo em conjunto com o diagrama NorMAS-ML de normas, possibilitando a integração das fases de requisitos e modelagem via modelo.

7. Conclusão e Trabalhos Futuros

Este artigo teve como objetivo propor um mapeamento entre *i** normativo e NorMAS-ML com o intuito de identificar quais as semelhanças e as diferenças conceituais existentes entre eles. Para tanto, foram analisadas conceitualmente as entidades previstas em *i** normativo e do diagrama NorMAS-ML de normas. Assim, foi proposto um conjunto de diretrizes para a transformação de modelos entre esse *framework* e essa linguagem. Adicionalmente, um exemplo de modelagem, baseado no sistema *SmartCD*, foi realizado com o intuito de ilustrar as diretrizes de transformação propostas. Além das diretrizes de transformação, a contribuição

desse artigo foi a integração entre as fases de requisitos e de modelagem previstas no processo de desenvolvimento de *software*, considerando o paradigma orientado a agentes. Mais especificamente, essa integração colabora para que o entendimento de sistemas complexos (SMANs) seja facilitada por meio da utilização de linguagem visual (diagramas) e para a diminuição de inconsistências que normalmente ocorrem na transição de fases do processo de desenvolvimento de *software*. Como trabalhos futuros, utilizando as diretrizes propostas neste trabalho, pode-se destacar: (i) a formalização das diretrizes baseando na abordagem de Melo et al. (2015) e (ii) a implementação de uma ferramenta capaz de ler os modelos em i* normativo e gerar o diagrama NorMAS-ML de normas correspondente.

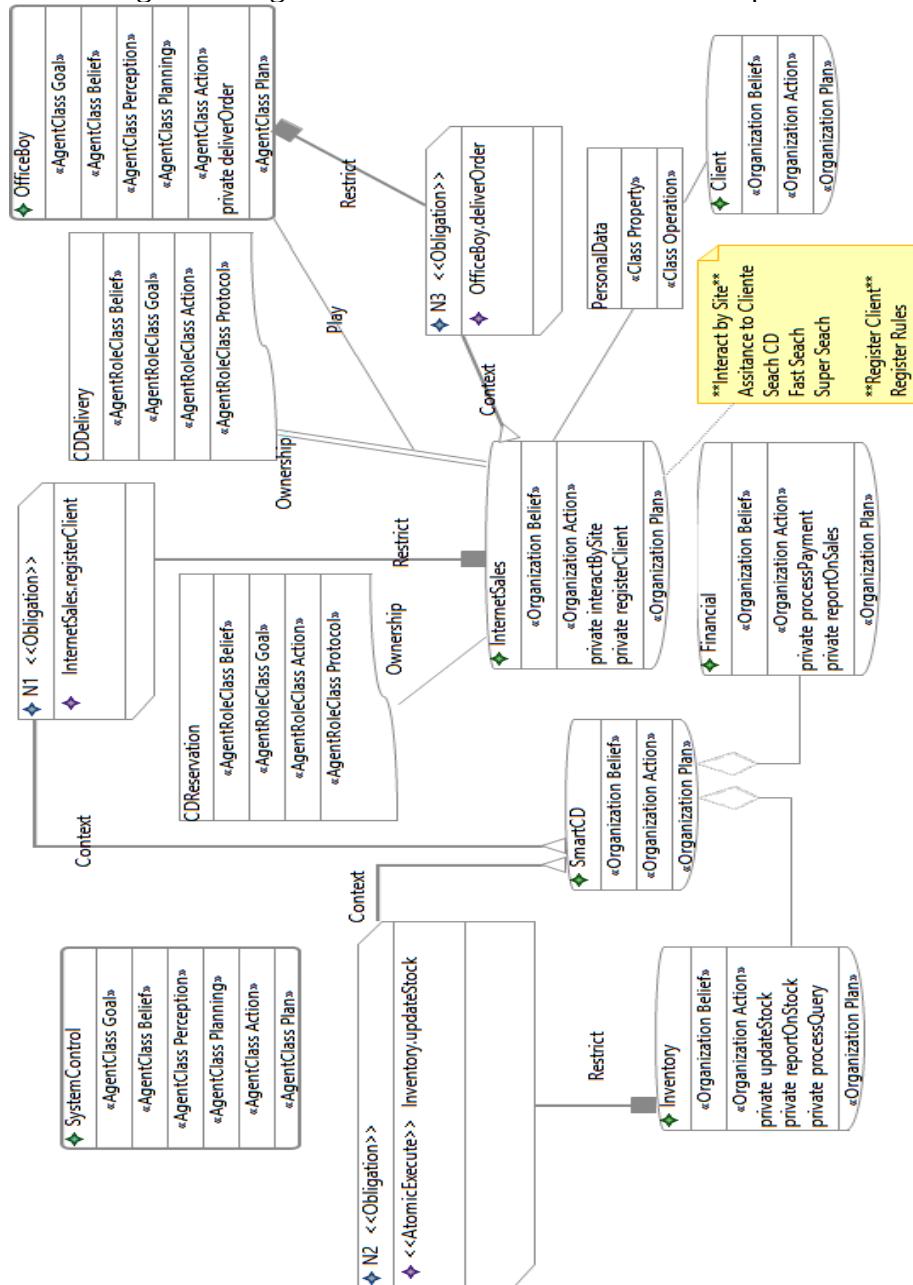


Figura 14. Diagrama NorMAS-ML de Normas para o SmartCD (Fonte: Próprios Autores)

Referências

Aguilar, J., Garrigós, I., Mazón, J. and Trujillo, J. (2010) An MDA approach for goal-

- oriented requirement analysis in Web engineering, *J. Univers. Comput. Sci.*, vol. 16, no. 17, pp. 2475–2494, 2010.
- Alencar, F., Gianchetti, G. and Pastor, O. (2009). From i* requirements models to conceptual models of a Model Driven Development process. In: *The Practice of Enterprise Modeling - Second IFIP WG 8.1 Working Conference, PoEM 2009, Proceedings*, pp. 99-114.
- Alencar, F.M.R., Pedroza, F., Castro, J.F.B. and Amorim, R.C.O. (2003) New Mechanisms for the Integration of Organizational Requirements and Object Oriented Modeling, VI WORKSHOP DE ENGENHARIA DE REQUISITOS, Piracicaba-SP , 2003.
- Boella, G., van der Torre, L. and H. Verhagen (2006) Introduction to normative multiagent systems, *Computational & Mathematical Organization Theory*, vol. 12, no. 2, pp. 71–79.
- Filho, C. G, Zisman, A. and Spanoudakis, G. (2003) Traceability Approach for i* and UML Models. In: *Proceedings of 2nd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'03)*, Portland.
- Franceto, S. (2005) Especificação e Implementação de uma Ferramenta para Elicitação de Requisitos de Software baseada na Teoria da Atividade. 2005. Disponível em: <<http://goo.gl/4wj9EZ>>. Acessado em 10 de outubro de 2017.
- Freire, E. S. S. and Cortés, M. I. (2016) Integrando Requisitos Organizacionais à Modelagem de Sistemas Multiagente Normativos. In: *10th Workshop-School on Agents, Environments, and Applications (WESAAC)*, 2016, Alagoas.
- Freire, E. S. S., Cortés, M. I., Gonçalves, E. J. T. and Lopes, Y. S. (2012) NorMAS-ML: A Modeling Language to Model Normative Multi-Agent Systems. In: *14th International Conference on Enterprise Information Systems (ICEIS)*, 2012, Wroclaw (Poland).
- Lucena, M., Castro, J., Silva, C., Alencar, F. and Santos, E. (2011) Stream: a strategy for transition between requirements models and architectural models. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 699-704. ACM, 2011.
- Melo, J., Sousa, A., Agra, C., Júnior, J., Castro, J. and Alencar, F. (2015) Formalization of Mapping Rules from iStar to Class Diagram in UML. In: *29th Brazilian Symposium on Software Engineering (SBES)*, 2015, Belo Horizonte.
- Richard, B. (2010) A deidealisation semantics for KAOS. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 267-274. ACM, March 2010.
- Siena, A., Maiden, N., Lockerbie, J., Karlsen, K., Perini, A. and Susi, A. (2008) Exploring the Effectiveness of Normative i* Modelling: Results from a Case Study on Food Chain Traceability. In: *CAiSE '08 Proceedings of the 20th international conference on Advanced Information Systems Engineering* Pages 182 – 196.
- Silva, V., Braga, C. and Figueiredo, K. (2010) A Modeling Language to Model Norms. In: *Workshop on Coordination, Organization, Institutions and Norms in agent systems at International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS10)*, Toronto, p. 25-32.
- Sommerville, I. (2011) Engenharia de Software. 9 ed. São Paulo: Pearson Addison- Wesley, 2011.
- Sun, Z., Wang, J., He, K., Xiang, S. and Yu, D. (2010) A Model Transformation Method in Service-Oriented Domain Modeling. In *Proceedings of the 2010 21st Australian Software Engineering Conference*, pages 107-116. IEEE Computer Society, 2010.
- Yu, E. (2002) Social Modeling and i*. In: Faculty of Information, University of Toronto, Toronto, Canada M5S 3G6, 2002.

An Agent-Based Model for Normative Hierarchical Organizations considering Goal Decomposition Process

Emmanuel S. S. Freire¹, Robert M. R. Jr², Mariela I. Cortés² and Gustavo A. L. Campos²

¹Teaching Department – Federal Institute of Ceará (IFCE)
Av. Prefeito Raimundo José Rabelo, 2717 – 62940-000 – Morada Nova – CE – Brazil

²Department of Computer Science – State University of Ceará (UECE)
Fortaleza – CE – Brazil

savio.freire@ifce.edu.br, robster@gmail.com, {mariela, gustavo}@larces.uece.br

Abstract. *The ideas about the structures and interactions that happen in human organizations have been used to model agent-based normative hierarchical organizations. Some works have used social control to regulate the agents' behavior to achieve an organizational goal, i.e., that is decomposed in a set of meaningful sub goals to the agents. Many organizational models have been defined to model these organizations, but they do not relate, more specifically, the agents in the hierarchical structure with the sub goals resulting from the decomposition process. This paper proposes an approach to bridge this gap, i.e., to model agent-based normative hierarchical organizations along with the goal decomposition process. In short, we extended a model found in Literature considering the goal decomposition defined in Moise+. We used UML diagrams to illustrate our extension.*

1. Introduction

Agent paradigm has been used to represent and simulate the interaction that occurs in real world. Moreover, this paradigm has been used with a solution to develop complex systems. As a result, the development of agent-centered systems has requested an effort of Software Engineering to provide methodologies, programming and modeling languages, and tools to support different steps of their development process.

One type of this system is a multi-agent system (MAS) which is composed by a set of agents ordered in an organization which interact between them to achieve a determinate goal. These systems can be regulated by norms to reach an organizational goal. Therefore, when norms and MAS entities are put together we have a normative multi-agent systems (NMAS). In these systems, an agent can decide whether it will comply or not an organizational norm. However, this agent can receive a reward or punishment, respectively.

Some authors [Vázquez and López y López 2007] [Dignum 2003] [Ferber, Gutknecht and Michel 2004] [Ferber and Gutknecht 1998] have researched in NMAS, more specifically in agent-based hierarchical organizations. These authors have analyzed the behavior of agents, the power relation and the interaction between agents to achieve an organizational goal and to follow the organizational norms. In an agent-based hierarchical organization the power relation exists in different level of the hierarchy. In

addition, in this kind of organization, an organizational goal can be divided into sub goals and the agent in a higher level is responsible for all organizational goals and has the power over his/her subordinates.

In this context, some organization models [Hübner, Sichman and Boissier 2002] [Ferber, Gutknecht and Michel 2004] [Dignum, 2003] [Ferber, Stratulat and Tranier 2009] have been proposed to represent organizations and theirs structures. However, none of them allow the modeling of these hierarchical structures completely. In addition, the division of a goal into sub goals and their dependencies are represented partially in these models. Considering this drawback in these organizational models, our paper has the objective to propose a framework to represent the goal decomposition process and the relation between sub goals created in this process for hierarchical organizations. Therefore, we extended the model proposed by Vázquez and López y López (2007) to represent the goal decomposition process defined by Hübner, Sichman and Boissier (2002).

This paper is organized as follows. We present the concepts related to hierarchical organizations, the agent-model proposed by Vázquez and López y López (2007) and Moise+ in Section 2. In Section 3 we discuss the related work. The extension of the model defined by Vázquez and López y López (2007) is detailed in section 4. An example to illustrate our extension is modeled in Section 5. In section 6 we discuss the benefits and the drawbacks of our extension. Finally, conclusions and future works are presented in Section 7.

2. Background

This section describes the main concepts needed to understand this work, including concepts related to hierarchical organizations, the agent-based model defined by Vázquez and López y López (2007) and the organizational model Moise+.

2.1. Hierarchical Organizations

A hierarchy or hierarchical organization is characterized by agents arranged in a tree-like structure, where agents located at the top of the tree have a more global view than those below them [Horling and Lesser 2005]. This structure is used to control larger groups of agents and to decompose task in subtasks. Consequently, these tasks can be performed more efficiently.

Fox (1979) described three types of hierarchical organizations: Simple hierarchy, Uniform hierarchy and Multi-divisional hierarchy. Keeping in mind that our paper will focus in agent-based uniform hierarchy organization, we will describe it in more detail. A uniform hierarchy has a distributed authority in each level of the tree. Consequently, the agent in a higher level is responsible for (i) achieving an organizational goal and (ii) checking the agents that are below him/her (subordinates). In other words, this agent is responsible for the organizational goal, i.e., for all the sub goals of his/her subordinates, over whom he has the power.

It is important to highlight that an organizational goal or sub goal can be decomposed into other sub goals [Hübner, Sichman and Boissier 2002], and each created sub goal can be divided in other sub goals. If necessary, this goal decomposition process and can happen several times. This process results in a tree-like structure and allows to divide tasks and responsibilities between agents in hierarchical organizations.

Only agents in higher levels of this type of organization may do this process. Thus, when an agent performs this process, this agent divides its own power with other agents which take some sub goals. Consequently, they start to have responsibility for the agents hierarchically below them.

2.2. The Agent-based Model defined by Vázquez and López y López (2007)

This section presents the main elements, structures, and concepts defined in the model of Vázquez and López y López (2007). The main elements in this model are norms, normative agents, resources, organizational goals, organizational services, contracts, position profiles, and organizational agents. Figure 1 presents the global view of this model represented in UML class diagram [OMG, 2017].

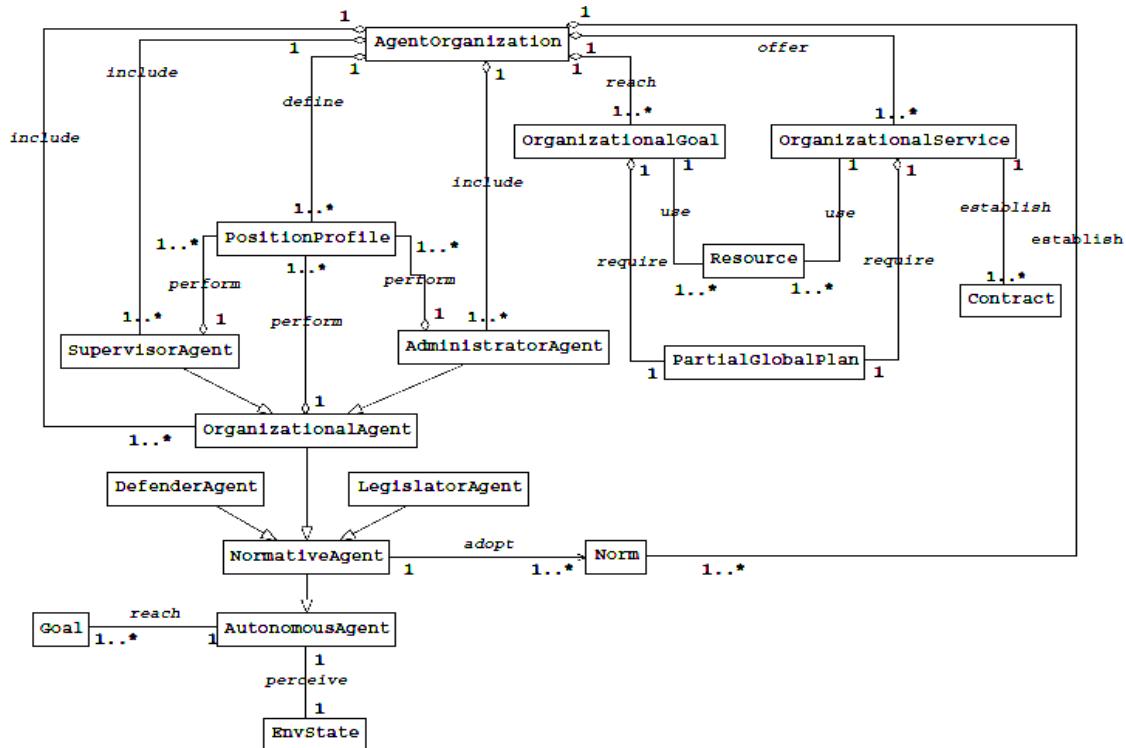


Figure 1. Global view of model [Vázquez and López y López 2007]

According to López y López, Luck and d'Inverno (2005), norms are used to regulate agents in a society and have the following elements:

- **Normative Goals**: they comprehend the goals defined by a norm;
- **Addressee Agents**: they represent that the agents must achieve the normative goals defined by a norm;
- **Beneficiaries Agents**: these elements represent that the agents could receive some benefits due to norm compliance;
- **Context**: it represents the conditions used to activate a norm;
- **Exceptions**: they represent events where an agent may decide to follow a norm or not;
- **Punishments**: these elements represent the penalties that an agent receives when it does not follow a norm; and

- **Rewards:** they are the benefits applied to an agent when he/she follows a norm.

In addition, López y López and Luck (2004) defined a set of organizational norms to regulate the behavior of an agent in NMAS [Boella et al. 2006]. Moreover, this set is composed by following norm types:

- **Defender Norms:** these norms are used to give rewards or to apply punishments to agents that follow or violate a norm, respectively;
- **Enforce Norms:** they help to enforce and to determine the fulfillment of the most recent set of norms;
- **Reward Norms:** these norms promote the fulfillment of norms by using rewards; and
- **Emitted Norms:** they allow the creation and the abolition of norms.

In this model, a Normative Agent knows the organizational norms. Additionally, there are agents called **Legislator Agent** which are entitled to create new norms or abolish previous norms in the system; and agents called **Defender Agent** which are entitled to apply rewards or punishments according to the norm compliance. Considering the administrative process functions defined by Chiavenato (2005), the model of Vázquez and López y López (2007) has the following agent types:

- **Organizational Agent:** it is a specialization of Normative Agent, and it must recognize and fulfill organizational norms;
- **Administrator Agent:** it is a specialization of Organizational Agent and Legislator Agent because it is entitled to generate plans, and to create or abolish norms in the system; and
- **Supervisor Agent:** it is a specialization of Organizational Agent and Defender Agent because it is entitled to make own subordinates to reach the organizational goals, to give rewards or to apply punishments.

These elements, norms and agents, can be put together in an organization. In the model of Vázquez and López y López (2007), the organization uses the position profile to identify a functional position in a hierarchy, to describe the set of obligations and rights in a position and to specify the superior (chief) and own set of inferior elements (subordinates) in each position. The position profile has the same function that an agent role. In addition, this organization has resources to operate to achieve its goals.

An organizational goal is a set of desired states. A state represents a situation which an organization comes across a period. A plan is the result of the decomposition of goals into sub goals, and it is depicted in PartialGlobalPlan. Additionally, a set of organizational services are provided by an organization. These services can be accessed by individual agents or other agent-based organizations. It is important to point out that an organizational service is guaranteed by means of contract. A contract is a set of obligation norms that represents benefits and obligations for own participants.

This model [Vázquez and López y López 2007] allows the representation of main elements that form a hierarchical organization for agent-based systems. In addition, it uses norms to control the behavior of its entities in different levels of hierarchy.

2.3. Moise+

Moise+ is an agent-centered organizational model defined by Hübner, Sichman and Boissier (2002). It was based on two main ideas: (i) Organizational Specification (OS) that defines the structure, the working and the norms that restrict the organization of agents; and (ii) Organizational Entity (OE) is the combination of an organizational structure and the set of agents that inhabit in it. More specifically, an OS has three dimensions: (i) Structural dimension: defines roles, interactions between roles and groups; (ii) Functional dimension: defines the organizational goal and its sub goals; (iii) Deontic dimension: defines the agents in the Structural dimension that are committed to the sub goals in the Functional dimension.

An agent playing a role can interact with other agents by three kinds of links: authority, communication and acquaintance links. In a scheme, it is possible to decompose a goal or a sub goal into sub goals using one of three operators, as depicted in Figure 2:

- **Sequence:** consider the goal g_0 was decomposed into three sub goals (g_1 , g_2 , g_3), meaning that g_0 will be achieved only if the g_1 , g_2 , and g_3 that are reached, sequentially in that order (the same occurs with sub goal g_2);
- **Parallel:** consider that sub goal g_3 was decomposed into two sub goals (g_{31} and g_{32}), meaning that g_3 will be reached when g_{31} and g_{32} are achieved, in any order of execution; and
- **Choice:** consider that sub goal g_1 was decomposed into two sub goals (g_{11} and g_{12}), meaning that g_1 will be reached if g_{11} or g_{12} is achieved, i.e., exclusively one of the sub goals.

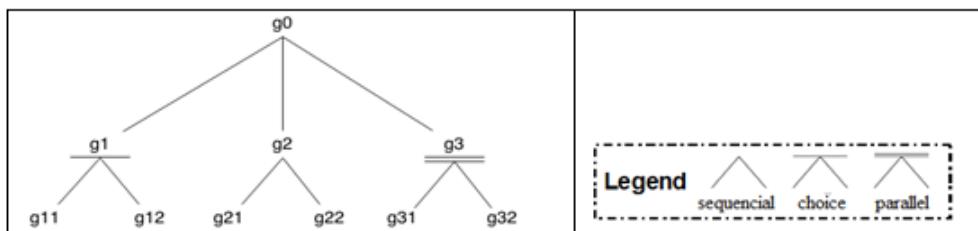


Figure 2. Operators for goal decomposition process

3. Related Work

This section involves works related to organization representation. Some organizational models have been proposed for NMAS [Hübner, Sichman and Boissier 2002] [Ferber, Gutknecht and Michel 2004] [Dignum 2003] [Ferber, Stratulat and Tranier 2009]. Our aim is to analyze four organizational models considering the support provided to the modeling of hierarchical organizations and the division goals into sub goals and their dependencies between them.

Hübner, Sichman and Boissier (2002) defined an organization model called Moise+. It supports the modeling of organizations composed by power levels (similar to hierarchical organizations), but it does not allow to apply sanction based on actions of agent. In this sense, Moise+ allows the description of permission and prohibition norms

for roles in the organizational context. In addition, CASE Moise API and Platform allows to create and specify organizations and to manage their entities.

AGR (*Agent-Group-Role*) [Ferber, Gutknecht and Michel 2004] is based on AALAADIN model [Ferber and Gutknecht 1998] that describe hierarchy organizations. In AGR, the agents can have their behavior regulated by interaction protocols. An organization modeled in this organizational structure is represented by congregations and colligations [Isern, Sánchez and Moreno 2011] of agents. These entities can communicate only with others owned same group. In the same way of Moise+, AGR does not allow sanction specifications.

In OperA [Dignum 2003], the description of obligation, permission and prohibition norms for agents, agent roles and agent groups is allowed in organizational context, interaction and scene transaction. Nevertheless, this model does not give support to model agent's structural aspects neither environments. In addition, norms can be applied to agents, in order to play a role as mean of a contract. In addition, every agent knows all the sanctions related to a contract. However, OperA does not allow the goal decomposition process.

MASQ (*Multi-Agent System based on Quadrant*) [Ferber, Stratulat and Tranier 2009] allows the sanction definition for each member associated in an organization, but it is not possible to describe perception mechanisms for these sanctions. In other words, punishments can be detailed and related with norms that restrict agent roles and the interactions inside a group of agents. However, this organizational structure does not have an associated sanction mechanism neither allows the goal decomposition process.

Unfortunately, all organizational structures presented and discussed in this subsection do not give totally support to model hierarchy organizations, all their structures and the goal decomposition process. Only Moise+ has the supporting to model the goal decomposition process.

4. Extending the Model of Vázquez and López y López (2007)

According to Vázquez and López y López (2007), a hierarchical organization is composed by a group of agents organized in a tree-like structure. A chief agent is responsible to divide a goal into sub goals and to choose agents situated in inferior levels to execute these sub goals. However, a sub goal could have a dependency with another sub goal. As a result, Hübner, Sichman and Boissier (2002) defined three goal operators, explained in Section 2: **Parallel, Sequence and Choice**.

To represent these goal decomposition operators in the model of Vázquez and López y López (2007), we defined new classes and relationships in this model. It is important to notice that our extension did not change the semantic and the structure of the previous version. In other words, we did a conservative extension in this model. Figure 3 presents the new version of the model depicted in UML class diagram.

In this context, we defined the classes *GoalRelation*, *SequentialNumber* and *GoalRelationTypes*. In Figure 4 (a), the *GoalRelation* class is a structure for each one kind of goal decomposition. It *has* relationship with *OrganizationalGoal* class, allows to identify the list of sub goals related with themselves. In Figure 4 (b), the associate class *SequentialNumber* defines the order that goals can be performed considering the

sequential decomposition goal through its *orderingNumber*. The *GoalRelationTypes* class (See Figure 4 (c)) is an enumeration class that has the kind of decomposition goals. It enables to identify the type of decomposition goal used by *GoalRelation* class through *hasType* relationship.

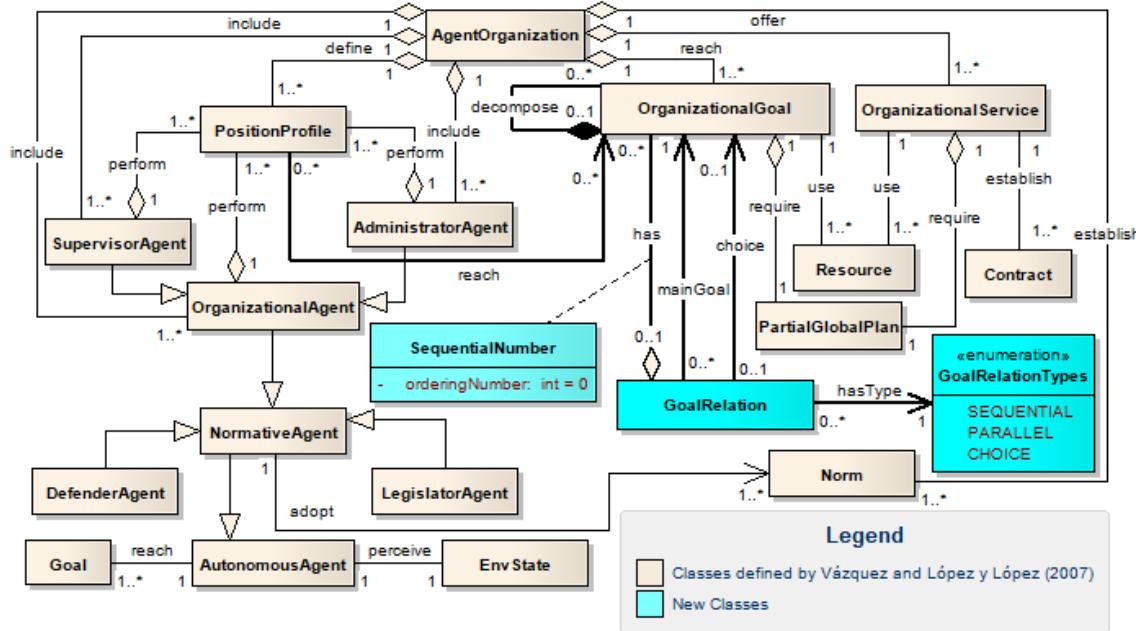


Figure 3. A new version of the model

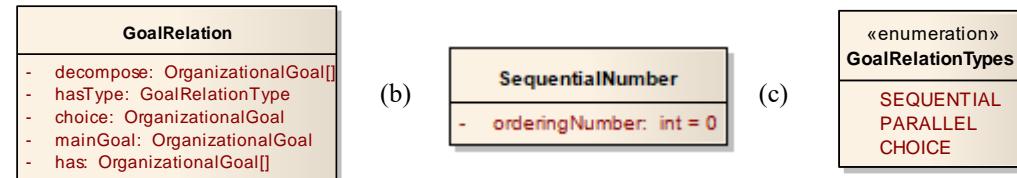


Figure 4. Structure of new classes

When the choice decomposition goal is used, the *choice* relationship in *OrganizationalGoal* class recognizes the sub goal that was chosen to execute. In its turn, the parallel decomposition goal does not need of a specified attribute or a relationship because of the list of sub goals (the self-association called *decompose* in *OrganizationalGoal* class) is already sufficient. In addition, *mainGoal* relationship allows to identify the goal (or sub goal) that was decomposed into sub goals.

In addition to the structure of the *OrganizationalGoal* class, we defined *reach* relationship between *PositionProfile* and *OrganizationalGoal* classes. This relationship represents the set of organizational goals which an organizational agent needs to achieve while plays a determinate position profile. In Vázquez and López y López's metamodel it was possible to identify what organizational goals were related to a position profile when we analyze the norms, because they identify the restricted entities (for instance, position profiles) and the regulated resources (for example, organizational goals). Thereby, the *reach* relationship defined in this new version of the metamodel take easy to identify the relation between *PositionProfile* and *OrganizationalGoal* classes,

allowing that the agent in a higher level to be responsible for achieving an organizational sub goal.

In short, our extension included the definition of three new classes (*SequentialNumber*, *GoalRelation*, *GoalRelationTypes*) and the definition of following relationships *decompose*, *reach*, *mainGoal*, *choice*, *has* and *hasType*. Consequently, these new elements are sufficient to represent the new concepts defined in goal decomposition.

5. Example of Application

To show the applicability of our extension, we used the context of a software development company. The functional positions defined in this company are: *Sponsor*, *Project Manager*, *Requirement Analysts* and *Software developers*. Figure 5 shows the UML use case diagram associated to these functional positions and their functionalities in this example. Figure 6 shows the defined goals. Figure 7 shows the goals and the dependency relation between them modeled through *OrganizationalGoal*, *SequentialNumber*, *GoalRelation* and *GoalRelationTypes* instances.

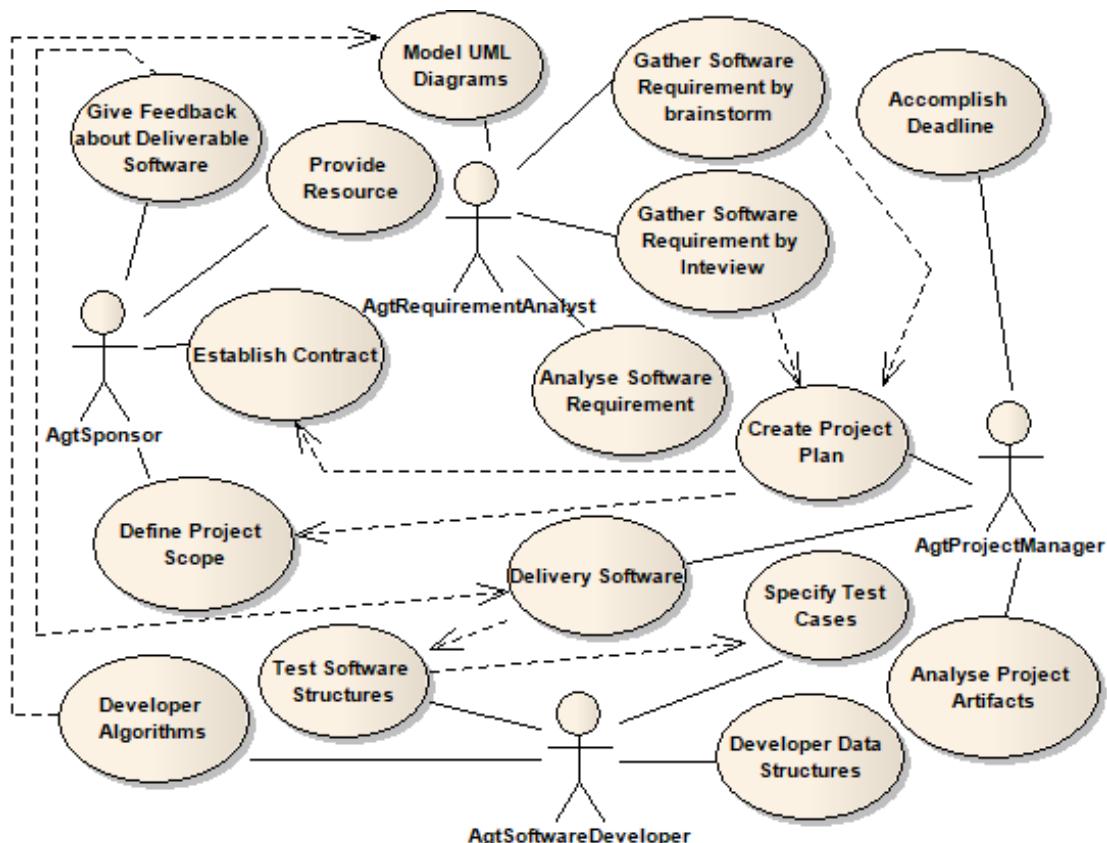


Figure 5. UML Use Case Diagram for Software Development Company

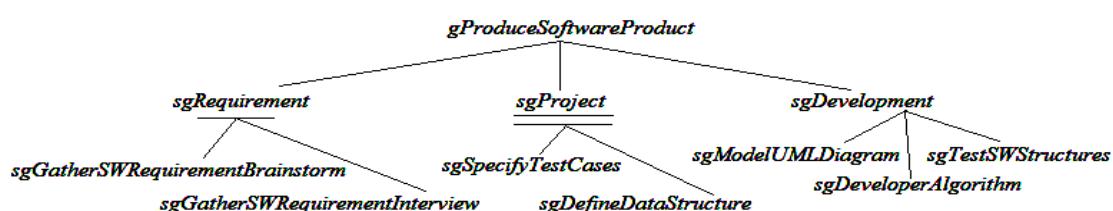


Figure 6. Goal, sub goals, and their dependencies represented in Moise+ syntax

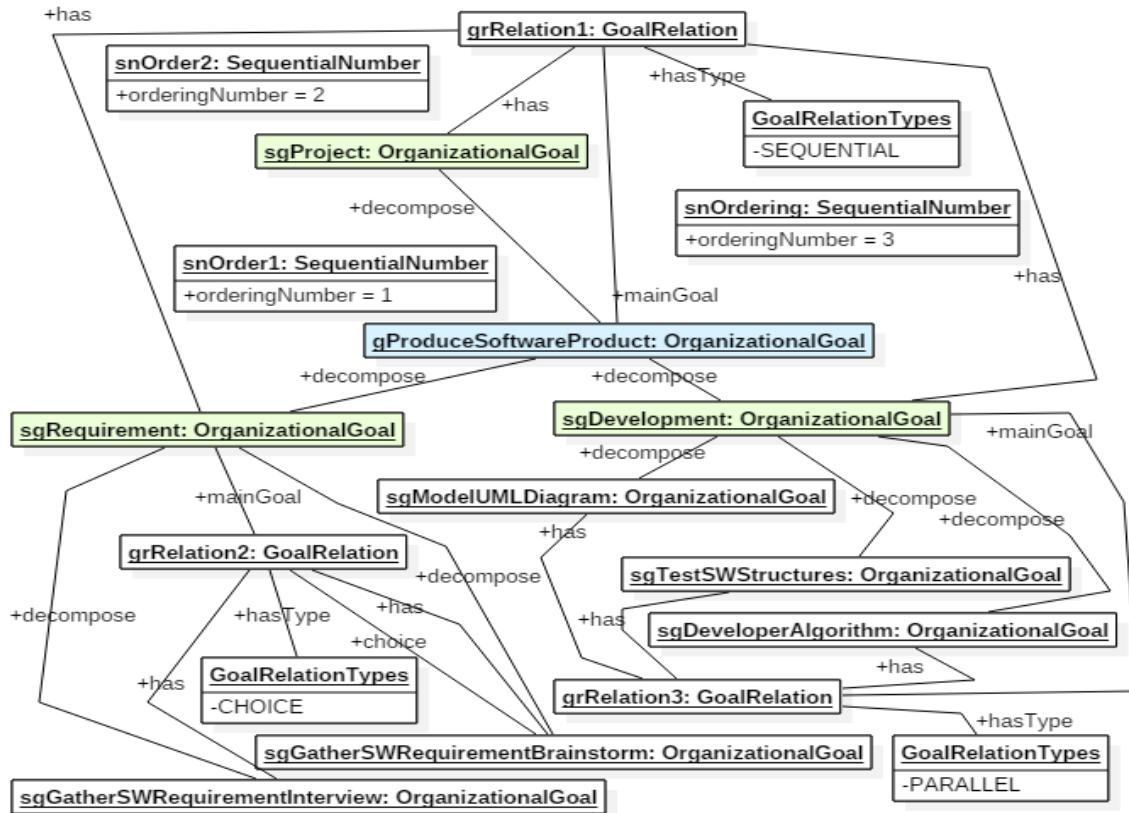


Figure 7. Modelling of goal dependencies

The *gProduceSoftwareProduct* goal was decomposed into *sgRequirement*, *sgProject*, and *sgDevelopment* sub goals, that must be executed in this order because they have a sequential goal relation. The *sgDevelopment* sub goal was decomposed into *sgModelUMLDiagram*, *sgDeveloperAlgorithm*, and *sgTestSWStructures* sub goals, that must be executed orderly because they are related by sequential goal relation. In contrast, the *sgGatherSWRequirementBrainstorm* sub goal must be executed exclusively because it was chosen in a choice goal relation with *sgGatherSWRequirementInterview* sub goal. These sub goals are part of the *sgRequirement* sub goal. Finally, *sgProject* sub goal was decomposed into *sgSpecifyTestCases* and *sgDefineDataStructure* sub goals and they must be executed in any order because they are related by parallel goal relation.

To model the execution of the goal decomposition process, we defined the *agtProjectManager* agent as the responsible for achieving the *gProduceSoftwareProduct* goal. In this paper, we just modeled the scenario about sequential goal relation involving this goal and its *sgRequirement* sub goal. The same process happens for *sgProject* and *sgDevelopment* sub goals. Figure 8 shows the UML sequence diagram describing the interactions between the instances in the system. Firstly, *agtProjectManager* agent uses *getState()* method to identify the state of the environment and recognize their responsibility for the *gProduceSoftwareProduct* goal. After this, the agent starts the goal decomposition process creating new sub goals and related them in *GoalRelation* instances. This process occurs while the decomposition is happening.

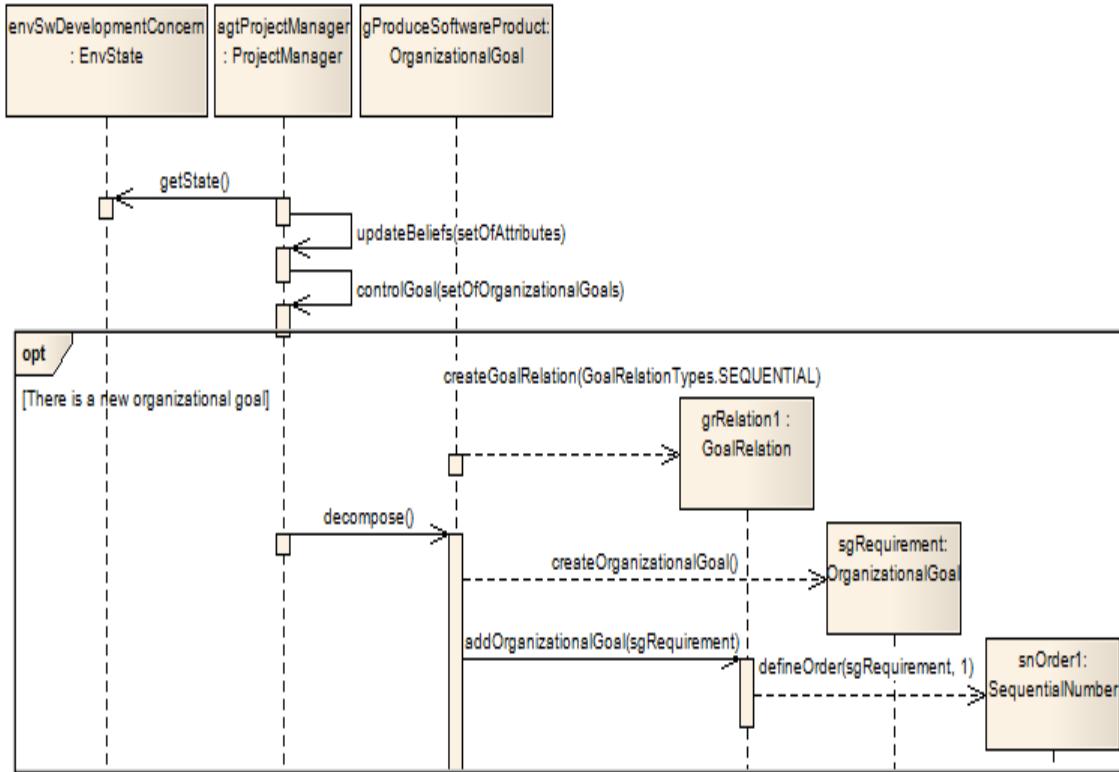


Figure 8. Modelling the goal decomposition process

6. Discussions

By means of our extension, it is possible to model the structure and the goal decomposition process for hierarchical organizations. An example of modelling was showed in section 5 using the context of a software development company. We choose the model of Vázquez and López y López (2007) because (i) it has all structures of hierarchical organizations, (ii) it is based on UML class diagram, allowing the extension and the adjustment of its entities, (iii) it has the definition of supervisor and defender agent that control the changes of norms in an organization, and (iv) it is flexible because normative agents can adopt new norms and update its set of norms.

The new version has the following advantages: (i) the syntax of our model is easy to understand and to use because it was based on UML, (ii) the designers can propose the goal decomposition considering the initial version of the organization focusing on the accomplishment of main organizational goal, and (iii) the division of goals/sub goals into sub goals allows that designers to understand the power and responsible level of their agents in a hierarchical organization. Our model presents the following drawbacks: (i) the designers need to spend time to understand the semantic of the entities in the model, and (ii) the modelling of a large system could ensue in a model of difficult understanding.

In this way, whether we compare our extension with related work, it is possible to notice that all organization models did not give support for the modelling of all entities in a hierarchical organization. For example, although very powerful, the organizational modelling language Moise+ does not allow the modelling of sanctions

based on agent's action. In addition, the goal decomposition process is used only to make plans that agent must be followed to reach an organizational goal. Our model allows the modelling of complex systems improving the understanding of the system that will be developed in technical and user visions, i.e., it is a contribution to the Software Engineering and Artificial Intelligence areas.

7. Conclusions and Future Works

This paper presented a model that allows the modelling of hierarchical organizations considering their structures and the goal decomposition process defined by [Hübner, Sichman and Boissier 2002]. Therefore, we extended the model of Vázquez and López y López (2007) defining three new classes (*SequentialNumber*, *GoalRelation*, *GoalRelationTypes*) and the relationships *decompose*, *reach*, *mainGoal*, *choice*, *has* and *hasType*. In addition, an example of application was modeled using our extension based on the context of a software development company. A UML use case diagram was modelled to present the main services of the organization and we used a UML object diagram to demonstrate the design of goal decomposition process considering three operators: sequential, parallel and choice. In addition, a UML sequence diagram was used to show the interaction between the entities while the goal decomposition process is happening.

As future works, we can suggest: (i) the implementation of our model using an agent framework, (ii) the formalism of our model using graph theory, and (iii) the analysis of the dynamic of norm compliance considering goal dependencies for individual and group of agents.

References

- Boella, G., van der Torre, L., and Verhagen, H. (2006). Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, 12(2):71–79.
- Chiavenato, I. (2005). Teoria Neoclássica da Administração. In: *Introdução à Teoria Geral da Administração*. São Paulo: Campus.
- Dignum, V. (2003). A model for organizational interaction: based on agents, founded in logic. PhD thesis, Utrecht University.
- Ferber, J. and Gutknecht, O. (1998). A meta-model for the analysis and design of organizations in multiagent systems. In *Proceedings of the Third International Conference on Multiagent Systems, ICMAS 1998, Paris, France, July 3-7, 1998*, pages 128–135.
- Ferber, J., Gutknecht, O., and Michel, F. (2004). From Agents to Organizations: An Organizational View of Multi-Agent Systems, pages 214–230. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Ferber, J., Stratulat, T., and Tranier, J. (2009). Towards an Integral Approach of Organizations in Multi-Agent Systems. In Dignum, V., editor, *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, pages 51–75. IGI Global.

- Fox, M. S. (1979). Organization structuring: Designing large complex software. Computer Science Technical Report CMU-CS-79-155, Carnegie-Mellon University.
- Horling, B. and Lesser, V. (2005). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, pages 281–316.
- Hübner, J. F., Sichman, J. S., and Boissier, O. (2002). A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems, pages 118–128. Springer, Berlin, Heidelberg.
- Isern, D., Sánchez, D., and Moreno, A. (2011). Organizational structures supported by agent-oriented methodologies. *J. Syst. Softw.*, 84(2):169–184.
- López y López, F. and Luck, M. (2004). A Model of Normative Multi-agent Systems and Dynamic Relationships, pages 259–280. Springer, Berlin, Heidelberg.
- López y López, F., Luck, M., and d'Inverno, M. (2005). A normative framework for agent-based systems. In Symposium on Normative Multi-Agent Systems, NORMAS 2005, part of the SSAISB 2005 Convention, University of Hertfordshire, Hatfield, UK, 12-15 April 2005. Proceedings, pages 24–35.
- OMG (2017). OMG Unified Modeling Language (OMG UML), Version 2.5.
- Vázquez, L. E. M. and López y López, F. (2007). An Agent-Based Model for Hierarchical Organizations, pages 194–211. Springer, Berlin, Heidelberg.

Um modelo de Multiagentes Normativos para Descrição de Personagens de Jogos Sérios

Jonathan Morris Samara¹², Cesar Augusto Tacla¹,
Sebastião Ribeiro Junior²⁵, Bruno Soares de Souza²
Klaus de Geus³, Rafael Bee³,
Sergio Scheer⁴

¹ Pós-Graduação em Engenharia Elétrica e Informática Industrial
Universidade Tecnológica Federal do Paraná - UTFPR
Av. Sete de Setembro 3165, CEP 80230-901 Curitiba PR.

²Divisão de Sistemas Elétricos - Institutos Lactec
Rodovia BR-116, Km 98, nº 8813, Jardim das Américas. Curitiba, Paraná.

³Companhia Paranaense de Energia
Rua Coronel Dulcídio, 800 - Batel - CEP 80420-170 - Curitiba - PR.

⁴Centro de Estudos de Engenharia Civil - Universidade Federal do Paraná - UFPR
Rodovia BR-116, Km 98, nº 8813, Jardim das Américas. Curitiba, Paraná.

⁵Departamento de Engenharia Elétrica - Universidade Federal do Paraná - UFPR
Rodovia BR-116, Km 98, nº 8813, Jardim das Américas. Curitiba, Paraná.

Abstract. *Research in the field of serious games has lately received significant attention of the academic community. Due to its importance in education and training, normative concepts and deontic logic have also played an important role in the field. Normative agents include computational representations of social, structural and deontic relations in the context of organizations in multiagents systems. The aim of this work is to discuss the development of an ontology used to create serious gaming characters as normative agents in serious games inspired by the definition of Moise+, in the context of an application aimed at training of live-line maintenance activities in energy substations. The implementation of this representation, using Moise+ and Unreal Engine 4, is presented, as well as the construction of an ontology that allowed the development of the game. Results indicate that the use of this specification is satisfactory for this kind of application. The results indicate the need to represent objects, tools and problems that must be solved by the characters in the game.*

Resumo. A investigação no campo de jogos sérios está atraiendo grande atenção da comunidade acadêmica. A lógica deôntica e sistemas normativos possuem uma grande importância no desenvolvimento de jogos sérios. Agentes normativos englobam representações computacionais das relações sociais, estruturais e deônticas no contexto de organização em sistemas multiagentes. O objetivo desse trabalho consiste em desenvolver uma ontologia para especificar personagens e suas missões como agentes normativos em jogos sérios inspirando-se na definição de organização de Moise+, no contexto de uma aplicação destinada ao treinamento de eletricistas de linha viva que atuam em

subestações. Foi feita a implementação de um protótipo de um jogo sério a partir da ontologia, com o uso do Moise+ e a Unreal Engine. Os resultados indicam que o uso da ontologia de especificação é satisfatória para esse tipo de aplicação. Os resultados também indicam a necessidade de representar objetos, ferramentas e problemas que devem ser resolvidos pelos personagens do jogo.

1. Introdução

As relações normativas e organizacionais são fundamentais para o funcionamento das atividades no âmbito profissional. Com base nisso, tratar essas relações em sistemas de treinamento, como Jogos Sérios, é algo fundamental para preservação dessas relações. A representação computacional de organizações é uma área de estudo que está contida em Sistemas Multiagentes Normativos [Boella et al. 2006]. As relações de objetivos, missões, grupos, funções, permissões e obrigações são definidas em modelos lógicos usados para representar estruturas organizacionais de agentes. Moise + é um modelo que tem como finalidade realizar esse tipo de representação [Hübner et al. 2002b]. Portanto há um interesse científico em entender como usar representações relacionadas a sistemas multiagentes normativos para criar personagens de jogos sério.

Esse trabalho tem por objetivo propor e avaliar por meio de um estudo de caso um modelo constituído de elementos necessários para o desenvolvimento de jogos sérios, mais especificamente, jogos sérios de treinamento de trabalhadores que realizam manutenção em linhas vivas. Uma particularidade do modelo é sua inspiração em sistemas normativos para especificar as missões, papéis, comunicação entre papéis e a normas por meio de operadores deônticos. Isto permite avaliar a conformidade da ação do trabalhador frente as normas operacionais e de segurança. A ontologia (ou modelo) deverá funcionar como uma ferramenta que guia, por meio de parâmetros de entrada, o desenvolvimento dos personagens do jogo. Como resultado final, o modelo permite especificar agentes e artefatos que devem constituir o jogo. Não há o propósito, nesse estudo, de desenvolver um gerador automático de jogos, mas sim um sistema que oriente os desenvolvedores a implementar o jogo. O modelo será uma ontologia de especificação de jogos sérios.

O problema utilizado para desenvolver essa representação é a manutenção em linha viva (manutenção em equipamentos elétricos energizados). Essas atividades necessitam de uma boa capacidade de planejamento e apresentam alto risco à vida de todos diretamente envolvidos. Outro ponto de interesse está no fato de que uma mesma manutenção pode ser solucionada de várias maneiras. Sendo assim, o modelo proposto nesse estudo deverá ser capaz de orientar o desenvolvedor a levar em consideração as seguintes restrições: relações sociais e profissionais entre as pessoas pertencentes ao grupo (ex. supervisor, executor), regras que devem ser seguidas por todos os agentes e pelo jogador, criação de regras que possam ser adequadas a todos os cenários possíveis, como tratar a violação dessas regras e quais as outras representações necessárias no contexto do jogo para que as regras continuem sendo adequadas.

2. Fundamentação Teórica

Os fundamentos teóricos dessa pesquisa consistem nos seguintes campos de conhecimento: Ontologia, Multiagentes Normativos e Jogos Sérios. As subseções a seguir disserão sobre cada um desses campos.

2.1. Ontologias

Do ponto de vista computacional o termo ontologia faz referência a um artefato de engenharia constituído por um vocabulário específico usado para descrever parcialmente um recorte da realidade. A ontologia também é constituída por suposições explícitas relativas ao significado pretendido para as palavras do vocabulário [Guarino 1998]. Dentro desse contexto, um Engenheiro de Ontologias está preocupado em identificar as entidades, em caracterizá-las segundo uma estrutura taxonômica e definir essas relações [Guarino et al. 2009].

2.2. Multiagentes Normativos

Um sistema multiagente é um sistema composto por entidades autônomas chamadas de agentes que interagem em um ambiente. Existem várias categorias de agentes. A primeira categoria é dos agentes reativos. Esses agentes são especificados por meio de regras. Uma segunda categoria consiste em agentes deliberativos. Esses agentes possuem controle de seus estados internos, informações passadas e também podem definir planos para atingir metas no longo prazo. Os agentes deliberativos de raciocínio prático com arquitetura BDI (*Beliefs, Desire, Intentions*) são caracterizados por representar e controlar seus estados internos por meio de crenças, desejos e intenções. Como consequência, esses agentes são capazes de elaborar planos para atingir um determinado objetivo [Nareyek 2001] [Woolridge and Wooldridge 2001].

Normas são entendidas como um princípio que guiam membros de um grupo tendo em vista o que é adequado e aceitável. Uma maneira de trabalhar as normas e agentes consiste no uso de lógica deônica que tem como por finalidade estudar as relações de obrigação, de permissão, de violação e da dinâmica da obrigação em relação ao tempo. Sendo assim, a lógica deônica pode representar as normas como regras ou condições que possibilitam o uso de raciocínios sobre isso [Boella et al. 2006].

Uma organização consiste em um conjunto de regras, relacionamentos e estruturas de autoridades e de governança que define o comportamento dos agentes [Hayden et al. 1999] [Horling and Lesser 2005]. No Moise+, uma organização é definida em três partes: Funcional (define o papel de cada agente, define as relações de ligação, define as relações de compatibilidade e define a estrutura dos grupos de papéis), Estrutural (define os objetivos e missões) e Deônica (define normas de ação sobre as missões) [Hübner et al. 2002a]. Na implementação do Moise+ no projeto JaCaMo, há dois módulos. O módulo S-Moise (parte Funcional, Estrutural e Deônica) define as relações sociais entre os agentes. O outro módulo é J-Moise, permite a integração da representação organizacional ao sistema multiagente que executa na plataforma Jason.

2.3. Jogos Sérios

Atualmente a definição mais usual para o conceito de jogos sérios é dada por [Susi et al. 2007]. Com base nesse autor, um jogo sério possui objetivos maiores (ex. treinamento, educação, simulação e entre outros) que apenas mero entretenimento [Breuer and Bente 2010].

Os jogos sérios e os jogos de entretenimento trabalham com funcionalidades similares ao de desenvolver tarefas, manter o foco, realizar simulação e realizar comunicação. Entretanto esses critérios são trabalhados com propósitos diferentes nos dois tipos de

jogos. Para um jogo sério, a tarefa tem como finalidade resolver o problema em foco enquanto que os jogos de entretenimento visam proporcionar uma experiência lúdica ao jogador. Em termos de manter foco, para um jogo sério, esse é um elemento primordial a ser desenvolvido, já na categoria de entretenimento isso é visto como uma forma de diversão. As simulações são vistas como essenciais para a ótica dos jogos sérios enquanto que um jogo de entretenimento entende as simulações como sendo um processo simples no que tange ao rigor de reproduzir com alta confiabilidade o que acontece no mundo real. A comunicação deve ocorrer de maneira natural nos jogos sérios enquanto que a comunicação ocorre visando perfeição nos jogos de entretenimento [Susi et al. 2007].

3. Estudos Relacionados

O estudo [Dignum et al. 2009] apresenta esforços para integrar uma plataforma de sistema multiagentes com alguma *game engine* com o objetivo de desenvolver personagens de jogos com comportamento mais natural. Para isso, os autores apresentam uma arquitetura de integração entre uma *Game Engine* e uma Plataforma de Agentes. Outro estudo que usa a mesma linha de desenvolvimento é o [Jepp et al. 2010]. O texto apresenta um framework para relacionar uma plataforma de desenvolvimento de jogos sérios com a *game engine*. O estudo proposto neste artigo difere de [Jepp et al. 2010], [Dignum et al. 2009] na abordagem usada para integrar os agentes com o jogo. Isso pois aqueles estudos estão preocupados em manter os agentes operando nas plataformas de sistemas multiagentes enquanto esse estudo pretende estabelecer critérios de representação para implementar agentes no código do jogo.

O estudo [Pons et al. 2012] apresenta um sistema de multiagentes que dinamicamente controla e orienta a ação de um jogo conforme as ações do jogador. A abordagem usada pelos pesquisadores consiste em descrever o jogo em termos de parâmetros, critérios e objetivos. Os parâmetros são todos os elementos que constituem e caracterizam o jogo (definições da física do jogo, recursos que são usados pelos jogadores, população de entidades artificiais). Os critérios são formas de combinar os parâmetros para obter o jogo. Os critérios devem possibilitar ao desenvolvedor a manipulação de todos os elementos presentes no jogo (gerando a formação de cenários possíveis). Os objetivos pedagógicos consistem em uma parte importante do jogo, pois são definições de performance sobre condições específicas que devem ser atingidas pelo jogador. Essas definições de performance são feitas em termos do critério. Um determinado objetivo consiste, portanto, em um critério específico que deve ser alcançado. Esse critério, então, é uma configuração de parâmetros que deve ser causada pelo jogador. Há um agente para cada especificação, esses agentes são: *Parameter-agent*, *Criterion-agent* e *Contrainst-agent*. O *Parameter-agent* é criado para cada parâmetro envolvido no cenário. O *Criterion-agent* é associado com cada critério usado para avaliar valores associados ao cenário. O *Contrainst-agent* ocorre para cada objetivo definido em relação a cada elemento do cenário. Esse agente computa a função de criticidade especificada pelo especialista em domínio. O objetivo é estruturado na função de criticidade (ex. quando a função atingir determinado valor, então o objetivo foi alcançado). Tanto o estudo [Pons et al. 2012] como este fazem uso de sistemas multiagentes em jogos sérios. A diferença reside no objetivo, pois enquanto que este estudo tem o interesse de usar essa ciência para especificar o comportamento dos *bots* do jogo, aquele estudo tem interesse em desenvolver controladores dinâmicos em relação ao comportamento do jogador.

O estudo [Tang and Hanneghan 2011] apresenta uma ontologia para especificar a modelagem de um jogo sério. O objetivo de desenvolvimento dessa ontologia reside no fato de que o projeto de jogos de computador necessita experiência na descrição das regras, da jogabilidade e da estética que compõem a experiência interativa. Isso complica o trabalho dos desenvolvedores inexperientes. Como consequência, a existência de um meta-modelo para documentar, especificar e desenhar jogos é algo de bastante utilidade. Os pesquisadores investigaram a área de desenvolvimento de jogos com a finalidade de encontrar uma estrutura hierárquica conceitual (classes) desse domínio. Tanto o estudo [Tang and Hanneghan 2011] como o estudo desenvolvido por esta pesquisa buscam pelo desenvolvimento de uma estrutura conceitual em termos ontológicos a fim de especificar jogos sérios. A diferença reside no fato de que esse estudo usa um modelo de organização de multiagentes para descrever um modelo organizacional de agentes (Moise+) na estrutura dos bots do jogo, já aquele estudo não tem preocupação com a parte social e normativa dos agentes (pelo menos com a mesma intensidade).

No estudo [Neri et al. 2012] o paradigma de agente é usado para o desenvolvimento de um jogo de estratégia de simulação. Os estudos obtêm como resultado um jogo de guerra onde todos os personagens são descritos por agentes cognitivos e reativos. Esse estudo se destaca em relação aos demais devido ao fato de que Moise+ está fortemente presente no jogo. Apesar de que tanto o estudo de [Neri et al. 2012] como este estudo fazem uso de sistemas multiagentes e de representação de conhecimento, este estudo está preocupado em desenvolver um modelo conceitual para ser implementado em qualquer *game engine* já aquele estudo não teve essa preocupação.

O estudo [Kobti and Sharma 2007] apresenta o desenvolvimento de multiagentes que são capazes de aprender e evoluir conforme as habilidades do jogador. O experimento desenvolvido pelos pesquisadores foi em um jogo simples que emprega o uso de agentes evolutivos. Os pesquisadores concluem que existe um alto potencial na utilização de agentes evolutivos nos jogos. Tanto o estudo [Kobti and Sharma 2007] como o estudo desta pesquisa fazem uso de sistemas multiagentes, porém aquele estudo visa agentes evolutivos enquanto que esse estudo está preocupado com o uso de agentes cognitivos.

4. Metodologia

Foi necessário seguir as seguintes etapas para alcançar os objetivos deste trabalho: levantamento de dados, análise dos resultados, modelagem, desenvolvimento da ontologia e desenvolvimento do jogo. A etapa levantamento de dados tem por objetivo a coleta de dados das atividades profissionais que são necessárias à construção do jogo sério. A atividade escolhida foi a manutenção de equipamentos elétricos energizados devido à disponibilidade de um engenheiro especialista da área em uma empresa de energia. O foco está nas manutenções que ocorrem em subestações de transmissão. Foram feitas três entrevistas com engenheiro especialista em manutenção em linha viva de uma certa concessionária de energia. Nas sessões foram feitas diversas perguntas com a finalidade de compreender os seguintes conceitos: ferramentas básicas, profissionais envolvidos, função de cada profissional relacionado a manutenção, tipos de manutenção existente em diferentes contextos, procedimentos envolvendo substituição de pingo aéreo, procedimentos envolvendo substituição de conector, procedimento envolvendo substituição de isolador de pedestal, quais são os perigos envolvidos e quais são os procedimentos postos em prática em situações críticas. Todas essas informações foram gravadas em um pouco

mais de 182 minutos de áudio.

Além da entrevista, os pesquisadores acompanharam a execução de um procedimento de manutenção em linha viva. Enquanto a manutenção estava sendo realizada, os pesquisadores observaram as atividades, filmavam as operações e tomavam nota dos ocorridos. Os vídeos obtidos possuem um pouco mais de 3 horas de gravação. Outro ponto que está contido no levantamento foi a consulta a documentação técnica normativa que é privativa à concessionária de energia. Nessa documentação foi possível obter informações sobre procedimentos de manutenção, conceitos, jargões, funções profissionais e hierarquias profissionais.

A análise do resultado consistiu em transferir os dados contidos em documentação, áudio e vídeo para um arquivo de texto. A informação, nesse arquivo, ficou disposta em forma de uma lista. O próximo passo consistiu em identificar quais eram as informações necessárias e prioritárias a serem representadas na ontologia. Isso consiste em responder as seguintes questões: dentre os tipos de manutenção existentes (troca do isolador de pedestal, troca do conector e troca do pingo aéreo), por qual delas o pesquisador deve começar? Quais elementos são necessários e suficientes para criar o modelo? Quais papéis são relevantes para criar um modelo consistente? O que permeia as respostas a essas indagações é a simplicidade. Sendo assim, optou-se por filtrar informações que possibilitem o desenvolvimento de um jogo operacional e funcional. Isso consiste em escolher o caso de manutenção mais simples, o número mínimo de agentes (porém suficiente) para que a manutenção tenha prosseguimento e a condição de problema mais simples possível. A modelagem é outra etapa da metodologia dessa pesquisa. Para isso, os pesquisadores buscaram pela documentação de referência do Moise+ [Hübner et al. 2002a]. Com base nas informações filtradas e com base nas especificações do Moise+, foi feita uma representação manual do procedimento de manutenção em linha viva escolhido. Nessa etapa, também, foram identificados outros elementos que devem fazer parte da representação, porém não são abrangidos pela representação organizacional de Moise+ (ex. espaço, objetos, problemas que os objetos podem ter, ferramentas).

A estrutura obtida da modelagem foi representada em uma ontologia construída conforme a metodologia de desenvolvimento *Methontology* [Fernandez-Lopez et al. 1997]. O ponto de interesse consiste em identificar elementos comuns a todos os casos em análise, permitindo a criação de uma ontologia que seja a mais genérica possível, i.e. que permita a descrição de outros casos de manutenção que serão realizados futuramente. Para cada caso a ser modelado, é feita uma análise para verificar se a ontologia permite sua representação. Se não, a ontologia é modificada a fim de poder representar o caso em questão sem perder a expressividade que permite representar os casos anteriores. Boa parte do desenvolvimento da ontologia consistiu em reproduzir a estrutura lógica do Moise+ em linguagem OWL, linguagem esta escolhida para implementar a ontologia. Ainda assim há outros conceitos que fazem parte de elementos que não estão presentes na ontologia. O motivo pelo qual deva existir elementos na ontologia além da estrutura lógica do Moise+ reside no fato de que esses elementos devem fazer parte da representação do jogo. Uma vez que se encontrou todas as classes importantes para descrever o modelo, foi feito o uso de lógica de predicados para encontrar todas as relações que devem existir. Ainda nessa etapa é feita uma validação com profissionais da área. Essa validação envolve a apresentação

de um diagrama da ontologia ao Engenheiro de Manutenção a fim de verificar se os conceitos e relações representados são adequados e cobrem o esperado.

A última etapa consistiu em usar a ontologia para desenvolver o jogo. Para esse estudo os pesquisadores optaram por usar a *Game Engine Unreal 4* (porém a implementação do jogo sério pode ser feito em qualquer outra plataforma). A linguagem de programação dessa *Game Engine* é C++. Outro aspecto relevante da Unreal consiste no fato de que os *bots* do jogo são implementados por meio de uma árvore de comportamentos. Essa árvore exibe o que deve acontecer tendo em vista estímulos do ambiente.

Os pesquisadores testaram várias consultas que podem ser feitas a ontologia. As tentativas revelaram qual é o sequenciamento de consultas que devem ser feitas com a finalidade de fornecer as informações relevantes para o desenvolvimento do jogo. A lista a seguir exibe formalmente a transferência de conhecimento da ontologia para desenvolvimento do jogo.

- Consultar todos os objetos e problemas presentes nesses objetos. Implementar no código do jogo os objetos e os problemas que eles apresentam.
- Consultar todas as ferramentas, todos os agentes e suas respectivas restrições deônticas em relação as missões e objetivos. Implementar no código de cada agente o que eles devem fazer em determinado objetivo. Os objetivos orientam ao desenvolvedor quando os eventos associados a cada agente, a cada ferramenta, a cada objeto e a cada problema devem ocorrer.
- A estrutura de código de cada agente deve permitir que o personagem seja comandado pelo jogador em vez das regras definidas pela ontologia. Nessa condição, as regras deixam de comandar o comportamento dos agentes para supervisionar o comportamento do jogador que comanda aquele respectivo personagem.

5. Resultados

As entrevistas, os vídeos e a documentação técnica permitem concluir que os eletricistas trabalham com dois métodos de atuação: ao potencial e a distância. O método ao potencial é usado quando o executor acessa o potencial com o uso de uma roupa condutiva mantendo distância do potencial da terra. O método à distância ocorre quando o eletricista acessa o potencial por meio de um bastão isolante, estando isolado do potencial do condutor, porém se mantendo ao mesmo potencial da terra. Todas as manutenções são definidas nos moldes do método ao potencial ou do método a distância.

Foi identificado papéis de Supervisão, Execução e Observação. O executor realiza os procedimentos manuais. O supervisor analisa e comanda o prosseguimento das atividades e o Observador analisa a manutenção sobre o aspecto de segurança. Antes da manutenção acontecer é feita uma análise da situação e planejamento por Técnicos e Engenheiros especialistas no assunto.

Na etapa de análise de dados, optou-se por trabalhar com a representação da substituição de um conector pelo método ao potencial. Esse tipo de manutenção apresenta um número de objetivos menor quando comparado aos demais. O número de agentes envolvidos é um total de sete, um observador, um supervisor e cinco executores. Optou-se por isso tendo em vista de que essa quantidade de elementos é realística. As ferramentas, disposição dos objetos e problemas foram pensados para que esteja de acordo com uma

manutenção que possa ser executado por sete profissionais por intermédio da troca do conector por meio do método ao potencial.

A ontologia possui 5 classes principais; *Agent*, *Object*, *Problems*, *Tool* e *Organization*. A classe *Agent* contém os agentes que atuam no jogo. A classe *Object* descreve todos os objetos no espaço de ação dos agentes. A classe *Problems* apresenta todos os problemas possíveis que deverão ser corrigidos. Esse problema deve estar relacionado com o objetivo onde ocorre a correção. A classe *Tool* possui todas as ferramentas necessárias para executar a manutenção. A classe *Organization* e todas as suas respectivas subclasses apresentam as especificações lógicas definidas no modelo de representação Moise+. As relações presentes na ontologia orientam o desenvolvedor a desenvolver os *bots* do jogo como agentes.

A ontologia possui 1197 axiomas, 33 classes, 59 propriedades de objeto, 27 propriedades de dados, 170 instâncias, 101 axiomas de subclases, 7 subpropriedades e 8 funções inversas. A lógica dessa representação é do tipo ALCRIF(D). Sendo assim, possui linguagem atributiva (o que permite fazer negação anatômica, tem interseção conceitual, possui restrições universais e possui quantificação existencial limitada), possui negação conceitual complexa, possui disjunção, possui propriedades inversas e possui propriedades do tipo dado [Baader et al. 2003].

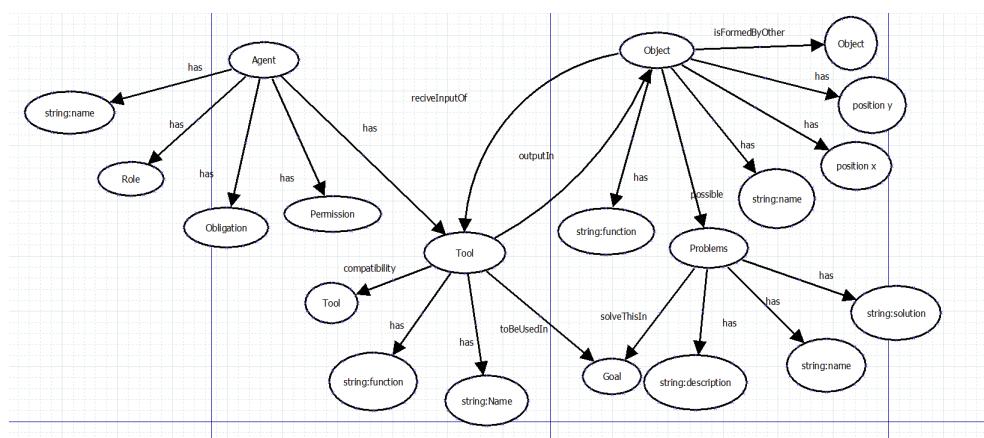


Figura 1. Descrição das classes Agent, Object, Problems e Tool.

A figura 1 apresenta o esquemático das relações de predicado entre as classes *Agent*, *Object*, *Problems* e *Tool*. O termo que está contido em um círculo, quando não recebe a expressão *string*: faz referência a uma classe. Quando recebe a expressão *string*, faz referência a um determinado tipo de dado. Os termos que estão fora dos círculos são os predicados. Classe para a qual a flecha sai é o Domínio e a classe na qual a flecha chega é a Imagem.

A lista a seguir apresenta as consultas que podem ser feitas a fim de poder implementar as soluções da ontologia no jogo. O primeiro elemento consiste por onde a consulta começa. O último elemento é o que deve ser fornecido ao usuário. Os elementos intermediários são consultas internas que devem necessariamente acontecer. A legenda da lista é Objetivo - *O*, Objetivo Paralelos - *OP*, Missão - *M*, Especificação Deôntica - *ED*, Função - *F*, Grupo - *G*, Compatibilidade - *C*, Link - *L*, Agente - *A*, Ferramenta - *Fe*, Objetos - *Ob*, Problema - *P*, Próximo Objetivo - *P.O.*

- $O \rightarrow OP.$
- $O \rightarrow M \rightarrow ED.$
- $O \rightarrow M \rightarrow ED \rightarrow F.$
- $O \rightarrow M \rightarrow ED \rightarrow F \rightarrow G.$
- $O \rightarrow M \rightarrow ED \rightarrow F \rightarrow C.$
- $O \rightarrow M \rightarrow ED \rightarrow F \rightarrow L.$
- $O \rightarrow A \rightarrow F.$
- $O \rightarrow A \rightarrow Fe.$
- $O \rightarrow Fe \rightarrow Ob.$
- $O \rightarrow Fe \rightarrow Ob \rightarrow P.$
- $O \rightarrow Fe \rightarrow Ob \rightarrow P.O.$

A partir da ontologia, os pesquisadores usaram as especificações para implementar o jogo. Foi possível obter uma versão operacional do jogo que reproduz a existência de todos os objetos, agentes, problemas e interações descritas na ontologia. As especificações do Moise+ também estão funcionando de forma operacional no jogo. A figura 2 apresenta o jogo em execução. Para uma primeira versão, os objetos foram representados como cubos, as ferramentas como cones e os problemas como "chamas azuis"(emissor de partícula simples). O que rege o comportamento dos bots são estruturas lógicas de regras baseadas no Moise+ (definida nas especificações deônticas). A lista a seguir exibe as regras de um dos *bots* implementados.

- *agentHasRole: roleExecutor*. Define o papel do agente.
 - *hasObligation: missaoProcedimentoDistanciaRoleExecutor*. Define a obrigação relacionada ao papel (*role*).
 - *hasPermission: missaoProcedimentoDistanciaRoleExecutor*. Define a permissão relacionada ao papel (*role*).
- *hasName: Executor01*. Define o nome relacionado ao agente.
- *hasTools: bastaoUniversal*. Define todas as ferramentas que o agente deve usar.

É importante ressaltar que o modelo representacional não leva em consideração as relações dinâmicas de adoção ou remoção de *schemas* como ocorre no JaCaMo.

6. Discussão

Ao descrever a manutenção em linha viva em uma ontologia inspirando-se em sistemas multiagentes normativos, tornou possível a modelagem das relações sociais, estruturais e deônticas. Sendo assim as relações normativas facilitaram o processo de definição do comportamento de cada agente. Isso, pois, as relações deônticas relacionam a função do agente em relação a missão. Esta, por sua vez, implica um conjunto de objetivos que devem ser atingidos pelo agente. A lógica deôntica estruturada nessa pesquisa não é útil apenas para delimitar o comportamento dos agentes, mas também possui utilidade na verificação do comportamento do jogador. Pois através da violação das normas do jogo torna-se possível verificar se o jogador está realmente preparado para assumir o cargo de interesse. Com base nessa linha de raciocínio, uma outra utilidade dessa aplicação implica identificar quais normas o jogador tem dificuldade de seguir.

No contexto da análise da manutenção é possível avaliar que a árvore de objetivos (definida na especificação estrutural) facilitou a identificação de procedimentos que são



Figura 2. Jogo em operação, jogador observa bot realizando um determinado procedimento

similares em diferentes manutenções. Como consequência há uma significativa redução no esforço necessário para realizar a descrição das atividades de novos casos (desde que esses casos se enquadrem nas mesmas restrições). Os objetivos exercem uma importância central dentro da representação presente na ontologia. Por esse motivo as consultas devem ser feitas sempre pelos objetivos. Uma consequência dessa situação consiste no fato de que os jogos resultantes dessa ontologia são orientados a objetivos. A relação entre os agentes e os objetivos ocorre por intermédio de lógica deôntica entre a função dos agentes e missões que encapsulam os objetivos.

A implementação da estrutura organizacional diretamente no código do jogo (sem utilizar Moise+) se deve ao fato de que ao analisar estudos que tentaram construir interfaces síncronas entre *Game Engines* comerciais com plataformas de Agentes como Jason, encontraram muitos problemas de implementação e de funcionamento [Dignum et al. 2009] [Jepp et al. 2010]. Como [Dignum et al. 2009] e como [Nareyek 2001] demonstram, a indústria de jogos eletrônicos, quando trabalham com agentes para descrever os personagens do jogo, fazem isso por meio de uma implementação no código do jogo. Esse é o motivo que levou à proposição da ontologia para especificação de um jogo sério de treinamento em manutenção. Outra vantagem de optar por essa escolha se deve ao fato de que o modelo proposto nesse estudo torna-se independente de uma plataforma específica de desenvolvimento de agentes ou de jogos. Futuramente, geradores de código específicos para cada plataforma poderão ser criados para geração de parte dos jogos sérios.

Os resultados indicam que a lógica de descrição (usando a sintaxe OWL-RDF) foi o suficiente para representar agentes normativos em jogos sérios. O uso do modelo de representação de uma organização inspirada em Moise+ foi o suficiente para tratar todas relações deônticas e sociais entre os agentes do estudo de caso. Porém, além disso, também foi necessário representar os agentes, as ferramentas, os objetos dispostos ao redor dos agentes e os Problemas que devem ser corrigidos pelos agentes. Todas essas representações se encontram em uma única ontologia em conjunto com as especificações sociais do Moise+. Ressalta-se que as instâncias podem ser de outro tipo de atividade

distinta do domínio elétrico.

7. Conclusão

O objetivo inicial deste trabalho consiste em desenvolver uma ontologia de especificação de jogos sérios que permite descrever personagens como agentes. O interesse nesse objetivo consiste em analisar quais representações são necessárias para realizar esse tipo de modelagem, quais linguagens são mais adequadas, qual é a capacidade de generalização dos resultados e qual é característica da lógica descritiva. Os resultados mostram que a lógica obtida possui alta capacidade de generalização, mostrou que os elementos do Moise+ foram suficientes para representar questões normativas do jogo, mostrou que foi necessário representar ferramentas, objetos e problemas para obter um modelo eficaz. Os próximos passos dessa pesquisa são: finalizar o desenvolvimento do jogo sério, validar o jogo por profissionais da área, implementar os outros casos de manutenção na ontologia, verificar modificações que devem ser feitas tanto na ontologia como no jogo e verificar como os profissionais de linha viva reagem ao jogo.

Referências

- Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.
- Boella, G., van der Torre, L., and Verhagen, H. (2006). Introduction to normative multi-agent systems. *Computation and Mathematical Organizational Theory, Special issue on Normative Multiagent Systems*, 12(2-3):71–79.
- Breuer, J. and Bente, G. (2010). Why so serious? on the relation of serious games and learning. *Journal for Computer Game Culture*, 4:7–24.
- Dignum, F., Westra, J., van Doesburg, W. A., and Harbers, M. (2009). Games and Agents: Designing Intelligent Gameplay. *International Journal of Computer Games Technology*, 2009:18.
- Fernandez-Lopez, M., Gomez-Perez, A., and Juristo, N. (1997). Methontology: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium*, pages 33–40, Stanford, USA.
- Guarino, N. (1998). *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 1st edition.
- Guarino, N., Oberle, D., and Staab, S. (2009). What is an ontology? In Staab, S. and Studer, R., editors, *Handbook on Ontologies*. Springer, second edition.
- Hayden, S. C., Carrick, C., and Yang, Q. (1999). A catalog of agent coordination patterns. In *Proceedings of the Third Annual Conference on Autonomous Agents*, AGENTS '99, pages 412–413, New York, NY, USA. ACM.
- Horling, B. and Lesser, V. (2005). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, pages 281–316.

- Hübner, J. F., Sichman, J. S., and Boissier, O. (2002a). A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In Bittencourt, Guilherme; Ramalho, G., editor, *Advances in Artificial Intelligence*, Lecture Notes in Computer Science, pages pp 439 – 448. Springer Berlin / Heidelberg.
- Hübner, J. F., Sichman, J. S., and Boissier, O. (2002b). A model for the structural, functional, and deontic specification of organizations in multiagent systems. In Bittencourt, G. and Ramalho, G. L., editors, *Advances in Artificial Intelligence*, pages 118–128, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Jepp, P., Fradinho, M., and Pereira, J. M. (2010). An agent framework for a modular serious game. In *2010 Second International Conference on Games and Virtual Worlds for Serious Applications*, pages 19–26.
- Kobti, Z. and Sharma, S. (2007). A multi-agent architecture for game playing. *2007 IEEE Symposium on Computational Intelligence and Games*, pages 276–281.
- Nareyek, A. (2001). Review: Intelligent agents for computer games. In Marsland, T. and Frank, I., editors, *Computers and Games*, pages 414–422, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Neri, J. R., Zatelli, M. R., Uez, D. M., and Callegaro, R. F. (2012). Aplicando conceitos de sistemas multiagentes na elaboração de um jogo de estratégia simulado. *RITA*, 19:34–58.
- Pons, L., Bernon, C., and Glize, P. (2012). Scenario control for (serious) games using self-organizing multi-agent systems. In *2012 IEEE International Conference on Complex Systems (ICCS)*, pages 1–6.
- Susi, T., Johannesson, M., and Backlund, P. (2007). Serious Games: An Overview. Technical report, GLS University of Wisconsin-Madison.
- Tang, S. and Hanneghan, M. (2011). Game content model: An ontology for documenting serious game design. In *2011 Developments in E-systems Engineering*, pages 431–436.
- Woolridge, M. and Wooldridge, M. J. (2001). *Introduction to Multiagent Systems*. John Wiley & Sons, Inc., New York, NY, USA.

Composição de Técnicas de Inteligência Artificial em uma Arquitetura Multinível para Emergência de Comportamentos em Agentes Cooperativos

João Rogério Vieira Neto¹, Jerusa Marchi¹

¹Departamento de Informática e Estatística – INE
Universidade Federal de Santa Catarina – UFSC
Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brasil

jrogerio.vn@gmail.com, jerusa.marchi@ufsc.br

Abstract. *Usually intelligent systems are complex systems where the accomplishment of the task demands the integrated and coordinated action of several agents. Besides that, the complexity of the individual actions can demands the application of several techniques. Digital games are among the problems presenting high complexity and whose solution demands the operation of several systems. This work presents a multilevel architecture implementation for composing cooperative agents. Each level of this architecture is developed using a specific artificial intelligence technique. In order to validate the proposal, the agents compete in a capture the flag game.*

Resumo. *Sistemas inteligentes são, em muitos casos, sistemas complexos, onde o cumprimento da tarefa pode demandar a ação integrada e coordenada de diversos agentes. Além disso, a complexidade das ações individuais pode demandar a utilização de diversas técnicas. Dentre a gama de problemas que apresentam alto grau de complexidade e cuja solução demanda diversos subsistemas destacam-se os jogos digitais. Este trabalho apresenta a implementação de uma arquitetura multinível para a composição de agentes cooperativos, que consiste em uma composição de técnicas de inteligência artificial. Para validar a proposta, os agentes competem em um jogo de captura à bandeira.*

1. Introdução

Em muitas soluções desenvolvidas utilizando técnicas de Inteligência Artificial (IA) a autonomia do sistema é uma qualidade desejada, onde o uso de agentes passa a ser comum. Além disso, quando a complexidade do sistema é alta somente um agente pode ser insuficiente para implementar a solução do problema, necessitando da ação coordenada de um grupo de agentes [Wooldridge 2002b, Jennings 2000]. Dependendo da complexidade das ações esperadas de cada agente, também torna-se necessária, a composição de diversas técnicas visando cobrir todos os aspectos do comportamento desejado. Por outro lado, jogos digitais são, em sua maioria, sistemas modulares e complexos, onde cada módulo é responsável por realizar tarefas específicas [McShaffry and Graham 2013, Schwab 2009], sendo portanto um excelente domínio para testar arquiteturas de agentes e a integração de técnicas IA.

Neste sentido, o objetivo deste trabalho é a implementação de uma arquitetura multinível onde cada nível utiliza uma técnica de inteligência artificial distinta. A ação coordenada dos níveis provê o comportamento inteligente individual e, em nível global, a ação coordenada dos agentes provê o comportamento inteligente do sistema.

O artigo está organizado da seguinte forma: na Seção 2 é apresentada a arquitetura multinível na qual a arquitetura implementada foi inspirada. Na seção 3 é descrito o cenário do jogo criado para instanciar os agentes. Na seção 4 apresenta-se a composição das técnicas escolhidas para a constituição dos agentes e o detalhamento de cada nível da arquitetura, bem como seu funcionamento global. A seção 5 apresenta a validação da implementação realizada, com os testes individuais e a integração final da arquitetura e sistema. Por fim, são feitas algumas considerações finais e são apresentadas algumas limitações encontradas.

2. Arquitetura Multinível

A arquitetura implementada é inspirada na arquitetura multinível proposta por Bittencourt [Bittencourt 1997] e integra as principais técnicas de IA, como Redes Neurais, Sistemas Fuzzy, Algoritmos Genéticos e Raciocínio Simbólico, cujo intuito é o desenvolvimento de um ser artificial autônomo, capaz de aprender com sua interação com o mundo, correlacionar fatos, armazenar e extrapolar situações vividas, tomar decisões e reorganizar seu próprio conhecimento. O modelo da arquitetura ampara-se nas seguintes hipóteses [Bittencourt and Marchi 2006]:

- Cognição é uma propriedade emergente de um processo cíclico e dinâmico baseado na interação de um conjunto de unidades funcionalmente independentes.
- Qualquer modelo da atividade cognitiva deve ser epistemologicamente compatível com a teoria da evolução.
- Aprendizado e atividade cognitiva estão fortemente relacionados, portanto, a modelagem cognitiva do agente deve depender do histórico do agente cognitivo.

Nesta proposta, Bittencourt estabelece a ação coordenada de três níveis de ciclos de pensamento/ações: um nível inferior denominado *nível reativo* que é composto por padrões extraídos de informações sobre o mundo externo, controles que produzem alguma ação neste mundo e uma população de cromossomos que unem percepção e ação. Tais cromossomos quando submetidos à um processo de seleção natural, no qual a função de fitness está associada com as emoções do agente, permite ao agente aprender e adaptar-se ao ambiente.

Um nível intermediário, denominado *nível instintivo* que possui um mecanismo de memória de longo prazo, de forma que quando o processo evolucionário avança no nível reativo e as situações começam a se repetir no mundo, torna-se possível identificar populações responsáveis por ações relevantes em uma determinada situação, abstrair suas propriedades e obter uma descrição geral de ação para uma determinada situação. Essas descrições são armazenadas pelo agente na memória de longo prazo e são acessadas mediante a percepção de situações similares.

Por fim, o terceiro e último nível, denominado *nível cognitivo* é encarregado da manipulação das descrições gerais obtidas no nível instintivo, aplicando funções cognitivas de dedução, abdução e indução. Este nível é constituído por duas atividades com-

plementares: o aprendizado através de descrições de situações relevantes, e a geração de novas estratégias de ação.

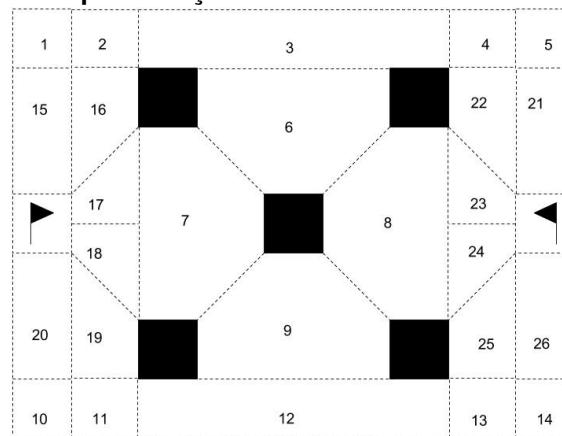
A proposta desta arquitetura é bastante ampla e genérica, o que torna sua implementação completa um sonho audacioso. Desta forma, para este trabalho, optamos por nos inspirar na idéia da coordenação das ações dos três níveis, porém fazendo uso de técnicas específicas voltadas ao domínio de um jogo de captura à bandeira. Desta forma, antes de introduzir as técnicas implementadas e a coordenação entre níveis, é importante definir o cenário de aplicação dos agentes.

3. Cenário de Aplicação

O sistema consiste em um jogo no estilo captura à bandeira, que acontece em uma pequena arena. Existem dois times, denominados time A e time B, cujos objetivos são a captura da bandeira inimiga e a defesa de sua própria bandeira. Vence a equipe que conseguir capturar a bandeira adversária primeiro. O mapa da arena, mostrado na figura 1, possui cinco obstáculos estáticos e diversas áreas delimitadas, denominadas regiões, por onde os agentes podem se movimentar. As áreas são enumeradas em uma sequência específica para ser usada na identificação da posição dos agentes e elaboração do plano. A caracterização das regiões é a seguinte:

- Região superior: áreas 1 a 5;
- Região central: áreas 6 a 9;
- Região inferior: áreas 10 a 14;
- Região de defesa do time A: áreas 15 a 20;
- Região de defesa do time B: áreas 21 a 26.

Figure 1. Representação das áreas e obstáculos da arena.



Cada time é composto por 3 agentes. Um agente pode atacar a bandeira adversária ou defender uma área, conforme suas competências (descritas na seção 4.1). Além disso, estabeleceu-se a figura de um agente líder, que define a estratégia do time e estabelece tarefas aos subordinados. As próximas seções apresentam a arquitetura específica desenvolvida para este domínio, descrevendo em linhas gerais as técnicas escolhidas e implementadas em cada nível, bem como a coordenação geral do sistema de agentes.

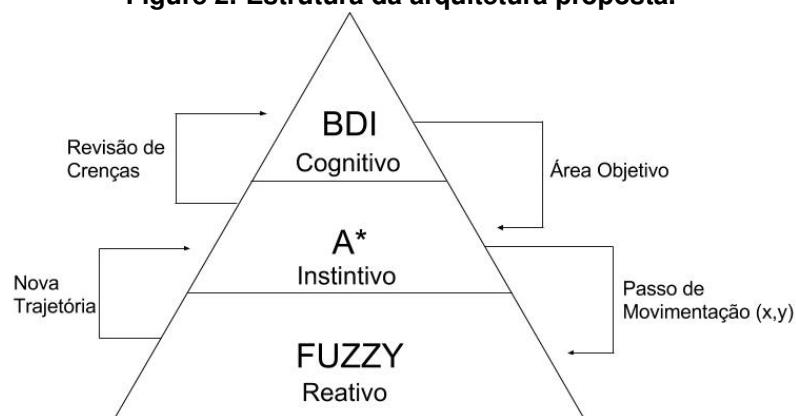
4. Modelagem dos agentes

A proposta do trabalho é a implementação de um sistema multiagente onde cada agente é definido por uma arquitetura em três níveis, conforme apresentado na seção 2, e cujos comportamentos devem atender às demandas do domínio de aplicação descrito na seção 3. Desta forma, é possível perceber que no nível cognitivo, o agente deve ser capaz de raciocinar sobre a estratégia do jogo, decidindo pelo ataque ou defesa; no nível instintivo, ele deve localizar-se na arena e coordenar seus movimentos conforme a decisão no nível cognitivo. De forma reativa, o agente deve buscar evitar colisões com os obstáculos estáticos, bem como deve *driblar* os oponentes ou deve colocar-se de forma a impedir que um agente adversário segue até a bandeira. Desta forma, tem-se para cada nível:

- Nível cognitivo: Para implementar o nível cognitivo foi escolhido o Modelo BDI - *Belief, Desire, Intention* [Wooldridge 2002a] - pela facilidade em estabelecer planos e definir o comportamento do agente em alto nível, bem como pela simplicidade em estabelecer novos objetivos dada a dinâmica do jogo, através de seu ciclo de raciocínio bem estabelecido.
- Nível instintivo: Neste nível é utilizada uma busca em grafos com algoritmo A*, que opera sobre grafos valorados realizando estimativa de custo para atingir um vértice destino. O algoritmo utiliza uma estratégia gulosa associada a uma função heurística para a estimativa de custo futuro [Russel and Norvig 2003], de forma a encontrar o melhor caminho para uma determinada área na arena.
- Nível reativo: O nível reativo foi implementado com o auxílio de um controlador fuzzy [Tanscheit 2003, Chen and Pham 2001], que recebe dados sobre a distância do agente a um determinado obstáculo e dependendo do comportamento (ataque ou defesa) impõe o próximo passo do agente.

A figura 2 ilustra os níveis da arquitetura e as técnicas implementadas em cada nível e as seções seguintes descrevem os detalhes de implementação de cada nível.

Figure 2. Estrutura da arquitetura proposta.



4.1. Nível Cognitivo

O nível cognitivo do agente é modelado utilizando *AgentSpeak* através do interpretador Jason, e suas ações são integradas com a aplicação Java utilizada para implementar a interface do jogo. Os agentes do sistema possuem crenças relativas às suas competências,

à sua posição atual e às ordens recebidas do agente líder. As crenças relativas às competências de ataque e defesa dos agentes são fixas, ou seja, durante toda a execução os agentes sempre vão defender ou atacar as mesmas áreas. As crenças relativas à posição atual e às ordens vindas do agente líder mudam no processo de revisão de crenças dos agentes.

A comunicação dos agentes ocorre por meio de troca de mensagens individuais e *broadcast*. A troca de mensagens entre os agentes do mesmo time é sempre individual, o *broadcast* só é emitido quando um agente muda sua crença de posição atual, fazendo-se necessário informar a todos os agentes do ambiente sobre sua nova posição. As tabelas 1 e 2 mostram as diferenças nas configurações de cada time, o que distingue as competências de cada um.

Table 1. Configuração do Time A

	Time A		
	Líder	Subord 1	Subord 2
Crença de defesa:	áreas 15 à 20	áreas 16,18 e 20	não
Crença de ataque	não	não	sim
Elabora plano	sim	não	não

O líder do time *A* fica observando a movimentação inimiga e identifica as áreas que precisam ser protegidas baseado na posição atual dos inimigos. Uma vez identificadas as áreas que devem ser protegidas, ele ordena que o subordinado 1 proteja a área que é capaz, enquanto o próprio líder protege outra. Enquanto o líder e o subordinado 1 estão sempre defendendo, o subordinado 2 está sempre focado em atacar.

Table 2. Configuração do Time B

	Time B		
	Líder	Subord 1	Subord 2
Crença de defesa:	áreas 21 à 26	áreas 22 e 23	24 e 25
Crença de ataque	não	sim	sim
Elabora plano	sim	não	não

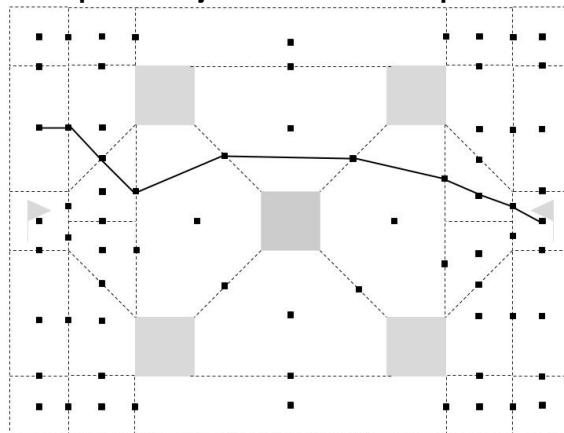
No caso do time *B*, o líder fica observando a movimentação inimiga, ao identificar as áreas que precisam ser protegidas, ele tenta alocar os subordinados para defendê-las, caso um subordinado não seja capaz de defender nenhuma delas ele ordena que esse agente ataque. Caso os dois agentes sejam capazes de proteger a região, ele ordena que os dois protejam e seu time ficará sem atacar enquanto executa essa estratégia de defesa.

Ao elaborar uma estratégia, o agente líder emite uma ordem para seu subordinado, que por sua vez, gera uma ação Java a ser executada na aplicação. Quando um agente da aplicação Java recebe uma ação vinda de seu modelo cognitivo em Jason, é um indicativo que a partir de agora ele possui uma posição alvo definida. Então inicia o processo deliberativo de sua ação, ou seja, essa posição é passada para o nível instintivo gerar uma trajetória até a área desejada.

4.2. Nível Instintivo

O objetivo do nível instintivo é gerar uma trajetória da posição atual do agente até uma posição de destino. Para efetuar essa tarefa é utilizado o algoritmo de busca A*, que opera percorrendo um grafo valorado gerado sobre as áreas da arena como exemplificado na figura 3.

Figure 3. Exemplo de trajetória através dos pontos de controle .



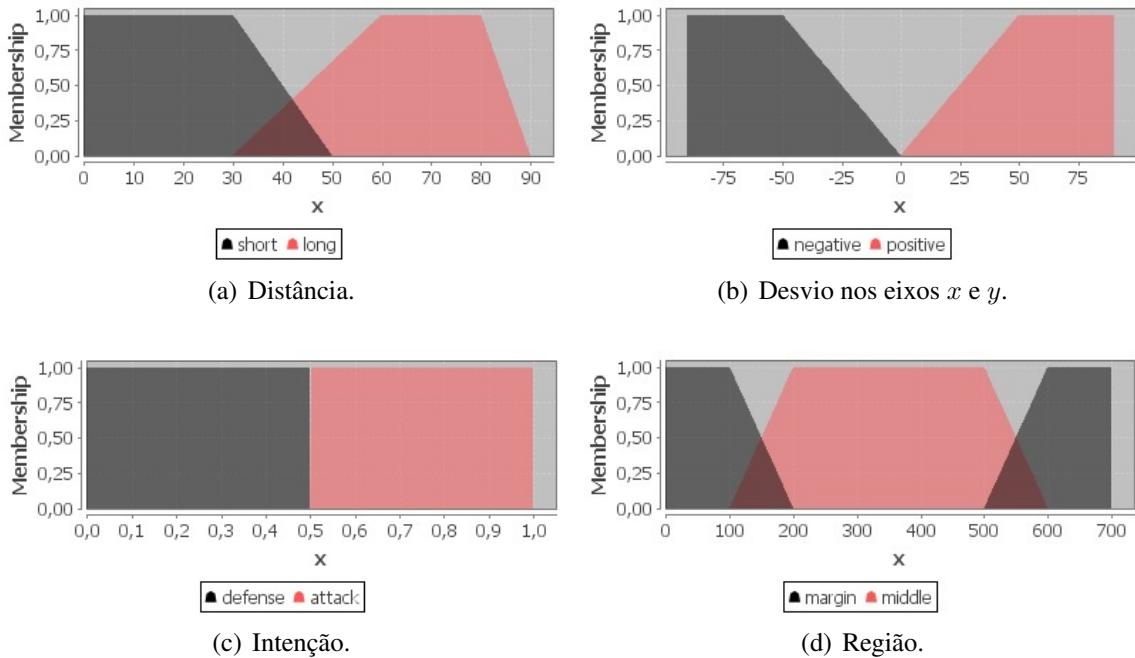
Para cada área da arena são definidos alguns pontos de controle. Esses pontos são coordenadas bidimensionais que representam posições intermediárias que o agente usa para se deslocar, ou seja, representam os vértices do grafo utilizado como base para o planejamento da trajetória. Há também a necessidade de se calcular a distância entre dois pontos em dois momentos, e para isso usa-se a distância Euclideana. O primeiro momento é na geração do grafo, onde o peso de cada aresta consiste na distância entre seus vértices. No segundo momento, cálculo da distância entre pontos é utilizado como função heurística para estimar a proximidade com a posição destino da trajetória.

A cada atualização temporal no sistema o agente deve realizar um passo de movimentação, ou seja, um deslocamento parcial dentro da trajetória. Sempre que um passo de movimentação é gerado, ele é enviado ao nível reativo para que esse faça eventuais alterações, e também é feita uma verificação para identificar uma transição entre duas áreas. Ao identificar uma transição entre duas áreas, o nível instintivo deve informar ao nível cognitivo, para que este faça a revisão de crenças e elabore uma nova estratégia.

4.3. Nível Reativo

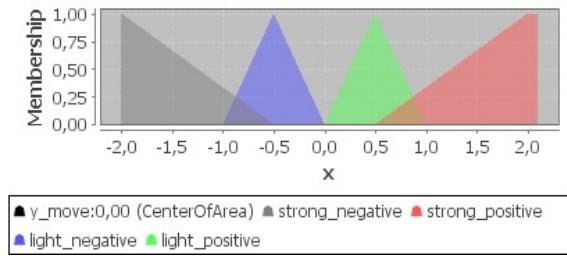
Para ser capaz de realizar as alterações no passo da trajetória, o nível reativo da arquitetura do agente utiliza um controlador fuzzy. A cada atualização da posição do agente, o nível recebe o passo de movimentação a ser aplicado e gera valores para as variáveis de entrada do controlador. O controlador é composto por cinco variáveis de entrada e duas de saída. As variáveis de entrada são apresentadas nas figuras 5(a), 5(b), 5(c) e 5(d), onde é possível ver os universos de discursos definidos para cada variável bem como os conjuntos fuzzy definidos para cada universo.

A variável distância trata da distância entre o agente e seu inimigo. A variável desvio é assinalada para os eixos x e y e indica a diferença na posição do agente com relação à trajetória estabelecida. A variável intenção define se o agente está atacando ou

Figure 4. Conjuntos fuzzy de entrada.

defendendo. A variável região define se o agente está no meio ou na borda de uma dada região.

As duas variáveis de saída representam o valor do passo de movimentação, sendo uma variável para cada eixo (x e y), apresentada na figura 5.

Figure 5. Conjunto fuzzy de saída: movimentação nos eixos x e y .

Por fim, as regras fuzzy cobrem três casos: ataque pela região central, ataque pelas regiões de borda e defesa. No caso do ataque o agente mantém o valor do passo em um eixo, enquanto gera um valor de desvio para o outro eixo. O valor da movimentação no eixo selecionado será sempre no sentido contrário ao deslocamento do inimigo, com o objetivo de fazer o desvio se distanciando dele, e será um movimento brusco ou suave dependendo da distância entre ambos. Para a situação de ataque pela região central da arena, o eixo y é selecionado para desvio e o passo é mantido para o eixo x . No caso de um ataque pelas regiões de borda, tanto a verificação do desvio, como o valor do passo de movimentação levarão em consideração o eixo x . Quando a intenção do agente é de defesa, o comportamento é mais simples e não depende de tantas variáveis de entrada. As regras que levam em conta essa intenção basicamente vão gerar valores para a variáveis de saída que farão o agente defensor bloquear o caminho do atacante, e seguí-lo em caso

de uma esquiva. O comportamento de defesa independe da região em que o agente se encontra.

O controlador tem um total de 16 regras, sendo 4 regras para cada caso. Por razões de espaço apresenta-se abaixo uma regra de cada conjunto, sendo a Regra 1 relativa ao ataque pela região central, a Regra 2 relativa ao ataque pela região de borda e a Regra 3 uma regra de defesa.

Regra 1: **IF** region **IS** middle **AND** distance **IS** short **AND**
yDeviation IS negative AND intention IS attack THEN
yMove IS strongPositive;

Regra 2: **IF** region **IS** margin **AND** distance **IS** short **AND**
xDeviation IS negative AND intention IS attack THEN
xMove IS strongPositive;

Regra 3: **IF** xDeviation **IS** negative **AND** intention **IS** defense
THEN xMove **IS** lightNegative;

4.4. Coordenação dos níveis funcionais da arquitetura

Dados os seus três níveis, as ações devem ser coordenadas de forma a assegurar o comportamento do agente. Desta forma há dois fluxos: o fluxo *top-down* e o fluxo *bottom-up*. O fluxo *top-down* inicia-se no nível cognitivo, que ao gerar uma intenção passa uma área alvo para o nível instintivo, que por sua vez, após gerar a trajetória envia o passo de movimentação para o nível reativo aplicar eventuais alterações. Eventualmente, o agente vai ter que refazer sua trajetória, além de fazer revisão de crença para alterar seu objetivo. Para isso é necessário definir também uma comunicação no sentido contrário (fluxo *bottom-up*) entre os níveis da arquitetura.

Quando o nível reativo é ativado, ele faz alterações no passo de movimentação a cada ciclo temporal da aplicação. Mas ele deve ser capaz de perceber quando não há mais necessidade de fazer alterações na trajetória. Então, quando o nível reativo identifica que a posição do agente ultrapassou a de seu inimigo no eixo definido, este indica ao nível instintivo que ele deve elaborar uma nova trajetória, partindo da posição atual do agente, e mantendo o mesmo destino. Por sua vez, o nível instintivo, após a aplicação de um passo de movimentação, deve sempre verificar se a mudança de posição resultou em uma transição para outra área da arena. Sempre que for identificada uma transição, o nível instintivo envia uma notificação para o nível cognitivo, para que este faça revisão de suas crenças, e adicione uma crença da sua nova posição atual.

Por fim, o nível cognitivo ao atualizar sua crença de posição atual, irá enviar uma mensagem *broadcast* aos demais agentes do ambiente, informando sua movimentação. Ação esta que fará com que o líder inimigo refaça o plano de ação de seu time, completando o processo inverso das ações da arquitetura e recomeçando o fluxo *top-down*.

5. Validação

Para realizar a validação da estrutura proposta, foram executados testes isolados de cada um dos níveis e testes de integração 2 a 2 e por fim a integração completa do sistema. Por razões de espaço, apresentaremos um cenário de teste do nível cognitivo, um cenário de teste de integração entre o cognitivo e o instintivo - onde é possível ver o agente criando a

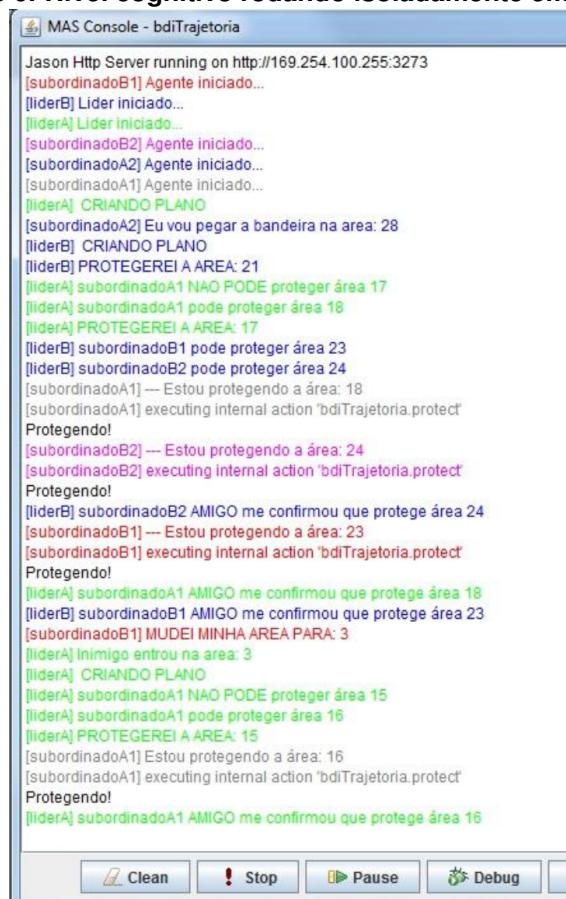
trajetória baseado no objetivo gerado no nível cognitivo e também a alteração da estratégia dos times; e um cenário de teste da arquitetura como um todo, onde é possível ver todas as suas propriedades, além de realizar todas as ações do cenário anterior, o agente nessa execução realiza alterações na trajetória de acordo com a necessidade, e posteriormente retoma sua trajetória.

Para as execuções a seguir são instanciados dois times, o time *A* está representado pela cor azul é instanciado no lado esquerdo da arena, enquanto o time *B* é representado pela cor vermelha e é instanciado ao lado direito da arena.

5.1. Cenário 1: Teste do Nível Cognitivo

O primeiro cenário a ser apresentado é do nível cognitivo funcionando isoladamente. Primeiramente os agentes são iniciados e os líderes de cada time elaboram um plano inicial, como na figura 6 são mostradas as ações acontecendo na saída da execução. Enquanto o líder *A* e subordinado *A*₁ defendem as áreas 17 e 18 respectivamente, o subordinado *A*₂ ataca a bandeira inimiga na área 28. Já o plano inicial elaborado pelo líder do time *B* consiste em ordenar ao subordinado *B*₁ defender a área 23, ao subordinado *B*₂ defender a área 24, enquanto ele próprio defende a área 21.

Figure 6. Nível cognitivo rodando isoladamente em Jason.



```

MAS Console - bdiTrajetoria
Jason Http Server running on http://169.254.100.255:3273
[subordinadoB1] Agente iniciado...
[liderB] Lider iniciado...
[liderA] Lider iniciado...
[subordinadoB2] Agente iniciado...
[subordinadoA2] Agente iniciado...
[subordinadoA1] Agente iniciado...
[liderA] CRIANDO PLANO
[subordinadoA2] Eu vou pegar a bandeira na area: 28
[liderB] CRIANDO PLANO
[liderB] PROTEGEREI A AREA: 21
[liderA] subordinadoA1 NAO PODE proteger area 17
[liderA] subordinadoA1 pode proteger area 18
[liderA] PROTEGEREI A AREA: 17
[liderB] subordinadoB1 pode proteger area 23
[liderB] subordinadoB2 pode proteger area 24
[subordinadoA1] — Estou protegendo a area: 18
[subordinadoA1] executing internal action 'bdiTrajetoria.protect'
Protegendo!
[subordinadoB2] — Estou protegendo a area: 24
[subordinadoB2] executing internal action 'bdiTrajetoria.protect'
Protegendo!
[liderB] subordinadoB2 AMIGO me confirmou que protege area 24
[subordinadoB1] — Estou protegendo a area: 23
[subordinadoB1] executing internal action 'bdiTrajetoria.protect'
Protegendo!
[liderA] subordinadoA1 AMIGO me confirmou que protege area 18
[liderB] subordinadoB1 AMIGO me confirmou que protege area 23
[subordinadoB1] MUDEI MINHA AREA PARA: 3
[liderA] Inimigo entrou na area: 3
[liderA] CRIANDO PLANO
[liderA] subordinadoA1 NAO PODE proteger area 15
[liderA] subordinadoA1 pode proteger area 16
[liderA] PROTEGEREI A AREA: 15
[subordinadoA1] Estou protegendo a area: 16
[subordinadoA1] executing internal action 'bdiTrajetoria.protect'
Protegendo!
[liderA] subordinadoA1 AMIGO me confirmou que protege area 16

```

Após os agentes executarem as ações definidas no plano inicial, o subordinado *B*₁ muda sua posição para área 3, emitindo uma mensagem *broadcast* para os demais agentes. Ao receber a mensagem o líder *A* elabora um plano de ação de seu time para

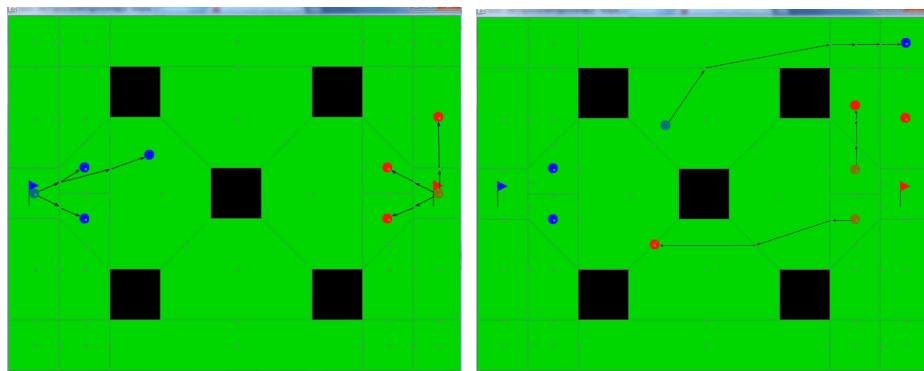
reagir à ação do agente inimigo. Finalizando com sucesso um ciclo de ações dentro do nível cognitivo.

5.2. Cenário 2: Teste de integração - níveis cognitivo e instintivo

Após os agentes serem instanciados, os líderes de cada time elaboram sua estratégia inicial. Como mostra a figura 7(a), o líder do time *A* e um de seus subordinados defendem as áreas centrais de acesso à sua bandeira, enquanto o outro subordinado se move em direção à região em que se encontram os inimigos. A estratégia do time *B* é totalmente defensiva, os agentes subordinados defendem as áreas centrais de acesso à sua bandeira, enquanto o líder defende uma das áreas de acesso superiores.

As áreas objetivo das trajetórias vêm do nível cognitivo. Para este cenário, o subordinado do time *A* que se desloca para a região inimiga, tem como objetivo a área do canto superior direito da arena. Sempre que este agente atravessa de uma área para outra é emitido um *broadcast* avisando os demais agentes da aplicação, então, no momento que este agente passa para a região superior da arena, o líder do time *B* identifica a necessidade de alterar o plano de ação. A alteração do plano de ação do time *B*, ilustrada na figura 7(b) consiste em ordenar ao subordinado *B*₁ que proteja a área de acesso superior, enquanto o subordinado *B*₂ deve atacar a bandeira inimiga.

Figure 7. Integração dos níveis cognitivo e instintivo.



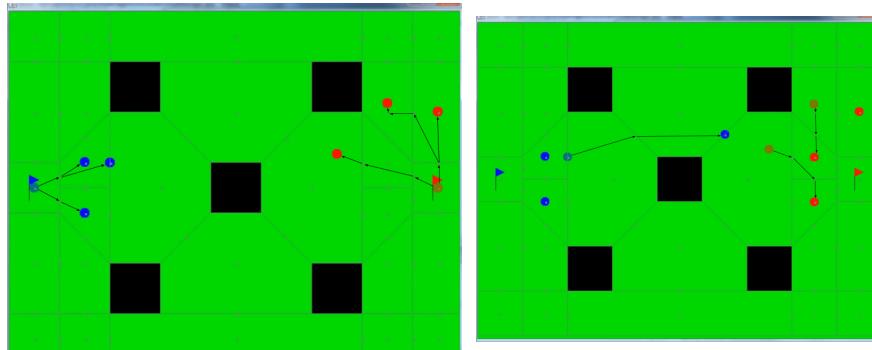
(a) Times executando a estratégia inicial.

(b) Time *B* alterando sua estratégia.

Portanto, o cenário mostra que os dois níveis da aplicação integrados geram o comportamento que se esperava dos agentes, mas ainda falta o terceiro nível da arquitetura para que a integração seja validada por completo.

5.3. Cenário 3: Integração completa dos três níveis

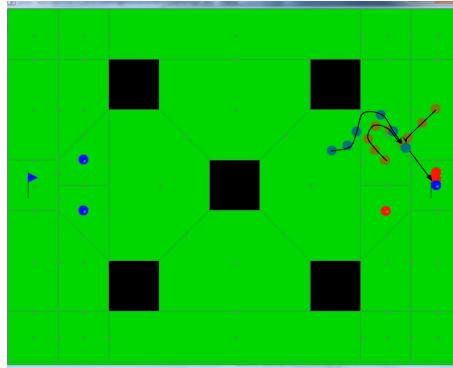
No último cenário, a estratégia inicial do time *B* consiste em defender as áreas superiores, enquanto um de seus subordinados ataca pela região central. O time *A* inicialmente defende as áreas de acesso centrais enquanto seu subordinado ataca pela região central como mostra a figura 8(a). Ao receber a informação via *broadcast* de que o subordinado do time *A* entrou na região central, o líder do time *B* reformula seu plano de ação, passando a ter uma postura totalmente defensiva, como mostra a figura 8(b), quando então ordena que seus dois subordinados defendam as áreas centrais de acesso à sua bandeira.

Figure 8. Integração dos três níveis.

(a) Times executando a estratégia inicial.

(b) Time *B* alterando sua estratégia.

O subordinado *A* persiste na sua movimentação de ataque pela região central, ativando por fim, o nível reativo, como mostra a figura 9. Neste momento, a cada ciclo temporal, o controlador *fuzzy* é executado e passo de movimentação é alterado conforme as regras nebulosas. Sendo assim, o subordinado do time *A* começa a se deslocar em direção contrária ao posicionamento de seu inimigo, enquanto o agente do time *B* começa a segui-lo. Ao verificar que conseguiu passar por seu inimigo, o agente do time *A* reformula sua trajetória com destino à bandeira inimiga. Nesse momento o líder do time *B* identifica a proximidade do inimigo e passa a segui-lo também, mas sem sucesso em alcançá-lo. Dessa forma o agente do time *A* consegue chegar à bandeira inimiga, finalizando a execução.

Figure 9. Agentes reagindo à movimentação inimiga.

Nesse cenário é possível perceber todas as propriedades da arquitetura. No sentido *top-down*, os times formulam suas estratégias e os agentes geram áreas de destino. Com um destino definido o nível instintivo gera uma trajetória, e o nível reativo altera o passo de movimentação quando identifica a proximidade de um inimigo. E no sentido *bottom-up*, quando o agente identifica que conseguiu desviar do inimigo, imediatamente indica ao nível instintivo que deve ser gerada uma nova trajetória. Por fim, ao transitar de uma área para outra, o nível instintivo comunica ao cognitivo a alteração da posição atual, que acontece quando o time *B* muda sua estratégia para uma postura totalmente defensiva.

6. Conclusão

O objetivo deste trabalho foi a implementação coordenada de técnicas distintas de IA em uma arquitetura multinível, onde cada nível torna-se responsável por parte do comportamento do agente e suas ações integradas permitem a elaboração de um comportamento inteligente mais elaborado. Como visto na validação, a proposta alcançou o comportamento esperado dos agentes. Foram fatores limitantes do trabalho alguns problemas decorrentes da integração Jason/Java. Dentre os problemas encontrados, estão: (i) interferência na comunicação dos agentes - o *loop* principal do jogo, implementado em Java, provocou problemas na troca de mensagens entre os agentes modelados em Jason. Ao elaborar um plano, os líderes emitiam mensagens, mas seus subordinados não as recebiam. Este problema foi solucionado através da criação de uma ação Java que os líderes emitem para interromper a execução do *loop*, retomando-a após realizada a comunicação. (ii) Referência nula completa ou incompleta na inicialização dos agentes - na ocorrência deste erro, os agentes do sistema não são, total ou parcialmente, inicializados, resultando em times incompletos ou na arena vazia. Este problema acontece de forma aleatória e não nos foi possível identificar a sua causa. No entanto, tais erros estão relacionados com a integração Java/Jason, e não com o funcionamento da arquitetura, que demonstrou-se estável em todas as execuções corretamente inicializadas.

Como trabalhos futuros, sugere-se a investigação mais profunda da integração Java/Jason visando sanar os problemas citados. Neste trabalho foi visto que é possível utilizar a arquitetura em um ambiente específico, mas não foi atribuído nenhum requisito de desempenho nas ações dentro das regras do jogo, o que poderia ser melhor explorado. Ainda é possível estudar melhor a composição das técnicas, com o objetivo de fazer implementações mais complexas e por fim, seria interessante inserir alguma forma de aprendizado com o objetivo de fazer com que a arquitetura apresente maior competitividade.

References

- Bittencourt, G. (1997). In the quest of the missing link*. *IJCAI*.
- Bittencourt, G. and Marchi, J. (2006). *Artificial Cognition Systems*, chapter An Embodied Logical Model for Cognition, pages 27–64. IDEA Group Inc.
- Chen, G. and Pham, T. T. (2001). *Introduction to Fuzzy Sets, Fuzzy Logic and Fuzzy Control Systems*. CRC Press.
- Jennings, N. R. (2000). On agent-based software engineering. *Artificial Intelligence*, 117.
- McShaffry, M. and Graham, D. (2013). *Game Coding Complete*. Course Technology, a part of Cengage Learning, 4th edition.
- Russel, S. J. and Norvig, P. (2003). *Artificial Intelligence*. Prentice Hall, Upper Saddle River, New Jersey, 2nd edition.
- Schwab, B. (2009). *AI Game Engine Programming*. Course Technology, a part of Cengage Learning, 2nd edition.
- Tanscheit, R. (2003). Sistemas fuzzy. *Anais de Minicursos do VI SBAI*.
- Wooldridge, M. (2002a). Intelligent agents. *Multi-Agent Systems and Applications II*.
- Wooldridge, M. (2002b). *An Introduction to Multiagent Systems*. John Wiley and Sons.

Impacto da Confiança em Simulação Baseada em Agentes para Cadeias de Suprimentos

André Domingues da Silva Jalbut , Jaime Simão Sichman

¹Laboratório de Técnicas Inteligentes (LTI)
Escola Politécnica (EP)
Universidade de São Paulo (USP)
andre.jalbut@gmail.com, jaime.sichman@usp.br

Abstract. Companies in supply chains have a goal to optimize their productivity, and hence their profits. One way to study the behavior of these chains is to simulate them using multi-agent systems; a model used in the literature is the Beer Game. In this work, we add multiple agents in the levels of the Beer Game to evaluate the local and global performance of the suppliers by using profiles based either on trust or on price. We measure the impact of using a selection policy based on trust in the agents' profit.

Resumo. Empresas em cadeias de suprimentos tem como objetivo maximizar suas produtividades, e consequentemente seus lucros. Uma maneira de estudar o comportamento destas cadeias é simulá-las utilizando sistemas multi-agentes; um modelo adotado na literatura é o Beer Game. Neste trabalho, adicionamos múltiplos agentes nos níveis do Beer Game para avaliar a eficiência local e global dos fornecedores, utilizando perfis baseados em confiança ou em preço. Medimos o impacto de usar tal política de seleção baseada em confiança no lucro dos agentes.

1. Introdução

A meta principal de um negócio é o lucro [Friedman 2007]. No contexto de empresas interagindo em uma cadeia de suprimentos (SC)¹, parcerias baseadas em relações de confiança podem ser mais lucrativas do que aquelas baseadas em mecanismos de oferta e demanda. Esta tese é alicerçada na constatação de que quanto maior a confiança de um consumidor em seus fornecedores, maior a responsividade destes, e portanto maior o ganho para a SC [Handfield and Bechtel 2002].

Cadeias de suprimentos são definidas como o conjunto de organizações, atividades, informação e recursos envolvidos na movimentação de um produto ou serviço de fornecedor para consumidor [Nagurney 2006]. O estudo do gerenciamento destas cadeias (SCM)² vem sendo alvo de interesse crescente visando obter vantagens competitivas para o mercado por meio de melhorias em seus processos [Croom et al. 2000]. A Figura 1 [Lambert and Cooper 2000] ilustra um exemplo de uma SC.

¹Do inglês *supply chain*.

²Do inglês *supply chain management*.

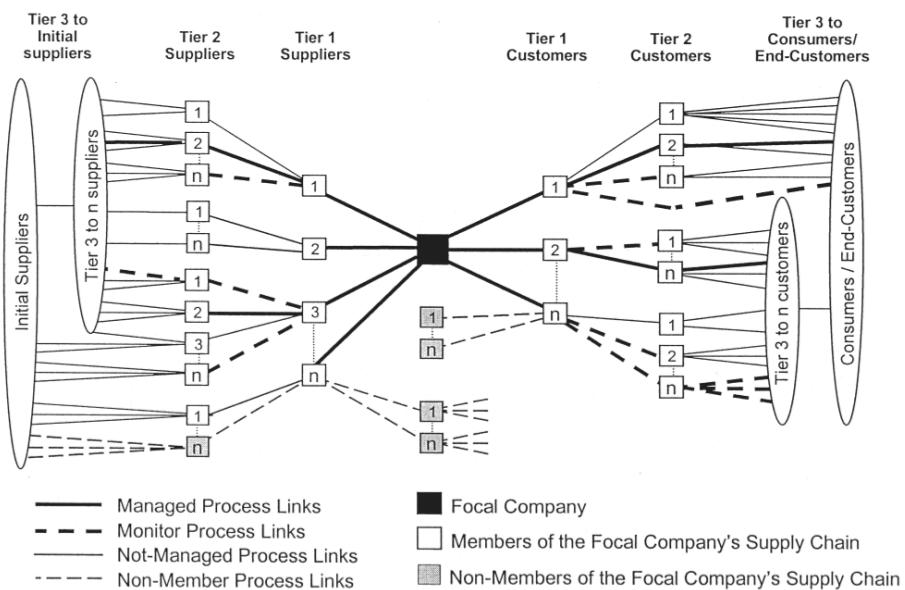


Figura 1. Exemplo de cadeia de suprimentos [Lambert and Cooper 2000].

Confiança é definida como a predisposição de um agente a se colocar em uma situação vulnerável em relação a outro esperando que este lhe proporcione algum benefício em contrapartida [Mayer et al. 1995]. Uma vez que uma SC é composta por firmas individuais colaborando para servir consumidores finais, sua eficácia é altamente dependente da confiança entre os parceiros da rede [Vlachos and Bourlakis 2006].

Outro conceito importante é o de sistemas multi-agentes (MAS)³. Este é um ramo da inteligência artificial que estabelece a representação de um sistema como a interação entre múltiplos agentes inteligentes, cada um avaliando alternativas diferentes e tomando suas próprias decisões, dentro de um contexto definido por restrições locais e externas [Wooldridge 2000].

Em [Swaminathan et al. 1998], os autores propõem simular SCM com o auxílio de modelagem baseada em agentes (ABM)⁴, o que permite que a performance da cadeia seja avaliada sob perspectivas organizacionais distintas. Em [De La Fuente and Lozano 2007], pesquisadores defendem a escolha de ABM para simular SCM por este último ser um problema fisicamente distribuído em cada agente pode considerar tanto interesses próprios como de toda a cadeia, e também um problema altamente complexo, influenciado pela interação entre diversas variáveis.

Uma forma de modelar o comportamento dos agentes em meio a uma SC é valendo-se da estrutura do *Beer Game* (seção 2), um jogo de tabuleiro criado no âmbito de SCM e bastante citado na literatura. Em [Kim 2009], por exemplo, utilizam-se as regras deste jogo em sua simulação para modelar a atuação dos agentes interagindo em uma SC.

Este trabalho tem por objetivo analisar o impacto da confiança no contexto de uma SC sujeita às normas do *Beer Game*, simulando empresas com perfis de atuação distintos interagindo entre si.

³Do inglês *multi-agent systems*.

⁴Do inglês *agent-based modeling*.

2. Beer Game

O *Beer Game*⁵ é um jogo de tabuleiro projetado por Forrester com dois objetivos: entender SCM e demonstrar na prática o chamado efeito chicote (BWE)⁶ [Sterman 1992]. Tal efeito, descrito inicialmente em [Forrester 1997], é um fenômeno que ocorre em SCs. Consiste na amplificação da variância dos pedidos feitos entre os níveis da cadeia, no sentido do consumidor final para a fábrica. Este efeito, demonstrado matematicamente em [Lee et al. 1997], é responsável segundo este último autor por aumentos drásticos em custos, entre eles o de matéria prima, o de manufatura, armazenamento e transporte.

No *Beer Game*, times de 4 jogadores competem entre si, sendo que cada time representa uma cadeia de suprimentos e cada membro do time exerce o papel de um de seus 4 níveis: fábrica, distribuidor, atacadista e revendedor. O objetivo de cada time é gerenciar o estoque em face de demanda externa desconhecida, de modo a minimizar os custos acumulados na soma dos níveis da cadeia. Cada equipe participante tem à sua disposição seu próprio tabuleiro. Nele, cada membro do grupo tem seu estoque e carregamentos a receber representados por marcadores, e pedidos anotados em folhas de papel viradas para baixo, como representado na Figura 2.

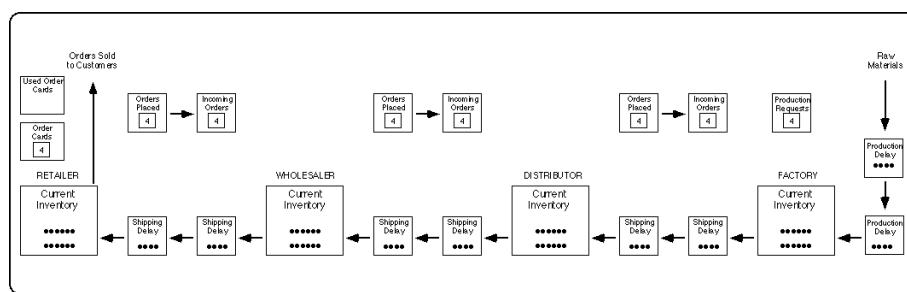


Figura 2. Disposição do tabuleiro para o *Beer Game* [Sterman 1992]

Pedidos demoram uma rodada para chegar ao destinatário. Cada jogador dimensiona e envia pedidos para o jogador que controla o nível superior, exceto a fábrica, que envia ordens para sua própria linha de produção. Cada jogador também recebe e deve atender aos pedidos recebidos do nível inferior, exceto o revendedor, que obtém seus pedidos de uma pilha de papéis virados para baixo. Esta pilha representa a demanda do consumidor final e possui a mesma ordenação e valores para todos os grupos.

Carregamentos demoram duas rodadas para chegar ao destinatário. Cada jogador retira de seu estoque e envia carregamentos para atender aos pedidos do jogador que controla o nível inferior. Cada jogador também recebe os carregamentos enviados pelo nível superior e os deposita em seu próprio estoque, exceto a fábrica, que recebe os carregamentos de sua própria linha de produção. Caso um jogador não disponha de estoque suficiente para atender o último pedido recebido, ele envia um carregamento com o restante do seu estoque, e a parcela devida é anotada para ser enviada na próxima rodada, somada ao próximo pedido a ser satisfeito.

O jogo se dá em um número fixo de rodadas, cada uma contendo uma sequência

⁵Em português *jogo da cerveja*.

⁶Do inglês *bullwhip effect*.

de ações que devem ser executadas simultaneamente por todos os membros de todos os grupos. Esta sequência é descrita no Algoritmo 1.

Algoritmo 1 Beer Game - Fluxo

- 1: **para cada** rodada **faça**
 - 2: Recebe carregamento do fornecedor
 - 3: Envia carregamento ao cliente
 - 4: Anota estoque em mãos ou total devido
 - 5: Recebe pedido do cliente
 - 6: Envia pedido ao fornecedor
 - 7: **fim para**
-

Ao final da 36^a rodada, o jogo é encerrado, e o grupo vencedor é determinado como sendo aquele que tiver acumulado, em todos os seus níveis e rodadas, a menor pontuação conforme definido na Equação (1):

$$\text{pontuação} = 0.5 * \text{estoque em mãos} + 1 * \text{total devido} \quad (1)$$

A demanda do consumidor é revelada: ela é fixa em 4 unidades por rodada até a 4^a rodada, e em 8 unidades por rodada da 5^a rodada em diante. Tendo-a como parâmetro, é possível observar o efeito chicote, com oscilações significativamente maiores em estoque e nos pedidos feitos entre os níveis superiores, crescendo no sentido da fábrica [Sterman 1989].

3. Estado da Arte

Os principais trabalhos que abordam confiança em SCM são listados na Tabela 1, juntamente com o indicador de desempenho que condiciona as escolhas de fornecedores, a função de utilidade utilizada e as políticas de decisão testadas para os agentes. Analisando esta tabela, destacam-se alguns pontos relevantes:

Tabela 1. Comparativo entre publicações similares

Trabalho	Indicador de performance	Políticas de agentes	Função de utilidade
[Akkermans 2001]	Volume recebido de entregas	Heterogêneas	Quantidade de trocas de fornecedor
[Schieritz 2003]	Volume recebido e rapidez de entregas	Homogêneas	Quantidade de trocas de fornecedor
[Lin et al. 2005]	Preço e tempo de entrega	Homogêneas	Custo, pontualidade, tempo de ciclo
[Kim 2009]	Proporção entre pedidos e entregas	Homogêneas	Flutuação nos níveis de estoque
[Hou et al. 2014]	Taxa de entregas pontuais	Homogêneas	Capital de trabalho e firmas ativas

- Em todos os trabalhos, cada empresa da SC corresponde a um agente simulado, mas apenas em [Kim 2009] o autor utiliza como modelo o *Beer Game*;
- Todos os trabalhos condicionam o desempenho de um fornecedor à quantidade e agilidade de suas entregas;

- Apenas em [Akkermans 2001] são usadas políticas de decisão heterogêneas em uma mesma SC simulada, com agentes que privilegiam desempenho em curto prazo de seus fornecedores, e outros que focam em desempenho a longo prazo;
- Todos os trabalhos analisam o impacto da atuação dos agentes sob o ponto de vista global da SC, mas nenhum deles registra ou compara o lucro individual de cada empresa representada.

4. Proposta

Esta pesquisa diferencia-se das demais ao buscar medir o desempenho individual de agentes com perfis de atuação distintos, usando ou não a noção de confiança, interagindo simultaneamente em uma mesma SC modelada segundo as regras do *Beer Game*. Para isso, o modelo utilizado, o fluxo, os perfis e políticas a serem avaliadas, são descritos a seguir.

4.1. Modelo de cadeia de suprimentos

De maneira similar ao que ocorre em [Hou et al. 2014], a cadeia simulada é composta por 5 níveis de profundidade: fábricas, distribuidores, atacadistas, revendedores e consumidores finais, e 20 agentes por nível. A cada rodada, cada um dos agentes de um nível poderá fazer seu pedido a um dos fornecedores presentes no nível superior, assim como deverá atender a eventuais pedidos feitos por cada um dos clientes presentes no nível inferior.

4.2. Modelo de agente

Neste trabalho, foram utilizados agentes cognitivos, cuja arquitetura subdivide-se em:

- **Percepção:** o agente detecta os carregamentos enviados por seu fornecedor e os pedidos enviados por seus clientes;
- **Raciocínio:** o agente raciocina e decide para quais clientes enviar carregamentos (entrega primeiro a quem deve mais) e dimensiona o pedido ao fornecedor com base no mecanismo de ancoragem e ajuste, descrito na seção 4.3;
- **Ação:** o agente envia carregamentos a seus clientes e um pedido a seu fornecedor.

4.3. Estratégia de pedidos por ancoragem e ajuste

Em [Sterman 1989], o autor propõe um modelo para caracterizar a forma como companhias dimensionam pedidos a fornecedores a fim de controlar seus próprios níveis de estoque. Tal modelo é baseado no mecanismo cognitivo de ancoragem e ajuste (A&A)⁷ descrito em [Tversky and Kahneman 1974], que propõe uma heurística para os pedidos a fornecedores de modo a:

1. ressuprir perdas esperadas do estoque;
2. reduzir a discrepância entre o estoque atual e o desejado;
3. manter uma linha de suprimento adequada para os pedidos já feitos, mas ainda não recebidos.

⁷Do inglês *anchor and adjustment*.

A estratégia para calcular a diferença IO_t no estoque desejado é traduzida na Equação (2), em que o pedido a ser feito é obtido pela soma de três parcelas:

$$IO_t = Le_t + AS_t + ASL_t \quad (2)$$

onde (i) Le_t são as perdas esperadas, calculadas por atenuação exponencial, conforme a Equação (3), (ii) AS_t é o ajuste do estoque, obtido da Equação (4), e (iii) ASL_t é o ajuste na linha de suprimento, que resulta da Equação (5).

$$Le_t = \Theta L_{t-1} + (1 - \Theta)Le_{t-1}, 0 < \Theta < 1 \quad (3)$$

$$AS_t = \alpha_s(S* - S_t), 0 < \alpha_s <= 1 \quad (4)$$

onde $S*$ é o estoque desejado, e S_t é o estoque em mãos no instante t.

$$ASL_t = \alpha_{sL}(SL* - SL_t), 0 < \alpha_{sL} <= 1 \quad (5)$$

onde $SL*$ é a linha de suprimento desejada e SL_t é linha de suprimento no instante t.

Por fim, a quantidade solicitada O_t nunca deve ser negativa, o que é expresso pela Equação (6), que recebe IO_t como parâmetro.

$$O_t = MAX(0, IO_t) \quad (6)$$

4.4. Modelo de capital financeiro

O modelo financeiro proposto é baseado no descrito em [Hou et al. 2014]. Para cada agente, a variação de capital ΔC após cada rodada é expressa pela equação (7):

$$\Delta C = c * O + \sum_i (v - p) * O_i - u * S \quad (7)$$

onde (i) c é preço de custo unitário do fornecedor, (ii) v é o preço de venda unitário praticado aos clientes, (iii) p é o custo de produção unitário, (iv) O é o tamanho do pedido feito ao fornecedor, (v) O_i é o tamanho do pedido feito por cada cliente i , (vi) u é o custo de armazenamento unitário do estoque, e (vii) S é o estoque em mãos ao término da rodada.

Todos estes valores podem variar por rodada, com os preços de custo de cada agente correspondendo aos preços de venda fixados pelo seu fornecedor. Estes preços podem variar entre valores v_{min} e v_{max} , calculados por nível, da seguinte forma:

1. O custo por unidade de estoque u por rodada é convencionado como sendo \$1;
2. O custo de produção p é considerado o mesmo para cada nível da cadeia, e parametrizado em função de u ;
3. Para a fábrica, as parcelas de lucro l_{min} e l_{max} são parametrizadas, e uma ou outra somadas a p para obter v_{min} e v_{max} , respectivamente;
4. Para cada nível inferior na cadeia, as parcelas de lucro l_{min} e l_{max} são somadas ao v_{max} do nível imediatamente acima, para obter os seus v_{min} e v_{max} .

4.5. Mecanismo de confiança

Para quantificar o nível de confiança de um cliente em seu fornecedor, utiliza-se o cálculo proposto em [Hou et al. 2014]. A cada rodada, o nível de confiança é dado pela proporção histórica entre os carregamentos entregues a cada rodada e os pedidos correspondentes feitos três rodadas antes. Tal defasagem compreende a soma dos tempos de transmissão de pedido ao fornecedor e de envio de carregamentos por este ao seu cliente, conforme mencionado na seção 2. Esta proporção é expressa pela equação (8):

$$Trust_{CFn} = \sum_1^n C_i / \sum_{-2}^{n-3} P_i \quad (8)$$

onde (i) n é a rodada atual, (ii) P_i é o pedido enviado ao fornecedor na rodada i , (iii) C_i é o carregamento recebido do fornecedor na rodada i , e (iv) $Trust_{CFn}$ é a confiança do cliente C no fornecedor F na rodada n .

4.6. Perfis de agentes

Cada nível da SC será composto por agentes que apresentam diferentes perfis. Cada um destes perfis será norteado por um objetivo que incorre em uma combinação de políticas de decisão. Dois perfis são propostos neste artigo:

- **Popular:** Um agente popular visa atrair o maior número possível de clientes por meio de preço baixo, e mantê-los por meio da entrega mais pontual possível de seus pedidos;
- **Ganancioso:** Um agente ganancioso tem por objetivo maximizar o seu lucro comprando barato, vendendo caro e reduzindo despesas.

Quanto à **escolha de um fornecedor**, tais perfis se comportam da seguinte maneira:

- **Popular:** privilegia fornecedores em que possui maior confiança, uma vez que atrasos em entregas de pedidos provocam escassez no estoque do agente em questão, o que pode inviabilizar as entregas a seus clientes. A regra utilizada é a de preferência por confiança (“preferred trust rule” [Hou et al. 2014]):
 1. O agente sorteia conjunto N de candidatos a fornecedor do nível superior;
 2. O agente escolhe, dentre os N sorteados, aquele em que mais confia;
 3. Se o fornecedor escolhido tiver um nível de confiança maior que o atual, o agente troca de fornecedor;
 4. Caso contrário, com chance ϵ , o agente permanece com o fornecedor atual, ou com chance $1 - \epsilon$ troca para um fornecedor escolhido ao acaso.
- **Ganancioso:** privilegia fornecedores mais baratos. A regra utilizada é a de preferência por preço (“preferred price rule” [Hou et al. 2014]). Esta regra é análoga a “preferred trust rule”, exceto que o critério de decisão é o preço mais barato.

Já em relação ao **gerenciamento de estoque**, os comportamentos de tais perfis são:

- **Popular:** tem como estoque desejado S^* o estoque de confiança parametrizado, a fim de prevenir-se contra picos de demanda que prejudiquem suas entregas;
- **Ganancioso:** não mantém estoque de segurança ($S^* = 0$), de forma a reduzir o custo de manutenção de seu estoque.

Finalmente, no que diz respeito ao **controle de preços**, tem-se os seguintes comportamentos:

- **Popular:** mantém sempre o preço no valor mais baixo, de forma a minimizar a chance de perder clientes focados em preço baixo;
- **Ganancioso:** mantém sempre o preço no valor mais alto, de forma a maximizar a sua margem de lucro.

5. Simulação e resultados

5.1. Ciclo de simulação

O Algoritmo 2 utilizado neste trabalho é uma extensão do Algoritmo 1 original do *Beer Game*.

Algoritmo 2 Simulação - Fluxo

- 1: Agentes inicializam estoque, e linhas de pedidos e suprimentos
 - 2: Clientes escolhem um fornecedor cada dentre os membros do nível superior de forma aleatória
 - 3: **para cada** rodada **faça**
 - 4: Clientes recebem carregamentos enviados há duas rodadas pelos fornecedores
 - 5: Fornecedores pagam o custo de produção e enviam carregamentos devidos aos respectivos clientes, total ou parcialmente, respeitando o critério de ordenação parametrizado
 - 6: Fornecedores registram estoque em mãos ou total devido
 - 7: Fornecedores pagam o custo do estoque em mãos
 - 8: Fornecedores atualizam preço de venda
 - 9: Fornecedores recebem pedidos feitos há uma rodada por clientes
 - 10: Clientes revisam nível confiança nos respectivos fornecedores
 - 11: Clientes decidem se mantêm ou trocam de fornecedor, e para qual trocar
 - 12: Clientes dimensionam e enviam pedido a seus fornecedores e pagam preço fixado por eles
 - 13: Fábrica dimensiona e envia ordem de produção
 - 14: Fornecedores recebem o pagamento de pedidos feitos na rodada atual por clientes novos, que compram pelo preço fixado
 - 15: **fim para**
-

5.2. Ambiente de simulação

O projeto foi implementado em ReLogo [Ozik et al. 2013], uma linguagem de domínio específica (DSL)⁸ para ABM. Esta ferramenta incorpora as bibliotecas e recursos do framework Repast Simphony [Collier 2003]. Além disso, foi programado e executado a partir da IDE Eclipse [Eclipse 2007]. Este ambiente foi escolhido devido à familiaridade do autor com a linguagem e também pelo fato do Repast dispor de um simulador com ferramentas úteis e intuitivas, a exemplo dos geradores de gráficos e das tabelas de estados atualizadas a cada passo para depuração.

5.3. Estado inicial

Os parâmetros para o estado inicial dos agentes são os mesmos presentes em [Edali and Yasarcanb 2014], e são mostrados na Tabela 2.

⁸Do inglês *domain specific language*.

Tabela 2. Parâmetros para o estado inicial dos agentes

Parâmetro	Valores
Estoques iniciais	todos os agentes iniciam a simulação com 12 unidades de estoque em mãos
Pedidos em atraso iniciais	todos os agentes começam sem pedidos em atraso a qualquer cliente
Pedidos em trânsito	todos os pedidos em trânsito e ordens de produção são inicializados com 4 unidades, que chegam ao destino na rodada posterior
Carregamentos em trânsito	todos os carregamentos em trânsito e linhas de produção são inicializados com 8 unidades, sendo que 4 delas chegam ao destino na rodada posterior e as outras 4 unidades após 2 rodadas

5.4. Parâmetros de entrada

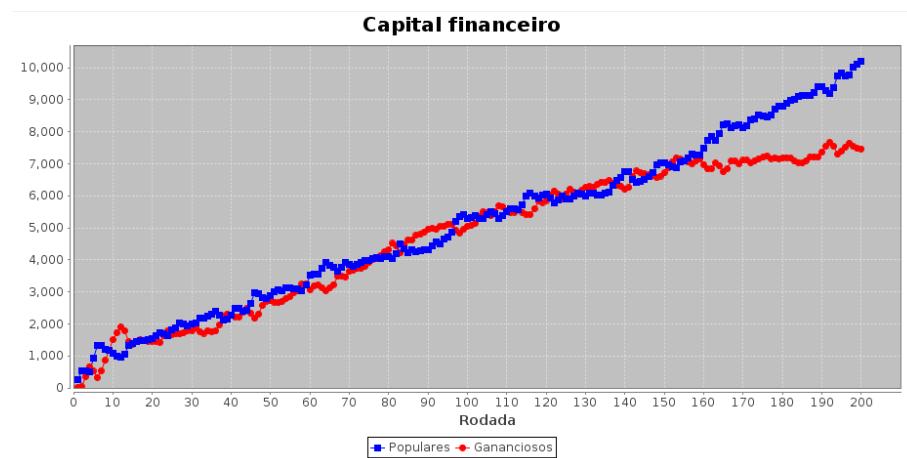
Nos experimentos, combinaram-se 50% de agentes com perfil Popular e 50% com perfil Ganancioso por nível da SC, para determinar qual estratégia prevalece no caso de ambas serem adotadas pelo mesmo número de agentes.

Outro parâmetro definido é a demanda dos consumidores finais. Ao contrário do que ocorre para os níveis superiores da SC, o dimensionamento de seus pedidos não segue a equação (2), e portanto esta política deve ser arbitrada. Para este trabalho, utilizou-se demanda aleatória: a cada rodada, cada consumidor final obtém o número de unidades solicitadas a partir de uma distribuição uniforme, definida no intervalo [0, 12], incluindo os extremos. Os demais parâmetros foram fixados de acordo com a Tabela 3.

Tabela 3. Parâmetros utilizados

Parâmetro	Descrição	Valores
maxStep	Duração da simulação em passos	200
L	Agentes por nível da SC	20
S*	Estoque desejado - (eq. 4)	24
p	Custo de produção - seção 4.4	10
l_{min}	Lucro mínimo da fábrica - seção 4.4	10
l_{max}	Lucro máximo da fábrica - seção 4.4	20
α_s	usado no ajuste de estoque - eq. 4)	0,5
β	$\alpha_s L / \alpha_s$ - usado no ajuste da linha de suprimentos - eqs. 4 e 5	1,0
Θ	usado na previsão de demanda - eq. 3	0,5
ϵ	usado na escolha de fornecedor - seção 4.6	0,9
N	usado na escolha de fornecedor - seção 4.6	5

5.5. Resultados obtidos

**Figura 3. Capital financeiro médio por perfil**

Os gráficos das Figuras 3 a 7 foram obtidos da execução disparada via interface gráfica. As estratégias apresentam rentabilidade semelhante, até por volta da rodada 160, em que a confiança nos agentes populares contribui para maior lucratividade destes frente aos gananciosos via migração de clientes (Figura 3).

A confiança média depositada em cada perfil (Figura 4) parte do mesmo patamar, mas logo os populares disparam à frente, devido ao estoque de segurança que os protege contra flutuações na demanda dos clientes.

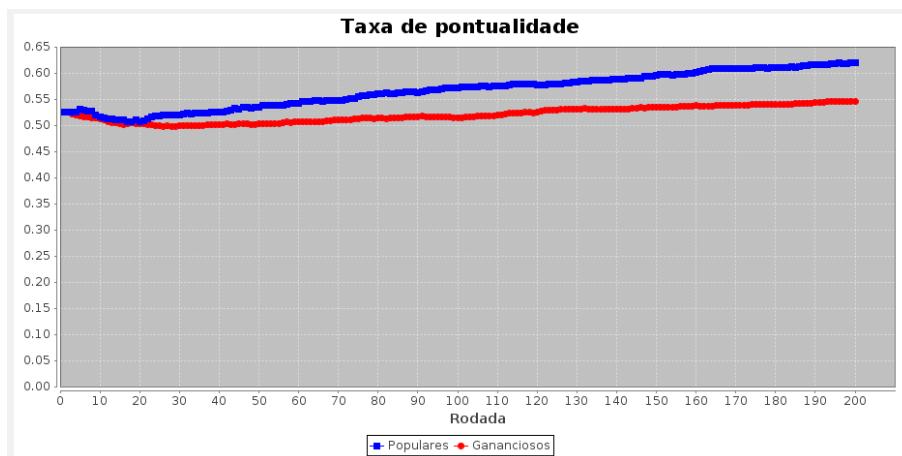


Figura 4. Confiança média por perfil

A diferença entre os perfis na média de clientes é ampliada conforme clientes populares percebem que fornecedores populares são mais confiáveis que fornecedores gananciosos (Figura 5).

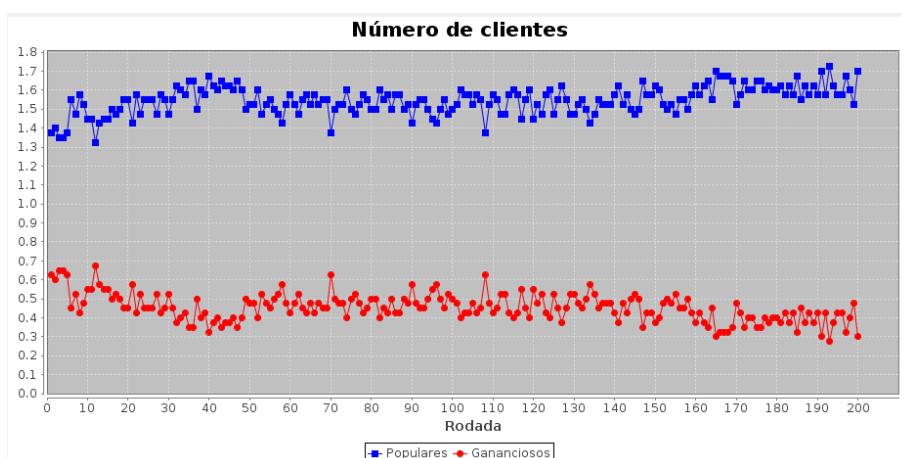


Figura 5. Média de clientes por perfil

A margem de lucro mantém-se praticamente constante para ambas as políticas (Figura 6), com gananciosos comprando barato e vendendo caro, portanto com margem maior que os populares, que vendem barato e compram pelo preço do fornecedor.

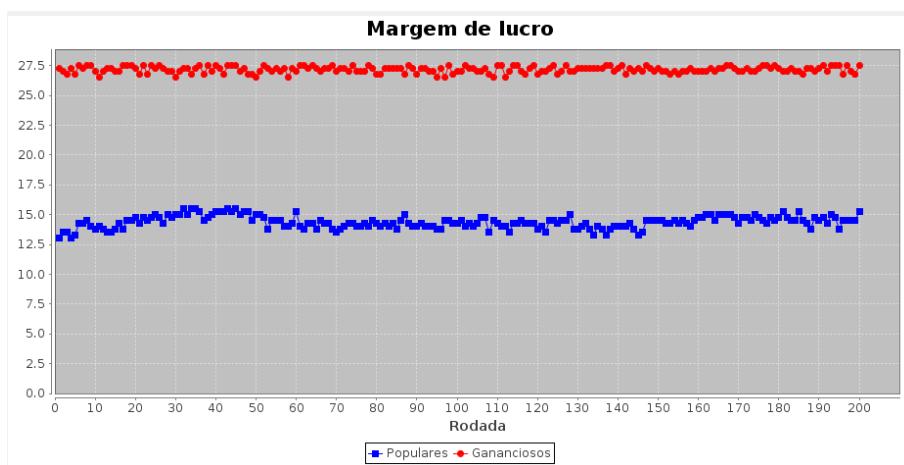


Figura 6. Margem de lucro média por perfil

No gráfico da média de pedidos feitos entre níveis (Figura 7), é possível observar o BWE: a amplitude da variação dos pedidos dos distribuidores para as fábricas é maior que a dos atacadistas para os distribuidores, que é maior que as dos revendedores para os atacadistas, que, por fim, é maior que a dos consumidores finais para os revendedores.

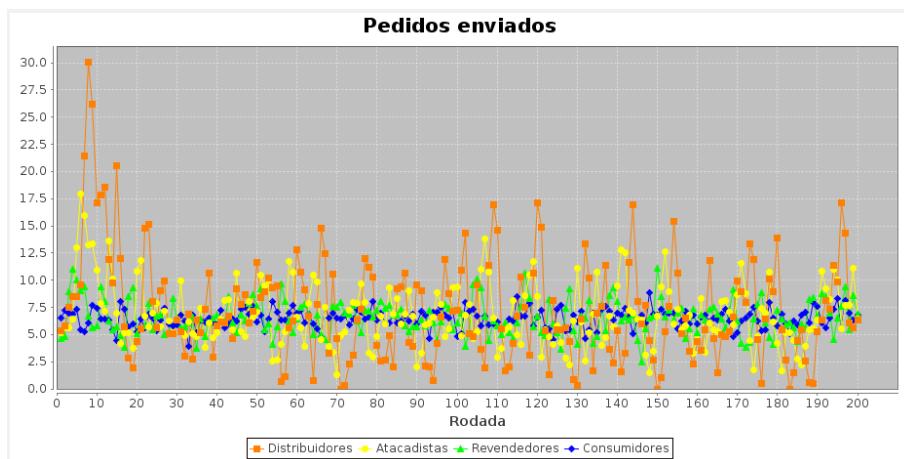


Figura 7. Média de pedidos enviados por nível da SC

6. Conclusões

Neste trabalho, sistemas multi-agentes foram utilizados para simular cadeias de suprimento segundo uma extensão do modelo do *Beer Game*. No experimento realizado, foi possível observar o efeito de estratégias de atuação distintas no lucro individual médio dos seus agentes, bem como a migração de clientes para fornecedores mais confiáveis à medida que o experimento avançava. Por fim, o efeito chicote foi reproduzido, com o aumento da variância dos pedidos no sentido do consumidor final para a fábrica.

Este estudo restringiu-se a analisar relações verticais, no caso confiança entre clientes e fornecedores. Futuramente, serão analisadas também relações horizontais entre agentes de um mesmo nível produtivo, podendo estes trocar mensagens contendo recomendações de fornecedores. Ademais, as simulações, por ora limitadas a uma

combinação específica de parâmetros de entrada, deverão ser alvo de testes de sensibilidade a fim de compreender a influência de cada variável do modelo no comportamento da cadeia.

Referências

- Akkermans, H. (2001). Emergent supply networks: system dynamics simulation of adaptive supply agents. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, page 11. IEEE Comput. Soc.
- Collier, N. (2003). RePast : An Extensible Framework for Agent Simulation. *The University of Chicago's Social Science Research*, 36:371–375.
- Croom, S., Romano, P., and Giannakis, M. (2000). Supply chain management: an analytical framework for critical literature review. *European journal of purchasing & supply management*, 6(1):67–83.
- De La Fuente, D. and Lozano, J. (2007). Application of distributed intelligence to reduce the bullwhip effect. *International Journal of Production Research*, 45(8):1815–1833.
- Eclipse, I. (2007). Eclipse foundation.
- Edali, M. and Yasarcanb, H. (2014). A Mathematical Model of the Beer Game. *JASSS*.
- Forrester, J. W. (1997). Industrial dynamics. *Journal of the Operational Research Society*, 48(10):1037–1041.
- Friedman, M. (2007). The social responsibility of business is to increase its profits. *Corporate ethics and corporate governance*, pages 173–178.
- Handfield, R. B. and Bechtel, C. (2002). The role of trust and relationship structure in improving supply chain responsiveness. *Industrial marketing management*, 31(4):367–382.
- Hou, Y., Xiong, Y., Wang, X., and Liang, X. (2014). The effects of a trust mechanism on a dynamic supply chain network. *Expert Systems with Applications*, 41(6):3060–3068.
- Kim, W. S. (2009). Effects of a trust mechanism on complex adaptive supply networks: An agent-based social simulation study. *JASSS*, 12(3).
- Lambert, D. M. and Cooper, M. C. (2000). Issues in supply chain management. *Industrial marketing management*, 29(1):65–83.
- Lee, H. L., Padmanabhan, V., and Whang, S. (1997). Information Distortion in a Supply Chain: The Bullwhip Effect. *Source: Management Science*, 43(4):546–558.
- Lin, F.-r., Sung, Y.-W., and Lo, Y.-P. (2005). Effects of trust mechanisms on supply-chain performance: A multi-agent simulation study. *International Journal of Electronic Commerce*, 9(4):9–112.
- Mayer, R. C., Davis, J. H., and Schoorman, F. D. (1995). An integrative model of organizational trust. *Academy of management review*, 20(3):709–734.
- Nagurney, A. (2006). *Supply chain network economics: dynamics of prices, flows and profits*. Edward Elgar Publishing.
- Ozik, J., Collier, N. T., Murphy, J. T., and North, M. J. (2013). The relogo agent-based modeling language. In *Simulation Conference (WSC), 2013 Winter*, pages 1560–1568. IEEE.
- Schieritz, N. (2003). Emergent structures in supply chains - A study integrating agent-based and system dynamics modeling. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences, HICSS 2003*.
- Sterman, J. D. (1989). Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment. *Management science*, 35(3):321–339.
- Sterman, J. D. (1992). Teaching Takes Off: Flight Simulators for Management Education. *OR/MS Today*, pages 40–44.
- Swaminathan, J. M., Smith, S. F., and Sadegh, N. M. (1998). Modeling supply chain dynamics: A multiagent approach. *Decision Sciences*, 29(3):607–631.
- Tversky, A. and Kahneman, D. (1974). Judgment under Uncertainty: Heuristics and Biases. *Science*, 185(4157):1124–1131.
- Vlachos, I. P. and Bourlakis, M. (2006). Supply chain collaboration between retailers and manufacturers: do they trust each other? In *Supply Chain Forum: An International Journal*, volume 7, pages 70–80. Taylor & Francis.
- Wooldridge, M. J. (2000). *Reasoning about rational agents*. MIT press.

Agente Inteligente Embarcado baseado em Sistemas Ciberfísicos no contexto da Indústria 4.0

Braian Konzgen Maciel¹, Mario Ricardo Nascimento Marques Junior¹,
Gabriel Machado Balota¹, Eder Mateus Nunes Gonçalves¹

¹Centro de Ciências Computacionais - C3 – Universidade Federal do Rio Grande (FURG)
Brasil, Rio Grande do Sul, Rio Grande
Av. Itália km 8 Bairro Carreiros – Rio Grande – RS – Brasil

Abstract. *Industry 4.0 or advanced manufacturing is promoting a new Industrial Revolution by expanding the application of computing and communication technologies. Cyber Physical Systems can be considered the main component for the development of this new revolution. A CPS constitution requires a combination of several methodologies and technologies, many of which are emerging. In this context, the theory of intelligent agents and multi agent systems can bring several benefits such as decentralization of control, modularity, adaptation to the environment, etc. This article aims to present an architecture for intelligent agents, a network in the CPS 5C architecture, to compose a cyberphysical system in the industrial environment.*

Resumo. *A Indústria 4.0 está promovendo uma nova Revolução Industrial através da aplicação de tecnologias computacionais e de comunicação. Sistemas Ciberfísicos (CPS) podem ser considerados como componente chave para o desenvolvimento desta nova revolução. A constituição do CPS requer a combinação de diversas metodologias e tecnologias, muitas delas emergentes. Neste contexto, a teoria de agentes inteligentes e sistemas multiagentes podem trazer diversos benefícios como descentralização do controle, modularidade, etc. Este artigo tem como objetivo propor uma arquitetura para agentes inteligentes embarcados, baseado na arquitetura CPS 5C, para compor um CPS no ambiente industrial.*

1. Introdução

A evolução do ambiente industrial é fundamental para o suprimento das demandas humanas. E no momento em que surgem novas necessidades e desafios, inovações tecnológicas precisam emergir para suprir tais exigências. Ao longo da história o cenário industrial passou por três grandes transformações: a 1^a Revolução Industrial teve como característica a mecanização da produção, com a invenção da máquina a vapor. A introdução da eletricidade e criação de linhas de montagem, caracterizaram a 2^a Revolução Industrial, viabilizando a produção em massa idealizada por Henry Ford. A 3^a Revolução Industrial introduziu elementos de eletrônica, tecnologia da informação e automação nesse cenário. Elementos como o Controlador Lógico Programáveis (CLP) e da Tecnologia da Informação (TI) aplicadas ao chão de fábrica [Rodrigues et al. 2016].

De acordo com [Kagermann et al. 2013], a introdução do programa alemão denominado *Industrie 4.0* deu início ao desafio que se apresenta como a de 4^a Revolução

Industrial, em que máquinas e componentes inteligentes podem comunicar-se de maneira autônoma. Assim, as decisões no chão de fábrica podem ser tomadas pelas próprias máquinas, a partir de informações fornecidas em tempo real. Outra característica importante, que pode ser destacada da Indústria 4.0, é a integração de diversas tecnologias relacionadas ao sistema, com foco na sua representação cibernetica. A Figura 1 demonstra essa evolução do sistema industrial com suas principais características.



Figura 1. Evolução da indústria. Fonte: Adaptado de [Posada et al. 2015]

Para [Zhou et al. 2015], a Indústria 4.0 é uma visão para o futuro pois atualmente enfrenta muitas dificuldades e desafios, incluindo desafios científicos, desafios tecnológicos, desafios econômicos, problemas sociais e questões políticas. Como exemplo de desafios científicos e tecnológicos pode-se citar o desenvolvimento de dispositivos inteligentes, a construção do ambiente de rede, a grande análise e processamento de dados e a manufatura digital.

Dentro deste novo mundo que está sendo idealizado, alguns conceitos ganham grande destaque e influenciam diretamente para o desenvolvimento da 4^a Revolução Industrial. Sistemas Ciberfísicos (*Cyber Physical Systems - CPS*) [Jazdi 2014] e Internet das Coisas (*Internet of Things - IoT*) [Shrouf et al. 2014] são alguns dos conceitos que vem contribuindo para que tecnologias já conhecidas e emergentes sejam aplicadas a manufatura industrial.

A constituição do CPS pode envolver o uso de várias tecnologias, como sistemas multiagentes (*Multi Agent Systems - MAS*), Arquiteturas Orientadas a Serviços (*Service-Oriented Architecture - SoA*), Computação em Nuvem (*Cloud Computing*), *Big Data*, *Machine-to-Machine* (M2M) e a Computação Visual (CV).

A integração entre diversas tecnologias visa contribuir para o CPS enfrentar desafios identificados nos princípios da Indústria 4.0. Sistemas multiagentes por exemplo, podem contribuir com flexibilidade, robustez, adaptação, configuração e controle distribuído para estes sistemas.

A capacidade de comunicação e transferência de dados entre diferentes dispositivos (sensores e atuadores) dentro de um ambiente industrial surge como um dos problemas a ser resolvido para a nova infraestrutura de indústria. Diversos padrões tem surgido para garantir que equipamentos de diferentes fontes realizem a coleta e transmissão de

dados de forma eficiente e segura.

Outro ponto que tem um grande destaque neste cenário é a descentralização do controle e incremento de complexidade para a realização de todas as operações. Dessa forma a necessidade de se desenvolver o comportamento autônomo do sistema através de abordagens como os sistemas multiagentes.

Este artigo tem como objetivo propor uma arquitetura para desenvolvimento de agentes inteligentes embarcados baseados em Sistemas Ciberfísicos. É tomada por referência a arquitetura 5C proposta por Lee[Lee et al. 2015] para composição de Sistemas Ciberfísicos. O design do agente inteligente é baseado nas características desta arquitetura e na teoria de sistemas multiagentes, com o intuito de viabilizar a construção de ambientes inteligentes compatíveis com o CPS.

Visto que a arquitetura 5C apresenta uma forma genérica para implementação de CPS em ambientes industriais, a proposta apresentada neste artigo possui maior detalhamento da implementação para uma aplicação específica.

2. Fundamentação

Esta seção apresenta alguns conceitos de técnicas, bem como tecnologias que estão ganhando destaque com o desenvolvimento da próxima revolução industrial.

2.1. Princípios da Indústria 4.0

A Indústria 4.0 está fundamentada sobre seis princípios básicos [Hermann et al. 2016]:

- Capacidade de operação em tempo real: consiste na aquisição e tratamento de dados de forma instantânea, permitindo a tomada de decisões dentro das restrições de tempo do ambiente;
- Virtualização: propõe a existência de uma cópia virtual das fábricas inteligentes, permitindo a rastreabilidade e monitoramento remoto de todos os processos por meio dos inúmeros sensores espalhados ao longo da planta;
- Descentralização: a tomada de decisões poderá ser feita pelo sistema ciberfísico de acordo com as necessidades da produção em tempo real. Além disso, as máquinas não apenas receberão comandos, mas poderão fornecer informações sobre seu ciclo de trabalho;
- Orientação a serviços: Utilização de arquiteturas de software orientadas a serviços aliado ao conceito de *Internet of Services*.
- Modularidade: Produção de acordo com a demanda, acoplamento e desacoplamento de módulos na produção, oferecendo flexibilidade para alterar as tarefas das máquinas facilmente.
- Interoperabilidade: Capacidade de máquinas, dispositivos, sensores e humanos de se conectar e comunicar através da Internet das Coisas e da Internet.

Para [Barbosa et al. 2016], o CPS e o IoT surgem como pontos fundamentais para a realização da 4^a Revolução Industrial. E utilizam-se de tecnologias conhecidas e emergentes como agentes inteligentes, sistemas multiagentes, arquiteturas orientada a serviço, big data, computação em nuvem entre outras, para o ingresso de seus sistemas nesta nova idealização de industria.

2.1.1. Internet das Coisas e Serviços

A IoT, é uma infraestrutura de rede global, dinâmica e auto-gerenciada, onde coisas físicas e virtuais possuem identidades, atributos físicos e personalidades virtuais. Essa infraestrutura é capaz de integrar vários dispositivos equipados com sensoriamento, identificação, processamento e comunicação em rede [Da Xu et al. 2014].

2.1.2. Sistema Ciberfísico

O CPS é responsável por conectar o mundo virtual com a realidade física, que integra capacidades de computação, comunicação e armazenamento, podendo operar em tempo real de forma confiável, segura, estável e eficiente.

No contexto da Indústria 4.0 os agentes inteligentes e sistemas multiagentes compartilham um terreno comum com o CPS. Eles viabilizam o CPS a alcançar gerenciamento de complexidade, descentralização, inteligência, modularidade, flexibilidade, robustez e capacidade de resposta em tempo real [Leitao et al. 2016].

2.1.3. Arquitetura CPS 5C

Entre as arquiteturas CPS, a arquitetura denominada 5C proposta por [Lee et al. 2015] tem grande destaque na literatura. Ela serve como um guia para desenvolvimento e implementação do CPS para aplicações industriais. Esta arquitetura está dividida em cinco níveis . Na figura 2 é possível visualizar estes níveis bem como suas respectivas funções.



Figura 2. Arquitetura CPS 5C. Fonte: Adaptado de [Lee et al. 2015]

2.1.4. MTConnect

A capacidade de conexão entre diferentes dispositivos surge com um dos principais desafios da Indústria 4.0. Diversos padrões e protocolos estão sendo desenvolvidos nos últimos anos, e o Instituto MTConnect[MTConnect] propôs uma solução para este problema.

O padrão MTConnect baseia-se em tecnologias padrão da Internet como HTTP (*Hyper Text Transfer Protocol*) e XML (*Extensible Markup Language*). Um sistema que implementa o protocolo MTConnect tem cinco componentes fundamentais: Dispositivo, Adaptador, Agente, Rede e Aplicação/Cliente conforme Figura 3. Os componentes mais importantes são Agente e Adaptador.

O Dispositivo pode ser qualquer tipo de ferramenta, processo, planta, máquina, controlador ou equipamento. O Adaptador, é responsável por realizar a “tradução” entre o formato proprietário do equipamento para o padrão MTConnect.

O Agente é o núcleo da aquisição de dados do MTConnect. Ele coleta dados através do Adaptador e os escreve em um documento XML. Seu trabalho principal é analisar os dados do equipamento e fornecer uma interface para a rede, então os aplicativos externos podem obter os dados de que precisam. Também possui um *buffer* para armazenamento temporário dos dados

A Rede é responsável por garantir a conectividade entre as fontes de dados (dispositivos) e consumidores de dados (aplicações/clientes), geralmente é uma rede Ethernet. Os clientes são softwares que se conectam ao agente a fim de requisitar informações com algum propósito.

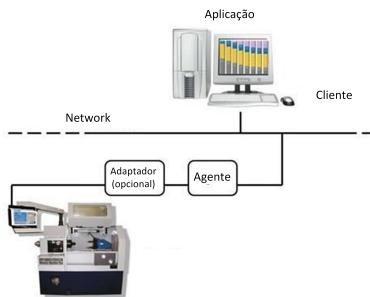


Figura 3. Arquitetura de um sistema MTConnect. Fonte:[MTConnect]

No contexto da Indústria 4.0 o MTConnect surge como solucionador para conexão entre dispositivos físicos de um CPS, atuando no mais baixo nível consegue garantir a aquisição de dados independente do formato ou protocolo de comunicação.

3. Trabalhos Relacionados

Nesta seção, são apresentados trabalhos que fazem utilização do paradigma de sistema multiagente, bem como tecnologias consideradas emergentes voltado para aplicações industriais para concepção de Sistemas Ciberfísicos.

3.1. Projeto GRACE

No projeto GRACE [Leitão et al. 2012] a arquitetura SMA-CPS desenvolvida, visualizada na Figura 4, foi aplicada em uma indústria de fabricação de máquinas de lavar. O seu principal objetivo, foi melhorar a produtividade e a qualidade dos produtos através da integração do controle do processo com a qualidade. Para isso foi implementado um sistema multiagente, no qual foi responsável pelos procedimentos dinâmicos de auto-adaptação, *loops* de controle e mecanismos de auto-otimização.

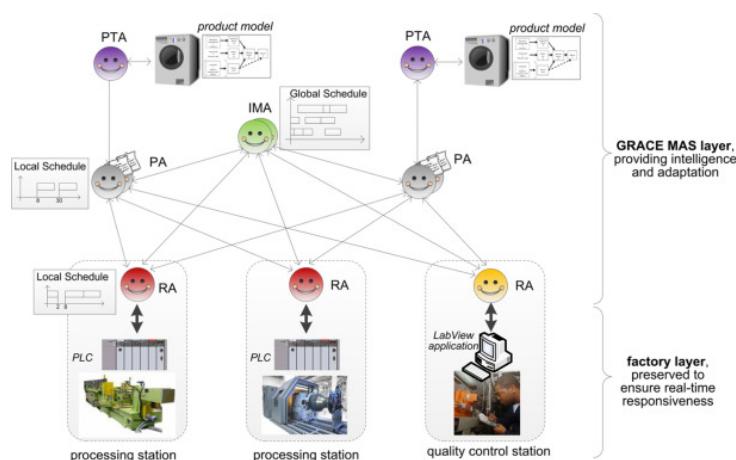


Figura 4. Sistema multiagente para integração de qualidade e controle de processos proposto em [Leitão et al. 2012].

Basicamente 4 tipos de Agentes foram projetados, *Product Type Agents* (PTA), *Product Agents* (PA), *Resource Agents* (RA), *Independent Meta Agents* (IMA)[Leitão et al. 2016]:

- PTA: representa os diferentes produtos que podem ser produzidos na planta;
- PA: representa a produção de instâncias de produto sendo produzidos ao longo da linha de produção;
- RA: representa os recursos da linha de produção, estações de processamento, estações de controle de qualidade e operadores;
- IMA: Os agentes coletam os dados do chão de fábrica de forma distribuída e realizam análise de dados em tempo real para ajustar dinamicamente as variáveis de produção, ou seja, os parâmetros de processamento, de operação e de inspeção.

Como funcionalidades destes agentes, pode-se destacar a adaptação contínua das estações de processamento e inspeção, a seleção de testes funcionais, a geração de avisos de qualidade e a parametrização do controlador. Os PA interagem continuamente com os RA ao longo da linha de produção para obter *feedback* relacionando à qualidade das operações realizadas (processamento e inspeção) sobre as máquinas de lavar. Outra função dos PA, é aplicar um algoritmo para adaptar o plano de ensaios a realizar na máquina de lavar de acordo com seu históricos de produção.

Toda solução de agente foi projetada utilizando a plataforma JADE, e distribuída em 8 computadores dispostos ao longo do chão de fábrica, sendo conectados por uma rede Ethernet, utilizando o protocolo TCP/IP para comunicação. Para o controle de baixo nível foram utilizados CLP's e os programas executados de acordo com a norma IEC 61131-3, onde tem como principal função garantir a resposta em tempo real.

Os autores ainda comentam que a integração de tecnologia SMA com dispositivos físicos como CLP, permitem construir componentes ciberfísicos, para integração de processos e controle de qualidade. Os resultados obtidos mostraram um conjunto de benefícios como: aumento de eficiência produtiva, otimização dos parâmetros do processo, bem como melhoria de qualidade do produto [Leitão et al. 2016].

3.2. Projeto GOODMAN

De acordo com os resultados alcançados em outros projetos de pesquisa europeus desenvolvidos nos últimos anos, como GRACE [Leitão et al. 2012], IDEAS [IDEAS] e *Self-Learning* [Self-Learning], foi lançado em 2017 o projeto GOODMAN [GOODMAN].

O SMA-CPS projetado é baseado na arquitetura GRACE, visualizado na Figura 4, e o sistema orientado a agentes inteligentes é utilizado para garantir defeito zero em vários estágios de fabricação. Esse projeto tem como objetivo desenvolver e implementar tecnologias que serão aplicadas em uma linha de produção, a fim de evitar a geração e propagação de defeitos. Também visa desenvolver uma estratégia de produção que garanta alta qualidade de produtos melhorando a eficiência de produção de todo o sistema.

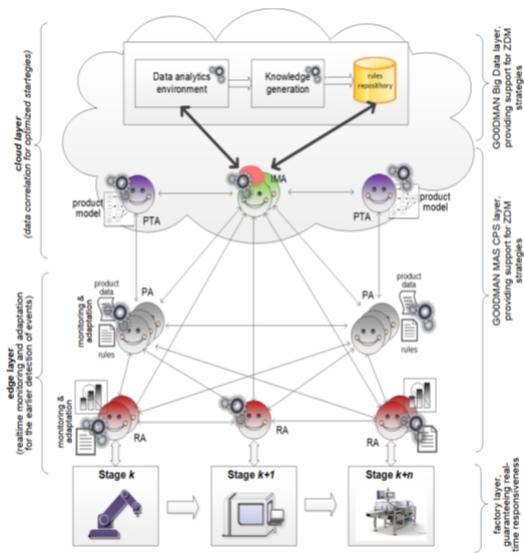


Figura 5. CPS baseado em agentes utilizado no projeto GOODMAN. Fonte: [GOODMAN]

A Figura 5 comprehende uma sociedade de agentes inteligentes distribuídos, autônomos e cooperativos que representam os componentes de produção dispostos ao longo do sistema de fabricação em vários estágios. Em alguns casos, esses agentes têm associação direta com um recurso físico, por exemplo, um robô de soldagem ou uma estação de controle de qualidade, mas em outros casos representam entidades lógicas, como análise de dados e ferramentas de auto-aprendizagem. Até os produtos que estão sendo produzidos ao longo da linha terão um agente inteligente associado, que transforma um produto passivo em um produto inteligente. Com a associação de cada estágio e recurso, dispostos ao longo do ambiente de fabricação, a um agente inteligente é formado então um sistema multi-agente, que coleta, armazena e processa dados de forma distribuída.

Esses agentes inteligentes são enriquecidos com algoritmos de análise de dados, executando parcialmente algoritmos mais simples em agentes de nível inferior ou invocando serviços analíticos de dados no nível da nuvem.

3.3. Arquitetura CPPS para integração de sistemas de produção heterogêneos

Em [Vogel-Heuser et al. 2014] é apresentada uma arquitetura para implementar componentes de CPPS (*Cyber Physical Production System*) baseado na tecnologia de MAS e

apoiada por modelos de conhecimento e ontologias. A arquitetura visa resolver o problema de um cenário de produção em que iogurte personalizado em massa é encomendado por um cliente e produzido por diferentes instalações de produção distribuídas.

A arquitetura proposta de CPPS foi desenvolvida com base no padrão existente da Foundation for Physical Agents (FIPA) e em abordagens de outros trabalhos. Dentro do composto de CPPS, cada sistema de produção é encapsulado por um CPPS que é representado por um Agente de Planta que propaga as capacidades da planta particular e agenda localmente as ordens de entrada. Para controlar a planta de acordo com o esquema de produção, este Agente de Planta pode ser uma entidade de software do próprio software de automação do sistema de produção (ver 6, Sistema de Produção B), ou implementado em hardware adicional (ver 6, Sistema de Produção A) para encapsular a planta de produção e fornecer sua interface para o CPPS. Estas opções permitem a implementação dos Agentes de Plantas em diferentes linguagens de programação e em diferentes plataformas heterogêneas.

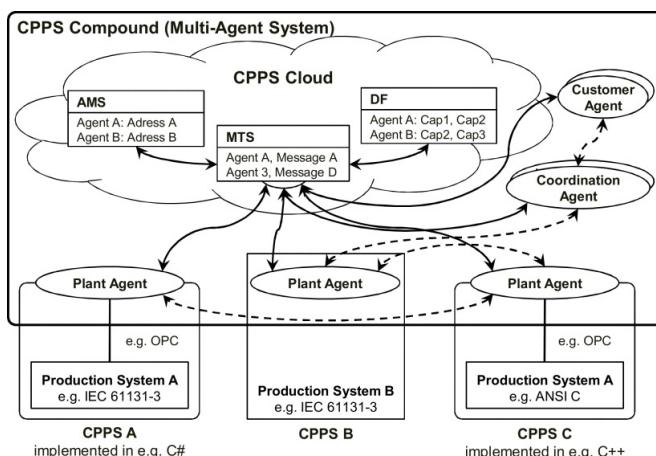


Figura 6. Arquitetura do CPPS proposto em [Vogel-Heuser et al. 2014].

A comunicação e a interoperabilidade entre os diferentes agentes e as entidades organizacionais da CPPS é realizada por uma linguagem comum que se baseia na Linguagem de Comunicação de Agentes (ACL), que também é padronizada pelo FIPA. A plataforma de agente desenvolvida e implementada suporta a especificação de um vocabulário específico de domínio e protocolos de comunicação como um primeiro passo para a implementação de uma ontologia.

Para avaliar a implementação da abordagem, foram introduzidas manualmente diferentes ordens no sistema utilizando o cliente web fornecido e observou-se manualmente a reação e comportamento dos diferentes sistemas de produção de laboratório. Verificou-se que o Agente de Planta de CPPS corretamente processa as ordens dos clientes e despacha as ordens de produção para os diferentes CPPS onde são executadas.

4. Proposta

Nesta seção apresentamos uma arquitetura para a construção de agentes inteligentes embarcados baseados nos conceitos e premissas do CPS e Indústria 4.0. A aplicação de sistemas ciberfísicos no contexto de manufatura tem se mostrado muito promissora, mas carece de arquiteturas, metodologias e estratégias para a sua concepção.

A estratégia adotada nessa proposta é baseada na teoria de agentes e sistemas multiagentes, que visa a construção de um ambiente formado por diversos agentes inteligentes, dotados de um comportamento autônomo. O agente inteligente embarcado deve ser capaz de perceber e interagir com o meio físico através de sensores e atuadores, assim como interagir com agentes virtuais através da rede. Ele também pode apresentar elementos de controle, através de algoritmos como o PID (*Proporcional Integral Derivativo*) e inteligência artificial, promovendo o controle distribuído e a descentralização.

4.1. Arquitetura de Software do Agente

A arquitetura de software proposta para o desenvolvimento do agente, está organizada em cinco módulos: configuração, inteligência, cibرنético, conversão e comunicação.

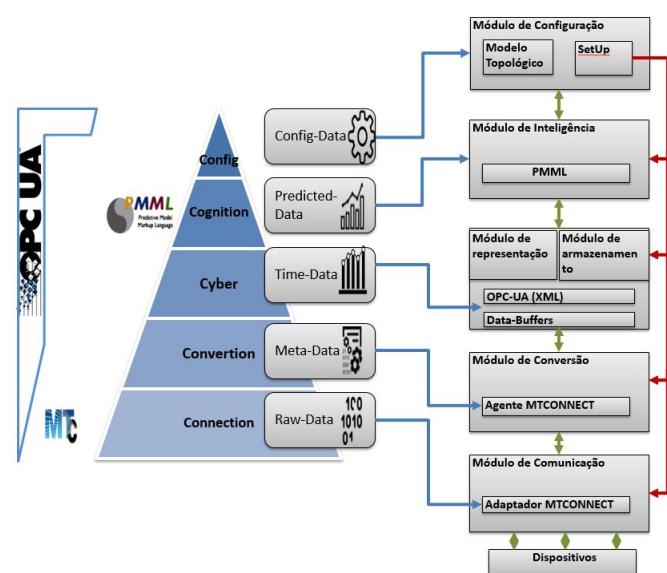


Figura 7. Arquitetura de Software em 5 módulos.

A Figura 7 apresenta a disposição, bem como funcionalidades dos módulos da arquitetura de software do agente.

- Módulo de Inteligência: responsável pelo mapeamento e rastreamento de padrões, comportamentos e o controle de qualidade dos dados. Utiliza algoritmos avançados para apontar falhas e promove o comportamento preditivo e cognitivo do sistema. Deste módulo emergem funções básicas de sistemas de manufatura, como controle, monitoramento, planejamento e escalonamento.
- Módulo Cibرنético: responsável pela gerênciа das informações do sistema de modo a representá-los em escala temporal por meio de inferências e previsões. Ele é formado por um *buffer*, que registra os dados do agente e por um adaptador de dados para bases de dados externas que habilita a aquisição de dados.
- Módulo de Conversão: responsável pela conversão dos dados coletados no módulo de comunicação em informação para o sistema, atribuindo semântica e algum tipo de tratamento para a garantia do provimento contínuo desta informação.
- Módulo de Comunicação: responsável por adaptar os diferentes protocolos de redes industriais para o padrão adotado no sistema e viabilizar a interoperabilidade por meio desse padrão. O módulo também disponibiliza o modelo com a descrição topológica do equipamento para os demais agentes por meio da rede.

4.2. Aplicação

Para desenvolvimento e implementação da arquitetura proposta, será utilizado como estudo de caso uma planta industrial didática do modelo PD3 Series da Smar. Essa planta contém diversas malhas de controle para automação de processos industriais e utiliza os mesmos instrumentos de campo que são desenvolvidos para aplicações em larga escala.

Os agentes serão conectados fisicamente na planta junto aos sensores e atuadores, conforme Figura 8. Para implementação dos agentes, serão utilizados computadores de um único módulo, ou *Single Board Computers* (SBC) como *Raspberry Pi* e *Odroid*. São dispositivos que apresentam os requisitos entendidos como necessários para este tipo de aplicação, como dimensionamento reduzido e alta capacidade de memória e processamento. Estes agentes embarcados são responsáveis por coletar as informações do ambiente bem como atuar sobre o mesmo.

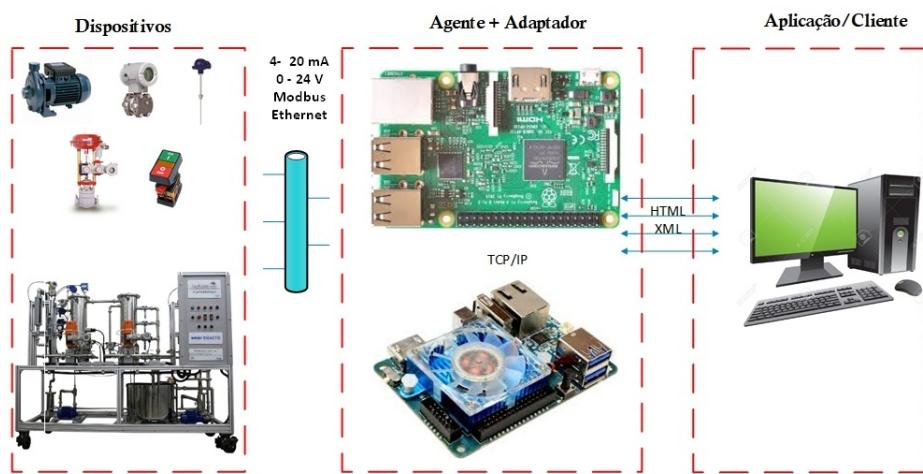


Figura 8. Arquitetura de hardware da aplicação.

Os módulos de software do agente serão implementados utilizando linguagens de programação como *C++*, *NodeJS* ou *Python*. As interfaces do módulo de configuração do agente serão baseadas em tecnologias web, visando a simplificação das operações de calibração e configuração pela rede. O protocolo padrão do módulo de comunicação será o MTConnect, e adaptadores de protocolos deverão ser implementados para a comunicação dos sensores e atuadores.

O modelo topológico será descrito através de XML (*eXtended Markup Language*), obedecendo o padrão MTConnect que estabelece um vocabulário e um conjunto de definições para a representação virtual dos componentes integrados ao agente.

5. Discussão e considerações finais

De acordo com os trabalhos relacionados, pode-se perceber, que uma grande capacidade de processamento era necessária para implementação da arquitetura desenvolvida. Em GRACE por exemplo, todo sistema era gerenciado por 8 computadores dispostos ao longo da linha de produção. A grande vantagem que pode ser elencada, analisado estes trabalhos relacionados, onde há a necessidade de potentes computadores para a implementação,

é a capacidade da arquitetura proposta nesse artigo ser implementada em dispositivos embarcados como Single Boards Computers.

Nota-se também que as arquiteturas apresentadas utilizam computadores conectados a CLPs ou outros dispositivos de controle industrial. Os agentes são executados em computadores e são responsáveis por mandar as ordens de controle para esses dispositivos que por sua vez realizam o controle propriamente dito. A arquitetura apresentada neste artigo propõe que um ou mais sensores e atuadores sejam conectados a cada dispositivo. Também prevê que na planta industrial sejam instalados vários desses dispositivos contendo um agente inteligente embarcado que será responsável pelo controle de um processo ou por parte dele.

O uso de paradigmas computacionais como *Fog* e *Cloud Computing*, assim como apresentado nos trabalhos relacionados, também é considerado na implementação da arquitetura proposta neste artigo. A virtualização, por exemplo, em determinados casos exige muito poder de processamento do dispositivo sendo necessário a implementação em nuvem, assim facilitando o acesso para os demais agentes realizar a virtualização completa do sistema.

Visto isso, entre as vantagens esperadas dessa abordagem baseada em agentes inteligentes, pode-se enumerar a padronização da comunicação, a promoção do controle distribuído, a autoconfiguração, a visibilidade e transparência dos dados assim como a interoperabilidade entre agentes e aplicações.

Referências

- Barbosa, J., Leitão, P., Trentesaux, D., Colombo, A. W., and Karnouskos, S. (2016). Cross benefits from cyber-physical systems and intelligent products for future smart industries. In *Industrial Informatics (INDIN), 2016 IEEE 14th International Conference on*, pages 504–509. IEEE.
- Da Xu, L., He, W., and Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243.
- GO0DMAN. Agent Oriented Zero Defect Multi-Stage Manufacturing. <http://go0dman-project.eu/>. Acessado 2 de Fevereiro, 2018.
- Hermann, M., Pentek, T., and Otto, B. (2016). Design principles for industrie 4.0 scenarios. In *System Sciences (HICSS), 2016 49th Hawaii International Conference on*, pages 3928–3937. IEEE.
- IDEAS. Instantly Deployable Evolvable Assembly Systems. <http://www.ideas-project.eu>. Acessado 2 de Fevereiro, 2018.
- Jazdi, N. (2014). Cyber physical systems in the context of industry 4.0. In *Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on*, pages 1–4. IEEE.
- Kagermann, H., Helbig, J., Hellinger, A., and Wahlster, W. (2013). *Recommendations for Implementing the strategic initiative INDUSTRIE 4.0: securing the future of German manufacturing industry; final report of the Industrie 4.0 working group*. Forschungsun-

- Lee, J., Bagheri, B., and Kao, H.-A. (2015). A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18–23.
- Leitão, P., Colombo, A. W., and Karnouskos, S. (2016). Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers in Industry*, 81:11–25.
- Leitao, P., Karnouskos, S., Ribeiro, L., Lee, J., Strasser, T., and Colombo, A. W. (2016). Smart agents in industrial cyber–physical systems. *Proceedings of the IEEE*, 104(5):1086–1101.
- Leitão, P., Rodrigues, N., Turrin, C., Pagani, A., and Petrali, P. (2012). Grace ontology integrating process and quality control. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, pages 4348–4353. IEEE.
- MTConnect. MTConnect Institute. <http://www.mtconnect.org>. Acessado 13 de Agosto, 2017.
- Posada, J., Toro, C., Barandiaran, I., Oyarzun, D., Stricker, D., de Amicis, R., Pinto, E. B., Eisert, P., Döllner, J., and Vallarino, I. (2015). Visual computing as a key enabling technology for industrie 4.0 and industrial internet. *IEEE computer graphics and applications*, 35(2):26–40.
- Rodrigues, L. F., de Jesus, R. A., and Schützer, K. (2016). Industrie 4.0: Uma revisão da literatura. *Revista de Ciência & Tecnologia*, 19(38):33–45.
- Self-Learning. Reliable Self-Learning Production Systems Based on Context Aware Services. <http://www.selflearning.eu>. Acessado 2 de Fevereiro, 2018.
- Shrouf, F., Ordieres, J., and Miragliotta, G. (2014). Smart factories in industry 4.0: A review of the concept and of energy management approached in production based on the internet of things paradigm. In *Industrial Engineering and Engineering Management (IEEM), 2014 IEEE International Conference on*, pages 697–701. IEEE.
- Vogel-Heuser, B., Diedrich, C., Pantförder, D., and Göhner, P. (2014). Coupling heterogeneous production systems by a multi-agent based cyber-physical production system. In *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*, pages 713–719. IEEE.
- Zhou, K., Liu, T., and Zhou, L. (2015). Industry 4.0: Towards future industrial opportunities and challenges. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on*, pages 2147–2152. IEEE.

Avaliação da testabilidade do modelo organizacional Moise⁺ baseada em Redes de Petri

Bruno Coelho Rodrigues, Eder Mateus Gonçalves

¹Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Rio Grande – RS – Brasil

brunocoelho.r@gmail.com, edergoncalves@furg.br

Abstract. *Organizational models in Multiagent Systems (MAS) are used to structure agents into groups, where members have roles to play and also constraints to obey. Even with this level of control over agents, unexpected behaviors may arise. To ensure the correct functioning of the system, software testing techniques can be employed as one of the strategies. This work aims to propose a method to evaluate testability in MAS using the Moise⁺ organization model, using Petri Nets (PN) as a tool for description and analysis. The method is based on a testability evaluation technique for BDI agents, which should be mapped to Petri nets. The initial result indicates the number of use cases required to ensure system coverage.*

Resumo. *Modelos organizacionais em Sistemas Multiagentes (SMA) são empregados para estruturar os agentes em grupos, onde os membros têm papéis a desempenhar e também restrições a obedecer. Mesmo com este nível de controle sobre os agentes, comportamentos inesperados podem surgir. Para garantir o correto funcionamento do sistema, técnicas de teste de software podem ser empregadas como uma das estratégias. Este trabalho tem por objetivo propor um método para avaliar a testabilidade em SMA que emprega o modelo de organização Moise⁺, utilizando Rede de Petri (RP) como ferramenta de descrição e análise. O método é baseado em uma técnica de avaliação de testabilidade para agentes BDI, que deve ser mapeado para Redes de Petri. O resultado inicial indica o número de casos de uso necessários para garantir a cobertura do sistema.*

1. Introdução

Segundo [Hübner et al. 2007] em uma sociedade de agentes comportamentos indesejados podem emergir no sistema. Para resolver este tipo de comportamento, os SMA podem ser representados como uma organização através de modelos organizacionais. Estes modelos coordenam os agentes em grupos e hierarquias fazendo com que eles sigam regras comportamentais específicas [Van Den Broek et al. 2005, Argente et al. 2006].

Mesmo possuindo estruturas complexas de organização, um SMA ainda possui poucas garantias de que seu funcionamento é correto, sendo este um dos obstáculos para uma maior adoção desta abordagem na indústria [Houhamdi 2011, Winikoff 2010]. Um modo de obter essa garantia é através de Teste de Software (TS). Os testes ajudam a medir a qualidade do software em termos de números de defeitos encontra-

dos [Graham et al. 2008]. Mas já se sabe que testar SMA não é uma tarefa trivial [Winikoff 2010].

[Athemana and Houhamdi 2012] abordaram os testes de comportamento de SMA utilizando um modelo para transformar diagramas de classe de agente e função do MaSE para o diagrama UML 2.0 e então para RP equivalentes. [Winikoff and Cranefield 2014] avaliaram a dificuldade de testar um SMA BDI utilizando técnicas de caixa-branca baseado em todos os caminhos, neste trabalho o autor concluiu que tentar garantir a correção do sistema testando o sistema como um todo não é viável. Em [Winikoff 2017] o objetivo é o mesmo do trabalho anterior, mas agora utilizando como métrica todas as arestas, e não mais todos os caminhos. Com esta nova métrica, concluiu-se que o número de testes necessários é suficientemente pequeno para ser viável.

O objetivo deste trabalho é identificar o número de casos de teste necessários para a cobertura total de uma especificação *Moise⁺*, e isto é feito através da contagem de caminhos na RP. É utilizando como base o teste de caixa-branca, semelhante ao apresentado em [Winikoff 2017], a GPT utilizada para representar programas BDI é bastante semelhante a especificação funcional encontrado no *Moise⁺*, e a Rede de Petri é empregada como ferramenta para modelagem e análise dos resultados, mas incluindo agora o nível social no SMA. O trabalho apresenta-se em uma etapa que os resultados são apenas iniciais, com a evolução da pesquisa podem ainda ocorrer mudanças no método.

O texto deste trabalho está organizado da seguinte forma. A seção 2 apresenta uma revisão sobre temas com importância para o entendimento do trabalho. Na seção 3 é feita uma revisão de trabalhos relacionados à testes e testabilidade em sistemas multiagentes. A seção 4 descreve a proposta do trabalho, e a seção 5 apresenta as conclusões do trabalho.

2. Referencial Teórico

2.1. Sistema Multiagente

De acordo com [Lesser 1999] sistemas multiagentes são sistemas computacionais em que dois ou mais agentes interagem ou trabalham em conjunto para executar um grupo de tarefas, ou para satisfazer um conjunto de metas. O comportamento destes agentes pode ser limitado através de políticas e/ou de uma organização, em que agentes específicos exercem certas funções dentro desta sociedade.

2.2. Organização em SMA

No início, os sistemas tinham apenas uma visão centrada nos aspectos individuais dos agentes, de modo que o SMA era projetado em termos de estados mentais dele, como as crenças, intenções e objetivos, conhecida como metodologia orientada a agentes. A partir de então os SMA evoluíram para uma metodologia orientada à organização, levando em conta suas principais metas, estrutura e normas sociais [Argente et al. 2006]. Os modelos de organização tornaram-se populares para coordenar entidades autônomas em sistemas abertos, descentralizados e dinâmicos. Estes modelos propõem uma regulação dos SMA por um conjunto de normas, planos, mecanismos e/ou estruturas formalmente especificadas para alcançar algum objetivo global desejado.

Existem vários modelos de organização, e eles estruturam o comportamento de entidades complexas em uma hierarquia de entidades encapsuladas, onde cada membro

tem funções a desempenhar [Argente et al. 2006]. A organização pode ajudar grupos de agentes simples a exibir comportamentos complexos e ajudar agentes sofisticados a reduzir a complexidade de seus raciocínios.

2.2.1. MOISE⁺

O modelo de organização *MOISE* (*Model of Organization for multi-agent SystEms*) foi inicialmente proposto por [Hannoun et al. 2000] e extensões posteriormente foram desenvolvidas, entre elas [Hübner 2003] e [Hübner et al. 2005]. Este modelo apresenta uma visão centrada na organização, e a organização é como um conjunto normativo de regras que restringe o comportamento dos agentes. Neste consideram-se três formas de representar as restrições organizacionais: papéis, planos e normas [Hannoun et al. 2000].

De acordo com [Hannoun et al. 2000] quando um agente entra em uma organização ele passa a ter que respeitar obrigações e interdições, e tem permissões relacionadas a esta organização. O *MOISE* é estruturado em três níveis: nível individual, nível coletivo e nível social. No nível individual as restrições são sobre as possibilidades de ação de cada agente. No nível coletivo a restrição está no conjunto de agentes que podem cooperar. No nível social, os enlaces organizacionais restringem os tipos de interação que os agentes podem ter com o sistema.

A Especificação Organizacional (OS) no *Moise*⁺ é formada com base nas especificações estrutural, funcional e deôntica [Hübner 2003]:

- *Especificação Estrutural (EE)*: no nível individual, os papéis têm a função de ser elo entre o agente e a organização. No nível social os papéis se relacionam com outros papéis através de ligações e compatibilidade. No nível coletivo os grupos representam um conjunto de agentes com maior afinidade e objetivos mais próximos. A Figura 1 representa o grupo *seleção* os papéis (docente, membro, funcionário secretário, presidente e candidato) e a ligação entre eles, este grupo forma uma sociedade (soc) de uma comissão de seleção para o ingresso em um curso de pós-graduação.

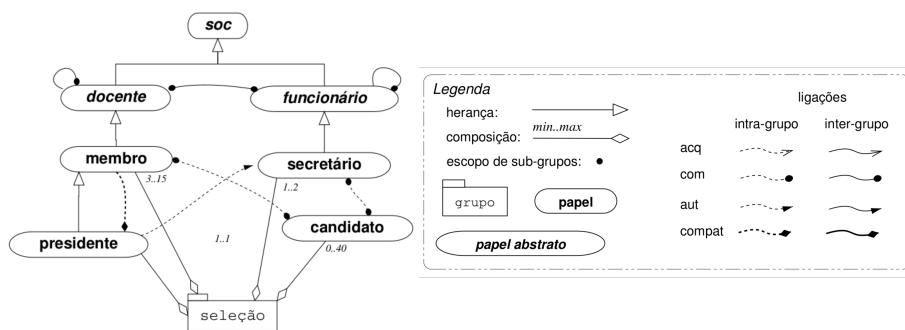


Figura 1. Especificação Estrutural [Hübner 2003].

- *Especificação Funcional (EF)*: é constituída por um conjunto de esquemas sociais e de uma relação entre preferência entre missões. Nos esquemas sociais a meta global é um conceito fundamental. No nível individual um esquema social é constituído por missões. Uma missão é o conjunto de metas globais que podem ser

passadas a um agente através de um de seus papéis. No nível coletivo um esquema social é uma árvore de decomposição de metas globais, a raiz é meta do esquema social e a decomposição de metas é feita através de planos. A Figura 2 representa o esquema social para o exemplo. A leitura do esquema social deve ser feita da esquerda para a direita, o operador *sequência* significa que a meta g_2 só pode ser realizada quando a meta g_1 for satisfeita. O operador *escolha* significa que a meta g_7 será satisfeita se uma e somente uma das metas g_8 ou g_9 for satisfeita. O operador *paralelismo*, significa que a meta g_4 será satisfeita quando ambas as metas g_5 e g_6 forem alcançadas, mas as duas sub-metas podem ser buscadas em paralelo.

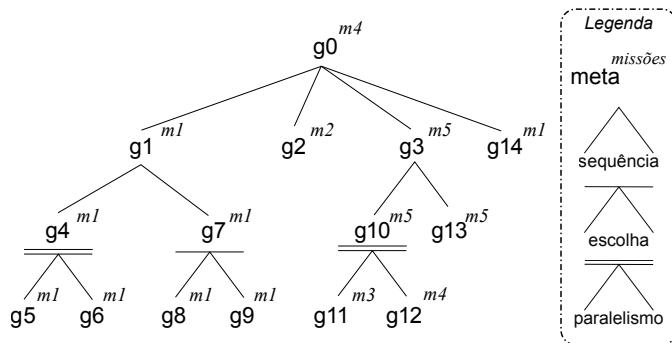


Figura 2. Esquema Social, adaptado de [Hübner 2003].

Os diferentes papéis do exemplo possuem diferentes missões, um agente no papel de candidato, tem a missão m_1 . Para cumprir esta missão o candidato deve ter como objetivo as metas ($g_1, g_4, g_5, g_6, g_7, g_8, g_9$ e g_{14}), a descrição das metas está na Tabela 1.

Tabela 1. Descrição das metas [Hübner 2003].

meta	descrição	meta	descrição
g_0	candidato é aceito no programa de pós-graduação	g_7	a inscrição está submetida
$g_1(Dt)$	a documentação é recebida no prazo	g_8	submissão eletrônica
g_2	a documentação está correta	g_9	submissão por correio
g_3	candidato é aprovado pela comissão	g_{10}	metas g_{11} e g_{12} são cumpridas
g_4	metas g_3 e g_4 são cumpridas	g_{11}	uma reunião está marcada
g_5	candidato tem toda a documentação necessária	g_{12}	um relator está indicado
g_6	candidato tem um orientador	g_{13}	o projeto do candidato é avaliado
g_{14}	formulário de matrícula preenchido é recebido		

- *Especificação Deôntica (ED):* é a especificação que relaciona a EE com a EF no nível individual, especificando quais as missões um papel tem permissão ou obrigação de realizar.

2.3. Teste de Software

O teste faz parte de um método de verificação e validação de software onde a verificação investiga se o software atende aos seus requisitos funcionais e não funcionais enquanto a validação é um processo mais amplo no qual o objetivo é garantir que o software atenda às

expectativas do cliente [Sommerville 2010]. O teste tem a função de medir a qualidade do software por pelo menos três termos: do número de defeitos encontrados, pelo rigor dos testes executados e pela cobertura do teste em relação ao sistema [Graham et al. 2008].

Segundo [Myers et al. 2011] é de grande importância ter um bom planejamento dos casos de teste, já que é impossível realizar um teste completo. Uma boa estratégia é tentar fazer o teste o mais completo possível dada as restrições de tempo e custo.

Para encontrar um equilíbrio entre o menor número de casos de testes e uma ótima cobertura do programa são adotadas estratégias de teste de software. Uma estratégia para testes de software fornece um roteiro que descreve as etapas a serem realizadas como parte do teste, quando as etapas são planejadas, realizadas, e quanto esforço, tempo e recursos serão necessários [Pressman 2005].

Diferentes abordagens podem ser utilizadas para identificar casos de testes, entre elas os testes estruturais, são baseados em uma análise dos detalhes procedurais, os caminhos lógicos através do software e as colaborações entre componentes [Jorgensen 2016, Pressman 2005].

2.4. Teste de Software em SMA

Os agentes e sistemas multiagentes possuem uma série de especificidades que tornam o processo de teste mais complexo e que devem abordar algumas questões que não eram preocupação no desenvolvimento de software orientado a objetos. [Rouff 2002, Houhamdi 2011, Nguyen 2009] citam as propriedade do agente ou SMA como ser distribuído e assíncrono, autônomos, trabalhar com envio de mensagens, possuir fatores ambientais e normativos.

Testar uma comunidade de agentes torna os objetivos de testes mais amplos, tendo que verificar se os agentes da comunidade trabalham juntos como projetado, testando a troca de comunicação entre eles, verificando se essa troca de mensagem realmente está ocorrendo entre os agentes previstos e com o ambiente, mas agora com um agravante de ter um número muito maior de interações.

2.5. Rede de Petri

Redes Petri (RP) é uma ferramenta gráfica e matemática para a descrição e análise de processos concorrentes, assíncronos e paralelos que surgem em sistemas distribuídos. Como ferramenta gráfica ela pode auxiliar na comunicação visual como um fluxograma e sendo uma ferramenta matemática, é possível estabelecer equações de estados, equações algébricas e outros modelos matemáticos que regem o comportamento dos sistemas [Murata 1989].

A Figura 3 representa uma RP simples. O grafo da rede modela as propriedades estáticas de um sistema, assim como um fluxograma representa as propriedades estáticas de um programa de computador. O grafo contém dois tipos de nós: os círculos chamados de *lugares* e as barras, chamadas de *transições*. Os nós são conectados por arcos direcionados de *lugares* para *transições* e de *transições* para *lugares* [Peterson 1977].

Além destes dois elementos, uma RP ainda possui propriedades dinâmicas que são resultantes da sua execução. O elemento que atribui esta propriedade dinâmica é a *ficha* [Peterson 1977]. As fichas são indicadas por um ponto em um lugar. A *ficha* pode

representar um recurso em uma posição, ou uma estrutura de dados que se manipula por exemplo [Cardoso and Valette 1997].

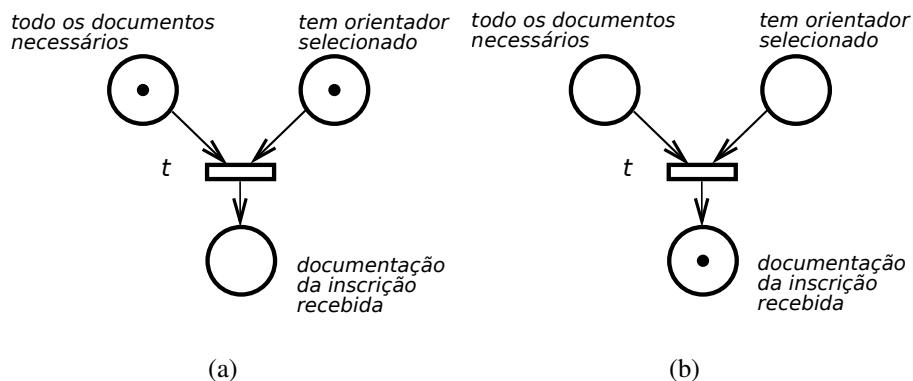


Figura 3. Rede de Petri

A dinâmica da RP atua da seguinte maneira. As fichas são movidas pelo disparo das transições associadas da rede. Na Figura 3(a) podemos observar que temos uma ficha no lugar *todos os documentos necessários* e outra ficha em *tem orientador selecionado*. A ocorrência do evento, associado à transição t só pode ocorrer se houver ao menos uma ficha em cada um destes lugares. O disparo da transição t retira uma das fichas vinculadas a cada um dos lugares de entrada e as posiciona no lugar de saída *documentação da inscrição recebida* Figura 3(b). A Figura 3 representa estados diferentes do mesmo sistema, uma evolução da rede.

3. Trabalhos Relacionados

Diversos trabalhos utilizam RP no domínio de SMA. Em [Köhler et al. 2001] Redes de Petri Coloridas (RPC) executáveis foram utilizadas para modelar a estrutura e o comportamento dos agentes. [Weyns and Holvoet 2002] relataram como principais motivos da utilização de RPC como ferramenta de modelagem, a visão conceitual clara sobre os agentes e o ambiente e o ótimo suporte a verificação e formalização. [Bai et al. 2004] apresentaram uma abordagem baseada em RPC para formar protocolos de interação flexíveis entre agentes. No trabalho [de Almeida et al. 2004] é introduzido um modelo formal para verificar os planos em um SMA, baseado na modelagem, simulação e verificação do modelo de RP Coloridas Hierárquicas (RPCH). [Poutakidis et al. 2009] apresentam ferramentas para geração de casos de teste para teste de unidade e outra para depuração e monitoramento de sistemas de agentes em execução. [Goncalves 2010] expõem um modelo de RP desenvolvida para especificar o conhecimento em agentes e SMA, independentemente de estruturas e formalismos de representação do conhecimento. [Miller et al. 2011] especifica, e mostra como medir, o grau de detalhes de um conjunto de casos de teste através de um agente de depuração que age como um oráculo, para avaliar a correção de um teste e utilizar a representação da RP do agente como suporte para medidas de cobertura de teste.

Em [Athamena and Houhamdi 2012] os autores utilizam o *Multiagent Systems Engineering* (MaSE), uma metodologia de engenharia de software orientada a agente, que é uma extensão da abordagem orientada a objetos, e propõem uma abordagem baseada em RP para o teste de comportamento de SMA. Para isso os autores desenvolveram

um modelo para a transformação dos diagrama de classe de agente e diagrama de função do MaSE, para o diagrama de sequência do modelo UML 2.0, e então propuseram um modelo para transformar os diagramas de sequencia em RP equivalentes e enfim podem ser aplicados as técnicas de testes automáticos no SMA. Segundo os autores as principais contribuições do trabalho foi propor um processo de teste completo e abrangente para o SMA e reduzir os efeitos colaterais na execução e monitoramento do teste, não utilizando agentes testadores ou agentes oráculos, o que pode influenciar no comportamento dos agentes ou desempenho do sistema.

Em [Winikoff and Cranefield 2014] o objetivo é avaliar o quanto difícil é testar um programa de agente na arquitetura BDI. Para isso o autor utiliza a técnica de testes de caixa-branca, baseado no fluxo de controle, um critério básico e muito longo para avaliar a adequação de um conjunto de testes em que todos os caminhos (*All-Path*) do programa sejam cobertos. Um motivo para a escolha de todos os caminhos é que os SMA geralmente envolvem ambientes não-episódicos, sendo que o comportamento de um determinado plano ou meta é geralmente sensível ao histórico do agente, precisando assim considerar as diferentes histórias possíveis.

Os eventos e planos podem ser visualizados como uma árvore em que cada objetivo tem como filhos as instâncias do plano que são aplicáveis a ele, e cada instância do plano tem como filhos os sub-objetivos que ele publica. Esta forma de visualização facilita a análise do número de caminhos através de um programa BDI. Com isso, o programa é visualizado como uma transformação de dados de uma árvore de planos-metas (finita) em uma sequência de execuções de ações. Assim, a questão de quanto grande é o espaço de comportamento para os agentes BDI é respondida derivando fórmulas que permitem calcular o número de comportamentos, bem-sucedidos e mal sucedidos (ou seja, falhou) para uma determinada árvore de planos-metas.

Neste artigo os autores concluíram que um teste completo em um sistema BDI não é viável. O tamanho de espaços de comportamento é muito grande e ainda se torna significativamente maior quando o sistema tem suporte à tolerância de falhas. Os autores também chegaram à conclusão de que o mecanismo de recuperação de falhas é eficaz para alcançar uma baixa taxa de falha real. E novas abordagens foram propostas para lidar com a testabilidade do sistema.

O trabalho [Winikoff 2017] retorna com o objetivo de avaliar se é possível obter garantias em um sistema multiagente através de testes, verificando a testabilidade de um programa, dando continuidade no trabalho anterior [Winikoff and Cranefield 2014], mas utilizando novas métricas. Este trabalho tem por objetivo quantificar quantos testes são necessários para testar um programa de agente BDI para satisfazer um critério considerando o critério de adequação do teste de todas as arestas, que é considerado como o mínimo geralmente aceito [Jorgensen 2016].

Uma das grandes contribuições dos autores é que a análise é genérica permitindo a aplicação a todos os programas que utilizam a arquitetura BDI. Para esta análise equações são derivadas para chegar a conclusão de quantos casos de teste (caminhos) são necessários para cobrir todas as arestas no grafo de fluxo de controle correspondente a um determinado programa BDI.

A Figura 4 apresenta um exemplo de um grafo de controle de fluxo de um pro-

grama BDI, este programa tem início em “S” e possui quatro ações $\alpha_1, \alpha_2, \alpha_3$ e α_4 . Caso alguma das ações forem bem-sucedidas então o programa executa “Y” e é encerrado em “E” concluindo-se com sucesso. Se a ação α_1 falhar ela avança para ação α_2 que pode ter sucesso ou a falha pode ocorrer novamente e assim a próxima ação seria executada. Este programa exigiria 5 testes para cobrir todas as bordas, um teste é onde as quatro ações falham ($S \rightarrow \alpha_1 \rightarrow \alpha_2 \rightarrow \alpha_3 \rightarrow \alpha_4 \rightarrow N \rightarrow E$) e os outros 4 testes é para quando uma ação é bem-sucedida e a anterior falhou.

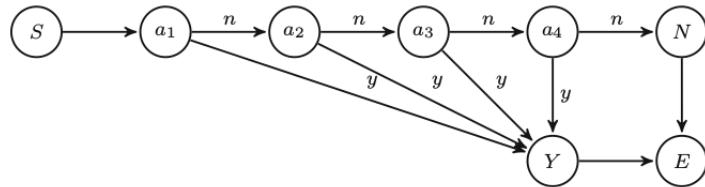


Figura 4. Controle de Fluxo [Winikoff 2017].

Para derivar equações que calculem o menor número de caminhos exigidos de um programa começando em S para chegar em E é necessário descobrir quantos destes caminhos são bem-sucedidos (passando por Y) e quantos falharam (passando por N). Os autores definiram $p(P)$ como o número de caminhos necessários para cobrir todas as arestas do grafo de fluxo de controle correspondente ao programa P , $y(P)$ para os caminhos que vão por Y e $n(P)$ para os caminhos que vão por N , então $p(P) = y(P) + n(P)$.

Logo após os autores consideram $P_1; P_2$, onde um subprograma P_1 é colocado em sequencia com P_2 Figura 5. O subprograma P_1 requer $p(P_1)$ testes para cobrir todas as arestas com $n(P_1)$ testes levando até a falha e $y(P_1)$ levando à uma execução com sucesso.

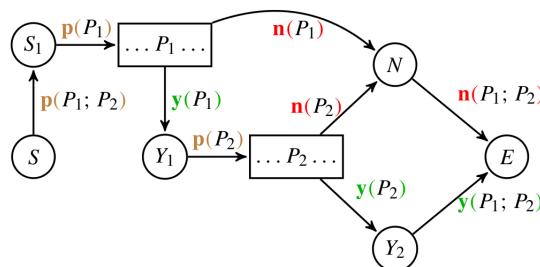


Figura 5. Controle de Fluxo de $P_1; P_2$ [Winikoff 2017].

A partir do exemplo da Figura 5 diversas equações são derivadas para os diferentes casos. Estas equações servem para determinar quantos testes são necessários para garantir uma cobertura adequada em relação ao critério todas as arestas. Então os autores implementam as equação em um programa Prolog que calcula os valores de $p(p)$, $y(P)$ e $n(P)$ para qualquer programa BDI. Para um programa BDI com 62 metas, 2 instâncias de planos aplicáveis e 2 submetas, os resultados encontrado para o [Winikoff and Cranefield 2014] foi 6.33×10^{12} testes que tiveram sucesso e 1.82×10^{13} testes que falharam, enquanto no trabalho [Winikoff 2017] o resultado encontrado é de 141 testes onde se considera todos os planos relevantes de uma meta.

Este trabalho chegou à conclusão de que o número de teste necessário para *Todas Areias* é muito menor que para a abordagem de *Todos Caminhos*, encontrando resultados onde é possível realizar os testes na prática. Outra conclusão é que permitir o tratamento de exceções não fez diferença significativa no número de testes.

4. Proposta

Em [Winikoff and Cranefield 2014] um grafo de controle de fluxo foi utilizado para avaliar a testabilidade de um sistema BDI. A proposta deste trabalho é avaliar a testabilidade de um SMA utilizando o *Moise⁺* como modelo de organização e empregando rede de Petri como ferramenta de descrição e análise. Partindo como base o trabalho de [Winikoff 2017] o grafo de controle de fluxo apresentado na Figura 4 foi transformado em uma RP Figura 6.

Como indicado pela *ficha* a rede começa no lugar **S**, a transição **t0** é disparada e então a ficha é movida para o lugar **A1**. Neste lugar tem-se uma escolha entre diferentes sequências, onde a transição **t6** é disparada quando é possível realizar a ação, indo assim para um lugar auxiliar **aux_A1**, sendo assim possível disparar a transição **t10** que leva até ao lugar **Y** que corresponde à execução bem sucedida do programa, e finalmente para **E** que é o encerramento do programa. A transição **t1** é acionada quando o agente não consegue executar a ação, direcionando assim para o **A2**, esta ação pode ser bem sucedida, indo para a transição **t7**, ou falhar e ir para a transição **t2**, operando semelhante ao apresentado no *lugar A1*.

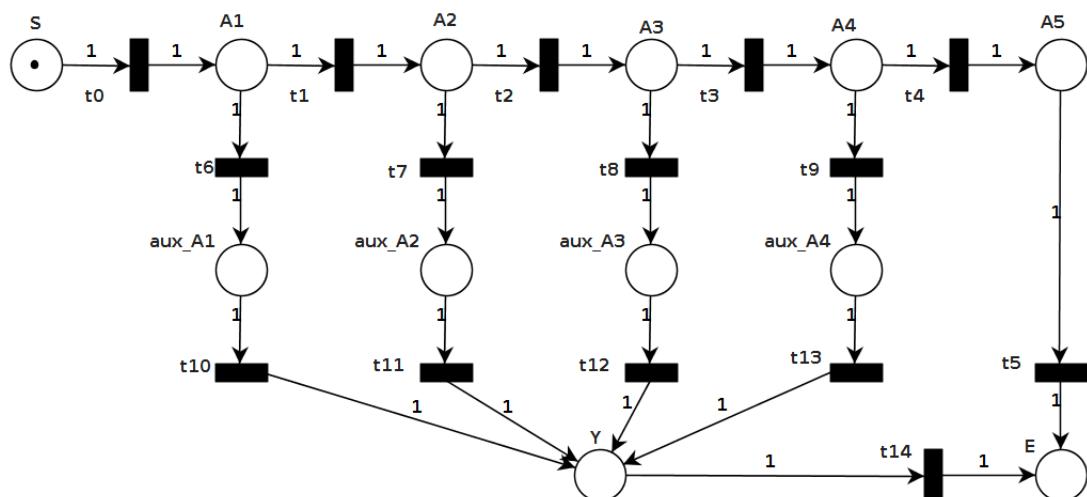


Figura 6. RP do grafo de controle de fluxo.

No método de [Winikoff and Cranefield 2014, Winikoff 2017] a avaliação de testabilidade é realizada apenas no nível de metas e ações. Para apresentar a proposta utilizando o *Moise⁺* será utilizado um exemplo exposto em [Hübner 2003]. Neste contexto temos que avaliar em múltiplos níveis que não estavam previsto em outros trabalhos, como missões, planos, metas e ações.

Primeiro é feita uma relação dos operadores apresentados na legenda da Figura 2 e sua RP equivalente, Figura 7. O operador de *sequência* representa uma cadeia de metas em sequência. A meta **G10** inicia a sequência, caso a transição **t1** for iniciada a

ficha é retirada de G10 e é colocada em **G13**, permitindo assim o seguimento da cadeia. No operador *escolha* apenas uma das metas, **G8** ou **G9** podem ser adotadas, para isso o lugar **G7** recebe uma restrição, quando uma transição for disparada e a ficha for inserida no lugar **G7**, a outra transição será desabilitada, não permitindo assim que a outra meta seja realizada. O operador *paralelismo* significa que as metas **G5** e **G6** podem progredir paralelamente e de modo assíncrono, mas a transição **t1** só será disparada quando as duas metas estiverem concluídas.

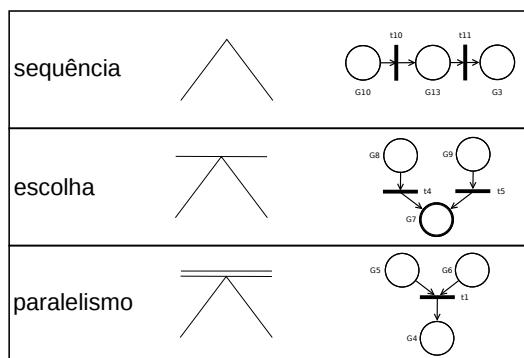


Figura 7. Relações dos operadores para Rede de Petri.

Iniciando pela árvore de metas globais da Figura 2, e utilizando os operadores da Figura 7 a árvore é transformada em RP para cada missão, ainda de maneira isolada Figura 8. O processo começa lendo a árvore da esquerda para a direita da parte inferior, e transformando as metas em lugares da RP, utilizando os operadores relacionados, até que um conjuntos de meta para finalizar a missão seja concluída.

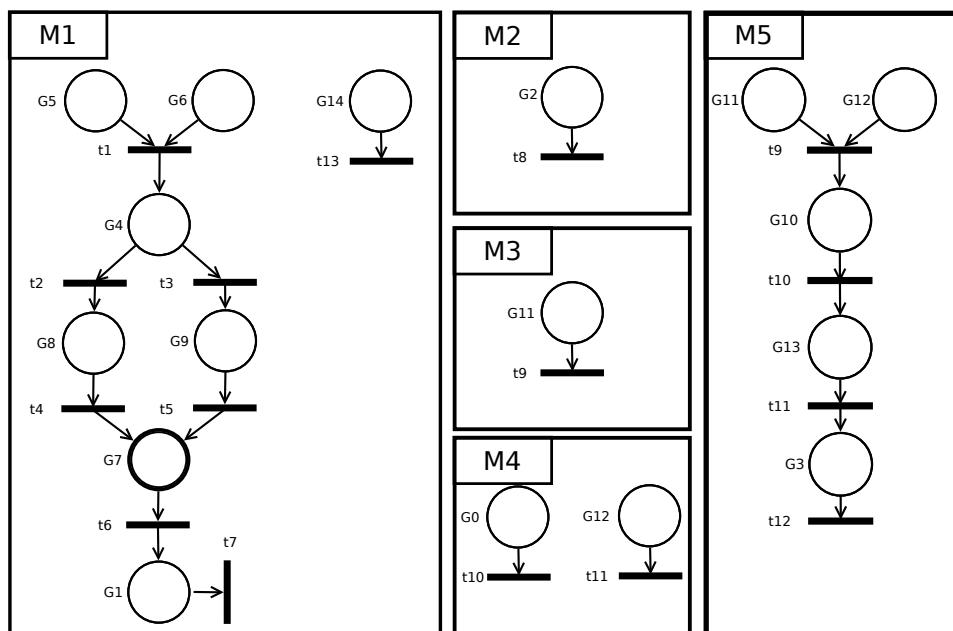


Figura 8. Transformação das missões do exemplo em RP.

5. Conclusões

Este trabalho apresenta uma proposta para avaliar a testabilidade de um SMA, que utiliza o modelo de organização *Moise*⁺ utilizando RP. O trabalho foi baseado inicialmente na abordagem proposta por [Winikoff 2017] onde as GPT representam a execução de um programa BDI e os caminhos (todos ou as arestas) representam o número de testes a serem realizados para a cobertura do sistema.

Até o momento é apresentada uma proposta para a transformação do esquema sociais, representado pela decomposição de metas globais em uma RP, formando assim a descrição das missões de maneira isolada. A próxima etapa é descrever um nível de descrição que estabelece as relações entre as missões, onde juntando as RP que representam m_1 , m_2 , m_3 , m_4 e m_5 o resultado é g_0 . Para cada meta os agentes ainda possuem planos e estes planos possuem ações. Esta relação de subníveis também será descrita e apresentada em redes de Petri.

Referências

- Argente, E., Julian, V., and Botti, V. (2006). Multi-agent system development based on organizations. *Electronic Notes in Theoretical Computer Science*, 150(3):55–71.
- Athamena, B. and Houhamdi, Z. (2012). A petri net based multi-agent system behavioral testing. *Modern Applied Science*, 6(3):46.
- Bai, Q., Zhang, M., and Win, K. T. (2004). A colored petri net based approach for multi-agent interactions. In *Proc. of 2nd International Conference on Autonomous Robots and Agents, Palmerston North, New Zealand*, pages 152–157.
- Cardoso, J. and Valette, R. (1997). *Redes de petri*. Editora da UFSC.
- de Almeida, H. O., da Silva, L. D., Perkusich, A., and de Barros Costa, E. (2004). A formal approach for the modelling and verification of multiagent plans based on model checking and petri nets. In *International Workshop on Software Engineering for Large-Scale Multi-agent Systems*, pages 162–179. Springer.
- Goncalves, E. M. N. (2010). An approach to specify knowledge in multi-agent systems using petri nets. In *Network and System Security (NSS), 2010 4th International Conference on*, pages 456–461. IEEE.
- Graham, D., Van Veenendaal, E., and Evans, I. (2008). *Foundations of software testing: ISTQB certification*. Cengage Learning EMEA.
- Hannoun, M., Boissier, O., Sichman, J., and Sayettat, C. (2000). Moise: An organizational model for multi-agent systems. *Advances in Artificial Intelligence*, pages 156–165.
- Houhamdi, Z. (2011). Multi-agent system testing: A survey. *International Journal of Advanced Computer Science and Applications*, 2(6):135–141.
- Hübner, J. F. (2003). *Um modelo de reorganização de sistemas multiagentes*. PhD thesis, Universidade de São Paulo.
- Hübner, J. F., Sichman, J. S., and Boissier, O. (2005). \backslash mathcal {SM} oise $^+$: A middleware for developing organised multi-agent systems. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 64–77. Springer.

- 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. *International Journal of Agent-Oriented Software Engineering*, 1(3-4):370–395.
- Jorgensen, P. C. (2016). *Software testing: a craftsman's approach*. CRC press.
- Köhler, M., Moldt, D., and Rölke, H. (2001). Modelling the structure and behaviour of petri net agents. *Applications and Theory of Petri Nets 2001*, pages 224–241.
- Lesser, V. R. (1999). Cooperative multiagent systems: A personal view of the state of the art. *IEEE Transactions on knowledge and data engineering*, 11(1):133–142.
- Miller, T., Padgham, L., and Thangarajah, J. (2011). Test coverage criteria for agent interaction testing. In *Agent-oriented Software Engineering XI*, pages 91–105. Springer.
- Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.
- Myers, G. J., Sandler, C., and Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.
- Nguyen, D. C. (2009). *Testing techniques for software agents*. PhD thesis, University of Trento.
- Peterson, J. L. (1977). Petri nets. *ACM Computing Surveys (CSUR)*, 9(3):223–252.
- Poutakidis, D., Winikoff, M., Padgham, L., and Zhang, Z. (2009). Debugging and testing of multi-agent systems using design artefacts. *Multi-Agent Programming*, pages 215–258.
- Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave Macmillan.
- Rouff, C. (2002). A test agent for testing agents and their communities. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 5, pages 5–2638. IEEE.
- Sommerville, I. (2010). *Software Engineering*. Addison-Wesley Publishing Company, USA, 9th edition.
- Van Den Broek, E. L., Jonker, C. M., Sharpanskykh, A., Treur, J., et al. (2005). Formal modeling and analysis of organizations. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 18–34. Springer.
- Weyns, D. and Holvoet, T. (2002). A colored petri-net for a multi-agent application. In *Proceedings of MOCA'02*, volume 561, pages 121–141.
- Winikoff, M. (2010). Assurance of agent systems: What role should formal verification play? *Specification and Verification of Multi-agent systems*, pages 353–383.
- Winikoff, M. (2017). Bdi agent testability revisited. *Autonomous Agents and Multi-Agent Systems*, pages 1–39.
- Winikoff, M. and Cranfield, S. (2014). On the testability of bdi agent systems. *Journal of Artificial Intelligence Research*, 51:71–131.

Modelo de Fusão de Dados com Incerteza para Consciência Situacional

Munyque Mittelmann¹, Jerusa Marchi², Aldo von Wangenheim¹

¹Programa de Pós-Graduação em Ciência da Computação

Departamento de Informática e Estatística

Universidade Federal de Santa Catarina (UFSC)

Florianópolis – SC – Brazil

²Departamento de Informática e Estatística

Universidade Federal de Santa Catarina (UFSC)

Florianópolis – SC – Brazil

mmittelmann@incod.ufsc.br, {jerusa.marchi, aldo.vw}@ufsc.br

Abstract. *Situation Awareness provides a theory for agents decision making in order to allow perception and comprehension of his environment. However, transformation of sensory stimulus in beliefs in order to favor the BDI reasoning cycle is still an unexplored subject. This paper presents a model to beliefs generation using fuzzy-bayesian inference. An example in robotics navigation and location is used to illustrate the proposal.*

Resumo. *A área de Consciência Situacional provê uma teoria que embasa a tomada de decisão em agentes, de forma a permitir a percepção e a compreensão do ambiente em que o agente está inserido. Contudo, a transformação de estímulos sensoriais em crenças que favoreçam o ciclo de raciocínio em agentes BDI ainda é uma área pouco explorada. Este trabalho apresenta um modelo para geração de crenças, utilizando inferência Fuzzy-Bayesiana. Para ilustrar a proposta, um exemplo em navegação e localização robótica é utilizado.*

1. Introdução

Na área de psicologia cognitiva, a avaliação da situação atual é considerada como um aspecto cognitivo necessário para que haja uma tomada de decisão efetiva [Wickens and Hollands 2000]. Neste sentido, a Consciência Situacional (*Situation Awareness - SA*) é definida como a percepção dos elementos no ambiente dentro de um volume espaço-temporal e a compreensão de seus significados e de sua projeção num futuro próximo [Endsley 1995].

Endsley [Endsley 1995] define um modelo de SA que incorpora três diferentes níveis de consciência: (i) a percepção de sinais, (ii) a compreensão e integração da informação e (iii) a projeção da informação em eventos futuros. Embora a abordagem original seja referente a SA humana, o modelo também é utilizado como uma justificativa para estruturar o processo computacional de consciência situacional [Kokar et al. 2009]. A diferença entre o processo de consciência situacional humano e o computacional é que o processo humano precisa ser medido e possivelmente suportado, enquanto que o processo computacional precisa ser definido e implementado [Kokar et al. 2009].

O primeiro nível de SA é a Percepção e envolve perceber os sinais referentes aos atributos e dinâmicas de elementos relevantes no ambiente. O procedimento de coleta e monitoramento de dados situacionais varia dependendo das particularidades do cenário de aplicação, podendo incluir redes de sensores, dispositivos e atuadores [Chen et al. 2009].

O nível intermediário de SA é a Compreensão, que é responsável por integrar e compreender o significado das informações percebidas, sendo composto pela agregação de situações (associação), medição das propriedades das situações (avaliação), estimativa de situações (inferência) e medições de capacidade, oportunidade e intenção da situação (avaliação de impacto) [Golestan et al. 2016].

Por fim, o nível de Projeção é responsável por projetar a informação percebida e compreendida em situações futuras. Trabalhos na área de agentes autônomos costumam utilizar a projeção para a tomada de decisão e planejamento do agente [Insaurralde and Petillot 2015].

Agentes autônomos usualmente fazem uso de múltiplos sensores para possibilitar a captura de diferentes aspectos do ambiente. Nestes casos, é necessário que o agente tenha a capacidade de realizar um processo conhecido como fusão de dados, que inclui a fusão dos dados de baixo nível produzidos por sensores físicos, o reconhecimento de entidades relacionadas ao contexto para formar uma figura unificada do ambiente e identificação das correlações existentes entre entidades e situações [Golestan et al. 2016, Steinberg and Bowman 2004].

A pesquisa na área de Fusão de Informação fraciona o processo de fusão de dados sensoriais em quatro níveis funcionais, que são detalhados na Seção 2.1 [White 1988, Steinberg and Bowman 2004]. Os níveis de fusão de dados podem ser relacionados aos níveis de SA propostos por Endsley [Endsley 1995], uma vez que a fusão de dados de baixo nível é inerente ao nível de Percepção do agente e o produto do processo de fusão de dados de alto nível corresponde a ter alcançado a Consciência Situacional do agente [Golestan et al. 2016].

Ao nível de Percepção, os dados capturados por sensores físicos normalmente são afetados por algum nível de imprecisão ou incerteza em suas medidas. Tratar a incerteza proveniente de dados sensoriais é um dos desafios que impulsionam a pesquisa em fusão de dados [Khaleghi et al. 2013]. Dependendo da natureza da incerteza presente nos dados, diferentes métodos podem ser aplicados. Para tratar simultaneamente a incerteza por aleatoriedade e por vagueza, pode-se employar a inferência Fuzzy-Bayesiana.

Neste trabalho, a inferência Fuzzy-Bayesiana é integrada a um modelo de fusão de dados com o objetivo de prover o nível de Percepção na Consciência Situacional de um agente. As percepções do agente são representadas por um conjunto de crenças, gerado como saída do modelo. As crenças de Percepção possuem grau de certeza e são propagadas para os níveis de Compreensão e Projeção.

O trabalho está estruturado como segue: na Seção 2 são apresentados os conceitos referentes aos níveis Funcionais de Fusão de Dados e a descrição da inferência Fuzzy-Bayesiana; na Seção 3 é apresentada a proposta de modelo paralelamente a um cenário exemplificando sua aplicação; a Seção 4 apresenta a implementação do exemplo em Java; por fim, na Seção 5 são apresentadas algumas considerações da pesquisa.

2. Conceitos Básicos

Apresenta-se nesta seção a distinção entre os níveis de fusão de dados e sua relação com os níveis de SA de Endsley [Endsley 1995], e posteriormente, descreve-se a inferência Fuzzy-Bayesiana empregada nesta pesquisa dentro de um modelo de fusão de dados.

2.1. Níveis Funcionais de Fusão de Dados

A classificação mais comum do processo de fusão de dados é o modelo JDL proposto por White [White 1988] que considera o processo de fusão em quatro níveis crescentes de abstração: Objeto, Situação, Impacto e Refinamento do Processo.

Como extensão do modelo JDL, Steinberg e Bowman [Steinberg and Bowman 2004] apresentam os níveis Funcionais de Fusão de Dados (FFD). Neste particionamento, uma entidade de interesse do sistema pode ser vista como (i) um indivíduo com atributos, características e comportamentos ou (ii) como um conjunto de componentes inter-relacionados. Os níveis FFD de Steinberg e Bowman [Steinberg and Bowman 2004] são definidos conforme segue:

- (0-FFD) Avaliação de Sinais: estimação e predição de sinais ou estados característicos;
- (1-FFD) Avaliação de Entidades: estimação e predição de parametrização de entidade e estados atributivos;
- (2-FFD) Avaliação de Situações: estimação e predição das estruturas de partes da realidade, como relação entre entidades e suas implicações nos estados de entidades relacionadas;
- (3-FFD) Avaliação de Impacto: estimação e predição do custo/benefício do sinal, entidade ou estados de situações;
- (4-FFD) Avaliação de Desempenho: estimação e predição do desempenho do sistema quando comparado aos estados desejados e medidas de efetividade.

A fusão de dados de baixo nível corresponde aos níveis 0-FFD, 1-FFD e 2-FFD, os quais são responsáveis pela identificação de entidades e suas correlações. Estes três níveis são análogos ao nível de Percepção de SA. O nível 2-FFD também inclui a estimação da implicação dos relacionamentos entre entidades em seus estados, podendo ser relacionado ao nível de Compreensão de SA. Os níveis 3-FFD e 4-FFD têm a função de prever custos, estados e desempenho do sistema, tendo correspondência, principalmente, com o nível de Projeção de SA.

Relacionamentos entre entidades no nível 2-FFD podem ter características implícacionais e são comumente modelados por meio de Modelos Gráficos Probabilísticos, como Redes Bayesianas, Redes Bayesianas Dinâmicas e Redes Fuzzy-Bayesianas [Koller and Friedman 2009, Golestan et al. 2016]. Com intuito de compreender a fusão de dados de baixo nível e o nível de Percepção de SA, este trabalho aplica inferência Fuzzy-Bayesiana em um modelo para agentes.

2.2. Inferência Fuzzy-Bayesiana

Conjuntos difusos são classes que possuem continuidade em seus graus de pertinência [Zadeh 1965]. Dados difusos usualmente não são descritos pela inferência Bayesiana padrão e, assim, é necessário modelá-los de modo a incorporar seu aspecto difuso antes de analisá-los por métodos estatísticos [Viertl 1987, Viertl 2008]. Os estudos de Viertl

[Viertl 1987, Viertl 1989, Viertl and Hule 1991, Viertl 1995, Viertl 2008], investigam a generalização da estatística Bayesiana para dados difusos, tais abordagens híbridas são conhecidas como inferências *Fuzzy-Bayesianas*.

Dentre as inferências *Fuzzy-Bayesianas* existentes, o modelo apresentado em [Brignoli et al. 2015] é uma extensão que visa incluir variáveis difusas não-dicotômicas¹. O modelo de inferência Bayesiana com entradas difusas proposto é baseado na superposição de estados. Brignoli et al. [Brignoli et al. 2015] define a probabilidade \tilde{P} sobre evidências imprecisas, conforme a seguinte equação:

$$\tilde{P}(H_i|E_jx_k) = \frac{P(H_i) \times \prod_{j=1}^m \sum_{k=1}^t (P(E_jx_k|H_i) \times (\varepsilon E_jx_k))}{\sum_{l=1}^n P(H_l) \times \prod_{j=1}^m \sum_{k=1}^t (P(E_jx_k|H_l) \times (\varepsilon E_jx_k))} \quad (1)$$

Na qual, H_i , com $1 \leq i \leq n$, é o vetor de n hipóteses; E_jx_k , com $1 \leq j \leq m$ e $1 \leq k \leq t$, é o vetor de m evidências E observadas nos estados x contínuos; t é a quantidade de estados; $P(H_i)$ é a probabilidade *a priori* sobre H_i ; $P(E_jx_k|H_i)$ é a probabilidade condicional de E_jx_k dado H_i ; e εE_jx_k representa o estado x_k da evidência E_j em relação a função de pertinência difusa. Assim, $\tilde{P}(H_i|E_jx_k)$ é o vetor das probabilidades de H_i ajustadas pela imprecisão sobre as variáveis E_j de estados contínuos x_k .

O modelo de inferência *Fuzzy-Bayesiana* de [Brignoli et al. 2015] é empregado nesta proposta. A inferência *Fuzzy-Bayesiana* é utilizada para tratar a incerteza por vagueza presente nas entidades observadas e para modelar os relacionamentos probabilísticos entre entidades.

3. Proposta

O trabalho propõe um modelo de Fusão de Dados com Incerteza para atingir a Consciência Situacional de um agente ao nível de Percepção. O modelo abrange os níveis de fusão de dados 0-FFD, 1-FFD e 2-FFD. Para esclarecer a proposta, é introduzido um cenário de exemplo na Seção 3.1. O cenário de exemplo é acompanhado em paralelo ao detalhamento da proposta. A Figura 1 introduz uma visão geral do modelo de Fusão de Dados, ilustrando sua relação com os modelos de SA e FFD.

No nível 0-FFD, é realizada a captura das medições sensoriais, são definidas as estruturas dos conjuntos de medição, mas não são tratadas suas semânticas. O nível 1-FFD é responsável pela estimativa de entidades e classificações. As entidades são agregadas e consideradas como observações, em seguida, são classificadas em relação a sua pertinência a conjuntos fuzzy do domínio.

O nível 2-FFD diz respeito às relações entre entidades e suas implicações. A implicação entre as entidades é realizada por meio da inferência Bayesiana com as entradas difusas do nível predecessor. Com base nos resultados da inferência, o modelo gera crenças, com grau de certeza, sobre a situação observada. Neste modelo, os níveis 0-FFD, 1-FFD e 2-FFD representam o nível de Percepção de SA. As crenças geradas pelo modelo são disponibilizadas para serem utilizadas nos níveis de Compreensão e Projeção de SA. Os estágios do modelo são especificadas nas Seções 3.2, 3.3 e 3.4.

¹ Considera-se como variável dicotômica aquela que é dividida logicamente em apenas dois conceitos, geralmente opostos.

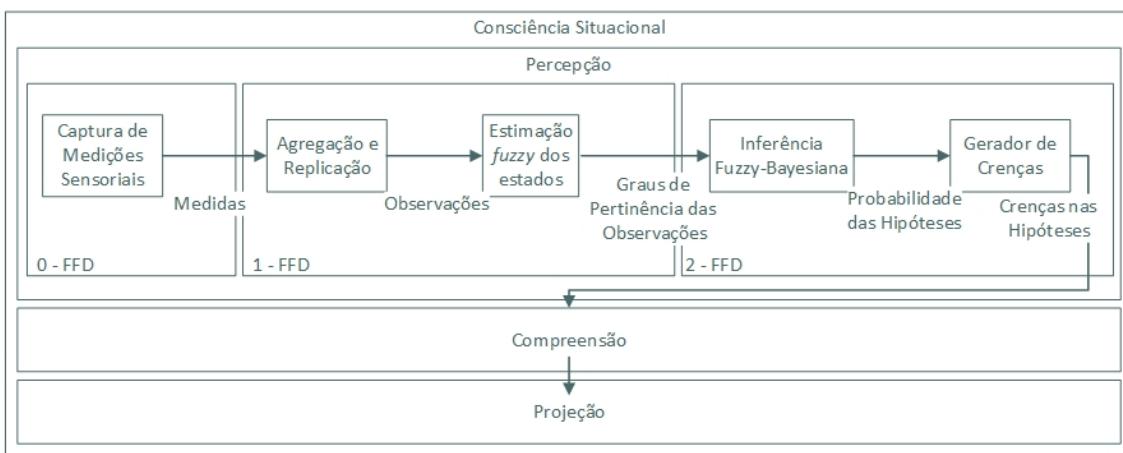


Figura 1. Modelo de Fusão de Dados para Consciência Situacional

3.1. Exemplo de Cenário de Aplicação

Considere a existência de um robô cujo objetivo é limpar os cômodos de uma casa. Como cada cômodo demanda equipamentos de limpeza distintos, o robô precisa saber em qual cômodo está. A casa é composta por dois quartos (Q_1 e Q_2 , respectivamente) e dois corredores conectando os quartos (C_1 e C_2 , respectivamente). A Figura 2(a), ilustra a distribuição dos cômodos Q_1 , Q_2 , C_1 e C_2 pela casa.

O robô caminha aleatoriamente pela casa e não conhece a disposição de seus cômodos. Para perceber o ambiente, o robô possui apenas sensores para captar sons, luminosidade e distância. Com estes sensores, o robô precisa descobrir, respectivamente, se a sonorização do ambiente é alta ou baixa (Figura 2(b)), se o ambiente está claro ou escuro (Figura 2(c)) e se o espaço entre as paredes é amplo ou apertado (Figura 2(d)). Observa-se que as informações sensoriais são classificadas por variáveis linguísticas provadas de imprecisão por vaguedade (alto e baixo, claro e escuro, amplo e apertado).

Antes de ser posto em funcionamento, o robô recebeu informações amostrais sobre mapeamentos anteriores do ambiente. Assim, ele conhece a probabilidade *a priori* de estar em cada cômodo, bem como as probabilidades condicionais de estar com sonorização alta ou baixa, em um ambiente claro ou escuro e em um espaço amplo ou apertado dado que está em algum cômodo específico. As Figuras 2(b), 2(c) e 2(d) ilustram, respectivamente, a distribuição da sonorização, iluminação e espaço pela casa em um determinado momento. Os valores das medições de som, luminosidade e distância podem variar dependendo do momento, seja por causa de falha sensorial ou de mudança no ambiente.

Munido apenas das informações sensoriais e das probabilidades *a priori* e condicionais, o robô deve decidir em qual cômodo está e, assim, ser capaz de determinar seus planos de ação. Durante a apresentação da proposta, o cenário do Robô Limpador é retomado de modo a exemplificar a aplicação e os níveis do modelo. A proposta é apresentada em três estágios, correspondentes aos níveis FFD abrangidos pelo modelo.

3.2. Nível de Avaliação de Sinais (0-FFD)

No nível 0-FFD, é realizada a estimativa de sinais ou medições. O Agente possui um conjunto $\mathcal{S}_T = \{S_{T_1}, \dots, S_{T_n}\}$ de sensores, onde T_i representa o tipo de sensor e $S_{T_i} =$

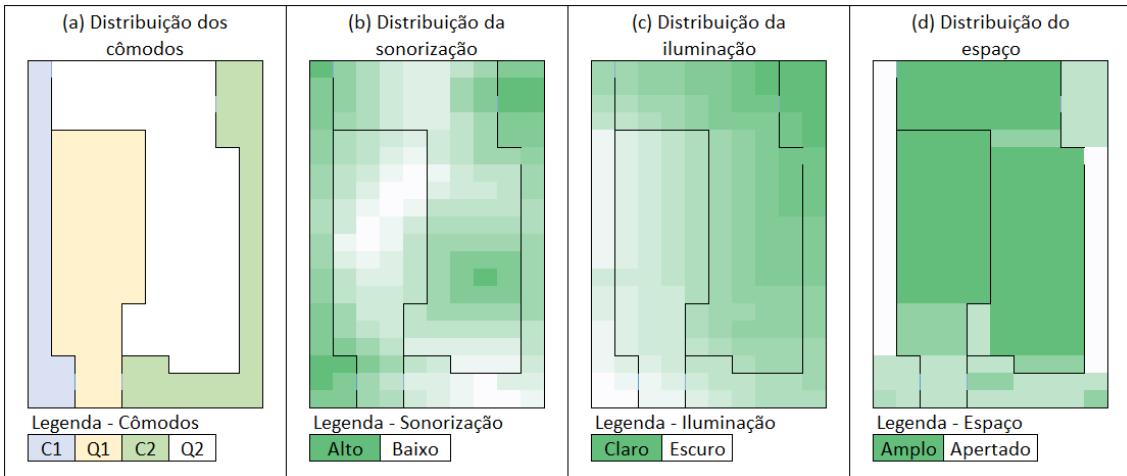


Figura 2. Cenário do Robô Limpador

$\{s_1, \dots, s_m\}$ representa um conjunto de sensores físicos de mesmo tipo. Para cada S_{T_i} são obtidos conjuntos M_i de medidas, de forma que $\mathcal{M} = \bigcup_{i \in T} M_i$ forma o conjunto de todos os sinais percebidos pelo agente.

Exemplo 1 O Robô Limpador da Seção 3.1 possui três tipos de sensores: luminosidade, distância e sonoro, ou seja: $\mathcal{S}_{T_{robo}} = \{S_{luz}, S_{distancia}, S_{som}\}$. O robô possui um sensor de luminosidade, dois de distância e um sonoro, assim $S_{luz} = \{s_{luz}\}$, $S_{distancia} = \{s_{d1}, s_{d2}\}$, com s_{d1} e s_{d2} sendo, respectivamente, os sensores de distância à direita e à esquerda do robô, e $S_{som} = \{s_{som}\}$. As medidas obtidas pelos sensores são $\mathcal{M}_{robo} = \{m_{luz}, m_{d1}, m_{d2}, m_{som}\}$. A Figura 3 ilustra o nível de Avaliação de Sinais para o Robô Limpador.

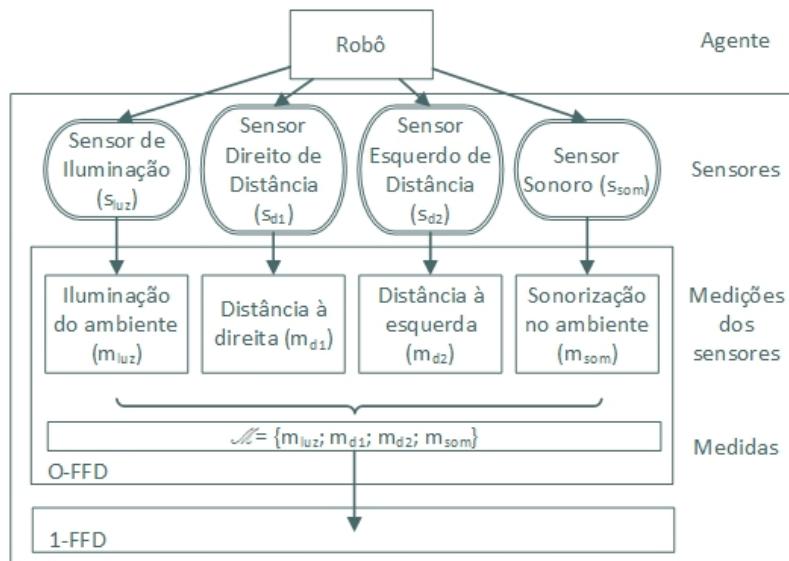


Figura 3. Nível de Avaliação de Sinais (0-FFD) no Cenário do Robô Limpador

3.3. Nível de Avaliação de Entidades (1-FFD)

O nível 1-FFD é responsável pela estimativa de evidências e suas classificações. Neste estágio, ocorre a transformação dos dados de medidas em evidências observadas. A

transformação das medidas é definida como a aplicação de uma função $f(x) \rightarrow \mathbb{R}$ sobre os mesmos, onde $f \in \{A, R\}$ é uma função de Agregação ou Replicação, $x \in 2^{\mathcal{M} \setminus \emptyset}^2$ é a aridade da função. Cada função $f(x)$ gera uma nova evidência observada $o_k \in \mathbb{R}$, na qual k representa um domínio (ex.: distância, iluminação, sonorização).

A função de Agregação A é realizada por meio de operações sobre uma ou mais as medições pertencentes a \mathcal{M} e é definida por $A(m_i, \dots, m_j) \rightarrow \mathbb{R}$, com $(m_i, \dots, m_j) \in 2^{\mathcal{M} \setminus \emptyset}$. A função de Replicação R realiza cópia de um elemento de \mathcal{M} , tal que $R(m_i) \rightarrow \mathbb{R}$, com $m_i \in \mathcal{M}$.

Sobre o_k é aplicado o operador de normalização, tal que $N : \mathbb{R} \rightarrow [0, 1]$. Tal operação visa facilitar o suporte de cada evidência observada, entretanto, não é necessário realizar a normalização. Para fins de simplificação da notação, continuaremos a denominar a evidência observada normalizada de o_k . Consideraremos $\mathcal{O} = \bigcup_{\forall k} o_k$ o conjunto global de evidências.

Exemplo 2 No cenário do Robô Limpador, as medições m_{d1} e m_{d2} são agregadas tal que $o_{espaco} = \|m_{d1}\| + \|m_{d2}\|$ ³. Ou seja, o espaço entre as paredes o_{espaco} é a soma da distância da parede à direita do robô com a distância da parede à esquerda do robô. As medições m_{luz} e m_{som} são replicadas como evidências observadas, tal que $o_{luz} = m_{luz}$ e $o_{som} = m_{som}$. Nota-se que $\mathcal{O}_{robo} = \{o_{luz}, o_{espaco}, o_{som}\}$ é o conjunto normalizado de todas as evidências observadas pelo robô.

Para cada evidência observada o_k é considerado um universo de discurso no intervalo $[min_k, max_k]$, denotado por \mathcal{A}_k , onde min_k e max_k , respectivamente, são os valores mínimos e máximos do domínio K. Para cada $o_k \in \mathcal{O}$ será calculado o valor de pertinência a algum conjunto fuzzy A_k , definido sobre $\mathcal{A}_k = \{A_{k_1}, \dots, A_{k_n}\}$. O grau de pertinência de o_k ao conjunto A_{k_i} é denotado por $\mu_{A_{k_i}}(o_k)$. O conjunto de todos os graus de pertinência $\mu_{A_{k_i}}(o_k), \forall A_{k_i} \in \mathcal{A}_k$ e $\forall o_k \in \mathcal{O}$ é denotado por: $\mu(\mathcal{O}) = \bigcup_{\forall k, \forall i} \mu_{A_{k_i}}(o_k)$.

Exemplo 3 Para cada evidência observada pelo Robô Limpador, foi definido um universo de discurso e calculados os graus de pertinência da evidência a cada conjunto do universo de discurso. Para o_{luz} , $\mathcal{A}_{luz} = \{\text{claro; escuro}\}$. Para o_{espaco} , $\mathcal{A}_{espaco} = \{\text{amplo; apertado}\}$. Para o_{som} , $\mathcal{A}_{som} = \{\text{baixo; alto}\}$. A seguir, para cada evidência o_k observada pelo robô, foi estimado seu grau de pertinência $\mu_{A_{k_i}}(o_k)$ para cada conjunto $A_{k_i} \in \mathcal{A}_k$ (Figura 4).

3.4. Nível de Avaliação de Situação (2-FFD)

O nível 2-FFD é responsável por estimar as relações entre as evidências e suas implicações. Os graus de pertinência das observações $\forall \mu_{A_{k_i}}(o_k) \in \mu(\mathcal{O})$, são considerados como as evidências da inferência Fuzzy-Bayesiana. Sendo $\mathcal{H} = \{H_1, \dots, H_n\}$ o conjunto de hipóteses sobre Z, Z como a situação na qual o agente está, $P(H_h) \rightarrow [0; 1]$ representa a probabilidade a priori da h-ésima hipótese e $P(o_k | H_h) \rightarrow [0; 1]$ é a probabilidade de observar o_k dada a hipótese H_h . A probabilidade da hipótese H_h ajustada pela imprecisão sobre todas as observações $o_k \in \mathcal{O}$ é denotada por $\tilde{P}(H_h | \mathcal{O})$ e definida aplicando a Equação (1) às definições do modelo, tal que:

² $2^{\mathcal{M} \setminus \emptyset}$ representa o conjunto das partes de \mathcal{M} sem o conjunto vazio \emptyset .

³ $\|x\|$ representa o módulo de um número $x \in \mathbb{R}$.

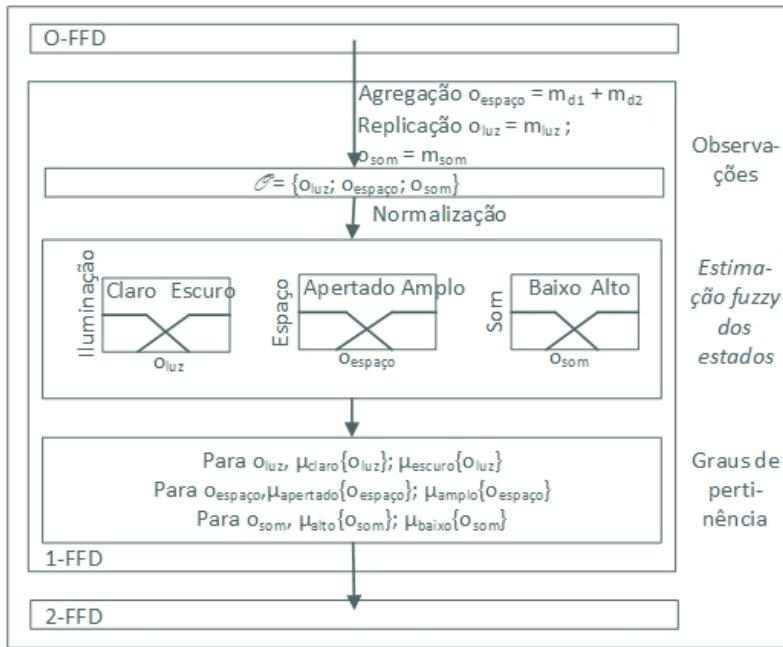


Figura 4. Nível de Avaliação de Entidades (1-FFD) no Cenário do Robô Limpador

$$\tilde{P}(H_h|\mathcal{O}) = \frac{P(H_h) \times \prod_{k=1}^{|\mathcal{O}|} \sum_{i=1}^n (P(o_k|H_h) \times \mu_{A_{k_i}}(o_k))}{\sum_{l=1}^{|H|} P(H_l) \times \prod_{k=1}^{|\mathcal{O}|} \sum_{i=1}^n (P(o_k|H_l) \times \mu_{A_{k_i}}(o_k))} \quad (2)$$

A probabilidade ajustada pela imprecisão $\tilde{P}(H_h|\mathcal{O})$ é estimada $\forall H_h \in \mathcal{H}$.

A situação avaliada no cenário da Seção 3.1 é a localização atual do robô, ou seja $Z_{robo} = Local$. As hipóteses são as possibilidades de localização, os cômodos da casa, então $\mathcal{H}_{robo} = \{H_{Q_1}; H_{Q_2}; H_{C_1}; H_{C_2}\}$. Assim, $P(H_h)$ representa as probabilidades a priori do robô estar em cada cômodo da casa. $P(o_k|H_h)$ são as probabilidades condicionais dos estados de uma observação, ou seja $o_k = luz, som$ ou $espaco$, sabendo-se que está em determinado cômodo (H_h).

Exemplo 4 Para o Robô Limpador, a inferência Fuzzy-Bayesiana descobre as probabilidades $\tilde{P}(H_h|\mathcal{O})$ de estar em cada cômodo $\forall H_h \in \mathcal{H}_{robo}$ dadas as evidências observadas pelo robô $\mathcal{O}_{robo} = \{o_{luz}, o_{espaco}, o_{som}\}$.

A última etapa do modelo é denominada Gerador de Crenças (GC). Considerase que o conjunto de hipóteses \mathcal{H} seja formado pelas implicações das relações entre as observações, ou seja, a conclusão do que o agente está observando. Assim, para cada hipótese $H_h \in \mathcal{H}$, o GC gera uma crença em lógica de predicados na forma: $Z(H_h, G_h)$, onde, Z é o símbolo de predicado que representa a situação atual acreditada pelo agente e G_h é o grau de certeza do agente em relação a hipótese H_h , dado por $G_h = \tilde{P}(H_h|\mathcal{O})$.

Exemplo 5 Para cada cômodo da casa, o GC do Robô Limpador gera uma crença com o grau de certeza que o robô tem de estar no cômodo. A Figura 5 ilustra o nível de avaliação de situação no cenário do Robô Limpador.

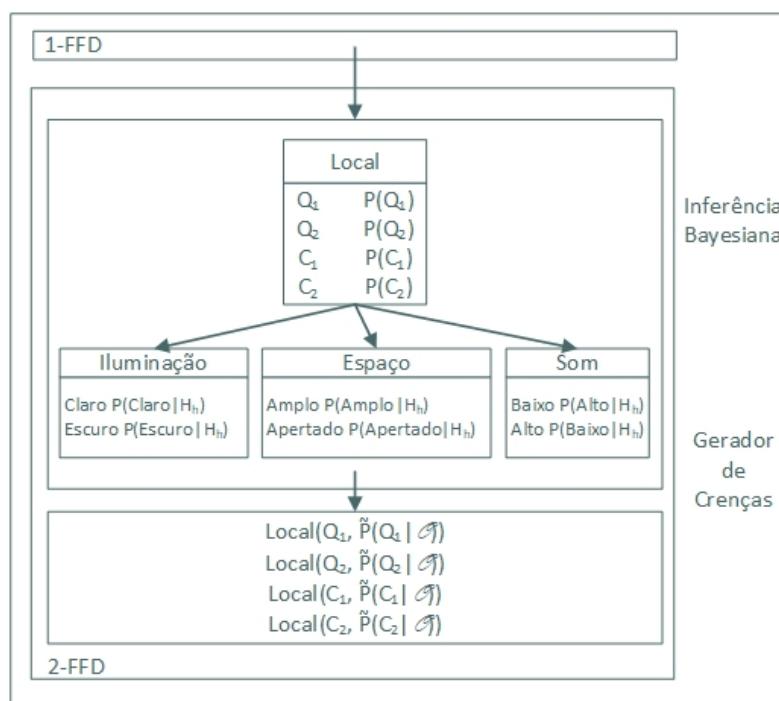


Figura 5. Nível de Avaliação de Situação (2-FFD) no Cenário do Robô Limpador

As crenças geradas pelo modelo são do nível de Percepção e possuem o propósito de fundamentar a integração de crenças distintas como crenças inferidas (nível de Compreensão) e sua projeção em crenças futuras (nível de Projeção). A obtenção de crenças situacionais aos níveis de Percepção, Compreensão e Projeção permite que o agente realize planejamento e tomada de decisão com base em sua situação atual. É intuitivo que o Agente deva considerar, durante a Compreensão e Projeção, a crença com maior grau de certeza. Entretanto, incluir as crenças para todas as hipóteses permite que o agente reaja a situações para as quais nenhuma crença tenha um grau de certeza razoável.

Exemplo 6 Considere um grau de certeza na crença C_1 de 51% e em C_2 de 49%. Estes valores indicam que o agente tem preferência em acreditar em C_1 . Entretanto, como a diferença do grau de certeza entre as crenças é baixa, existe grande incerteza em relação a situação. A propagação desta incerteza permite considerá-la nos planos do agente. Suponha que o agente deva realizar a ação A_1 quando está na situação descrita por H_1 . Ao invés de executar A_1 imediatamente, o agente pode reagir de acordo com a incerteza: esperar uma nova informação sensorial, consultar outro agente, entregar o controle da ação a um humano, etc.

4. Implementação do Robô Limpador

O exemplo explorado ao longo deste trabalho foi implementado em Jason⁴ com um agente BDI. A Figura 6 apresenta os planos do agente e a seguir é detalhado seu funcionamento.

Por questões de limitação de espaço, são omitidos os detalhamentos das funções (i) limpar (Agente), (ii) andar (Agente, Passos), (iii) esperar (Agente, Tempo), (iv) atualizarCrenças (Agente),

⁴Jason é um interpretador de AgentSpeak, disponível em <http://jason.sourceforge.net/>.

```

1 //Regra 1
2 sabe_local(robo) :- local(_, X) & X >= 0.6.
3
4 // Meta inicial
5 !limparAmbiente(robo).
6
7 //Plano 1
8 ⊕ +!limparAmbiente(robo): true
9 ⊕     <- !verificarLocal(robo)
10 ⊕      limpar(robo)
11 ⊕      andar(robo, 1).
12
13 //Plano 2
14 ⊕ +!verificarLocal(robo): true
15 ⊕     <- -local(_,_)
16 ⊕         atualizarCrenças(robo).
17
18 //Plano 3
19 ⊕ +local(Comodo, Grau): Grau >= 0.6
20 ⊕     <- selecionarMateriais(robo, Comodo).
21
22 //Plano 4
23 ⊕ +local(_, _): not sabe_local(robo)
24 ⊕     <- esperar(robo, 5)
25 ⊕         !verificarLocal(robo).

```

Figura 6. Base de Crenças e Planos do Robô Limpador

(v)selecionaMateriais(Agente, Comodo). Onde em (i) o Agente verifica se sua posição atual precisa ser limpa, neste caso, procede a limpeza; (ii) o Agente se locomove Passos posições; (iii) o Agente espera Tempo segundos; (iv) o Agente atualiza suas crenças conforme o modelo proposto que retorna o conjunto de crenças no formato local(Comodo, Grau), ou seja, engloba o processo de obter as medições do ambiente, estimá-las como evidências fuzzy e gerar crenças com os graus de certezas obtidos por meio da Equação (2); (v) o Agente seleciona os materiais de limpeza adequados para o Comodo.

A Regra 1 (Figura 6 - linha 2) é verdadeira caso o robô acredite estar em determinado local com grau de certeza maior ou igual a 0,6. O robô possui a meta inicial !limparAmbiente(robo). Para atingir esta meta, executa o Plano 1 (linhas 8 a 11), que ativa a intenção de !verificarLocal(robo) e executa as ações limpar(robo) e andar(robo, 1). A intenção !verificarLocal(robo) ativa o Plano 2 (linhas 14 a 16), que remove as crenças de localização anteriores e atualiza as crenças de acordo com o modelo proposto.

Quando o robô possui a crença local(Comodo, Grau) com grau de certeza de pelo menos 0,6, o Plano 3 (linhas 19 e 20) é executado. O Plano 3 executa a ação selecionarMateriais(Comodo). Quando o robô não possui nenhuma crença com grau de confiança maior que 0,6, é executado o Plano 4 (linhas 23 a 25), que realiza a ação esperar(5) e ativa a intenção de verificar sua localização novamente.

No início da execução, o Robô tem as seguintes percepções geradas pelo modelo: local(C1, 0.3) ; local(C2, 0.2) local(Q1, 0.2) ;

local (Q2, 0.3). Como não possui grau de confiança alto em nenhuma das suas crenças, a ação do robô é aguardar 5 segundos e atualizar suas crenças. Após atualizar suas crenças, as novas percepções do robô são: local(C1, 0.1) ; local(C2, 0.2) local(Q1, 0.6); local (Q2, 0.1). Acreditando estar em Q_1 com 0.6 de certeza, o robô seleciona os materiais para a limpeza de Q_1 , realiza a limpeza e locomove-se para outra posição (Código 1).

```
[robo] doing: esperar(robo, 5)
[robo] doing: atualizarCrencas(robo)
[robo] doing: selecionarMateriais(robo, q1)
[robo] doing: limpar(robo)
[robo] doing: andar(robo, 1)
```

Código 1. Execução do Robô Limpador como Agente BDI

5. Considerações Finais

Com o intuito de obter a consciência situacional de um agente em nível de Percepção, este trabalho apresentou um modelo de fusão de dados. Para tratar a incerteza inerente de dados sensoriais, o modelo emprega inferência *Fuzzy-Bayesiana*. A saída do modelo é um conjunto de crenças com grau de certeza e são propagadas para os níveis de consciência situacional de Compreensão e Projeção, de modo a possibilitar o posterior raciocínio, planejamento e a tomada de decisão do agente. O uso da inferência Bayesiana permite estimar a probabilidade mesmo no caso de não serem observadas todas as evidências do ambiente, assim, se um sensor falhar, as crenças do agente ainda serão formadas.

Contudo, a abordagem *Fuzzy-Bayesiana* não permite a geração de crenças de saída difusas, pois não é possível usar a rede como um regressor. Esta característica, afeta agentes que precisem fazer uso da crença difusa em um controlador. Além disso, a crença difusa poderia ser empregada como entrada para outras redes *Fuzzy-Bayesianas* ou em sistemas *fuzzy*. Faz-se necessário ainda, considerar que o ambiente observado pelo agente possui características variantes no tempo. Esta variação pode ser introduzida no modelo através da incorporação de medições passadas à observação presente. Neste sentido, como trabalhos futuros, pretende-se explorar o uso de inferência de Redes Bayesianas Dinâmicas no modelo proposto. Para exemplificar a necessidade de explorar a continuidade nos dados, usaremos novamente o exemplo do Robô Limpador. Suponha que o robô estime sua localização a cada k segundos e consiga apenas locomover-se um passo durante este intervalo. Considerando que as características de ambos os corredores C_1 e C_2 são similares, poderia ocorrer de o robô deduzir que sua localização em um instante t seja C_1 e no instante $t + 1$, seja C_2 . Devido à disposição dos cômodos no cenário (Figura 2), sabe-se que não é possível movimentar-se de C_1 para C_2 em apenas um passo, assim, a dedução do robô seria inverossímil. Utilizando observações probabilísticas contínuas, o robô teria informações para julgar que, ao estar em C_1 no instante t , a probabilidade de continuar em C_1 ou mesmo de entrar em um dos quartos Q_1 ou Q_2 no instante $t + 1$ é maior do que a probabilidade de ter atingido o corredor C_2 .

Como trabalhos futuros, pretende-se implementar o modelo em um cenário robótico real, no qual seja possível explorar a tomada de decisão do agente e a escalabilidade da abordagem.

Referências

- Brignoli, J. T., Pires, M. M., Nassar, S. M., and Sell, D. (2015). A fuzzy-Bayesian model based on the superposition of states applied to the clinical reasoning support. *IntelliSys 2015 - Proceedings of 2015 SAI Intelligent Systems Conference*, pages 210–219.
- Chen, L., Nugent, C., and Al-Bashrawi, A. (2009). Semantic Data Management for Situation-aware Assistance in Ambient Assisted Living. In *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '09, pages 298–305, New York, NY, USA. ACM.
- Endsley, M. R. (1995). Toward a Theory of Situation Awareness in Dynamic Systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):32–64.
- Golestan, K., Soua, R., Karray, F., and Kamel, M. S. (2016). Situation awareness within the context of connected cars: A comprehensive review and recent trends. *Information Fusion*, 29:68–83.
- Insaurralde, C. C. and Petillot, Y. R. (2015). Capability-oriented robot architecture for maritime autonomy. *Robotics and Autonomous Systems*, 67:87–104.
- Khaleghi, B., Khamis, A., Karray, F. O., and Razavi, S. N. (2013). Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28–44.
- Kokar, M. M., Matheus, C. J., and Baclawski, K. (2009). Ontology-based situation awareness. *Information Fusion*, 10(1):83–98.
- Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.
- Steinberg, A. N. and Bowman, C. L. (2004). Rethinking the JDL Data Fusion Levels. *NSSDF Conference Proceedings*, (c):1–18.
- Viertl, R. (1987). Is it Necessary to Develop a Fuzzy Bayesian Inference ? In Viertl, R., editor, *Probability and Bayesian Statistics*, pages 471–475. Springer US, Boston, MA.
- Viertl, R. (1989). Modeling of Fuzzy Measurements in Reliability Estimation.
- Viertl, R. (1995). Statistics with Fuzzy Data. In Della Riccia, G., Kruse, R., and Viertl, R., editors, *Proceedings of the ISSEK94 Workshop on Mathematical and Statistical Methods in Artificial Intelligence*, pages 33–49, Vienna. Springer Vienna.
- Viertl, R. (2008). Fuzzy Bayesian Inference. *Smps*, pages 10–15.
- Viertl, R. and Hule, H. (1991). On Bayes ' theorem for fuzzy data. *Statistical Papers*, 32:115–122.
- White, F. E. (1988). A model for data fusion. In *Proc. 1st National Symposium on Sensor Fusion*, volume 2, pages 149–158.
- Wickens, C. D. and Hollands, J. G. (2000). *Engineering Psychology and Human Performance*. Prentice Hall, Upper Saddle River, New Jersey, 3 edition.
- Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8(3):338–353.

Análise de Dilemas em Natyasastra: Um Sistema de Resolução Dramática para Autorregulação de Processos de Trocas Sociais em Sistemas Multiagentes

Nelson de Faria Traversi , Renata G. Wotter, Graçaliz P. Dimuro , Diana F. Adamatti

¹Universidade Federal do Rio Grande – FURG
R. Visc. de Paranaguá, 102 - Centro, Rio Grande – RS – Brazil

{20kfurg, gracaliz, dianaada}@gmail.com

Abstract. This paper presents an implementation for two selected social dilemmas of Drama Theory, in the context of the dramatic game of self-regulation of social exchange processes in multi-agent systems, called NATYASAстра. In order to improve social exchanges and the agents' gains while their strategies evolve, the evolutionary model used in Natyasastra has been modified, incorporating two dilemmas defined by Drama Theory. The selected dilemmas were those of induction and cooperation. The main objective is to analyze the influence of these dilemmas on the behavior of agents and the evolution of their strategies of social exchanges, aiming at the regulation of the exchange processes.

Resumo. Este artigo apresenta uma implementação para alguns dilemas sociais selecionados da Teoria do Drama, no contexto do jogo dramático de autorregulação de processos de trocas sociais em sistemas multiagentes, denominado de NATYASAstra. Tendo em vista a melhora das trocas sociais e o ganho dos agentes enquanto suas estratégias evoluem, o modelo de evolução utilizado em Natyasastra foi modificado, incorporando os dilemas definidos pela teoria do drama. Os dilemas selecionados foram o de indução e o de cooperação. O objetivo principal é analisar a influência desses dilemas nos comportamentos dos agentes e na evolução de suas estratégias de trocas sociais, visando a regulação dos processos de trocas.

1. Introdução

Este trabalho situa-se no contexto das áreas de Trocas Sociais de Piaget [Piaget 1995], Teoria dos Jogos [von Neumann and Morgenstern 1944], Teoria do Drama [Howard 1994] e Dilemas Dramáticos [Azar et al. 2014].

A Teoria das Trocas Sociais de Piaget [Piaget 1995] tem sido usada como base para a análise das interações em sistemas multiagente. Essas interações são chamadas trocas de serviços, que são avaliadas pelos agentes que interagem, gerando assim um conceito de valores para as trocas sociais que serão valores qualitativos e subjetivos [Dimuro and Costa 2006].

A Teoria dos Jogos [von Neumann and Morgenstern 1944] define que, normalmente, em um jogo, as preferências e oportunidades dos jogadores são fixas e por não existir comunicação entre os jogadores estes valores não são alterados. Assim, os jogadores tentam prever o resultado de um jogo como jogadores racionais.

A Teoria do Drama [Howard 1994] foi criada como uma extensão da teoria dos jogos, onde as preferências e opções dos personagens (jogadores) podem mudar devido à pressão proveniente das negociações do pré-jogo. Ao contrário da teoria dos jogos, a teoria do drama permite que os personagens comuniquem-se uns com os outros antes do jogo, criando suas próprias estratégias de jogo, como também o resultado esperado, sem a necessidade de prever um resultado. Também é descartada a hipótese de que os jogadores tem ideia do que querem, o que os outros querem, e o que eles e os outros podem fazer sobre isso e que todas as opções e preferências são fixas [Howard 1994].

Os Dilemas Dramáticos [Howard 1994] pertencentes à Teoria do Drama, dizem que quando os personagens se tornam familiares com suas posições e planos em um confronto de interesses, seis dilemas podem ocorrer: o dilema da Cooperação, Confiança, Dissuasão, Indução, Ameaça e Posicionamento. Para cada personagem, uma melhora em potencial significa uma mudança na situação futura, mas essa mudança pode não melhorar a situação do agente.

Baseado inicialmente no Jogo de Autoregulação de Processos de Trocas Sociais (JAPTS) [Macedo et al. 2014], onde os agentes, possuindo diferentes estratégias de trocas sociais, efetuam trocas e a partir da taxa de sucessos evoluem seu vetor de ajuste, posteriormente acrescentando trocas mais justas e equilibradas, tentando aumentar o número de trocas de sucessos. [Wotter 2016] acrescentou sentimentos, emoções e reputação ao JAPTS, obtendo o Natyasastra, um jogo cujo objetivo é a aproximação de simulações de interações sociais ao contexto de realidade trazido pela Teoria do Drama [Howard 1994]. Mas ainda assim, Natyasastra não incorporou todos os conceitos da Teoria do Drama, visando uma melhoria foram incorporados dois dilemas dentre os 6 dilemas existentes para uma melhora na simulação: Dilema de Cooperação e Dilema de Indução [Azar et al. 2014].

Este artigo está estruturado em 4 seções. Na seção 2, é apresentada a base teórica deste trabalho. A seção 3 apresenta os dilemas de Natyasastra e os resultados em três cenários. A seção 4 mostra conclusões e trabalhos adicionais desta pesquisa.

2. Embasamento Teórico

2.1. Trocas Sociais

A Teoria de trocas sociais de Piaget tem sido utilizada como a base de interações em sistemas multiagente, com as trocas de serviços em que agentes participantes geram o conceito de valores nessas interações [Piaget 1995]. Seguindo a definição dada por Piaget: Trocas são qualquer interação entre dois agentes em que dadas as ações de um agente A, acabe prestando serviços ao outro agente B, gerando crédito para A e débito para B, e em uma segunda parte tal crédito de A é cobrado e B presta serviços a A.

2.2. Teoria dos Jogos

Descrito inicialmente por Von Neumann e Oskar Morgenstern em [von Neumann and Morgenstern 1944] a Teoria dos jogos delimita a informação compartilhada em um jogo, com seus participantes conhecendo apenas suas estratégias e desconhecendo a estratégia dos outros. Baseando nas informações disponíveis os agentes tomam suas decisões racionalmente de forma a minimizar as perdas. Dessa forma foi

dado primeiro passo no desenvolvimento da teoria dos jogos envolvendo a construção de uma descrição formal e matemática do jogo. A teoria dos jogos estuda casos em que existam conflitos de interesses, buscando indicar melhores soluções. Em determinadas condições acaba conduzindo ao objetivo desejado.

2.3. JAPTS

Em [Macedo et al. 2014] Luis Felipe Macedo introduziu o Jogo de Autorregulação de Processos de Trocas Sociais - JAPTS, onde agentes possuem diferentes estratégias de trocas sociais e podem evoluir suas estratégias ao longo do tempo, com intuito de criar interações mais equilibradas e justas, visando o aumento de trocas de sucesso. O JAPTS é considerado um jogo de informação incompleta, já que os agentes não possuem informações sobre as estratégias de outros agentes, possuindo estratégias diferentes e considerando tanto os aspectos de curto e longo prazo da interação. Evoluem suas estratégias de forma a maximizar sua taxa de sucesso dada pela função fitness, a qual avalia o ganho individual do agente. Definindo estratégias de trocas sociais iniciais (por exemplo egoísmo e altruísmo), agentes possuem a habilidade de evoluir suas estratégias a partir de um vetor de evolução. Tal vetor possui um número de posições igual ao número de possibilidades de evoluções possíveis contidas no Vetor de Ajuste. Inicialmente cada possível evolução possui uma chance igual de ser escolhida aleatoriamente, mas conforme são passados os ciclos, as chances de cada opção vão variando.

2.4. Teoria do Drama

Criada como uma extensão da Teoria dos jogos por Nigel Howard [von Neumann and Morgenstern 1944], onde as opções e preferências dos jogadores podem mudar por influência das negociações do pré jogo. Ao contrário da teoria dos jogos em que jogadores tentam prever o resultado de um jogo contra jogadores “racionais” a teoria do drama permite que os jogadores conversem entre si antes do jogo, construindo um jogo para si e também o resultado que esperam dele, sem a necessidade de prever o resultado. A Teoria do Drama possui cinco estágios para sua resolução dramática, sendo eles: definição de cena, acúmulo, clímax, resolução e desfecho.

2.4.1. Definição de Cena

Na definição de cena é criado o ambiente fechado E, onde os agentes podem interagir com intuito de decidir suas estratégias. Nenhuma nova informação deve vir de fora da cena, embora agentes possam criar novas informações de forma criativa ou entre si. Caso novas informações de fora da cena forem inseridas, significa que o episódio foi interrompido.

2.4.2. Acúmulo

Nessa Fase ocorre a seleção ou re-seleção de $F = (Q, P)$ classe E. Cada personagem i toma uma posição P_i . Assim uma partição A é criada. Cada elemento de A é um subconjunto não vazio de personagens que tomam a mesma posição. Os personagens de C do novo frame F tomam ou retomam posições P_i e aspirações de um futuro comum onde,

sabendo a posição dos outros personagens é possível formar grupos que compartilham uma posição em comum. Resultando ao final dessa fase um frame $F = (Q, P)$ e um grupo de posições $P = (P_i | i \in C)$.

2.4.3. Climax

Caso todos os personagens tenham a mesma posição, ocorre um equilíbrio e a fase 3 é ignorada seguindo para próxima parte. Entretanto, caso não seja alcançado um equilíbrio, a fase 3 ocorre. Nessa fase, cada grupo instala-se em um ponto fixo qualquer que é comum a todas as políticas do grupo. Nesse momento, ocorre um confronto e os paradoxos geram emoções. Os personagens podem adotar novas políticas ou um novo ponto fixo é escolhido, sem mudar os paradoxos que enfrentam, ou retorna-se para a fase 2, onde novamente suas preferências podem mudar, e assim o conceito associado ao ponto fixo também mudado, de forma a resolver os paradoxos, para finalmente poder avançar para fase 4.

2.4.4. Resolução

Quando o equilíbrio é alcançado tanto na fase 2, quanto com a ação da fase 3, o jogo se move para fase 4. Nesta parte não existe liberação emocional, e simplesmente desacordos escondidos são revelados, muitas vezes mostrando que o equilíbrio não foi realmente alcançado na fase 2.

2.4.5. Desfecho

Finalmente, na fase Desfecho o ambiente fechado é aberto, novas informações de fora do grupo podem ser inseridas. Isso ocorre pela preocupação com um novo plano.

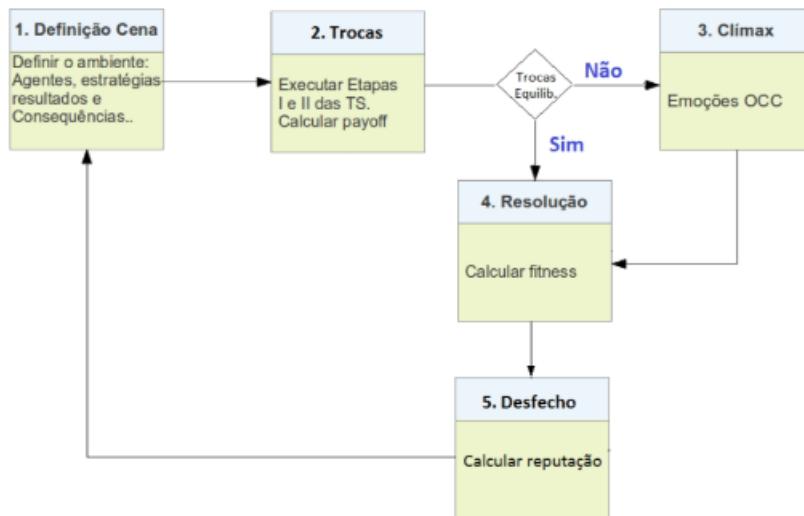
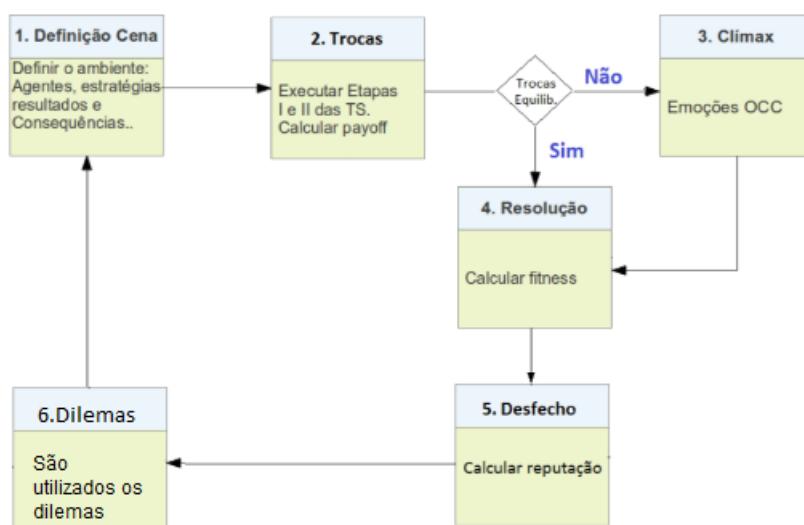
2.5. Natyasastra

Criado por Wotter [Wotter 2016] como uma extensão do JAPTS, adicionando a ele a teoria do Drama. Baseado no modelo de resolução dramática, o modelo do Natyasastra utiliza as 5 fases da resolução dramática da teoria do drama como mostrado na figura 1, utilizando conceitos da teoria do drama aplicados ao JAPTS.

3. Dilemas em Natyasastra

Fazendo uso dos dilemas de cooperação e indução propostos em [Azar et al. 2014], foi feita sua implementação e estudo das comunidades de agentes com diversos grupos. Simulações com mesmos grupos de agentes usadas no Natyasastra [Wotter 2016] foram usados para que uma análise baseada em trabalhos anteriores fosse possível. Os Dilemas ocorrem quando os agentes se tornam familiarizados com suas posições no confronto. Ou seja, no final da fase 5 retornando a fase 1 na resolução dramática, como mostrado na figura 2.

Dentre os seis dilemas existentes: Cooperação, Confiança, Dissuasão, Indução, Ameaça e Posicionamento, tendo em vista os cenários previamente estipulados no tra-

Figure 1. Resolução Dramática em Natyasastra [Wotter et al. 2016].**Figure 2. Resolução Dramática em Natyasastra com Dilemas.**

balho Natyasastra, os dois dilemas a seguir foram escolhidos visando ter um maior impacto.

3.1. Dilema de cooperação

É um dos dilemas mais comuns entre trocas de dois agentes. Tal dilema ocorre quando um jogador A coopera com trocas com agente B sem saber do posicionamento de B. O dilema ocorre porque A está constantemente duvidando da cooperação de B. Na implementação deste dilema, os agentes foram separados em grupos de confiança, forçando grupos mais inseguros a se adaptar à comunidade para prosseguir com suas trocas, enquanto grupos mais confiáveis não necessitam de tantos ajustes.

3.1.1. Dilema de Indução

Ocorre quando membros da comunidade A possuem uma posição e comportamento com desempenho superior aos demais, o que acaba ameaçando membros da sociedade B que se tornam incapazes de competir nas trocas por recursos. Membros do grupo B acabam aceitando atitudes do grupo A e tentam incorporá-las ao seu grupo. Nesta implementação do dilema, tal ocorre com agentes que possuem uma taxa de sucesso inferior aos melhores da sociedade, por possuírem uma taxa de sucesso inferior acabam sendo influenciados pelos melhores do grupo modificando seu vetor de probabilidades apenas, assim acrescentando peso maior às escolhas dos membros do grupo com maior taxa de sucesso.

3.2. Cenários de Análise dos Dilemas

As simulações dos Dilemas em Natyasastra foram realizadas a partir de três cenários distintos utilizados no Natyasastra. Cada cenário utilizou 30 agentes para cada estratégia utilizada. A reputação dos agentes foi setada com 100%, com objetivo de ser fiel as simulações originais do Natyasastra, mantendo as estratégias iniciais e numero de agentes de cada estratégia inicial.

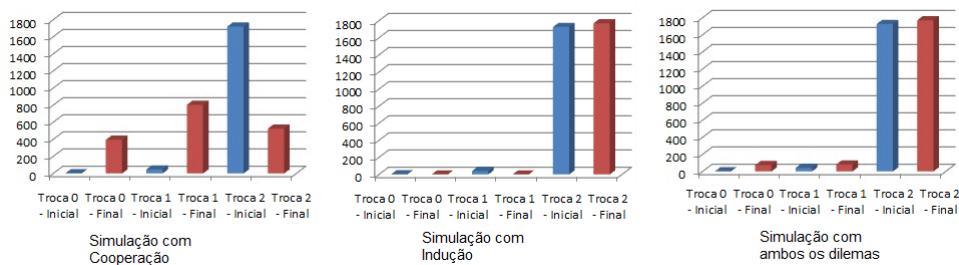
É importante enfatizar que, apesar de ter o mesmo começo, a evolução da sociedade é inicialmente dada de forma aleatória, e à medida que os ciclos continuam, as estratégias começam a tomar forma, no entanto, houve casos em que o início da evolução tomou um curso prejudicial.

Nas simulações, construímos tabelas contendo a média das trocas completas (Trocas 2) quando um agente realizou um serviço e recebeu um serviço em troca, trocas incompletas (Trocas 1) quando apenas um agente ofereceu algo e o outro recusou-se a trocar em troca , e, quando não ocorrem trocas (Trocas 0).

3.2.1. Cenário Altruísta

No cenário altruísta, utilizamos 30 agentes altruístas fortes que têm um alto valor de investimento e esperam um baixo valor em retorno e 30 altruístas fracos que têm um valor de investimento um pouco menor e esperam um valor de retorno um pouco maior que os agentes altruístas, totalizando 60 agentes.

Figure 3. Média das simulações no Cenário Altruísta.



A Figura 3 mostra os resultados das simulações altruísticas, divididas em 3 sessões, cooperação, indução e ambas. Nas simulações desse cenário usando o dilema de cooperação, houve uma perda nas trocas totais no final da simulação. Esse comportamento deve-se ao fato de que os agentes altruístas já estão normalmente dispostos a fazer trocas, no entanto, enfrentando o dilema para cooperar, acabaram dificultando sua cooperação.

Com o dilema de indução, houve um ganho nas trocas totais, superando a simulação sem os dilemas, o que se deve ao fato de que, com tantas trocas bem-sucedidas sendo realizadas por tantos agentes dispostos a fazer trocas, acaba gerando um número de agentes de grande influência, que levam a sociedade a um caminho com mais trocas.

No entanto, a maior surpresa ocorreu quando ambos os dilemas foram usados. Neste jogo, ao contrário do que era esperado, o dilema da cooperação, juntamente com o dilema da indução, aumentou consideravelmente o número de trocas, atingindo 90% dos casos, completando 100% das trocas.

3.2.2. Cenário Egoísta

No cenário egoísta, usamos 30 agentes egoístas fortes com baixo valor de investimento e alto valor de retorno esperado e 30 agentes egoístas fracos com um valor de investimento um pouco maior e esperando um valor de retorno um pouco menor do que os fortes agentes egoístas.

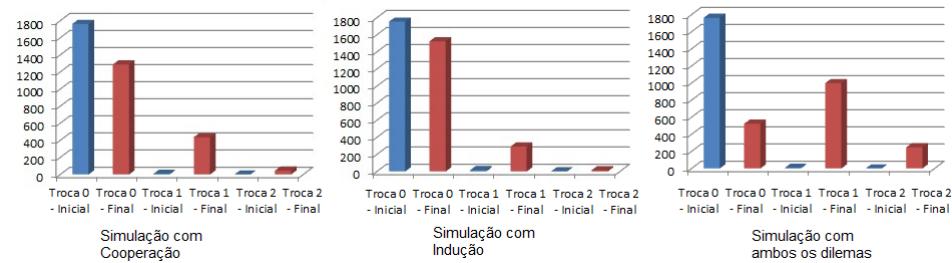
Na figura 4 é exposto os resultados das simulações egoísticas, divididas em 3 sessões, cooperação, indução e ambas.

Nas simulações que utilizam o dilema da cooperação, é possível notar que o número de trocas completas é muitas vezes superior ao jogo sem o uso de dilemas, esse comportamento ocorre porque vários agentes entram no grupo de reforço para melhorar suas trocas.

Nas simulações que utilizam os dilemas de indução, o número de trocas totais ainda era maior do que o jogo sem o uso de dilemas, mas não era mais eficiente do que o dilema de cooperação. Esse resultado decorre do fato de que não há agentes modelo ideais para influenciar a população.

Com o uso de ambos os dilemas, foram obtidos várias trocas completas, 5 vezes maiores do que o número obtido usando o dilema de cooperação, esse comportamento

Figure 4. Média das simulações no Cenário Egoísta.



deve-se ao fato de que o dilema de cooperação permite que um agente exemplar surgisse para o grupo , e esse agente acaba influenciando toda a população através do dilema da indução.

3.2.3. Cenário com todas estratégias

Neste último cenário, as cinco estratégias estão presentes, 30 agentes de cada um totalizando 150 agentes, com o objetivo de mostrar um ambiente heterogêneo, contendo estratégias que promovem trocas como os altruístas e altruístas fracos, estratégias que inibem tais trocas como egoísta e egoísta fraca, e agentes racionais que têm o valor da oferta igual ao valor esperado.

Figura 5 mostra os resultados das simulações com todas as estratégias, divididas em 3 sessões, cooperação, indução e ambas.

No cenário com a influência do dilema da cooperação, o número de trocas completas diminuiu em média 1024 trocas, enquanto o número de trocas incompletas aumentou em 2157 trocas, como mostra a figura 5 o dilema da cooperação não é tão eficaz em estratégias que beneficiam as trocas.

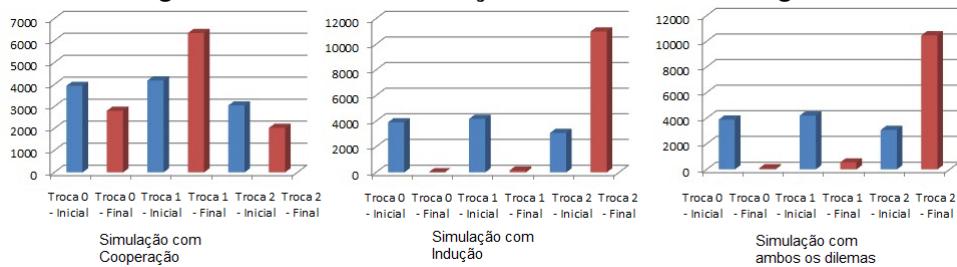
No cenário com a influência do dilema de indução, obteve-se um ótimo ganho nas trocas completas, que em 80 % das simulações atingiram 100 % de bolsas completas como mostrado na figura 5, esse comportamento é devido à existência de agentes com boas condições físicas e evoluções que servem de exemplo para toda a sociedade.

No último cenário, usando os dilemas de cooperação e indução, o ganho no comércio atingiu 100% de trocas completas em 70 % das simulações em seu ciclo final. O dilema da indução não foi tão eficaz em ambientes que não possuem agentes que servem como bons modelos como o segundo cenário. Em simulações que têm estratégias que inibem as trocas, o dilema da cooperação conseguiu obter uma melhora nas trocas.

Quanto pior o ambiente para as trocas, melhor será a ação do dilema da cooperação, melhorando as trocas e a aptidão nesses casos, no entanto, quanto mais favorável a cena para a troca, menos o dilema da cooperação é efetivo.

4. Conclusões e Trabalhos Futuros

Este trabalho teve como objetivo estudar como a cooperação e a indução influenciaram o cenário das trocas sociais entre os agentes, com base na teoria do drama complementando o drama da Natyasastra, além de obter a análise do impacto gerado pelos dilemas

Figure 5. Média das simulações com Todas Estratégias.

escolhidos nas simulações.

Com base na análise das simulações apresentadas neste trabalho, é possível deduzir que o dilema da cooperação tem um efeito negativo nas sociedades propensas a fazer trocas, o que representa um obstáculo adicional para a tomada de decisões, no entanto, tem um efeito benéfico em sociedades egoísticas que já sofrem com a dificuldade de fazer trocas e acabam usando o dilema da cooperação para melhorar suas estratégias.

O dilema da indução, por outro lado, tem um efeito positivo em sociedades propensas a trocas, onde há vários exemplos a serem seguidos e onde todos estão tentando melhorar seu já elevado número de trocas, mas em sociedades egoísticas, esse dilema teve pouco efeito, um dos fatores para isso é a falta de um modelo de agente, bem como a falta de dedicação para iniciar novas trocas na sociedade egoísta, sempre esperando receber muito sem colaborar com a sociedade.

A verdadeira surpresa ocorreu quando os dois dilemas interagiram na mesma sociedade, um dilema complementando as falhas do outro acabou aumentando o número de trocas completas em todos os cenários. Como trabalho futuro, será realizada a implementação dos restantes dilemas, possibilitando um estudo mais detalhado sobre seus impactos e relações nas sociedades.

References

- Azar, A., Khosravani, F., and Jalali, R. (2014). Drama theory: A problem structuring method in soft or (a practical application: Nuclear negotiations analysis between islamic republic of iran and the 5+ 1 group). *The International Journal of Humanities*, 19(4):1–14.
- Dimuro, G. P. and Costa, A. C. R. (2006). Interval-based Markov decision processes for regulating interactions between two agents in multi-agent systems. In Dongarra, J., Madsen, K., and Wasniewski, J., editors, *Selected Papers of the 7th International Conference on Applied Parallel Computing, PARA'04, Lyngby*, number 3732 in LNCS, pages 102–111, Berlin. Springer.
- Howard, N. (1994). Drama theory and its relation to game theory. part 1: Dramatic resolution vs. rational solution. *Group Decision and Negotiation*. Kluwer Academic Publishers., pages 187–206.
- Macedo, L. F. K., Dimuro, G. P., Aguiar, M. S., and Coelho, H. (2014). An evolutionary spatial game-based approach for the self-regulation of social exchanges in MAS. In Schaub, T., Friedrich, G., and O’Sullivan, B., editors, *ECAI 2014 – 21st European*

- Conference on Artificial Intelligence, Proceedings*, number 263 in Frontier in Artificial Intelligence and Applications, pages 573–578, Netherlands. IOS Press.
- Piaget, J. (1995). *Sociological Studies*. Routledge, London.
- von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Wiley, New York.
- Wotter, R. G. (2016). Natyasastra: Um jogo dramático de autorregulação de processos de trocas sociais baseado na teoria do drama. Master's thesis, Programa de Pós-graduação em Computação da Universidade Federal do Rio Grande.
- Wotter, R. G., Adamatti, D. F., and Dimuro, G. P. (2016). Self-regulation of social exchange processes: A model based in drama theory. In Bajo, J., Escalona, M. J., Giroux, S., Hoffa-Dkabrowska, P., Julián, V., Novais, P., Sánchez-Pi, N., Unland, R., and Azambuja-Silveira, R., editors, *Highlights of Practical Applications of Scalable Multi-Agent Systems. The PAAMS Collection: International Workshops of PAAMS 2016, Sevilla, Spain, June 1-3, 2016. Proceedings*, pages 161–172. Springer International Publishing, Cham.

Segregação Racial e Valorização do Território: uma modelagem baseada em agentes

Gustavo L. Lima¹, Nelson F. Traversi¹, Diana F. Adamatti¹

¹Programa de Pós-Graduação em Computação

Universidade Federal do Rio Grande (FURG)

{gustavolameirao,20kfurg,dianaada}@gmail.com

Abstract. To understand how residential segregation of ethnic groups happens, Thomas Crombie Schelling has created a model to show how people take into account how many individuals similar are around. With the advancement of technology, a model was created in the NetLogo tool based on the Schelling model. This contributed to new simulations being made. The objective of this study is to add a new behavior in the agents, improving the model. Individuals will take into account, in addition to the neighborhood, the valuation of the position where they will stay. The results obtained in the simulations show a similarity when compared to racial distribution maps of large cities.

Resumo. Para entender como acontece a segregação residencial de grupos étnicos, Thomas Crombie Schelling criou um modelo para mostrar como as pessoas levam em consideração quantos indivíduos similares estão a volta. Com o avanço da tecnologia, foi criado um modelo na ferramenta NetLogo se baseando no modelo de Schelling. Isto contribuiu para que novas simulações fossem feitas. O objetivo deste estudo é acrescentar um novo comportamento nos agentes, aprimorando o modelo. Os indivíduos levarão em conta, além da vizinhança, a valorização da posição onde irão ficar. Os resultados obtidos nas simulações apresenta uma similaridade quando comparado com mapas de distribuição racial de grandes cidades.

1. Introdução

Thomas Crombie Schelling foi um economista que ganhou um prêmio Nobel em economia em 2005. Um dos seus trabalhos (Schelling 1981) buscou entender como acontecia a segregação. O estudo de Schelling ajudou no estudo da segregação residencial de grupos étnicos, ou seja, na forma com que as pessoas escolhem suas moradias nas cidades. Como exemplos reais desse estudo, pode-se verificar cidades como Nova York e Chicago, nos EUA. A Figura 1 apresenta distribuição racial em Nova York em 2010 e a Figura 2 mostra Chicago em 2000. Pode-se analisar que em Nova York toda a cidade é bem valorizada, podendo ser vista a clusterização dos grupos de cada etnia. Já em Chicago (Figura 2), é possível notar que em locais com grande valor agregado, como no centro dessa cidade, existe uma grande diversidade de etnias. É possível observar que no centro da cidade existe um comportamento heterogêneo das etnias. Conforme se afasta do centro, começa a ocorrer a segregação étnica, separando em bairros cada uma das etnias.

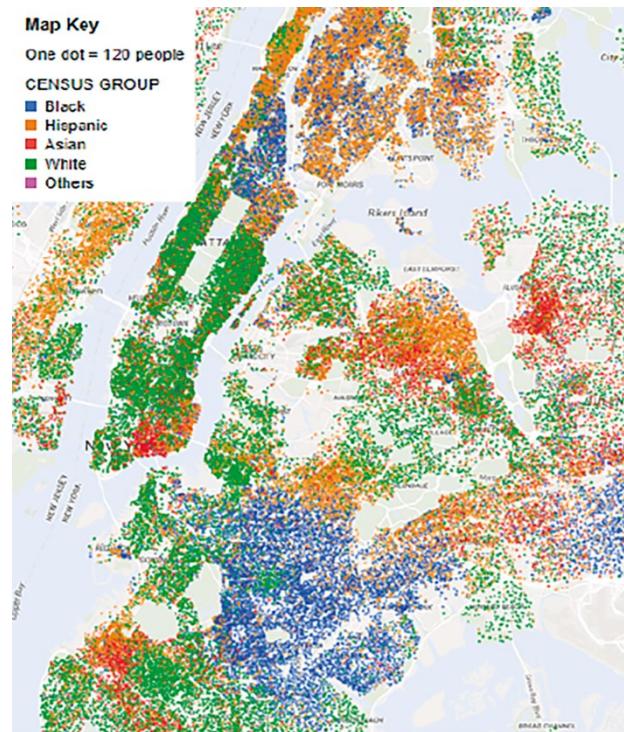


Figura 1. Mapa da segregação racial em Nova York [The New York Times, 2015]

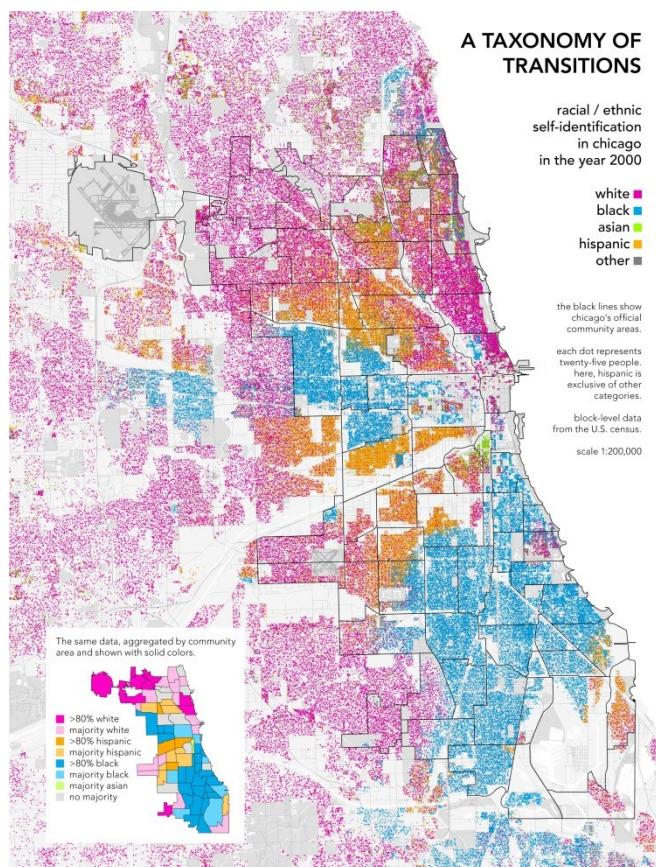


Figura 2. Mapa da segregação racial em Chicago [Rakin 2009]

Utilizando o avanço do poder computacional, Wilensky (Wilensky, 1997) criou um modelo similar ao proposto por Schelling na ferramenta NetLogo. Neste modelo, é possível realizar grandes simulações utilizando agentes que simulam as pessoas escolhendo onde irão morar. Este modelo mostra como os incentivos e as tendências de indivíduos se agruparem com indivíduos semelhantes leva a segregação.

No modelo original de Wilensky, agentes ocupam células de um espaço retangular. Os agentes são distribuídos pelo cenário, ocupando as posições aleatoriamente. Cada célula pode ser ocupada por apenas um agente. Além disso, cada célula possui 8 outras células ao seu redor (formando uma vizinhança). Os agentes pertencem a um dentre dois grupos. Cada agente verifica os agentes ao seu redor para constatar se a relação de amigos (proporção entre os agentes do mesmo grupo e o total de agentes a sua volta) é maior ou igual ao valor de aceitação. Se essa condição for satisfeita, o agente permanece no mesmo local. Caso contrário, o agente move para as células vazias em busca de um local onde a condição seja satisfeita.

Este estudo propõe adicionar outro comportamento que podem influenciar as pessoas na hora de escolher onde irão morar. Além da vizinhança, os agentes irão levar em conta possíveis valorizações nos terrenos (posições). Estas valorizações podem ocorrer em pontos específicos, ou pode ocorrer a valorização de um ponto central e, por consequência, uma valorização das residências ao seu redor.

Foi adicionado um valor de aceitação da localidade do agente. Este valor determina qual a qualidade mínima para que o agente seja feliz em determinada localidade. Os agentes levam este valor relacionado a sua posição, além dos vizinhos similares do modelo original, na hora de decidir onde irão ficar. É possível observar que, utilizando o valor de qualidade mínima baixo, a clusterização se aproxima do modelo original, da mesma forma que, ao utilizar um valor de qualidade mínima alto, os agentes consideram mais importante a localização do que a vizinhança. Vale ressaltar que o valor de qualidade mínima é igual para todos os agentes.

Este artigo está dividido em seções. A seção 2 apresenta o modelo original do NetLogo, que foi utilizado como base para o desenvolvimento do estudo proposto. As modificações feitas na extensão do modelo são apresentadas na seção 3. A seção 4 mostra os resultados obtidos nas simulações, em conjunto com sua análise. Por fim, a seção 5 trata das conclusões obtidas no estudo e dos trabalhos futuros.

2. Segregação - Modelo Base do NetLogo

O modelo criado Wilensky foi implementado no ambiente multiagente NetLogo. A interface da simulação é mostrada na Figura 3.

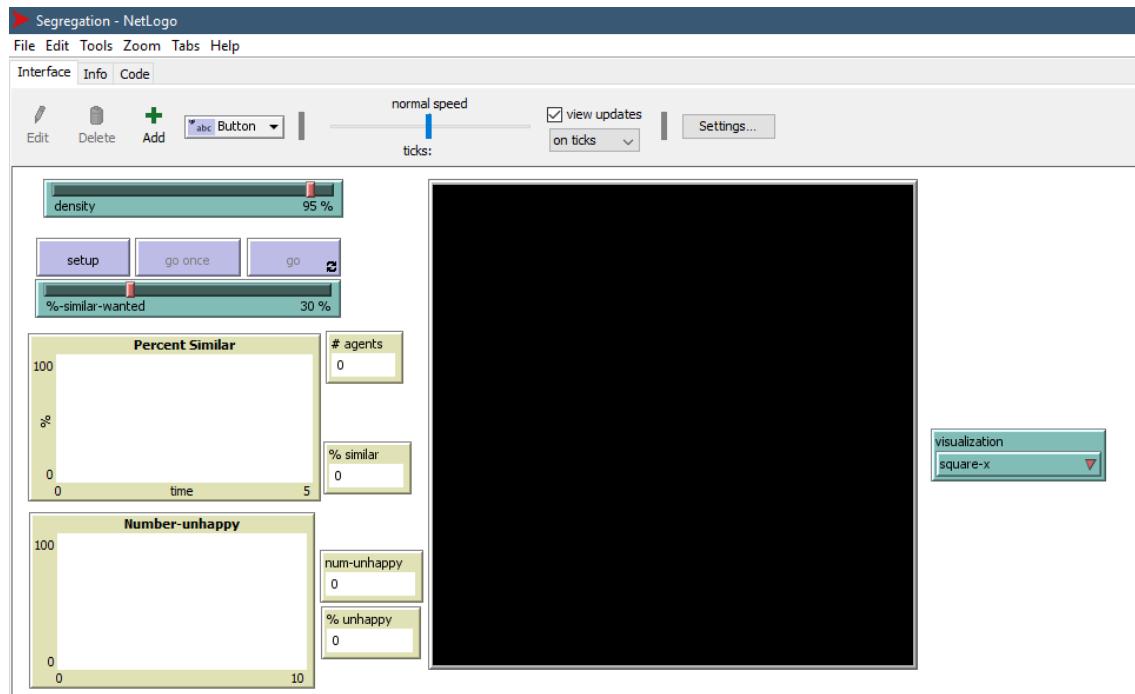


Figura 3. Interface do modelo no NetLogo

Neste modelo, existe um grupo de agentes que pertencem ao grupo verde ou vermelho. A felicidade de cada agente é determinada pela quantidade de agentes similares (do mesmo grupo) em sua volta. Se essa relação for maior ou igual a porcentagem de similaridade desejada (determinada pelo slider *%-similar-wanted*), o agente fica feliz. Caso contrário, o agente fica infeliz. Os agentes infelizes irão se mover para as células desocupadas até atingir o estado de felicidade. Se todos os agentes atingirem a felicidade, a simulação é finalizada.

Na simulação, é possível escolher a densidade (quantidade de agentes que serão utilizados) através do slider *density*, a porcentagem de similaridade desejada (*%-similar-wanted*). Como resultado, além da simulação através dos quadrados, também são mostrados 2 gráficos, um que trata da porcentagem média de vizinhos com a mesma cor para cada agente e outro que trata do número de agentes infelizes. Por fim, existem monitores para o número total de agentes, a porcentagem de similares, o número e a porcentagem de agentes infelizes.

Através das simulações, é possível constatar que quanto maior o nível de similaridade desejado, maior a segregação produzida. Para verificar tal constatação, foram realizadas três simulações, utilizando density 90% e %-similar-wanted em 25, 50 e 75 %. Os resultados são apresentados nas Figura 4 (1), (2) e (3), respectivamente.

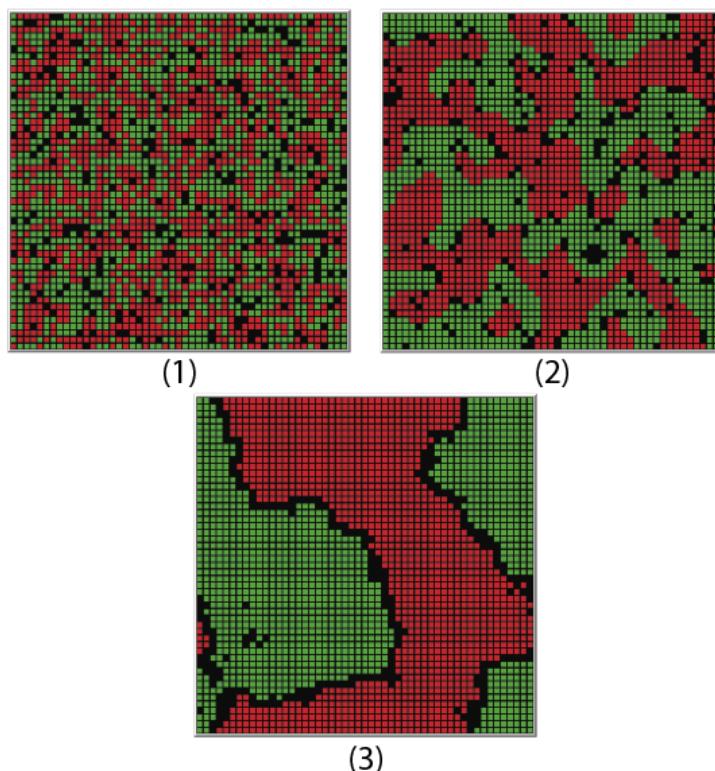


Figura 4. Resultados obtidos no modelo original do NetLogo

Também foi possível observar que se utilizar um grau de densidade muito alto e uma similaridade também alta, fica difícil de encontrar um fim para a execução da simulação, pois a exigência dos agentes fica muito alta, sendo necessário que se desloquem muito para encontrar um ponto que atenda sua exigência. Porém, como a densidade é muito alta, não existe espaço para os agentes se moverem. Se for utilizado valores similares aos anteriores, ao usar densidade 90%, não existe convergência da simulação para valores de similaridade acima de 80%.

3. Extensão do Modelo

No modelo desenvolvido por Wilensky, o principal fator determinante da escolha de onde o indivíduo irá residir é a vizinhança, ou seja, a quantidade de agentes similares ao seu redor. A extensão proposta visa introduzir um novo comportamento dos agentes. Além da sua vizinhança, os agentes também levarão em conta a localização e valorização de onde irão ficar. Esta proposta se baseia no fato de que os centros das grandes cidades são heterogêneos, ou seja, existem pessoas de diversas culturas residindo próximas, por conta da posição destas residências. Um exemplo é o bairro Liberdade na cidade de São Paulo, que é predominante habitado por imigrantes asiáticos. Entretanto, ao se afastar do centro, bairros com pessoas com alguma similaridade começam a ser formados.

Além do centro da cidade, existem outros pontos que podem ter sua localização valorizada, como casas próximas a grandes *shoppings*, aeroportos, dentre outros. Por esse motivo, os pontos de valorização de cada cidade são diferentes.

Ainda existem pontos específicos que são valorizados, como determinada residência que tem um alto valor, tratando de um comportamento de apenas um ponto, e não uma região.

O modelo proposto busca simular o comportamento dos indivíduos na escolha de onde irão viver, levando em consideração a vizinhança (modelo de Schelling) mas também levando em conta a valorização da posição, ou seja, um indivíduo pode acabar optando por morar em um lugar onde há poucos vizinhos similares a ele, se este lugar for bem valorizado.

Em um primeiro momento, o centro do mapa representa o centro de uma cidade, que é o ponto mais desejado e portanto é por onde uma cidade deve começar a ser formada.

Visto que mesmo tendo um valor mais elevado ao centro, algumas sociedades podem valorizar outros pontos da área. No modelo é possível criar outros pontos de valorização no mapa, onde o ponto central é escolhido e automaticamente os pontos a volta também são valorizados. O grau de valorização decai conforme as posições vão ficando mais afastadas do ponto escolhido. Além disso, também é possível escolher um valor aleatório para cada terreno, que pode variar entre zero a outro valor escolhido, possibilitando assim a criação de cenários mais complexos para o estudo.

Como o modelo se trata de uma extensão do modelo da biblioteca do NetLogo de Wilensky (que se baseia no modelo de Schelling), existe uma série de similaridades na interface e no funcionamento. Todos os controles apresentados na interface do modelo do NetLogo estão presentes neste modelo (controle de densidade, porcentagem de similaridade desejada, os gráficos, etc), mas a diferença está no controle das variáveis de valorização dos terrenos.

Um dos itens adicionados a interface é o *good-position*. Esta variável se comporta de maneira similar a similaridade desejada: quanto maior o valor, os agentes serão mais exigentes na escolha posição, da mesma forma que quanto menor o valor, menos exigentes eles serão. Outra diferença desta variável ao controle de similaridade é que este valor decai conforme o tempo passa (mais precisamente conforme o número de ciclos passa). A intensidade deste decaimento é controlada pelo *slider decay*. O gráfico *good-position* apresenta o decaimento da variável *good-position* conforme o tempo.

Outro comando adicionado a interface é o *switch prioritize-location*. Este switch determina o peso dado entre a valorização da posição e os vizinhos na escolha de onde o agente irá ficar. Se este *switch* estiver desativado, os agentes irão dar o mesmo valor para as duas variáveis. Ao ativá-lo, os agentes passarão a dar mais valor para a localização do ponto do que para a quantidade de vizinhos similares.

O *slider position-starting-value* trata da valorização inicial de cada terreno. Este valor é aleatório entre zero até a escolha. Isto serve para que cada posição tenha um valor inicial diferente.

O botão *area-value-increase* serve para criar um ponto de valorização. Quando o usuário clica neste botão e depois em um ponto do *grid*, será criado um círculo de valorização. O centro do círculo terá uma super valorização (determinada pelo *slider area-value-amount*) e os valores a sua volta também serão valorizados. A proporção da valorização decai conforme os pontos vão se distanciando do centro do círculo. O

tamanho do círculo é determinado pelo *slider area-value-increase-area*. A nova interface gráfica do simulador é apresentada na Figura 5.

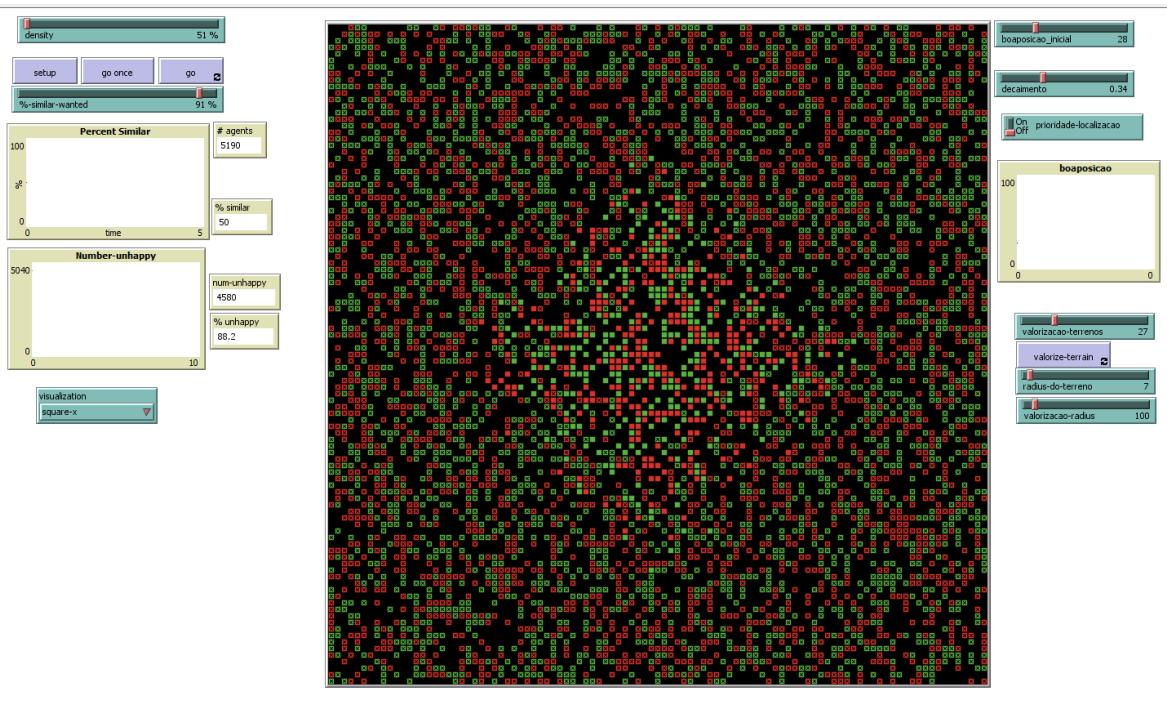


Figura 5. Interface do modelo desenvolvido

4. Análise dos Resultados

De forma a validar a extensão proposta, um conjunto de simulações foi realizado. Em um primeiro momento, foram criadas simulações onde os parâmetros originais foram mantidos iguais (para a simulação em questão foi utilizado densidade 90%, similaridade 90%), além dos novos parâmetros *good-starting-position* (valor 100) e decaimento (0.20). As combinações de alterações das demais variáveis inseridas ao modelo foram testadas e o resultado das segregações obtidas foram analisadas. O primeiro conjunto de testes é formado por cinco cenários:

1. Apenas o centro do grid valorizado;
2. Centro do grid valorizado e valorizações de pontos específicos (gerados aleatoriamente);
3. Centro do grid valorizado e várias zonas de valorização (valor inferior ao centro);
4. Centro do grid valorizado, várias zonas de valorização (raio pequeno e valorização bem inferior ao centro) e valorizações de pontos específicos (gerados aleatoriamente);
5. Centro do grid valorizado, várias zonas de valorização (raio grande e valorização levemente inferior ao centro) e valorizações de pontos específicos (gerados aleatoriamente).

A segregação formada por cada um dos casos é mostrada na Figura 6 e pode-se concluir em cada um dos cenários que:

1. Esta simulação possui apenas a valorização do centro. É possível ver que toda comunidade se concentrou no centro do mapa, formando uma cidade bem clusterizada.
2. Nessa simulação são utilizados valores aleatórios nos imóveis, em conjunto com o valor central da cidade. O comportamento dos agentes se fixando em certos lugares antes da “civilização” alcançá-los é observado.
3. Este caso se baseia no fato de que o centro da cidade é o mais desejável. Entretanto, existem quatro pontos nos extremos do *grid*, com valorização equivalente a metade do valor central. Nessa simulação, o centro ainda assim foi a escolha predominante, porém foram formados quatro novos bairros isolados com cores predominantes em cada um deles.
4. No quarto cenário, além do grande valor central e dos pontos nos extremos vistos nas simulações anteriores, foi adicionado um ruído, na forma de valores adicionais em cada uma das posições. Isto permitiu que certos agentes começassem a atingir o estado de felicidade mesmo sem possuir vizinhos similares a ele, ou seja, apenas pela qualidade do lugar. Tal simulação demonstra que quando existe um imóvel com alta valorização este pode ser escolhido, mesmo que seja em um bairro pouco desejado.
5. Nesta última simulação, são atribuídos valores específicos inferiores ao centro, além de valores aleatórios a pontos específicos do mapa. Este caso é basicamente a junção de todas as variáveis criadas. É possível notar que a clusterização se mantém padronizada, tomando apenas o formato que engloba o centro do mapa e seus quatro pontos de supervalorização.

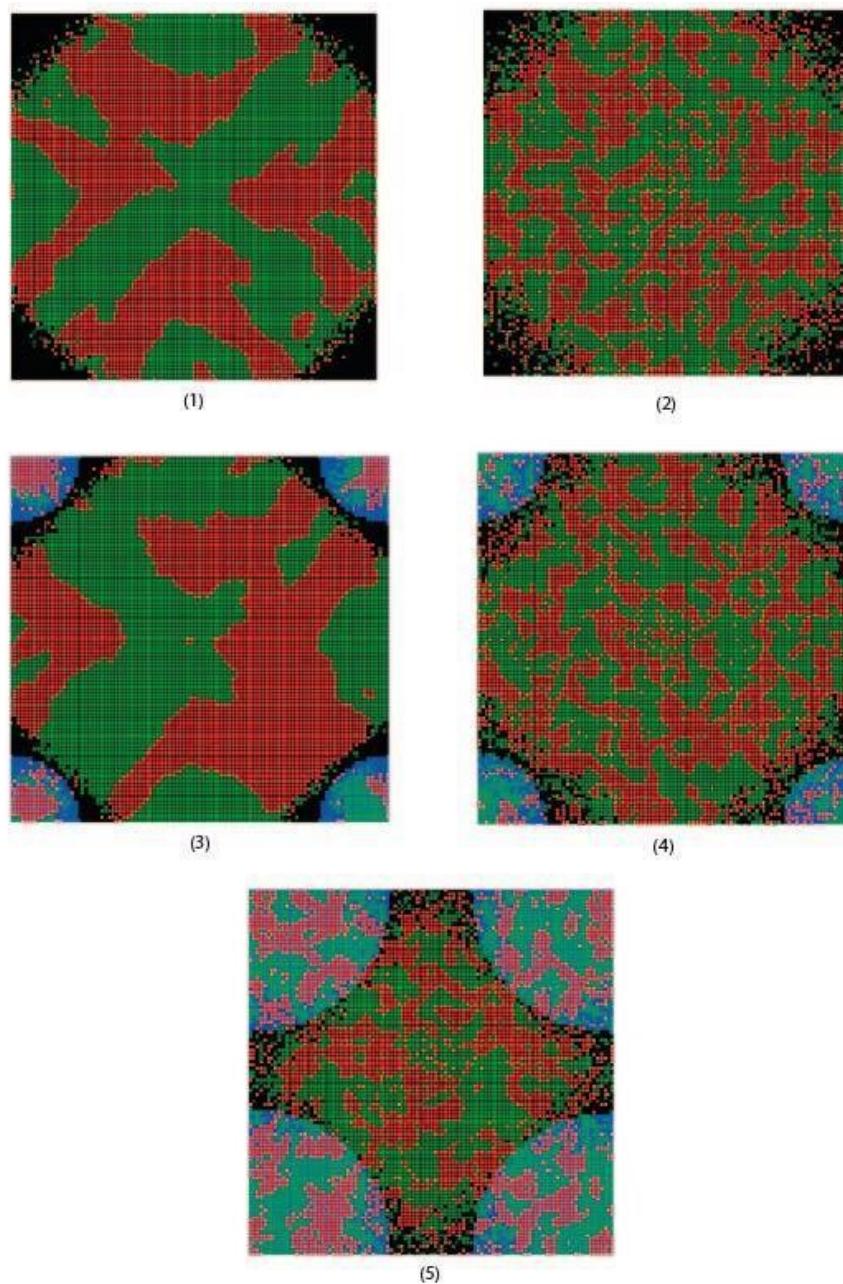


Figura 6. Resultados obtidos no primeiro conjunto de testes

O segundo conjunto de testes foi realizado para averiguar se é possível direcionar o crescimento de uma cidade para determinado ponto através do uso das variáveis criadas.

A Figura 7 apresenta o crescimento progressivo da segregação ao longo da simulação é mostrado, que utiliza apenas o centro da cidade e um círculo de valorização. Para essa simulação, foram utilizados os seguintes valores para as variáveis: densidade 50%, similaridade buscada 90%, *area-value-amount* 80%, *position-starting-value* 25 e *area-value-increase-area* 95 *area-value-amount*. É possível observar que a clusterização, nesta simulação, ocorre do centro do mapa em direção ao centro do círculo.

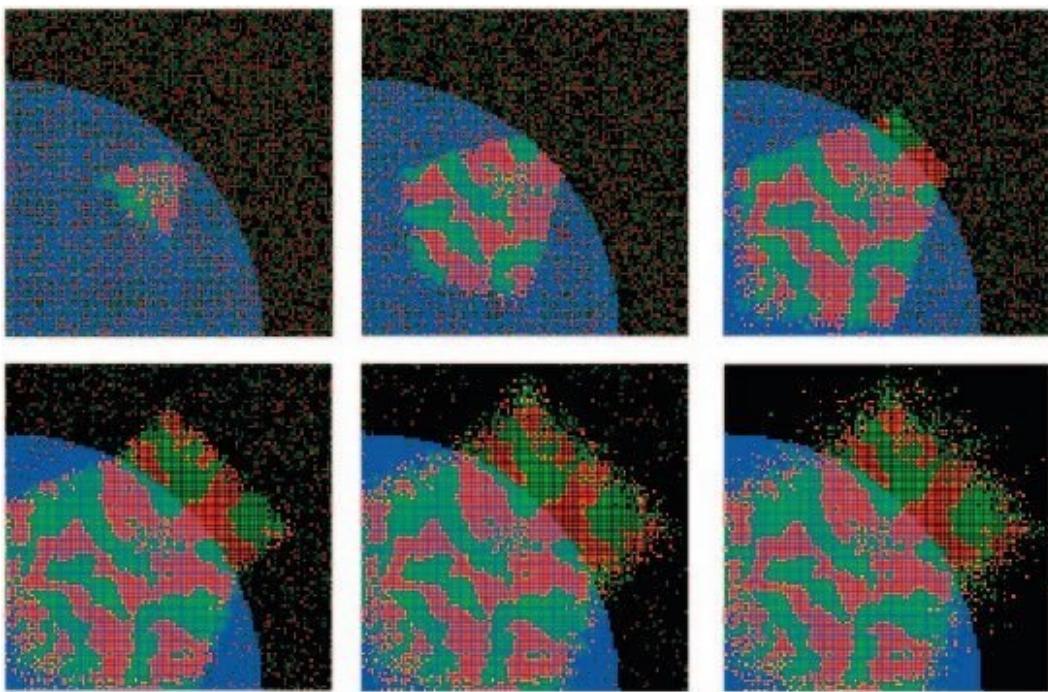


Figura 7. Resultados obtidos no segundo conjunto de testes

Por fim, foi criado um conjunto de teste, similar ao anterior, porém utilizando a valorização do centro do círculo maior do que o centro do mapa. Neste caso, os pontos do círculo passam a se tornar tão bons para os agentes que qualquer residência naquele raio é considerada boa. Assim, o centro do círculo passa a se tornar heterogêneo. Porém, é interessante observar que mesmo o centro sendo heterogêneo, a formação dos bairros (clusterização) começa a se formar conforme os pontos vão se distanciando do centro do círculo. O resultado desta simulação é apresentado na Figura 8.

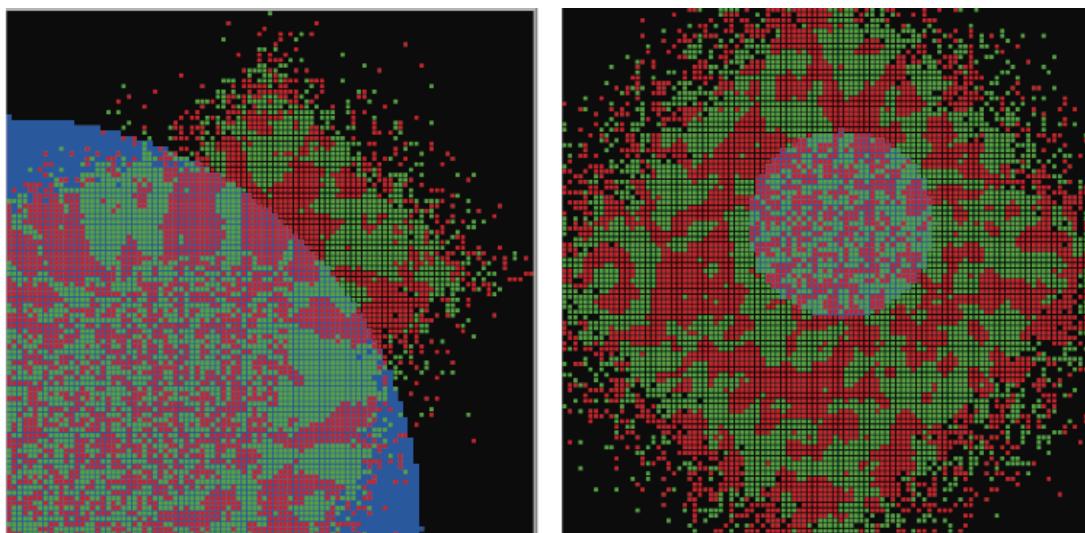


Figura 8. Resultados obtidos no terceiro conjunto de testes

5. Conclusões e Trabalhos Futuros

O modelo de segregação social de Schelling, ajudou no estudo da segregação residencial de grupos étnicos, ou seja, na forma com que as pessoas escolhem suas moradias nas cidades. Um exemplo real pode ser visto na Figura 1.

Este estudo buscou introduzir um novo comportamento que poderia influenciar na escolha do local de moradia das pessoas/agentes no modelo proposto por Wilensky, baseado no modelo de Schelling. O principal ponto sugerido é que, além das pessoas similares a volta, a valorização da localidade analisada pela pessoa poderia também influenciar em sua escolha.

Para verificar a influência de cada variável inserida no novo modelo, foram realizadas simulações, apresentando e comparando seus resultados. Após analisar as simulações e comparar com mapas de etnias de grandes cidades, é possível notar que o modelo final com localidades super valorizadas acabou se comportando como as cidades reais, em especial a cidade de Chicago, mostrada na Figura 2. Esta cidade tem seu centro com grande diversidade e periferias mais segregadas, mostrando que a localidade e valor atribuído a imóveis é uma grande variável na organização das sociedades atuais. O modelo também mostrou como é possível direcionar o crescimento da segregação das cidades, através do uso de áreas valorizadas por algum ponto central (grandes *shoppings*, aeroportos, dentre outros).

A criação deste modelo no NetLogo fez uso da computação para que fosse possível existir um modelo de fácil compreendimento e utilização, aproveitando do poder computacional existente para que seja possível criar inúmeros tipos de simulações com base no modelo.

Para trabalhos futuros, pretende-se incluir áreas de desvalorização do território ao modelo. Estas áreas serão representadas através de um círculo vermelho no mapa, visando permitir criar áreas de difícil acesso ou impróprias para serem habitadas (como morros, lagos, rios e bairros mais perigosos).

Além disso, pretende-se incluir também áreas que são desejadas apenas por um determinado grupo de agentes e é ignorado por outros.

Por fim, como o modelo original trata apenas de dois grupos (modelo binário), outra proposta de expansão do modelo é através da criação de novos grupos, de maneira similar ao que acontece com as diversas raças apresentadas nos infográficos das cidades de NY e Chicago, anteriormente citadas.

Referências

- Rakin, Bill (2009). “Chicago Boundaries”. Disponível em: <<http://www.radicalcartography.net/index.html?chicagodots>>. Acesso em: jan. 2018.
- Schelling, Thomas C. (1981). “The Strategy of Conflict.” Harvard University Press. 309p.
- The New York Times (2015). “Mapping Segregation”. Disponível em: <<https://www.nytimes.com/interactive/2015/07/08/us/census-race-map.html>>. Acesso em: jan. 2018.

- Wilensky, U. (1997). NetLogo Segregation model. <http://ccl.northwestern.edu/netlogo/models/Segregation>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Wilensky, U. (1999). NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Aplicando árvore de decisão para a localização de padrões no modelo Sugarscape do NetLogo

Vágner de Oliveira Gabriel¹, Cleo Zanella Billa¹, Diana Francisca Adamatti¹,
Fabiana Lorenzi²

¹Centro de Ciências Computacionais - Universidade Federal do Rio Grande - FURG
Av. Itália, Km 8 - Campus Carreiros – 96203-900 – Rio Grande, RS – Brasil

²Universidade Luterana do Brasil (ULBRA)
Av Farroupilha, 8001 Bairro São José – 92420280 – Canoas, RS – Brasil

vdeoliveiragabriel@gmail.com, {cleobilla, dianaadamatti}@furg.br
fabilorenzi@gmail.com

Abstract. Sugarscape is a model that simulates populations in an environment with limited resources, where they must move around seeking resources for their survival. NetLogo software has an implemented model of Sugarscape, which simulates populations of agents who need to collect sugar to stay alive in the environment. This work aims to apply the decision tree technique on data collected in simulations carried out in the Sugarscape model of NetLogo. Thus, the purpose is to find patterns that define which attributes are important in determining whether an agent will live or die in the simulation environment.

Resumo. O Sugarscape é um modelo que simula populações em um ambiente com recursos limitados, onde as mesmas devem se locomover buscando recursos para a sua sobrevivência. O software NetLogo possui um modelo implementado do Sugarscape, o qual simula populações de agentes que necessitam coletar açúcar para se manterem vivos no ambiente. Este trabalho tem como objetivo, aplicar a técnica de árvore de decisão sobre dados coletados em simulações realizadas no modelo Sugarscape do NetLogo. Dessa forma, o propósito é localizar padrões que definam quais atributos são importantes para determinar se um agente irá viver ou morrer no ambiente de simulação.

1. Introdução

O Sugarscape é um modelo amplamente utilizado em simulações baseadas em agentes. Este modelo apresenta como finalidade a simulação de sociedades que necessitam de recursos, os quais estão distribuídos em regiões de um ambiente em que a sociedade está localizada [Epstein and Axtell 1996]. Portanto, é de extrema relevância coletar estes recursos disponíveis no ambiente para que seja possível a sobrevivência dos agentes.

O software NetLogo possui implementado o Sugarscape¹, onde os agentes possuem a necessidade de coletar açúcar para sobreviverem no ambiente em que estão situados. O agente é modelado no sugarscape com as seguintes características: açúcar individual que o deixa vivo, metabolismo que consome o seu açúcar individual, visão de pontos onde estão distribuídos os açúcares e quantidade de açúcar existente no local.

¹<http://ccl.northwestern.edu/netlogo/models/Sugarscape1ImmediateGrowback>

A mineração de dados surge como uma ferramenta para localizar padrões significativos que não podem ser localizados apenas ao consultar as informações em um banco de dados [Tan et al. 2006]. Na mineração de dados existem distintas técnicas para a localização de padrões em base de dados com um número elevado de informações. Uma das técnicas mais conhecidas para classificação se chama árvore de decisão, a qual será utilizada neste trabalho.

Este trabalho tem como objetivo realizar uma mineração de dados utilizando o software Weka aplicando árvore de decisão sobre dados gerados no modelo de simulação de recursos emergentes sugarscape. Com a aplicação da árvore de decisão sobre os dados que serão coletados no modelo Sugarscape, se torna possível encontrar um padrão nas simulações e identificar os atributos mais determinantes para a sobrevivência do agente no ambiente.

Este trabalho está organizado da seguinte forma. Na seção 2 é apresentada a fundamentação teórica do trabalho, onde são conceituados os principais componentes estudados para a realização deste trabalho. Na seção 3, é descrita a metodologia utilizada para o desenvolvimento do trabalho. Na seção 4 é apresentado o resultado obtido no trabalho. Na seção 5, são realizadas as considerações finais e na seção 6 são apresentadas as referências estudadas para o desenvolvimento deste trabalho.

2. Fundamentação Teórica

Nesta seção, serão apresentados os componentes que serviram como base teórica para o desenvolvimento deste trabalho. Esta seção está dividida em duas subseções, onde a primeira subseção aborda uma breve introdução sobre agentes, sistemas multiagente e o modelo Sugarscape do NetLogo. A segunda subseção é utilizada para a definição de árvore de decisão.

2.1. Sistemas Multiagente

Agentes podem ser definidos como entidades reais ou virtuais, as quais estão situadas em um ambiente, onde se comunicam e interagem com outros agentes visando atingir seus objetivos [Rezende 2003]. Os agentes podem trabalhar isoladamente ou formar sociedades, resultando assim um sistema multiagente. Segundo [Wooldridge and Jennings 1994], agentes inteligentes possuem as seguintes características:

- Autonomia: os agentes possuem controle total de suas ações e estado interno, uma vez que, trabalham sem ligação direta com seres humanos;
- Habilidade sociais: se comunicam com outros agentes através do uso de um agente de comunicação de linguagem;
- Reatividade: Percebem alguma alteração no ambiente e agem sobre a mesma em um curto espaço de tempo;
- Pró-atividade: o agente pode raciocinar e tomar uma iniciativa.

2.1.1. Modelo Sugarscape

O modelo Sugarscape foi apresentado por [Epstein and Axtell 1996], onde é utilizado amplamente na comunidade de simulações baseadas em agentes. O Sugarscape simula

populações em um ambiente com recursos limitados, no caso do modelo introduzido por Epstein e Axtell, os agentes necessitam de açúcar para a sua sobrevivência.

No ambiente do Sugarscape (Figura 1) existem regiões com um nível elevado de açúcar e outras com um nível baixo, ou até mesmo sem açúcar. Os agentes se locomovem no ambiente e são capazes de coletar açúcar devido as suas propriedades como visão e metabolismo, sendo a visão utilizada para enxergar os lugares que possuem açúcar e o metabolismo para consumir o açúcar coletado. A visão e o metabolismo variam em cada agente, tornando dessa forma a população da simulação heterogênea. Conceituando o modelo sugarscape de um forma geral, os agentes dependem do açúcar para a sua sobrevivência, porém o metabolismo pode consumir todo o estoque de açúcar individual levando o agente a morte. Dessa forma, se torna necessário que o agente esteja posicionado em uma região com níveis altos de açúcar.

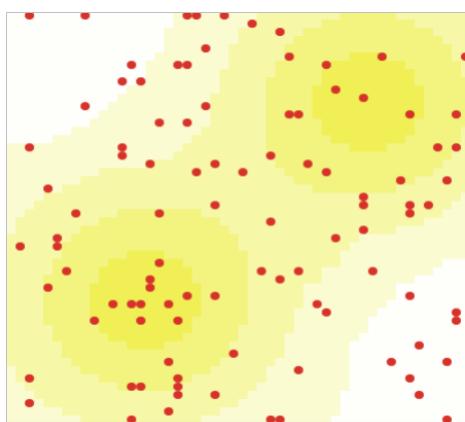


Figura 1. Modelo Sugarscape (NetLogo)

2.2. Árvore de Decisão

A mineração de dados pode ser definida como o processo de descoberta automática de informações relevantes ao usuário em grande depósito de dados [Tan et al. 2006]. Em mineração de dados são introduzidos os conceitos de funcionalidade, sub-funcionalidade, tarefas e o conceito de técnicas. Tarefas são as especificações do que o usuário está pesquisando, tipo de regularidades e padrões, como exemplo: a compra elevada de um produto em um supermercado apenas no mês de abril. Já técnicas, são caracterizadas na especificação dos métodos que garantem a descoberta de padrões que o indivíduo está buscando. As principais técnicas em mineração de dados são: estatísticas, técnicas de aprendizado de máquina e técnicas baseadas em crescimento-podavalidação [De Amo 2004].

As funcionalidades, em mineração de dados, definem como os tipos de padrões ou relacionamentos que interligam os registos e suas variáveis podem ser utilizados na mineração. Como exemplos de funcionalidades pode-se citar a análise descritiva e a análise de prognóstico. Este trabalho utiliza apenas uma sub-funcionalidade da análise de prognóstico, a sub-funcionalidade de classificação, focando apenas na sua definição.

[De Amo 2004] define classificação como um processo para encontrar um conjunto de modelos que definem e distinguem classes ou conceitos visando predizer as classes de objetos que ainda não foram classificadas. Árvores de decisão e redes neurais são técnicas usualmente utilizadas em tarefas de classificação.

A árvore de decisão pode ser caracterizada como um fluxograma que possui similaridade com uma árvore. Este fluxograma se constitui de nós internos, os quais representam testes em atributos e cada folha representa a distribuição dos registo [da Costa Côrtes et al. 2002]. Em [Monard and Baranauskas 2003], é apresentado um exemplo que demonstra uma árvore de decisão para diagnosticar um paciente.

No exemplo (Figura 2), as elipses servem como um teste em um atributo para um conjunto de dados de pacientes. Os retângulos representam classes, os quais referenciam os diagnósticos.

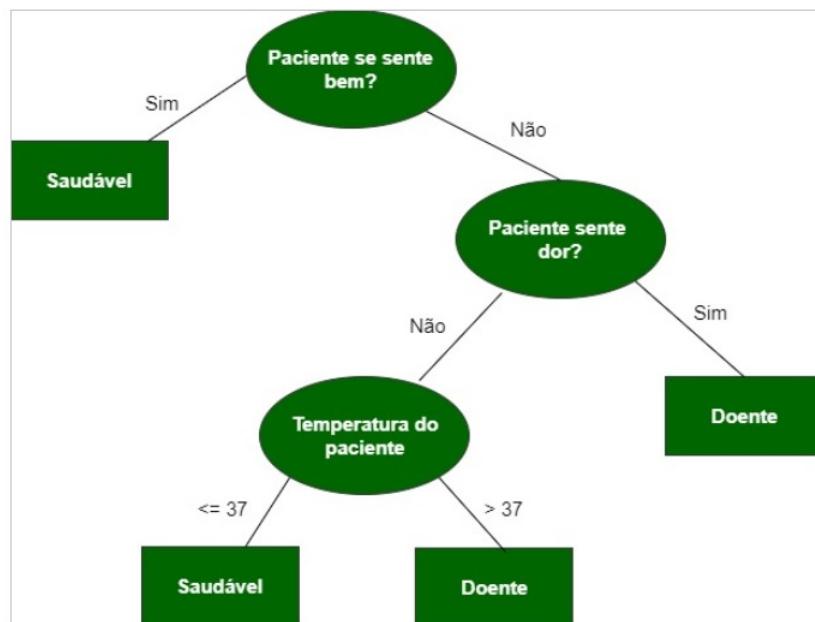


Figura 2. Árvore de Decisão de diagnóstico de um paciente (Adaptado de Monard e Baranauskas (2003))

No exemplo apresentado por Monard e Baranauskas (2003) (Figura 2), existem duas classes que são a classificação do paciente (Saudável/Doente). Existem também os conjuntos de dados que vão classificar se o paciente está saudável ou doente. Este conjuntos são:

- Paciente se sente bem?
- Paciente sente dor?
- Paciente sente dor?

3. Materiais

Nesta subseção serão conceituados os softwares utilizados para o desenvolvimento desse trabalho.

3.1. NetLogo

A ferramenta NetLogo² (Figura 3), é um ambiente de modelagem programável onde é possível simular fenômenos naturais e sociais. Esta ferramenta se adequa bem para modelar sistemas que apresentam uma complexidade maior em seu desenvolvimento, sendo

²<https://ccl.northwestern.edu/netlogo/>

composta ainda por uma linguagem de programação simples voltada para a modelagem e para simulação de fenômenos naturais e sociais [Wilensky 1999]. O NetLogo se adapta bem para a modelagem de sistemas que apresentam uma complexidade elevada e que evoluem ao longo do tempo. A ferramenta possibilita aos desenvolvedores passar instruções a centenas ou milhares de ‘agentes’, que operam de forma independente, interagindo entre si ou com o ambiente.

O NetLogo possui uma extensa documentação, trazendo alguns modelos em sua biblioteca, sendo uma coleção de simulações pré-escritas que podem ser usadas e modificadas. NetLogo em um geral é uma ferramenta poderosa e simples o suficiente para possibilitar que estudantes e professores simulem suas pesquisas abrangendo diversos campos de conhecimento.

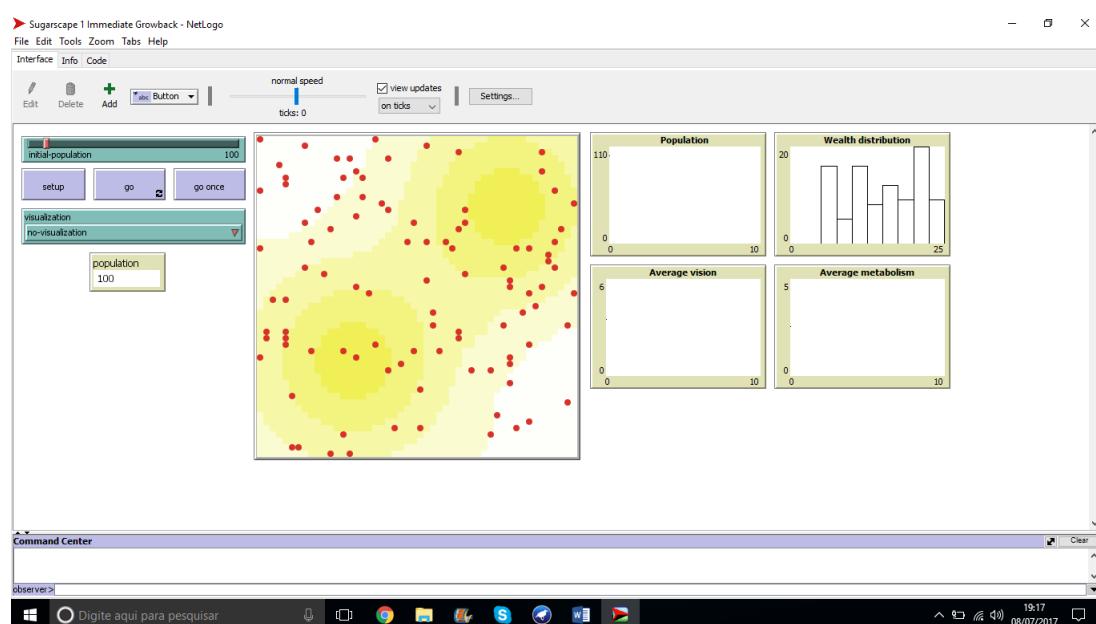


Figura 3. Ferramenta NetLogo

3.2. Weka

O software Weka³ (Figura 4), possui uma coleção de algoritmos de aprendizagem de máquinas, estes usados para a mineração de dados buscando localizar padrões entre outras características em uma gigantesca base de dados. O Weka foi desenvolvido na Universidade de Waikato na Nova Zelândia e fornece métodos para a mineração de dados, como classificação, regressão, agrupamento, regras de associação e visualização [Hall et al. 2009].

Neste trabalho foi adotada a utilização do software Weka pelo fato desta ferramenta ser de fácil manuseio e possibilitar a utilização de árvore de decisão para a classificação dos dados através do uso do algoritmo J48.

4. Metodologia

Na primeira etapa do trabalho foi utilizado o software NetLogo para realizar as simulações do modelo Sugarscape, o qual já está instanciado como um modelo de exemplo do soft-

³<https://www.cs.waikato.ac.nz/ml/weka/>

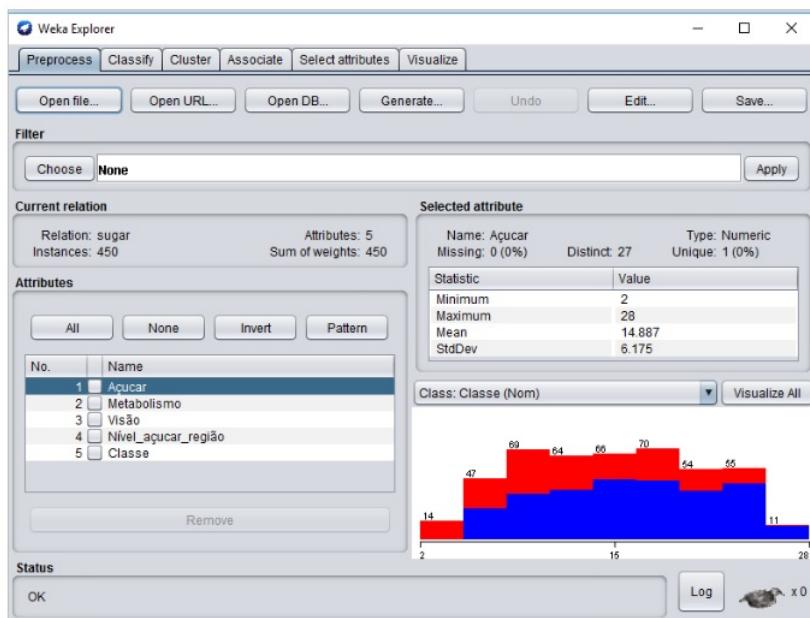


Figura 4. Software Weka

ware. Algumas alterações foram realizadas no modelo incluso no NetLogo, objetivando facilitar a coleta dos dados das simulações. A Figura 5 ilustra o código implementado para salvar em um arquivo txt os seguintes dados: id do agente, seu nível de açúcar, seu nível de metabolismo, seu nível de visão e a quantidade de açúcar existente na região onde o agente está posicionado.

```

to-report crialista
  set lista []
  set lista lput who lista
  set lista lput sugar lista
  set lista lput metabolism lista
  set lista lput vision lista
  set lista lput max-psugar lista
  report lista
end

to escrever [lista2]
  file-open "dados.txt"
  file-print reduce [ ( word ?1 "," ?2 ) ] lista2;;
  show reduce [ ( word ?1 "," ?2 ) ] lista2;;
  file-flush
  file-close
end

```

Figura 5. Código implementado no NetLogo

A primeira parte do código da Figura 5 foi desenvolvida com a finalidade de criar uma lista (vetor) e salvar as seguintes informações:

- Who: o *Id* (identificador) do agente;
- Sugar: nível de açúcar individual do agente;
- Metabolismo: nível de metabolismo que o agente possui;

- Vision: nível de visão do agente;
- Max-psugar: nível de açúcar existente na região onde o agente está posicionado;

O segundo trecho do código apresentado na Figura 5 foi desenvolvido com a finalidade de salvar em um arquivo txt, este chamado de “dados”, os arquivos salvos na lista.

Foram realizadas 20 simulações contendo 50 agentes no ambiente, onde foram coletados inicialmente os dados descritos na Figura 5 dos 50 agentes no ambiente, estes dados foram gerados de forma aleatória pelo software.

Das 20 simulações, a média de ticks (medida de tempo no software NetLogo) foi de 100. Após a finalização de cada simulação foram coletadas as informações de quais agentes sobrevivem no decorrer da simulação e colocadas na linha correspondente a cada agente classificando ele como “vive” ao final da simulação ou “morre”. Ao final da primeira etapa foi possível coletar uma base de dados contendo 1.000 dados. A Figura 6 ilustra os dados salvos no arquivo txt.

```
7,3,5,3,vive
19,4,1,0,morre
22,4,3,4,vive
18,4,6,4,vive
11,3,1,2,morre
20,1,2,1,vive
16,2,1,2,vive
10,1,4,0,morre
13,1,5,3,vive
23,2,2,2,vive
```

Figura 6. Dados após finalização de cada simulação

A segunda etapa consiste em preparar o arquivo de dados para um padrão que o software Weka consiga reconhecer. Dessa forma, foi necessário alterar o arquivo dados.txt para a extensão arff, esta reconhecida pelo software Weka. Para a leitura do arquivo foi necessária a criação de um cabeçalho, este ilustrado na Figura 7.

No cabeçalho tem a relação chamada sugar, e os atributos correspondentes a cada coluna do arquivo dados. A primeira coluna corresponde ao açúcar do agente, a segunda corresponde ao metabolismo do agente, a terceira corresponde a visão do agente, a quarta corresponde ao nível de açúcar do agente. Estas quatro primeiras colunas são do tipo numérico, uma vez que, os dados inclusos nelas são números. A coluna cinco é a classe do agente que representa a definição se ele vive ou morre ao final na simulação.

Após a adaptação dos dados coletados para um tipo de arquivo, o qual tornou-se reconhecido pelo software Weka, foi possível gerar a árvore decisão, a qual é apresentada na seção resultados.

```

@relation sugar
@attribute Açucar numeric
@attribute Metabolismo numeric
@attribute Visão numeric
@attribute Nível_açucar_região numeric
@attribute Classe {vive,morre}
@data
7,3,5,3,vive
19,4,1,0,morre
22,4,3,4,vive
18,4,6,4,vive

```

Figura 7. Alteração do arquivo para extensão arff

5. Resultados

Aplicou-se a técnica de árvore de decisão sobre os dados coletados como saída nas simulações realizadas no modelo *Sugarscape*. Dessa forma, localizando os seguintes Padrões:

- Se o nível de açúcar da região for ≤ 3 , o metabolismo for ≤ 3 , o nível de açúcar da região for ≤ 1 , o metabolismo ≤ 1 , o nível de açúcar da região ≤ 0 , o agente morre.
- Se o nível de açúcar da região for ≤ 3 , o metabolismo for ≤ 3 , o nível de açúcar da região for ≤ 1 , o metabolismo ≤ 1 , o nível de açúcar da região > 0 , o agente vive.
- Se o nível de açúcar da região for ≤ 3 , o metabolismo for ≤ 3 , o nível de açúcar da região for ≤ 1 , o metabolismo > 1 , o agente morre.
- Se o nível de açúcar da região for ≤ 3 , o metabolismo for ≤ 3 , o nível de açúcar da região for ≥ 1 , o metabolismo ≤ 2 , o agente vive.
- Se o nível de açúcar da região for ≤ 3 , o metabolismo for ≤ 3 , o nível de açúcar da região for > 1 , o metabolismo > 2 , nível de açúcar da região for ≤ 2 , a visão for ≤ 5 , o agente morre.
- Se o nível de açúcar da região for ≤ 3 , o metabolismo for ≤ 3 , o nível de açúcar da região for > 1 , o metabolismo > 2 , nível de açúcar da região for ≤ 2 , a visão for > 5 , o agente vive.
- Se o nível de açúcar da região for ≤ 3 , o metabolismo for ≤ 3 , o nível de açúcar da região for > 1 , o metabolismo > 2 , nível de açúcar da região for > 2 , o agente vive.
- Se o nível de açúcar da região for ≤ 3 , o metabolismo for > 3 , visão ≤ 5 , o agente morre.

- Se o nível de açúcar da região for ≤ 3 , o metabolismo for > 3 , visão > 5 , o nível de açúcar da região for ≤ 1 , o agente morre.
- Se o nível de açúcar da região for ≤ 3 , o metabolismo for > 3 , visão > 5 , o nível de açúcar da região for > 1 , o agente vive.
- Se o nível de açúcar da região for > 3 , o agente vive.

A Figura 8, ilustra o resultado obtido em formato de árvore, o qual foi gerado pela ferramenta Weka tendo como entrada os resultados coletados das simulações no NetLogo sobre o modelo *Sugarscape*.

A árvore (Figura 9), mostra que o atributo mais determinante para a sobrevivência de um agente é região onde ele está posicionado. Se a região possuir mais que três de nível ele vive, caso o contrário sua sobrevivência depende de um conjunto de outras situações.

```
J48 pruned tree
-----
Nivel_açucar_região <= 3
|   Metabolismo <= 3
|   |   Nivel_açucar_região <= 1
|   |   |   Metabolismo <= 1
|   |   |   |   Nivel_açucar_região <= 0: morre (7.0)
|   |   |   |   Nivel_açucar_região > 0: vive (14.0)
|   |   |   Metabolismo > 1: morre (32.0)
|   |   Nivel_açucar_região > 1
|   |   |   Metabolismo <= 2: vive (141.0/6.0)
|   |   |   Metabolismo > 2
|   |   |   |   Nivel_açucar_região <= 2
|   |   |   |   |   Visão <= 5: morre (40.0/4.0)
|   |   |   |   |   Visão > 5: vive (7.0/2.0)
|   |   |   |   Nivel_açucar_região > 2: vive (39.0/1.0)
|   |   Metabolismo > 3
|   |   |   Visão <= 5: morre (82.0)
|   |   |   Visão > 5
|   |   |   |   Nivel_açucar_região <= 1: morre (4.0)
|   |   |   |   Nivel_açucar_região > 1: vive (7.0/2.0)
Nivel_açucar_região > 3: vive (77.0)

Number of Leaves :      11
Size of the tree :      21
```

Figura 8. Árvore de decisão contendo os padrões localizados sobre os resultados das simulações do modelo Sugarscape

6. Considerações Finais

Com a aplicação da árvore de decisão se tornou possível localizar os atributos que são mais relevantes para a sobrevivência do agente no modelo de *Sugarscape* disponível no software NetLogo, além de apresentar um conjunto de situações que podem determinar a sobrevivência ou morte de um agente no ambiente.

A aplicação de árvores de decisão pode ser uma grande ferramenta para a localização de padrões gerados em simulações. Atualmente, existem diversos modelos de simulação baseados em agentes que geram um grande número de resultados. A utilização de mineração de dados para analisar e encontrar padrões nestes resultados pode ser de grande valia.

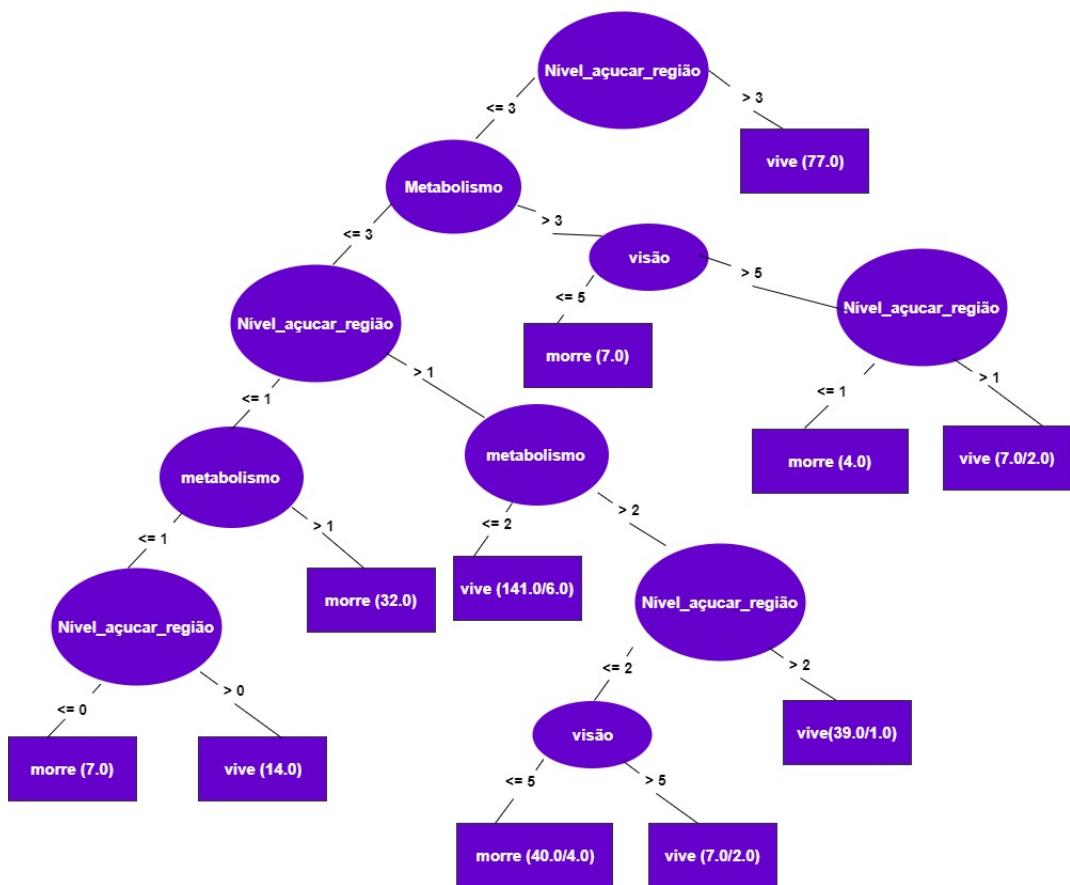


Figura 9. Fluxograma em formato de árvore contendo os padrões localizados sobre os resultados das simulações do modelo Sugarscape

Futuramente, se espera aplicar outras técnicas de mineração de dados sobre o modelo *Sugarscape*, visando testar sua eficiência e comparar os novos achados com os resultados obtidos neste trabalho.

Referências

- da Costa Côrtes, S., Porcaro, R. M., and Lifschitz, S. (2002). *Mineração de dados-funcionalidades, técnicas e abordagens*. PUC.
- De Amo, S. (2004). Técnicas de mineração de dados. *Jornada de Atualização em Informática*.
- Epstein, J. M. and Axtell, R. (1996). *Growing artificial societies: social science from the bottom up*. Brookings Institution Press.
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.
- Monard, M. C. and Baranauskas, J. A. (2003). Indução de regras e árvores de decisão. *Sistemas Inteligentes. Rezende, SO Editora Manole Ltda*, pages 115–140.
- Rezende, S. O. (2003). *Sistemas inteligentes: fundamentos e aplicações*. Editora Manole Ltda.

- Tan, P.-N. et al. (2006). *Introduction to data mining*. Pearson Education India.
- Wilensky, U. (1999). Netlogo.
- Wooldridge, M. and Jennings, N. R. (1994). Agent theories, architectures, and languages: a survey. In *Intelligent agents*, pages 1–39. Springer.

Using agent-based artificial financial market to analyse market manipulation

Luiza P. Biasoto¹, Everton R. Reis¹, Jaime S. Sichman¹

¹Laboratório de Técnicas Inteligentes (LTI)
Escola Politécnica da Universidade de São Paulo (EPUSP)
Av. Prof. Luciano Gualberto, 158 trav. 3 – 05508-010 – São Paulo – SP – Brazil
lpbiasoto@gmail.com, everton.rreis@gmail.com, jaime.sichman@usp.br

Abstract. This work aims to evaluate price manipulation provided by investors with great amount of capital and its overall effect in the stock market. In order to do so, we have created an artificial financial market using NetLogo. The experiments were carried out in a closed environment, with technical analysis speculators and other three different groups of agents, each one with a unique investment strategy. This work provides inputs for the creation of an artificial financial market, in which other diverse agent strategies could be added, and evidences of a market manipulation caused by excess demand.

Resumo. Este trabalho visa estudar a manipulação de preços por grandes investidores e seu efeito geral no mercado de ações.. Para isto, criou-se um mercado financeiro artificial utilizando NetLogo. Foram efetuados experimentos em um ambiente fechado, com especuladores que utilizam análise técnica e outros três diferentes perfis de agentes com estratégias de investimento únicas. Este trabalho provê inssumos para a criação de um mercado financeiro artificial, no qual poderiam ser adicionadas outras diversas estratégias para seus agentes, e evidências de uma manipulação de mercado causada por excesso de demanda.

1. Introduction

Financial markets are highly volatile. They consist of a heterogeneous environment, in which each investor has distinct interests and investing profiles. Furthermore, markets are strongly affected by unexpected news. Since it represents a complex environment, financial market modeling becomes extremely attractive for a multi-agent approach.

Financial markets are defined as markets where investors can buy and sell securities (stocks, bonds), commodities (agricultural products, precious metals) and currencies (dollar, euro...). They are divided in different types, such as [CVM, 2013]:

- Capital markets;
 - Stock markets, which consist of financing common stocks; a single share of stock represents fractional ownership of a corporation in proportion to the total number of shares; stock owners receive dividends equivalent to organization's profit;

- Bond markets, which consist of financing private and public bonds, such as Debentures or Public Treasure Titles; its securities are known to have a fixed income and refer to a debt issued by its participants;
- Commodity, derivatives, future, foreign exchange and other markets.

Our work is focused in stock markets, where trades are intermediated by a centralized exchange authority. These markets are increasingly being conducted by electronic exchanges.

Electronic trading has promoted an important increase in high-frequency trading and in using complex algorithms that attempt to predict the market behavior. It enhanced speculative trading and market manipulation as well [Angel and Mccabe, 2013]. Spoofing and layering, which creates artificial demand conditions, insider trading and other abusive actions are reduced by market regulation. An organization called CVM (Comissão de Valores Mobiliários) was created in 1976 in order to supervise, regulate, discipline and develop the Brazilian securities market. The responsibility to supervise and prevent fraudulent and abusive transactions, though, belongs to the exchangers, brokers and other organizations involved.

Financial markets are political and economic thermometers for a nation. When they are down, it strongly indicates that the country's economy is not performing as it should , and the opposite indicates that its performance is considered by the market as positive. Similarly, a poor political administration may reflect in a financial crisis. In a coordinated move, investors could create a false perception of economic and/or administrative issues within companies and the nation itself.

In this work, our objective is to analyse how investors with great amount of capital could influence asset pricing. We created an agent-based artificial financial market using NetLogo to this goal. We believe that it is possible to manipulate the market only by excess demand, and having more capital facilitates this intention. We designed a strategy used by “big” investors, i.e., institutional investors that can strongly influence the asset price and manipulate the market. Three other groups of investors were designed with distinct strategies. We observed the performance of the strategy used by big investors alongside the other investors’ strategies.

In the next section, we present the market environment, its transaction mechanism and some other important concepts discussed throughout this work. In section 3, we show some literature concerning the discussion of essential features used to build the artificial market. Section 4 presents the simulation framework, detailing the four different groups of investors, other system parameters, algorithms and the logic behind the functioning of the artificial financial market. Finally, the simulation results are presented and discussed in Section 5, and we describe our conclusions and future work in Section 6.

2. Stock Exchanges

Trading in stock markets can basically occur following two types of trading systems, which are executed mainly electronically: *Agency Markets*, such as New York Stock Exchange (NYSE), Toronto Stock Exchange or BOVESPA (Brazil), or *Dealer Markets (Over-The-Counter)*, such as BM&F (Brazil), where the trading occurs via a dealer

network [Vishwanath and Krishnamurti, 2009]. In 2008, BM&F merged with BOVESPA creating BM&FBOVESPA and in 2017 with CETIP (entity responsible mainly for the Brazilian private bonds) creating the B3 organization, which now coordinates most of the Brazilian financial market.

The stock markets' transaction mechanism of a single asset works as follows: (i) the investor sends a market or limit order to buy a certain quantity of a stock to an electronic platform that belongs to a brokerage firm; (ii) the broker then sends the order to the centralized exchange that have all the orders organized in an *order book*; (iii) the exchange intermediates the transaction through a *clearing* procedure, finding another investor which is willing to sell the stock at the same price, if it is a limit order, or execute it immediately at any price, if it is a market order; and (iv) the exchange finally executes both transactions, sending the results back to their specified brokers [Vishwanath and Krishnamurti, 2009]. This mechanism follows a simple supply and demand rule: when there is excess demand, the asset price tends to rise, and vice-versa.

The value of a stock can be estimated by two different approaches. *Fundamental analysis* considers macroeconomic (overall economy) and microeconomic (company's financial conditions and management) factors to determine the intrinsic asset value. On the other hand, *technical analysis* attempts to forecast the market behavior based on the statistics of the trading activity, such as price returns and trading volume. This analysis is commonly used in combination with charts of the asset price to find past patterns [Reilly and Brown, 2011]. Investors who use technical analysis of financial data to predict future market trends are called *chartists*.

Multiple indicators have been developed in an attempt to better predict the assets' future price movements, such as moving averages, trend lines and momentum indicators. Some of them are primarily focused on identifying the current market trend, and others determine its strength and the probability of its continuation.

The *risk* of an asset is strongly dependent on the volatility of its expected return. Investments can be classified as low, medium and high-risk investments. The lower the risk, the safer the bet, but the lower the return, and vice-versa. Investors have different preferences regarding acceptable risk for their investments. These preferences originate different investor's profiles.

3. Related Work

In the literature, we can find some research about financial market manipulation through multi-agent simulation. Regarding transaction taxes, there is a contradiction between models found in the literature. As [Marchesi et al. 2008] noted, the introduction of transaction taxes may increase asset price returns volatility and reduce market volume, mostly when there are chartist/technical traders present on the market. However, a different result was found by Buss et al.[Buss et al. 2016] and Westhoff and Dieci [Westhoff and Dieci 2006]. In our experiments, we used technical traders and no taxes, as there is no conclusion about its effect. [Moore et al., 2018] identified a price manipulation in the bitcoin market through suspicious trading activity in the Mt. Gox Bitcoin currency exchange. The bitcoin market is known for having no regulations at all and, thus, it is a golden pot for market manipulation. In our work, there were no

regulatory measures or transaction taxes modeled. We consider an environment that is optimal for speculators, like those observed in cryptocurrencies.

[Immonen 2017] proposed a robust and complex agent-based framework, based on dynamical systems models, contributing with agents that use both technical and fundamental strategies. We created a simple model by applying only technical analysis, as shown in Section 4.

A model based on partially cooperative agents in a world of risks was designed by [de Castro and Sichman 2012]. It assumes that investors have different preferences concerning their investments' risks and minimal returns, which were represented by an investor description model. Our traders agents were inspired by these representation of investor description models. A multi-agent architecture, named COAST, was proposed, where coaches negotiate with each other in order to define the best money allocation among different assets. The model used a financial market simulator called AgEx, that uses real data from NASDAQ stock exchange.

[Reis et al. 2016] used NetLogo integrated with R, a language and environment for statistical computing, along with machine learning techniques, to select the best feature or group of features that could better predict the market behavior using empirical data from BOVESPA stock exchange. Their results were used to select the technical indicators to use in our chartist agents. We applied in our work the exponential moving average (EMA), which is commonly used by chartist and chartist traders.

[Raberto et al. 2001] built the Genoa model, in which the clustering effect between investors was studied. As well as Genoa's, there is no money-creation in our model, the investors have a finite amount of cash and a finite quantity of assets, and the asset pricing is guided by simple supply and demand rules.

For more information regarding agent-based artificial financial markets, [LeBaron 2000] and [Cavalcante et al. 2016] reviewed and commented the existent literature and included a discussion about the future directions of this research field.

4. Framework

4.1. Environment and Design

We designed intelligent agents assigned with the role of investors and divided them into four groups with distinct investment profiles: (i) chartist traders, that use exponential moving average (EMA), (ii) random walkers, traders that will randomly send buy or sell orders) and (iii) “buy-and-hold” traders, that buy the assets in the beginning and just hold them until the end of simulation, as we focus on the effect caused by the asset price manipulation.

Each of them is capable of sending either buy or sell orders of a single asset and have a unique trading strategy. Limiting the trading to one asset only makes it simpler to observe these strategies without losing efficacy of the study. The investors can only send one order at a time: the investor must wait until its execution or expiration to send another order.

We assume a closed environment, in which there is no money creation and the agents start with a finite amount of cash and quantity of the asset. Thus, an agent can

only buy the asset with its available cash. These assumptions impose some constraints on our model and makes it more realistic.

The model is guided by simple trading rules, following the mechanism of an ordinary order book. The clearing price was determined using the methodology described by [Wurman et al. 1998], which modeled a double auction mechanism that admit multiple buyers and sellers. Let L be the total amount of active single-unit orders, M is the amount of sell orders and $N = L - M$ is the amount of buy orders. Wurman determines the *Mth-price* as M th highest price among all L orders, which plays the role as the clearing price of our order book.

We also designed a mechanism to simulate orders sent at the market price, i.e. orders sent at the lowest ask price or at the highest bid price. They are prioritized and executed before the others in the order book and do not influence the asset's price. This is useful when simulating the market reaction to a sudden drop or rise in the asset's price.

4.2. Strategies

As described in Section 4.1, we designed four different strategies assigned to four different groups of investors, which are described below. *Chartists* have been prioritized in relation to investors who follow a fundamentalist analysis, since in a speculative scenario the macroeconomic and microeconomic factors are neglected and the investment decisions are strongly influenced by the price trends. *Random-walkers* were designed to provide stock liquidity to the system and the *buy-and-holders* form the control group when compared to the others. These three are usually the common groups found in the market and their design is the simplest way to analyse market manipulation by a big investor.

Each strategy tells the investors whether they must send buy or sell orders, at which price and frequency of trading. With the exception of the *big-investor*, the quantity of stocks of each order sent by the investors was fixed to 1. By fixing this value, we assume a uniform order sending, reducing the standard deviation of the average wealth of each group of investors. Then, the rate of orders sent by one group of investors relies on the total number of investors present on this group and their trading countdown defined in Section 4.3.

4.2.1. Buy-and-holders

The buy-and-holders start with the predetermined cash and asset quantity and keep them until the end, without sending any buy or sell order during the entire period. The results of this group are used as a control group compared to the others.

4.2.2. Random-walkers

The random-walkers have an equal chance of either sending a buy order, sell order or holding its position. Thus, random-walkers tend to create a white noise in the asset's price return series, as noted by [Raberto et al. 2001] in their work. They also provide liquidity to the asset, by supplying the system with enough buy and sell orders, then allowing the other groups to apply their strategies.

The order's price definition considered a volatility factor. The price $p(t+1)$, at the time $t+1$, considers the value obtained by a random-normal function $N(\mu, \sigma_t)$, where μ is the distribution mean value and σ_t is the standard deviation of the asset's price calculated within the range of the 20 previous time steps. The price $p(t+1)$ is then defined as follows:

Buy orders:

$$p_b(t+1) = P(t) * N(\mu_b, \sigma_t) \quad (1.1)$$

Sell orders:

$$p_s(t+1) = P(t) * N(\mu_s, \sigma_t) \quad (1.2)$$

Where $P(t)$ is the asset's price at time t .

We set $\mu_b = 1.01$ and $\mu_s = 0.99$. By forcing the mean value of $p_b(t+1)$ to be slightly greater than $P(t)$ and the mean value of $p_s(t+1)$ to be slightly lower than $P(t)$, we are stimulating the trading environment and consequently increasing the volatility of the system. A similar volatility factor in the price formation is also found in [Raberto et al. 2001].

4.2.3. Chartists

A chartist sends a buy or a sell order relying on the trending denoted by the Momentum Strategy, here designed with exponential moving averages to predict the market behavior through moving average crossovers.

As the purpose of a chartist is to react to the changes of the market's behavior, all the orders sent by this group are at the market price and, thus, are prioritized against the others. The strategy employed with this indicator is described in Section 4.4.

4.2.4. Big-investor

The *big-investor* strategy is more complex than the others', though it is still simple: he buys in the lower prices to sell in the higher ones. This strategy relies on the Trend Indicator as well, but have a different duration than the one used by the chartists.

As the Momentum Strategy changes, the *big-investor*'s strategy starts. If it tends to the buying position, the *big-investor* sends a high share of buy-orders (85-95% of all sell-orders in the order book) until reaching 50% of the strategy duration time. There is then a small pause in which only *chartists* and *random-walkers* trade, and when it reaches the last 25% - when the prices are even higher, the *big-investor* starts sending sell-orders in smaller portions (20-30% of all buy-orders in the order book), until all the orders bought are sold or the asset's price reaches the value when the strategy was initiated.

In this way, the investor buys at a low price, starts a high trend and sells later, when it reaches higher prices. This strategy makes the speculation profitable for the investor.

4.3. Trading Countdown

The Trading Countdown was designed to control the market volatility, i.e., in order to control the frequency in which orders are sent by the investors. Each investor starts with a countdown at its maximum value. The countdown is then decreased by 1 for each time step. When the countdown reaches 0, the investor is enabled to send another order.

The maximum value varies accordingly to each group of investors. The *big-investor*, as well as the *chartists*, have a maximum countdown value of 5 time steps. After a few trials, this value was found to be the optimal for a proper market reaction, volatility control and strength of the *big-investor's* manipulation. On the other hand, the random-walkers have a countdown with a random maximum value of one of the following prime numbers: 2, 3, 5, 7 and 11. By setting the trading countdown of the *random-walkers* randomly to prime numbers, they provide the stock with enough liquidity for every time step, enabling the stock trading.

4.4. Moving Average Crossovers

The moving average crossover strategy denotes the market's tendency of buying, selling or holding the investors' position. It can assume three different values: -1 (selling position), 0 (holding position) and 1 (buying position). Exponential moving averages are calculated for long and short historical periods. When they cross each other, the indicator changes.

The exponential moving average (EMA) at time t is defined as [Kirkpatrick II and Dahlquist, 2010]:

$$EMA(t) = (P(t) - EMA(t-1)) * WM + EMA(t-1) \quad (2.1)$$

$$WM = \frac{2}{N_{EMA}-1} \quad (2.2)$$

Where $P(t)$ is the asset's price at time t , WM is the weighting multiplier, N_{EMA} is the period covered by the moving average and $EMA(t-1)$ is the exponential moving average at time $t-1$. The strategy holds the buying or selling position for a determined period before going back to the holding position. This period is set by a system parameter.

When the ascending short term EMA intersects the long term EMA from below, the strategy assumes the buying position (1), and when the descending short term EMA intersects the long term EMA from above, it assumes the selling position (-1). **Algorithm 1** describes the implemented logic in NetLogo. We called it a Momentum Strategy and it is indicated as MS, since it compares the current price in relation to the past price.

As the Momentum Strategy changes its value, it resets the *chartists'* and *big-investor's* countdown, as they promptly react to the market movement.

Algorithm 1 Calculate Momentum Strategy

```
1: to calculate-momentum-strategy
2:   if  $EMA_{short}(t-1) < EMA_{long}(t-1)$  and  $EMA_{short}(t) > EMA_{long}(t)$ 
3:     [ set MS 1
4:       ask chartists [ set countdown 0 ] ;resets the chartists' countdown
5:         if big-investor-active? ;system parameter that indicates whether the
           big-investor is enabled to send orders
6:         [ set big-investors-buy true ] ;initialize the big-investor's buying
           strategy
7:         set duration MS-duration ] ;system parameter that sets the duration of the
           tendency
8:     else if  $EMA_{short}(t-1) > EMA_{long}(t-1)$  and  $EMA_{short}(t) < EMA_{long}(t)$ 
9:       set MS -1
10:      ask chartists [ set countdown 0 ] ;resets the chartists' countdown
11:        if big-investor-active? ;system parameter that indicates whether the
           big-investor is enabled to send orders
12:        [ set big-investors-sell true ] ;initialize the big-investor's selling
           strategy
13:        set duration MI-duration ] ;system parameter that sets the duration of the
           tendency
14:      if duration = 0
15:        [ set MS 0 ;after reaching duration 0, updates MS to the neutral position
16:          set duration MS-duration ]
17:        if duration > 0
18:          [ set duration duration - 1 ] ;updates tendency duration
19: end
```

4.5. Order Expiration

As in real markets, investors don't let an order linger for too long. They tend to cancel and update their bets by sending another order. This is controlled by an Order Expiration parameter. As the orders reach the lifespan limited by this parameter, they are removed from the system and the investors are free to renew their bets.

4.6. Parameters and Interface

Chartists, *random-walkers* and *buy-and-holders* start with 80 shares of asset and \$1000 amount of money. The *big-investor* starts with 100.000 times more wealth than the other investors, i.e., 8.000.000 shares of the asset and \$100.000.000 amount of money. Thus, if the asset's starter price is 10, then the initial wealth of the *big-investor* is 180 million and the others' is 1.800.

When investors run out of stocks, they have 25% chance of sending a buy-order and 75% of holding their position. Similarly, when they run out of money, they have 25% chance of sending a sell-order.

The total number of investors, with the exception of the *big-investor*, is 200. The ratio of the *chartists* is 0.05, *random-walkers*' is 0.90 and *buy-and-holders*' is then 1 - 0.90 - 0.05 = 0.05. The more *chartists* present in the market, the higher the volatility of

the stock price. Analogously, the more *random-walkers* in the market, the higher the liquidity of the stock. The order expiration time is set as 15 ticks. Every tick is defined as an increase of one time step: $t + 1$.

The short term N_{EMA} is 15 ticks and the long term N_{EMA} is 100 ticks. The duration of the tendency set by the Momentum Strategy is 150 ticks. Regarding the big-investor, there is a switch that turns on/off his strategy, whose duration is parameterized as 100 ticks.

The screen interface is divided in 4 quadrants. The top quadrants are occupied by the *chartists*. In the bottom-left quadrant are the *random-walkers* and in the bottom-right are the *buy-and-holders*. The big-investor lies on the middle line. When sending an order, the investors hatch a turtle-agent that holds the order attributes and links it to them. Then, the orders are sent to the middle of the screen, where the asset lies. The orders are matched and, after updating the wealth of their respectively investors, they are deleted. The orders that do not match with any other stay in the middle until they are resolved or expired.

The initial exponential moving averages and initial asset price are calculated based on the historical closing prices of the Brazilian stock PETR4, from Petrobras, obtained from BOVESPA between the periods of 1st June 2017 to 3rd August 2017.

4.7. Meta-algorithm

For every time step, the **algorithm 2** is executed.

Algorithm 2 Go

```

1: to go
2:   ask orders [ expire ] ;orders expiration procedure
3:   ask chartists [ send-order ] ;regular order sending procedure
4:   ask random-walkers [ send-random-order ] ;random-order sending procedure
5:   ask big-investor [ send-big-order ] ;big-investor's order sending procedure
6:   ask orders [ move-to stock ] ;moves all the orders to the middle of the screen
7:   ask stock [ close-trades ] ;matches the orders and execute the trades
8:   ask investors [ update-wealth ] ;updates the wealth of all investors
9:   ask stock [ update-stock-price ] ;updates the stock-price
10:  calculate-indicators ;calculates mean wealth sd, EMAs and momentum strategy
11:  update-labels ;updates stock, investors and other interface labels
12:  tick
13: end

```

5. Results and Discussion

We ran 20 trials for 3 different scenarios: in the first one, there were only random-walkers trading. Next, we added the *chartists* and finally the big-investor, when all of three groups traded. Wealth variation, stock price, standard deviations and stock returns where observed and recorded.

Random-walkers send buy or sell orders arbitrarily, increasing or decreasing the stock price with equal probability. Thus, they make the time series unpredictable. By definition, a white noise process has serially uncorrelated errors with expected mean

equal to zero. A random walk time series is non-stationary because its covariance is time dependent. In **Figure 1**, we observe the logarithmic plot of the price returns of Scenario 1 (when only random-walkers are trading). In Scenarios 2 and 3, it is possible to observe some trends created by the *chartists* and the *big-investor*, especially around timestep 375.

Table 1. Description of the scenarios and its participants.

	Buy-and-holders	Random-walkers	Chartists	Big-investor
Scenario 1	Active	Active	Inactive	Inactive
Scenario 2	Active	Active	Active	Inactive
Scenario 3	Active	Active	Active	Active

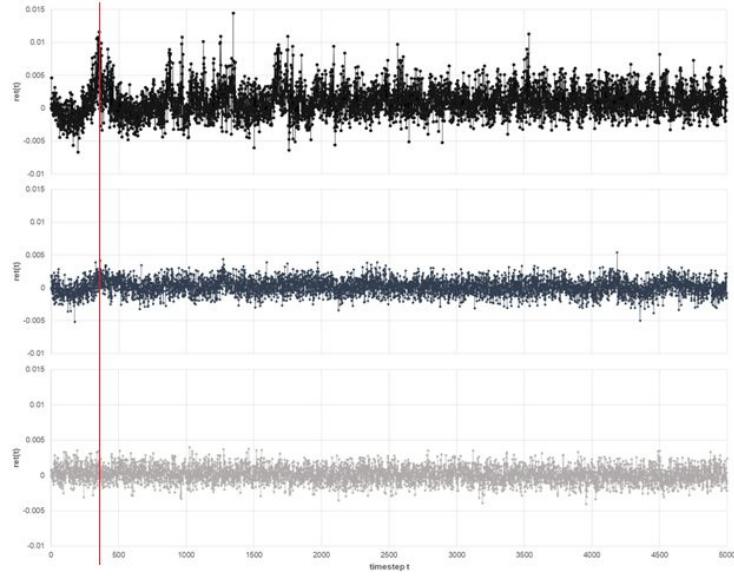


Figure 1. Logarithmic plot of price returns function $ret(t) = \log(P(t)) - \log(P(t-1))$. The black dots concerns the Scenario 3, the blue gray dots are from the Scenario 2 and the light gray dots are from the Scenario 1.

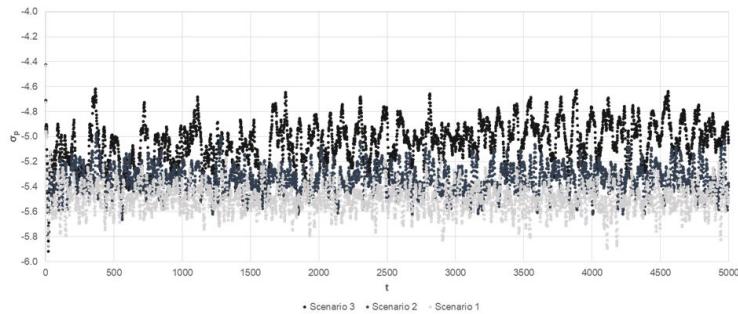


Figure 2. Logarithmic plot of the stock prices' standard deviation for the three different scenarios.

The introduction of the strategies from the *chartists* and the random-walkers enhanced the volatility of the time series and, thus, the risk of the stock returns. In **Figure 2**, the standard deviations $\sigma_p(t)$ of the three different scenarios are logarithmic plotted. The standard deviation $\sigma_p(t)$ of the price $P(t)$ is calculated based on the

previous 20 timesteps. As observed, the time series standard deviation rises with the introduction of the other market players.

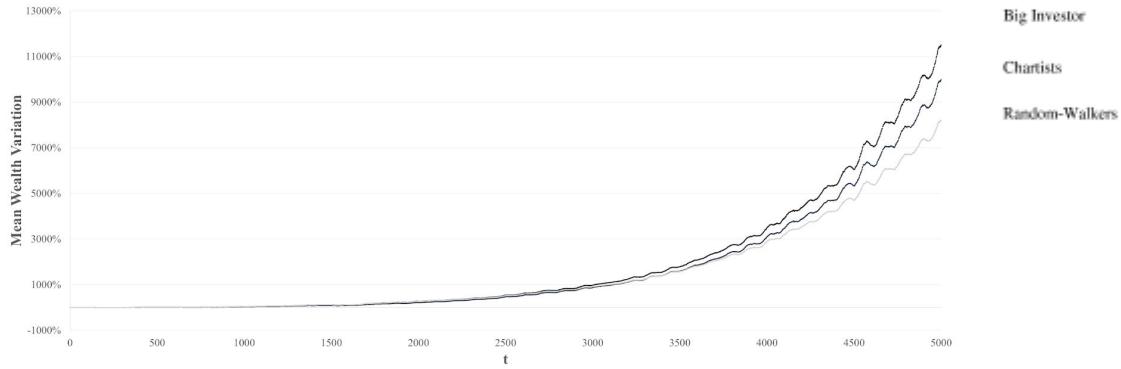


Figure 3. Overall performance of three groups in Scenario 3: big-investor, chartists and random-walkers.

The overall performance of the three different groups is observed in **Figure 3** – the curve for the *buy-and-holders* was suppressed since it is very similar to the *big-investor*'s curve, and final results after 5000 time steps are shown in **Table 2**. The *big-investor* and the *buy-and-holders* had the most profitable returns as the stock price rose significantly. Both multiplied their wealth by 115. After them, the *chartists* were able to grow their fortune by almost 100 times and the random-walkers by 82 times, this one with a standard deviation of 27 times. The stock price rose from 13.3 to 2985.1, 22384% higher.

Table 2. Final results after 5000 time steps on the Scenario 3.

	Buy-and-holders	Random-walkers	Chartists	Big-investor
Mean Wealth Variation	11506.5% ± 0.0%	8203.3% ± 2658.4%	9991.5% ± 241.2%	11512.6% ± 0.0%

As discussed in section 3, there were no regulatory measures introduced in our system, which is a perfect environment for speculation. In a regulated market, the manipulation would be more subtle. The strength of our artificial market manipulation relies on the period of the EMAs (velocity of the market response) and other parameters, such as how much *chartists* are trading on our artificial market (volatility) and the total number of investors (liquidity). It is possible to observe the fat tails formed in the time series in **Figure 4** and the exponential growth that market manipulation created on the stock price time series.

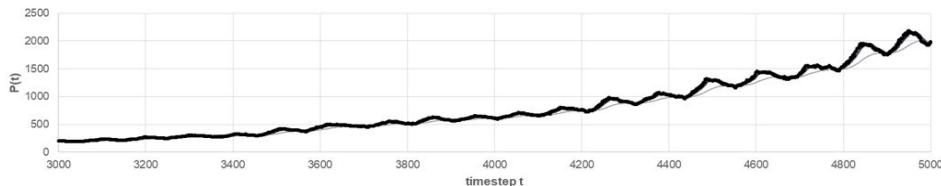


Figure 4. Price $P(t)$ time series from one of the Scenario 3 trials starting at time step 3000.

6. Conclusions

An artificial market was successfully built using NetLogo. It was possible to emulate different investment profiles and introduce technical analysis within its strategies. The financial market is unpredictable, but it can be manipulated to a certain level by investors with a great amount of capital in the absence of regulation. During our research, we identified scenarios that are very close to real situations, such as cryptocurrencies markets, in which no regulatory measures are applied.

The big investor is able to manipulate the market by injecting a great amount of capital in one stock, giving an artificial sensation of growth in this particularly stock, which is followed by the investment of other market players and creating a chain reaction. Since big investors hold a large amount of stock shares, they can handle its price negatively as well, by selling a huge amount of shares of this same stock in a short time span.

The *buy-and-holders*, compared to *chartists* and *random-walkers*, take advantage on the market manipulation because the stock price was rising. However, on an opposite manipulation, i.e., forcing a price decrease, their strategy could be the worst one. The different reaction timing to the market changes from the *chartists* could explain why this group underperformed when compared to the *big-investor* strategy.

Furthermore, NetLogo seems a great tool for the construction of our artificial financial market, enabling the creation of several intelligent agents, with an easy and concise programming language and meeting system performance.

Future steps of our work include defining better-designed big-investor strategies, introducing new investor profiles based on expected risk and return, applying different technical analysis indicators, as well as a fundamentalist approach, and inserting multiple stocks in our artificial market.

References

- Angel, J. and McCabe, D. (2013). “Fairness in financial markets: The case of high frequency trading”, *Journal of Business Ethics*, v. 112, n. 4, p. 585-595, 2013.
- Asparouhova, E., Bossaerts, P. and Plott, C. (2003) “Excess demand and equilibration in multi-security financial markets: the empirical evidence”, *Journal of Financial Markets*, vol. 6, Elsevier Science B.V., p. 1-21.
- Buss, A., Dumas, B., Uppal, R. and Vilkov, G. (2016) “The intended and unintended consequences of financial-market regulations: A general-equilibrium analysis”, *Journal of Monetary Economics*, vol. 81, Elsevier B.V., p. 25-43.
- Castro, P. A. and Sichman, J. S. (2009) “Agex: a financial market simulation tool for software agents”, In: Aalst W., Mylopoulos J., Sadeh N. M., Shaw M. J., Szyperski C., Filipe J., Cordeiro J. (eds) LNBIP, vol 24, Springer, Berlin, p. 704–715.
- Castro, P. A. and Sichman, J. S. (2013) “Automated asset management based on partially cooperative agents for a world of risks”, *Appl Intell*, vol 38, Springer, Berlin, p. 210-225.

- Cavalcante, R. C., Brasileiro, R. C., Souza, V. L. F., Nobrega, J. P. and Oliveira, A. L. I. (2016) “Computational Intelligence and Financial Markets: A Survey and Future Directions”, *Expert Systems With Applications*, vol. 55, Elsevier Ltd., p. 194-211.
- CVM, C. d. V. M. (2013) “Mercado de valores mobiliários brasileiro.” [S.l.]: Comissão de valores mobiliários, 1st ed., ISBN 978-85-67896-00-7.
- Immonen, E. (2017) “Simple agent-based dynamical system models for efficient financial markets: Theory and examples”, *Journal of Mathematical Economics*, vol. 69, Elsevier B. V., p. 38-53.
- Kirkpatrick II, C. D. and Dahlquist, J. A. (2010) “Technical analysis: the complete resource for financial market technicians”. FT press.
- LeBaron, B. (2000) “Agent-based computational finance: Suggested readings and early research”, *Journal of Economic Dynamics & Control*, vol. 24, Elsevier Science B.V., p. 679-702.
- Marchesi, M., Mannaro, K. and Setzu, A. (2008) “Using an artificial financial market for assessing the impact of Tobin-like transaction taxes”, *Journal of Economic Behavior & Organization*, vol. 67, Elsevier B.V., p. 445-462.
- Moore, T., Gandal, N., Hamrick, J. T. and Oberman, T. (2018) “Price Manipulation in the Bitcoin Ecosystem”, *Journal of Monetary Economics*, <https://doi.org/10.1016/j.jimoneco.2017.12.004>.
- Raberto, M., Cincotti, S., Focardi, S. M. and Marchesi, M. (2001) “Agent-based simulation of a financial market”, *Physica A*, vol. 299, Elsevier Science B.V., p. 319-327.
- Reilly, F. K. and Brown, K. C. (2011) “Investment analysis and portfolio management”. Cengage Learning.
- Reis, E. R., Castro, P. A. and Sichman, J. S. (2016) “Enhancing Classification Accuracy Through Feature Selection Methods”, XIII Encontro Nacional de Inteligência Artificial e Computacional, SBC ENIAC, Recife – PE, Brazil.
- Vishwanath, R and Krishnamurti, C. (2009) “Investment Management: A Modern Guide to Security Analysis and Stock Selection”. Springer-Verlag Berlin Heidelberg, 1st ed., p. 13-29.
- Westerhoff, F. H., and Dieci, R. (2006) “The effectiveness of Keynes–Tobin transaction taxes when heterogeneous agents can trade in different markets: A behavioral finance approach”, *Journal of Economic Dynamics & Control*, vol. 30, Elsevier B. V., p. 293-322.
- Wurman, P. R., Walsh, W. E and Wellman, M. P. (1998) “Flexible double auctions for electronic commerce: theory and implementation”, *Decision Support Systems*, vol. 24, Elsevier Science B.V., p. 17-27.

Simulador de NoCs modelado como um SMA

Gustavo L. Lima¹, Nelson F. Traversi¹, Odorico M. Mendizabal¹, Cristina Meinhardt², Diana F. Adamatti¹, Eduardo W. Brião³

¹Centro de Ciências Computacionais – C3, Programa de Pós Graduação em Computação -PPGComp, Universidade Federal do Rio Grande (FURG)

²Universidade Federal de Santa Catarina (UFSC)

³Instituto Federal do Rio Grande do Sul - Campus Rio Grande (IFRS)

{gustavolameirao, 20kfurg, dianaada}@gmail.com

odoricomendizabal@furg.br

cristina.meinhardt@ufsc.br

eduardo.briao@riogrande.ifrs.edu.br

Abstract. This work presents the development of a NoC simulator modeled as a multiagent system. NoC improves the transmission of messages in SoCs. This simulator aims to provide the user with a high degree of environment abstraction to simulate a NoC, allowing to define the number of messages, packets and transmission speed among routers, thus facilitating the modeling of new systems. In addition, the simulator calculates the routers load, number of messages that passed through each router and the number of lost messages, allowing to observe real-time information. As a case study, the XY routing protocol was modeled and analyzed.

Resumo. Este trabalho apresenta o desenvolvimento de um simulador de NoCs modelado como um sistema multiagente. Nocs melhoram a transmissão de mensagens em SoCs. Este simulador visa proporcionar ao usuário um ambiente com alto grau de abstração para simulação de uma NoC, permitindo definir o número de mensagens, pacotes e velocidade de transmissão entre os roteadores, assim facilitando a modelagem de novos sistemas. Além disso, o simulador calcula a carga nos roteadores, o número de mensagens que passaram por cada roteador e o número de mensagens perdidas, permitindo observar parâmetros da simulação em tempo real. Como estudo de caso, foi modelado e analisado o protocolo de roteamento XY.

1. Introdução

O constante avanço nas tecnologias de fabricação permite que sejam desenvolvidos circuitos integrados cada vez menores e mais eficientes. Além disso, estes circuitos também se tornam cada vez mais complexos, através da integração de diversos componentes (como processador, memórias) dentro de um único chip. Esta evolução levou a criação de sistemas completos dentro de um chip, sendo estes denominados SoC (*System-on-a-chip*). Um SoC tem inúmeras aplicações, principalmente em sistemas

embarcados e *smartphones*. As vantagens, quando comparadas aos sistemas antigos compostos por componentes individuais, são relacionadas principalmente com a melhora no desempenho, redução nos atrasos de comunicação e nos custos de fabricação, uma vez que tudo estará em um chip. Devido à complexidade de comunicação criada nos SoCs, foi proposta a adoção de redes internas nos chip, ou NoCs (*Network-on-Chip*) para realizar a comunicação entre os componentes do sistema (HEMANI, 2000).

O desempenho e confiabilidade das NoCs são afetados por diversas características, por exemplo, topologia, algoritmo de roteamento e controle de fluxo. Para melhorar a compreensão sobre o efeito das configurações associadas às características, é interessante testar o comportamento das NoCs em simuladores. Os simuladores de NoC na literatura podem ser classificados por vários fatores, como os parâmetros de entrada, tipo de resultados obtidos a partir de simulação, tipo de licença, interface de usuário, entre outros. Uma pesquisa abrangente sobre simuladores de NoC é apresentada em (BEN, 2013) (ALALAKI, 2017). Em (BEN, 2013), os simuladores são classificados de acordo com os parâmetros disponíveis e as métricas de saída. Em (ALALAKI, 2017), uma comparação entre é mostrada, de acordo com outras características como framework utilizado, topologias suportadas, suporte para GUI, ferramentas de benchmark, data de lançamento e operação síncrona / assíncrona.

O objetivo deste estudo é desenvolver um simulador de NoCs modelado na forma de um SMA, com foco nos algoritmos de roteamento. Algoritmos de roteamento tem a função de determinar qual caminho os pacotes irão seguir para que a troca de mensagens entre os nodos de destino e origem se comuniquem. O simulador tem a capacidade de simular e avaliar cenários com alto grau de abstração, independência do *hardware*, configuração versátil e obtenção de indicadores de uso de componentes como resultados de saída. Além disso, o simulador está preparado para que os usuários e desenvolvedores possam implementar novos algoritmos de roteamento e os avaliar. Como caso de uso, o algoritmo de roteamento XY foi escolhido. A escolha deste algoritmo se deve a sua simplicidade e vasta presença na literatura sobre NoCs. Além disso, características importantes do algoritmo puderam ser constatadas com a simulação.

Dentre os simuladores encontrados, concentrarmos nossa pesquisa em simuladores que oferecem a possibilidade de escolher e analisar algoritmos de roteamento. Em seguida, levamos em consideração a configuração dos parâmetros e as métricas de saída disponíveis. Por exemplo, comparamos os recursos dos simuladores para descrever o tamanho do NoC, o tamanho dos buffers, a geração de tráfego, a licença de uso e a interface do usuário. Três simuladores principais foram identificados: Noxim (CATANIA, 2015), DARSIM (LIS, 2010) e NS-2 (NS-2, 2007). Tanto o Noxim quanto o DARSIM apresentam parâmetros de entrada configuráveis similares aos escolhidos no nosso simulador, como tamanho de *buffers*, da rede e dos pacotes. As principais diferenças são relacionadas a geração de tráfego, onde o nosso é capaz de simular tráfego aleatório ou manual, e na existência de uma interface gráfica, que facilita a utilização da criação do cenário, além de ser possível observar comportamentos de saída durante a execução. Já o NS-2 embora seja usado para NoCs, não é um simulador específico de NoC. Ele é um simulador de redes de computadores, onde usuários podem criar abstrações para adaptar o funcionamento para simular uma NoC.

A Seção 2 deste trabalho apresenta os principais conceitos sobre NoCs. A Seção 3 mostra detalhes sobre a implementação do simulador (como os parâmetros de entrada e indicadores). A Seção 4 apresenta os resultados obtidos em simulações. Por fim, a Seção 5 apresenta as conclusões do trabalho.

2. Network-On-Chip

A complexidade criada pela integração de vários sistemas em um chip faz com que seja repensada a forma com que estas partes se comunicam. Algumas arquiteturas de comunicação, como barramentos ou fios dedicados, quando usadas nos SoCs, oferecem baixa escalabilidade. Os barramentos apresentam limitações de transmissão, pois apenas uma palavra pode ser transmitida por ciclo de clock. Conexões com fios dedicados também não são uma boa opção, devido a baixa reusabilidade e flexibilidade (BRIAOU, 2008).

Como alternativa para estas limitações, surgem outras propostas de como realizar esta comunicação, as NoCs (Network-On-Chip) (HEMANI, 2000). De maneira geral, a comunicação é feita através da criação de uma grande rede, composta por nodos de processamento conectados a nodos de chaveamento (através de uma interface de rede), e os nodos de chaveamento interligados por *links* físicos. Os nodos de processamento, denominados núcleos ou *Intellectual Property cores*, têm a função de executar as ações que o sistema deve realizar. Já os nodos de chaveamento, os roteadores, são responsáveis por realizar a transferência de mensagens entre dois cores que desejam se comunicar, através de ligações físicas, ou seja, através dos *links*.

A forma com que os roteadores são dispostos depende da topologia adotada. Um exemplo de organização é apresentado na Figura 1 (TOPÍRCEANU, 2013). Na Figura, é apresentada uma NoC 3x3 destacando seus principais componentes, os Núcleos de Processamento, as *Interfaces* de rede, os roteadores e os *links* físicos.

Trocas de mensagens entre dois pontos da NoC podem ocorrer por diversos caminhos. Este fato faz com que seja necessário adotar estratégias de roteamento, de maneira similar ao que acontece em uma rede de computadores, para determinar o caminho por onde a troca de mensagens ocorrerá. O uso de roteamento faz com que a comunicação nas NoCs seja mais eficiente quando comparado com os barramentos convencionais, podendo ser observado qual o menor caminho entre os pontos, ou qual caminho está menos congestionado. Além disso, as decisões de roteamento dos pacotes que trafegam pela NoC podem ser distribuídas, auxiliando assim na escalabilidade do sistema.

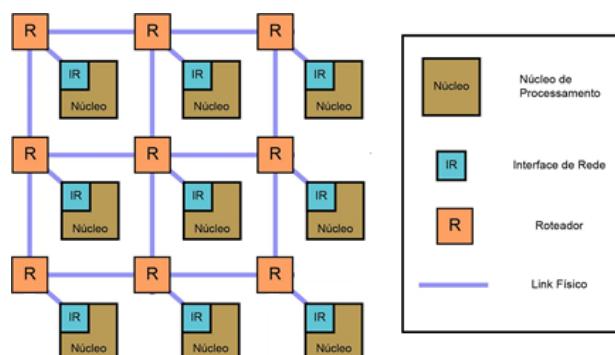


Figura 1: Organização de uma NoC

A Interface de Rede é responsável por fazer a ligação entre os processadores e a rede, tornando assim a comunicação mais transparente para o processo. A comunicação é dividida em duas partes: *back-end* e *front-end*. O *back-end* é responsável por fazer a comunicação com os núcleos, tanto do processo de envio, quanto no recebimento das mensagens. Esta estrutura permite a reutilização de multiprocessadores, bastando apenas mudar detalhes nesta camada. O *front-end* tem por objetivo transmitir os dados utilizados em serviços de alto nível da camada de transporte, como transferência de dados confiáveis e dados sobre qualidade de serviço (QoS) (NUNES, 2015).

A comunicação entre componentes da NoC é realizada pelo envio e recebimento de mensagens. Porém, como as mensagens que trafegam são grandes e de tamanhos variados, estas são divididas em pacotes menores, com tamanho menor ou igual a largura de banda do Enlace (NUNES, 2015). Cada pacote é dividido em três partes:

1. *Header* (cabeçalho) - contém as informações para rotear a mensagem pela rede.
2. *Payload* (carga útil) – é o corpo da mensagem.
3. *Trailer* (terminador) - comando que indica o término da mensagem.

Os roteadores são componentes cruciais dentro das NoCs. São responsáveis por transmitir os dados que trafegam por essa rede. A maioria dos roteadores de alto desempenho implementam uma rede de interconexão do tipo *crossbar* para permitir que múltiplas conexões entre portas de entrada e saída aconteçam simultaneamente. Como os roteadores recebem uma grande quantidade de mensagens, devem prover filas eficientes onde os pacotes serão armazenados até serem processados e enviados aos seus destinatários. As mensagens podem ser guardadas nos buffers de entrada, nos buffers de saída, em ambos ou até mesmo dentro do roteador (HENNESSY, 2008).

De forma geral, o roteamento é implementado utilizando uma máquina de estados finita ou uma tabela de encaminhamento, armazenada dentro da unidade de controle de roteamento no roteador. Na primeira opção, os dados de roteamento presentes no cabeçalho da mensagem são processados pela máquina de estados, e esta determina para qual saída a mensagem deverá ir. Já na segunda opção, as informações de roteamento presentes no cabeçalho da mensagem são utilizadas como um endereço de acesso a tabela de roteamento, carregada na inicialização do roteador, que contém as saídas a serem utilizada. A unidade de controle de roteamento também é responsável pela arbitragem, pois os pacotes podem chegar de maneira simultânea as portas de entrada, podendo dois pacotes requerer a mesma saída. Cabe ao roteador determinar quem terá prioridade ao acesso na porta requerida. A arbitragem pode ser centralizada ou distribuída. Na abordagem centralizada, todo o processo de distribuição dos pacotes às portas de saída é feito por um agente centralizador. Já na abordagem descentralizada, cada porta de entrada possui um módulo de roteamento, juntamente com um módulo de arbitragem ligado à sua respectiva porta de saída.

3. Desenvolvimento de um Simulador de NoC no NetLogo

O simulador de NoCs foi implementado no ambiente de criação de ferramentas multiagente chamado NetLogo (WILENSKY, 1999). Cada nodo (roteador) da NoC é representado por um agente. A definição do agente inteligente mostra similaridades. O roteador é uma entidade real, que opera em um ambiente, a rede NoC. Ele se comunica

com outras entidades: os demais roteadores. Tem o objetivo de realizar a comunicação de forma eficiente, possui recursos próprios tais como os *buffers* de entrada/saída e processamento. Além disso, é capaz de perceber mudanças no ambiente, por exemplo detectar congestionamento com seus vizinhos. E possui apenas uma representação parcial do ambiente completo sendo capaz de realizar serviços como enviar mensagens para outros roteadores. O mapeamento com as principais semelhanças entre agentes e roteadores é apresentado na Tabela 1.

Na implementação do simulador, a topologia inicial escolhida para a distribuição dos agentes foi uma malha 2D, formando uma grade. Os agentes foram organizados nesta grade distanciados uns dos outros por uma unidade de medida. Em seguida, criou-se os *links* entre cada um dos agentes dentro de uma distância 1. Dessa forma, as partes que compõem a grade foram conectadas apropriadamente. A malha formada é quadrada, e tem dimensão NxN, onde N é um valor escolhido pelo usuário. Para permitir a criação de diferentes cenários de simulação, este simulador disponibiliza ao usuário uma série de características configuráveis, sendo estas:

- **Número de roteadores:** Com base no valor N escolhido, será gerada uma NoC contendo N linhas e N colunas.
- **Tamanho do buffer:** Define em número de pacotes o tamanho do *buffer* de cada roteador.
- **Quantidade de pacotes por mensagem:** as mensagens são divididas em pacotes, e cada pacote possui um tamanho padrão. Esta opção define quantos pacotes são necessários para compor uma mensagem.
- **Pacotes por ciclo:** define quantos pacotes cada roteador é capaz de enviar em cada ciclo. Por exemplo, se for escolhido 3, a cada ciclo os roteadores irão enviar no máximo 3 pacotes, mesmo que tenham mais ou menos mensagens armazenadas em seus buffers.
- **Enviar mensagens aleatoriamente:** quando selecionado, este parâmetro realiza simulações onde remetente e destinatário de mensagens são escolhidos aleatoriamente.
- **Raio de mensagens:** determina o valor do raio onde cada roteador de origem irá escolher um roteador destino para enviar mensagens.
- **Completamente aleatório:** quando selecionado, o raio de escolha de pares de roteadores, remetente e destinatário, é totalmente aleatório, podendo escolher qualquer roteador da rede.
- **Número de ciclos para envio de novas mensagens:** define a frequência com que novas mensagens serão enviadas.
- **Mensagens a serem enviadas:** define um total de mensagens que devem ser enviadas em uma simulação.
- **Dados para envio de mensagem manual:** em um cenário onde determinadas mensagens definidas precisam ser enviadas, é possível escolher o agente origem através do campo Agente-Origem e as coordenadas X e Y do destinatário através dos campos Mensagem-X/Mensagem-Y. Após escolher os valores, basta disparar a mensagem através do botão mensagem-individual.

Tabela 1. Características de agentes e roteadores

Agente	Roteador
Entidade Real/Virtual	Entidade Real
Ambiente	Rede NoC
Comunicação com outros agentes	Ligaçāo entre os roteadores
Objetivos/Satisfações	Comunicação eficiente
Recursos Próprios	Buffers de entrada, processamento, dentre outros
Percebe o ambiente	Detecta congestionamento entre vizinhos
Representação parcial	Somente se relaciona com vizinhos conectados
Competência/Prestação de serviços	Envio da mensagem ao destino desejado

Finalizando a configuração dos parâmetros de entrada, a NoC é montada através do botão *setup*, e a simulação é executada através do botão *go*. A simulação é medida por ciclos. Cada vez que o *go* é executado, contabiliza-se 1 ciclo de execução. A função *go* é repetida indefinidamente até atingir uma condição de parada. Nessa função, cada roteador é processado sequencialmente. No processamento do roteador, este verifica se tem mensagens armazenadas em seu *buffer*, e transmite até x mensagens para o próximo roteador de acordo com o algoritmo de roteamento. x é o número máximo de mensagens que um roteador consegue encaminhar por ciclo. Quando um roteador recebe um pacote no *buffer*, primeiramente é verificado se existe espaço disponível para o armazenamento no *buffer*. Caso não exista, o pacote é descartado e é contabilizado como perdido. Se existir espaço, o pacote é colocado no final da fila de ciclo *buffer* para ser processado. A fila do *buffer* segue o modelo FIFO (*First in, First Out*). Ao fim, é calculado quanto tempo levou desde o início da simulação e o recebimento do último pacote, para fins de medição de desempenho do simulador.

Além destas informações, os roteadores armazenam outros dados como: Quantidade de ciclos que ficou em cada cor, onde a cor está relacionada a taxa de ocupação dos *buffers* do roteador; Quantidade total de pacotes processados pelo mesmo; Quantidade de pacotes perdidos pelo mesmo; e Taxa de ocupação geral, sendo determinada como a média da porcentagem de ocupação de seus buffers ao longo da execução como um todo.

A função *transmit* é responsável por determinar a forma com que os pacotes são roteados. Os algoritmos de roteamento nas NoCs têm o papel de determinar o caminho pelo qual uma mensagem deve trafegar para chegar ao seu destino. Existe uma série de fatores que diferenciam estes algoritmos e que são utilizadas para classificar os mesmos. Os algoritmos de roteamento mais conhecidos são o XY, NL, NF e WF (CARARA, 2004). Nesta primeira versão do simulador, a função *transmit* implementa o algoritmo XY (CARARA, 2004), devido a sua simplicidade e aparecer frequentemente na literatura. Entretanto, novos algoritmos de roteamento podem ser acrescentados sobrescrevendo a função *transmit*. Permitindo com o simulador a avaliação do algoritmos de roteamento e seu impacto na NoC simulada. O Algoritmo XY é um algoritmo determinístico, isto significa que para cada entrada determinada, o algoritmo

sempre produzirá a mesma saída. Neste algoritmo, os pacotes são roteados na direção X até atingir a coordenada X destino, em seguida são roteados na direção Y até atingir Y destino. Um exemplo de funcionamento é mostrado na Figura 2, onde cada quadrado representa um roteador, cada número identifica o roteador, e as setas indicam o caminho que a mensagem fez para ir do roteador 1 ao 10 (CARARA, 2004).

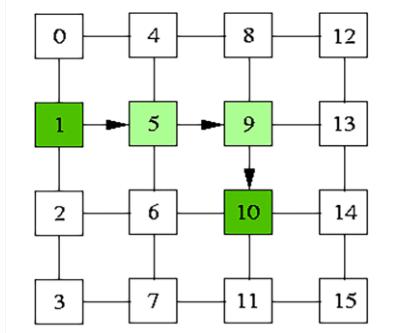


Figura 2: Estrutura do algoritmo XY

O simulador também disponibiliza uma série de dados da simulação, tanto durante quanto ao final da execução. Dentre os indicadores, é interessante destacar:

- **Coloração dos roteadores:** durante cada ciclo, os roteadores recebem a cor referente a sua taxa de ocupação.
- **Contabilização de pacotes:** É possível verificar a quantidade de pacotes enviados, recebidos (ao chegar no destino), perdidos (por não existir buffers para o armazenamento) e trafegando pela rede (que ainda não atingiram seu destino, mas estão em movimento).
- **Total do número de saltos:** contabiliza por quantos roteadores os pacotes passaram.
- **Número médio de saltos:** define-se uma média de quantos saltos os pacotes deram até chegar ao destino.
- **Tabela de ciclos:** tabela com a quantidade de ciclos que o roteador ficou em determinada cor.
- **Tabela de taxa de ocupação:** tabela com a taxa de ocupação de cada roteador ao longo da execução.
- **Tempo de execução:** desde o início da simulação até o recebimento da última mensagem.

A coloração dos registradores é realizada após processar as mensagens, mediante um cálculo realizado para determinar a porcentagem de ocupação do *buffer* do roteador. Com base nesta porcentagem, o simulador oferece uma saída visual da utilização do roteador, colorindo de acordo com a classificação da capacidade ocupada dos buffers apresentada na Tabela 2.

Tabela 2. Representação de cor conforme ocupação do buffer

Cor	Ocupação
Azul	0~1% da capacidade do buffer ocupada
Verde	1~30% da capacidade do buffer ocupada
Amarelo	30~60% da capacidade do buffer ocupada
Vermelho	60~99% da capacidade do buffer ocupada
Preto	100% da capacidade do buffer ocupada

4. Estudo de Caso: Algoritmo XY

Para averiguar o funcionamento da ferramenta desenvolvida, foram criadas simulações variando os principais atributos de entrada. Vale ressaltar que, para o estudo de caso, o algoritmo de roteamento utilizado na ferramenta foi o XY. Dois grupos de simulações foram propostos. No grupo 1, foram realizados testes variando o raio de escolha de roteadores para envios de mensagens, a quantidade de pacotes por mensagens, a quantidade de buffers e a quantidade de pacotes processados por ciclos em cada roteador. No grupo 2, foi criado um caso padrão (o mesmo do primeiro grupo) e foi variado o tamanho da NoC. A metodologia de cada experimento e os resultados são detalhados na sequência deste trabalho.

4.1. Grupo 1

A partir de uma NoC 20x20, foram enviadas 1000 mensagens. O raio da escolha de roteadores foi variado para 3 (pequeno), 9 (médio) e 30 (grande), tal valor de raio determina a distância máxima possível que o roteador destino pode ser escolhido a partir do roteador de origem. Dentro de cada raio, também foram criados 4 casos de teste, variando mais características, totalizando 12 casos de teste. Os casos são:

- **Caso padrão:** 10 pacotes por mensagens, *buffers* de tamanho 100 em cada roteador e cada roteador processa 10 pacotes por ciclo.
- **Caso 1:** 50 pacotes por mensagens, *buffers* de tamanho 100 em cada roteador e cada roteador processa 10 pacotes por ciclo, simulando um ambiente com mensagens maiores.
- **Caso 2:** 10 pacotes por mensagens, *buffers* de tamanho 500 em cada roteador e cada roteador processa 10 pacotes por ciclo, representando um ambiente com roteadores com mais *buffers*.
- **Caso 3:** 10 pacotes por mensagens, *buffers* de tamanho 100 em cada roteador e cada roteador processa 100 pacotes por ciclo, simulando um ambiente onde os roteadores com maior capacidade de processamento.
- **Caso 4:** 10 pacotes por mensagens, *buffers* de tamanho 15 em cada roteador e cada roteador processa 10 pacotes por ciclo, representando um ambiente onde roteadores têm menos *buffers*, para simular perdas de pacotes.

Na Tabela 3 é possível observar os valores de tempo de simulação, quantidade de pacotes enviados e recebidos, encontrados nas simulações dos 5 casos.

Tabela 3. Resultados obtidos nos casos com raio 3, 9 e 30. Relação entre tempo de simulação (medido em segundos), número de pacotes enviados e número de pacotes perdidos.

Caso	Raio 3	Raio 9	Raio 30
Padrão	16,015s//10000//0	22,064s//10000//0	31,452s//10000//0
1	55,574s//50000/0	100,381s//49970//30	159,896s//49900//100
2	45,446s//10000//0	52,107s//10000//0	60,652s//10000//0
3	15,607s//10000//0	23,67s//10000//0	30,949s//10000//0
4	15,194s//9970//30	15,092s//9665//335	22,179s//8975//1025

Para ilustrar a quantidade de mensagens manipuladas por cada roteador, foi escolhido o caso 2. O tipo de gráfico escolhido foram os mapas de calor. Estes mapas são utilizados para mostrar a intensidade da ocorrência de alguma situação. Os mapas de calor deste caso nos raios 3, 9 e 30 são apresentados nas Figuras 3, 4 e 5, respectivamente. Nestes gráficos, os pontos onde ocorreram a maior manipulação de mensagens estão com a coloração mais acentuada. Vale ressaltar que houve uma normalização nos gráficos. O ponto máximo da escala de cada gráfico foi escolhido com base no maior valor apresentado na representação em questão.

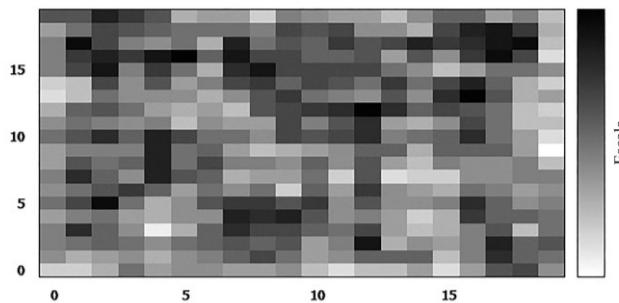


Figura 3: Estrutura do algoritmo XY

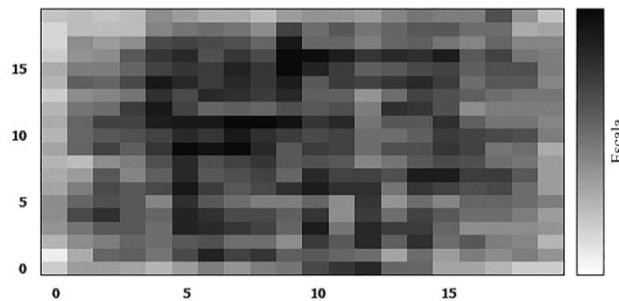


Figura 4: Estrutura do algoritmo XY

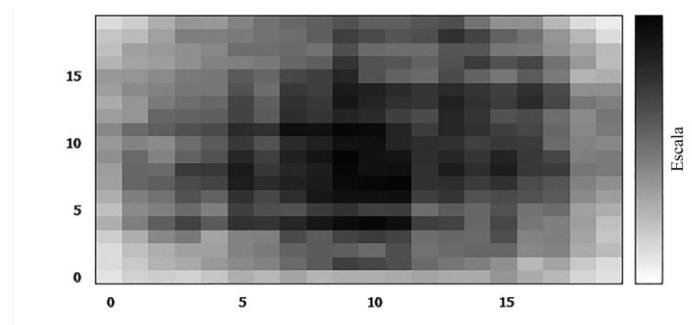


Figura 5: Estrutura do algoritmo XY

Ao comparar o comportamento do mapa de calor de quantas mensagens passaram pelos roteadores em um mesmo caso, variando o raio (exemplo: caso 1 nos raios 3, 9 e 30), é possível perceber que conforme o raio aumenta, acontece uma concentração de mensagens manipuladas no centro, como mostra a Figura 5. Este comportamento é esperado para esse algoritmo de roteamento, devido ao fato de que, ao utilizar um raio pequeno, os roteadores trocam mensagens apenas com roteadores vizinhos. Porém, ao aumentar o raio, existe troca de mensagens entre roteadores que podem estar muito distantes. Por esse motivo, as mensagens passam pelo meio, e há uma maior carga nestes roteadores centrais ilustrada na Figura 5. É importante ressaltar que, embora seja mostrada apenas a comparação do caso 2, este comportamento se repete em todos os casos.

Além desse comportamento, também foi possível observar que boa parte das variáveis não influenciam diretamente no desempenho do sistema, mas sim na perda de pacotes. Nas simulações aconteceram poucas perdas de pacote, pois foram ambientes balanceados e projetados para que isso ocorresse. O caso que teve maior variação de tempo, comparado aos demais, foi o caso 1, que utiliza um maior número de pacotes por mensagem. De maneira simplificada, mais pacotes por mensagem significa mensagens maiores. Esse caso fica mais pesado ainda conforme o raio aumenta devido ao fato de que podem ser feitas comunicações entre roteadores mais distantes, ou seja, vários pacotes precisam trafegar a rede inteira para chegar ao destino. Como são vários pacotes por mensagem, os roteadores intermediários terão seus buffers ocupados e irão precisar manipular vários pacotes que atravessam a rede por grandes distâncias.

4.2. Grupo 2

Na segunda bateria de testes, a partir do mesmo caso padrão anterior e com o mesmo número de 1000 mensagens, variou-se o tamanho da NoC e a forma de escolha dos roteadores para as trocas adotando o modo totalmente aleatório. Os tamanhos escolhidos para a NoC foram 3x3, 9x9, 20x20 e 50x50. A relação do tamanho da NoC com o tempo de execução é mostrada na Figura 6. Este grupo de testes foi simulado para testar a escalabilidade do sistema, conforme o tamanho da rede aumenta. Conforme observado no gráfico, o aumento é linear. Mesmo no maior dos casos, onde a NoC é 50x50, o tempo de simulação ficou em torno de 101,253 segundos.

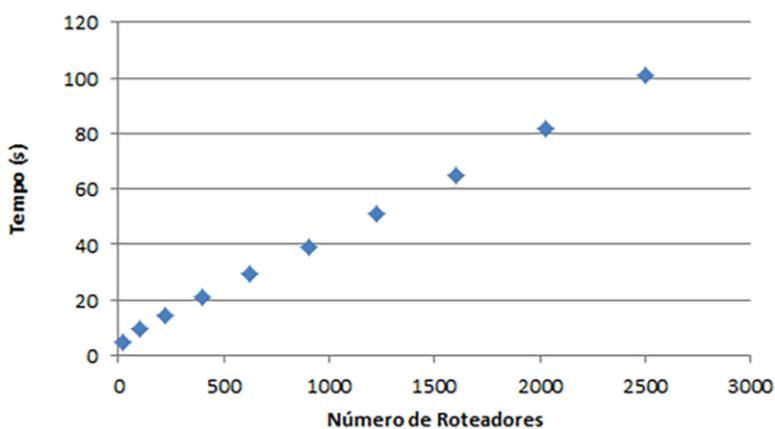


Figura 6: Estrutura do algoritmo XY

5. Conclusões

Este estudo mostrou as etapas da construção de um simulador de NoCs desenvolvido em um ambiente multiagente. Para demonstrar o potencial de aplicação do simulador, foram apresentados os resultados obtidos em simulações variando diversos parâmetros, como tamanho da rede, o raio de escolha de roteadores, tamanho das mensagens, quantidade de buffers, capacidade de processamento dos roteadores, dentre outros. Foi possível observar comportamentos nas simulações, como o da concentração de mensagens que passam pelo centro da NoC. Estes comportamentos são esperados por conta do algoritmo de roteamento utilizado, o XY.

Em síntese, conclui-se que o simulador possui uma alta versatilidade, onde o usuário pode criar uma NoC descrevendo vários parâmetros de maneira simples e obter indicadores. Dessa forma, a ferramenta pode auxiliar os desenvolvedores de NoCs a obter os parâmetros adequados para a sua aplicação, de maneira eficiente e simples. Acredita-se que estudos envolvendo as áreas de NoC e SMA são relevantes e que este trabalho possa contribuir neste sentido. Como trabalhos futuros, pretende-se implementar novos algoritmos de roteamento, para que seja possível comparar o desempenho destes e ver qual se adapta melhor a NoC desenvolvida e realimentar o simulador com parâmetros mais acurados.

Referências

- ALALAKI, M.; AGYEMAN, M. A Study of Recent Contribution on Simulation Tools for Network-on-Chip (NoC), International Journal of Computer Systems, vol. 11 no.4, 2017.
- ALI, M. et. al. Using the NS-2 Network Simulator for Evaluating Network on Chips (NoC), in International Conference on Emerging Technologies, 2006, pp. 506 - 512.
- BRIAQ, E. W. Métodos de Exploração de Espaço de Projeto em Tempo de Execução em Sistemas Embuçados de Tempo Real Soft Baseados em Redes-em-Chip. Tese de doutorado (Ciência da Computação). UFRGS. Porto Alegre. 2008.
- BEN, A.; SAOUD, S. A Survey of Network-On-Chip Tools, International Journal of Advanced Computer Science and Applications, vol. 4 no. 9, 2013.

- CARARA, E. A. Uma exploração arquitetural de redes intra-chip com topologia malha e modo de chaveamento Wormhole. Trabalho de Conclusão de Curso, 2004.
- CARDOZO, E.; MAGALHÃES, M.F. Redes de computadores: modelo OSI, 2012. Disponível em: < <ftp://ftp.dca.fee.unicamp.br/pub/docs/eleri/apostilas/osi.pdf> >. Acesso em: ago. 2017.
- CATANIA, V. et. al. Noxim: An Open, Extensible and Cycle-accurate Network on Chip Simulator, in Application-specific Systems, Architectures and Processors (ASAP), 2015 IEEE 26th International Conference on, Toronto, 2015.
- HEMANI, A. et. al. Network on chip: An architecture for billion transistor era. In: IEEE NORCHIP CONFERENCE, 2000. Procedding... [S.1.:s.n.], 2000.
- HENNESSY, J. L.; PATTERSON, D. A. Arquitetura de Computadores: Uma abordagem quantitativa. Elsevier, 4 edition, 2008.
- HENNESSY, J. L.; PATTERSON, D. A. Organização e Projeto de Computadores – A Interface Hardware Software. 3 edition, 2005.
- LIS, M. et. al. DARSIM: a parallel cycle-level NoC simulator, in IEEE Asian SolidState Circuits Conference, Saint Malo, 2010.
- NS-2 website. Available: nsnam.isi.edu/nsnam/index.php/Main_Page, 2007.
- NUNES, C. A. K. Uma estratégia para redução de congestionamento em sistemas multiprocessadores baseados em NOC. 2015. Dissertação (Mestrado em Ciência da Computação). Centro de Informática, UFPE, Pernambuco.
- TOPÍRCEANU, A. Networks On Chip, 2013. Disponível em: < <https://sites.google.com/site/alexandrutopirceanu/research/networks-on-chips>>. Acesso em: ago. 2017.
- WILENSKY, U. (1999). NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Transporte de Agentes Cognitivos entre SMA Distintos Inspirado nos Princípios de Relações Ecológicas

Vinicius S. de Jesus¹, Fabian Cesar P. B. Manoel¹, Carlos Eduardo Pantoja^{1,2}, Jose Viterbo²

¹Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET-RJ)
Campus Maracanã – 20271-110 – Rio de Janeiro – RJ – Brasil

²Universidade Federal Fluminense (UFF)
Campus Praia Vermelha – 24210-346 – Niterói – RJ – Brasil

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

Abstract. In biology, living beings are able to establish relationships that can be classified according to the behavior of those involved. Agents are autonomous entities capable of making decisions, cognitive reasoning and socializing with other agents in a Multi-Agent System (MAS). Some agents are able to move to other MAS, thus being able to relate to agents, in a similar way to living beings. This work aims to propose protocols inspired by biological relations with the purpose of exploring the movement of agents to a MAS embedded in a physical and autonomous device to another MAS in a different device. Three protocols will be addressed: predatism, mutualism and inquiline, where the transference is made with the goal of dominating, exchanging knowledge and surviving in the destination MAS, respectively. These protocols aim to preserve and/or share the indispensable knowledge obtained during the existence of the agents. In this case, an MAS can use one of the proposed protocols to migrate to another embedded system. Finally, some initial experiments will be presented, in which two prototypes (one leader and one host) were created where the leader is damaged and the predatism relationship is activated to preserve the acquired knowledge.

Resumo. Na biologia, os seres vivos são capazes de estabelecer relações que são classificadas de acordo com o comportamento dos envolvidos. Agentes são entidades autônomas capazes de tomar decisões, raciocínio cognitivo e socializar com outros agentes em um Sistema Multi-Agente (SMA). Alguns agentes são capazes de se moverem para outros SMA, podendo, assim, se relacionar com outros agentes, de forma similar aos seres vivos. Este trabalho tem como objetivo propor protocolos inspirados nas relações biológicas com a finalidade de explorar a movimentação de agentes de um SMA embarcado em um dispositivo físico e autônomo para outro SMA em um dispositivo distinto. Serão abordados três protocolos: predatismo, mutualismo e inquilinismo, onde a transferência é feita com o objetivo de dominar, trocar conhecimentos e sobreviver no SMA de destino, respectivamente. Estes protocolos visam preservar e/ou compartilhar os conhecimentos obtidos durante a existência dos agentes. Neste caso, um SMA pode utilizar um dos protocolos para migrar para outro sistema embarcado. Por fim, serão apresentados alguns experimentos iniciais, nos quais foram criados dois protótipos (um líder e um hospedeiro) onde o líder é danificado e a relação de predatismo é acionada para preservar os conhecimentos adquiridos.

1. Introdução

Para a ecologia, relações ecológicas é o estudo dedicado a explicar a forma que os seres vivos se relacionam a fim de manterem suas vidas [Tissot-Squalli 2009]. Existem subdivisões estabelecidas para as relações ecológicas tais como: mutualismo, inquilinismo e predatismo. O mutualismo é o tipo de relação ecológica onde os seres envolvidos possuem características que beneficiam os outros seres da relação, sendo considerada positiva para todos; o inquilinismo é a relação em que apenas um dos seres envolvidos se beneficia da relação, sem causar prejuízo a outro ser; já o predatismo é a relação em que um dos seres se beneficia, prejudicando o outro.

Os Sistemas Multi-Agentes (SMA) [Wooldridge 2009] são sistemas compostos de agentes autônomos, onde estes podem reagir a estímulos, tomar decisões e interagir com outros agentes. Os SMA são oriundos da sociologia e etologia [Hübner et al. 2004] e, por isso, se baseiam na ideia de que um indivíduo isolado não é tão forte quanto um conjunto de indivíduos. Dessa forma, pode-se considerar que a abordagem de agentes é um paradigma que estuda o comportamento de um grupo de agentes que busca a resolução de problemas coletivamente ou compete por um objetivo comum. Então, ao se falar de SMA, leva-se em consideração o relacionamento entre as suas partes e, assim como podem existir diferentes espécies se relacionando em um habitat, podem existir diferentes agentes em um SMA, inclusive, SMA diferentes se relacionando em um mesmo ambiente. Os SMA podem ser abertos ou fechados [Chebout et al. 2016], onde os sistemas abertos são capazes de se comunicarem com outros SMA, enquanto que, os fechados não. Os sistemas abertos também permitem a entrada e saída de agentes, estes agentes com essa capacidade de se moverem para diferentes SMA são chamados de agentes móveis.

O Ambient Intelligence (AmI) [Augusto Wrede et al. 2010] é um ambiente que faz uso de técnicas e sistemas computacionais alinhados a dispositivos físicos para interagir de forma ubíqua e imperceptível com os usuários visando auxiliá-los em tarefas específicas no dia-a-dia. A Internet of Things (IoT) [Zhang et al. 2012], é a interligação de diversos dispositivos autônomos e proativos em uma mesma rede e na internet com o objetivo de resolver problemas do dia-a-dia das pessoas. Para o desenvolvimento de aplicações para a IoT existe um *middleware* chamado de ContextNet [Endler et al. 2011] que proporciona a comunicação entre dispositivos ou nós móveis visando atividades com a geração de contexto e que suportam milhares de conexões simultâneas. Para o desenvolvimento de SMA existem diversos *frameworks*, como o Jason [Bordini et al. 2007] e Jade [Bellifemine et al. 2007], que possuem interpretadores de linguagens de programação orientadas a agentes que são utilizadas para criação de AmI integrados com a IoT.

Através de uma pesquisa na literatura de SMA abertos e agentes móveis foi possível observar que os principais trabalhos não focam na transferência de agentes entre diferentes SMA, mas sim, na segurança dos SMA, como nos trabalhos THOMAS [Ossowski et al. 2007] e em modelo de confiança entre agentes e SMA (FIRE) [Dong-Huynha et al. 2004], ou até mesmo, no consenso e adaptação dos agentes em um novo SMA, como, em análise de estratégias intra-equipe [Sanchez-Anguix et al. 2012] e estratégia de agrupamento de sistemas bio-inspirado em colônias de formigas [Calvo 2012]. Sendo assim, deixando uma lacuna na literatura que este trabalho pretende explorar.

Os trabalhos citados focam em soluções sobre o relacionamento entre os agentes de um SMA e agentes externos. Sendo assim, o objetivo deste trabalho é proporcionar o transporte de agentes de um SMA para outro baseado nos conceitos de relações ecológicas da biologia [Tissot-Squalli 2009]. Para isso, foram propostos 3 protocolos para a transferência de agentes entre SMA distintos, conhecidos como predatismo, inquilinismo e mutualismo, onde permitem dominar, sobreviver e trocar experiências com outros SMA, respectivamente. Será realizado um experimento com dois protótipos de carros que possuem sensores ultrassônicos, motores e LEDs, cada um com um SMA embarcado, onde um carro ativará o protocolo predador e, portanto, predará o outro carro para dar continuidade as suas atividades.

O trabalho está estruturado da seguinte maneira: na seção 2, o referencial teórico será apresentado; na seção 3, o protocolo de transferência de agentes será apresentado; na seção 4, os testes executados serão expostos; na seção 5, os Resultados e discussões serão abordados; na seção 6, os trabalhos relacionados serão discutidos; na seção 7, a conclusão e trabalhos futuros serão exibidos; e, por fim, as referências bibliográficas serão apresentadas.

2. Referencial Teórico

Nesta seção, serão explorados os conceitos básicos para a compreensão do trabalho, tais como, o *framework* Jason, a arquitetura ARGO, assim como as tecnologias utilizadas para o desenvolvimento de tais ambientes e sistemas (*middleware* ContextNet).

2.1. Desenvolvimento de SMA

O Jason [Bordini et al. 2007] é um *framework* que interpreta uma linguagem orientada a agentes chamada AgentSpeak em Java para a programação de SMA e, também, implementa o modelo Belief-Desire-Intention (BDI) [Bratman 1987], permitindo assim, a programação de agentes reativos e cognitivos. O modelo BDI é composto basicamente por crenças, desejos e intenções: as crenças são informações tidas como verdades pelo agente; os desejos são as motivações que um agente possui para realizar algum objetivo e intenções são ações que o agente se compromete a realizar.

O ARGO [Pantoja et al. 2016b] é uma arquitetura customizada que permite programar agentes com a capacidade de interagir e controlar microcontroladores através do *framework* Jason. Os agentes com esta capacidade podem ser chamados também de agentes robóticos [Matarić 2007], pois, são eles que fazem a interação direta com o *hardware* fornecendo leitura de sensores, controle de acionamentos e interação com o meio físico. A arquitetura customizada ARGO é formada por dois principais módulos: o *middleware* Javino e os filtros de percepções. O Javino [Lazarin and Pantoja 2015] permite a comunicação entre linguagens de alto nível e controladores que possuem uma conexão serial, por exemplo, o Arduino; além disso, o Javino possui uma estrutura de identificação de erro que impede a perda de dados nas mensagens enviadas ou recebidas em tempo de execução. Os filtros de percepções [Stabile Jr. and Sichman 2015] são mecanismos que fazem com que o agente bloquee as percepções do ambiente com base em filtros projetados pelo desenvolvedor em tempo de design, porém, os filtros podem ser alterados em tempo de execução pelo agente que possui o arquivo.

2.2. Tecnologias para a Internet das Coisas

O ContextNet [David et al. 2012] é um *middleware* criado para a IoT e serviços de contexto, que visa aplicações colaborativas, coordenação de atividades de entidades móveis e compartilhamento de informações. As entidades móveis podem ser qualquer dispositivo móvel, por exemplo, veículos, *smartphones*, *tablet*, *notebook* ou até mesmo robôs autônomos com conexão a redes IoT. Entre as tecnologias utilizadas para o desenvolvimento do ContextNet, o *Scalable Data Distribution Layer* (SDDL), [Vasconcelos et al. 2013], é uma camada do ContextNet responsável pela capacidade de comunicação e de distribuição de contexto. As demais funcionalidades do ContextNet foram desenvolvidas em módulos de *software* no topo da camada de distribuição. Além disso, o ContextNet permite a entrada de dispositivos em sua rede e o tratamento adequado.

O *middleware* ContextNet [Endler et al. 2011] permite o controle de milhares de nós móveis conectados ao mesmo tempo. Em [David et al. 2012] simula-se uma situação para o rastreamento em tempo real de milhares de cargas sendo transportadas com o posicionamento sendo atualizado. Por conta disso, o *middleware* ContextNet apresenta características que podem ser exploradas para a conexão entre diversos SMA embarcados e prover um ambiente para as relações biológicas propostas neste trabalho.

3. Protocolo de Transferência de Agentes

Nesta seção serão apresentadas a metodologia do protocolo de transferência de agentes, as possíveis relações estabelecidas entre SMA de origem e SMA de destino, além da sua implementação e tecnologias utilizadas. O protocolo de transferência de agentes permite a transferência de SMA inteiros ou agentes considerando a relação que será estabelecida com o SMA de destino, que são baseadas nos conceitos de relações ecológicas oriundas da biologia: Inquilinismo, Mutualismo e Predatismo. Na relação de Inquilinismo, o SMA inteiro se transfere para outro SMA com o intuito de fazer parte do mesmo; no Mutualismo, um agente ou o SMA de origem sai com a intenção de adquirir e transmitir novos conhecimentos, e posteriormente, voltar; já na de Predatismo, o SMA de origem visa a preservação da integridade de todos os seus agentes, assim, quando esta relação é ativada, o SMA predador se transfere para outros SMA com o intuito de dominar e transferir todos os seus agentes.

Para acontecer à transferência de um agente ou SMA para outro SMA existem duas condições: os SMA precisam estar conectados em uma rede IoT utilizando o *middleware* ContextNet para estabelecer uma conexão entre os diferentes SMA, dando a eles e aos seus agentes identificadores no ambiente; e a segunda condição é que o SMA de destino precisa permitir a entrada do novo SMA ou do agente transmitido.

O transporte de um agente é realizado enviando um arquivo do estado mental atual do agente para outro SMA. Primeiramente, é preciso criar um arquivo com as crenças, desejos e intenções atuais do agente (para isso, é preciso acessar a base de crenças, desejos e intenções, assim como a biblioteca de plano deste agente e acrescentar tudo que foi adquirido em tempo de execução); segundo, quando o agente chega ao SMA de destino, a sua permissão de entrada é avaliada, e é enviada uma mensagem de notificação para o SMA de origem; terceiro, se a notificação indicar que a entrada foi permitida, o SMA de origem apaga todas as informações dos agentes no sistema de origem.

Para a transmissão dos agentes ou SMA planeja-se criar uma ação interna no Jason (.moveOut), onde são passados três parâmetros: o primeiro é o identificador do SMA de destino no ContextNet; o protocolo de transferência de agente utilizado (Inquilinismo, Mutualismo ou Predatismo); e o terceiro é o agente que será enviado caso o protocolo utilizado for o mutualismo e deseja-se enviar somente um agente. Com isso, o SMA de destino pode conhecer a intenção da transferência dos agentes e permitir ou não a entrada destes, contudo uma ontologia mais apropriada se faz necessário. A figura 1 apresenta as etapas para o protocolo de Predatismo.

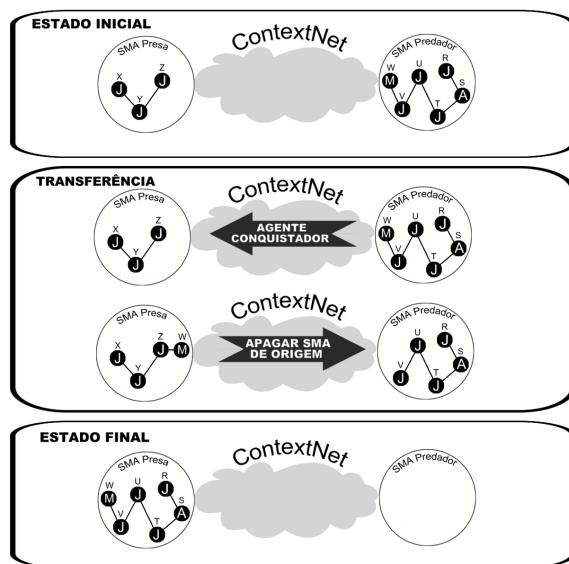


Figure 1. Relação de Predatismo e a dominação do SMA.

O protocolo de transferência de agentes foi proposto compreendendo a necessidade de mais mecanismos para a preservação da integridade dos conhecimentos dos agentes de um SMA, pois, os conhecimentos adquiridos por um agente em tempo de execução podem ser importantes, uma vez que, este pode adquirir ou transmitir conhecimentos, realizar desejos, intenções ou planos. Com isso, todas as relações do protocolo consideram a preservação dos conhecimentos dos agentes e em manter o agente sendo executado. A implementação do protocolo de agentes será feita como uma extensão da arquitetura de agentes presentes no *framework* Jason para possibilitar a utilização dos protocolos de transferência de agentes com as relações de inquilinismo, mutualismo e predatismo.

4. Testes executados

Nesta seção serão abordados os testes executados para avaliar o funcionamento dos protocolos de transferência de agentes aplicados em protótipos físicos no mundo real. Os testes de viabilidade dos protocolos de transferência de agentes foram realizados com o intuito de verificar a possibilidade de mover um agente de um SMA para outro, que consistiu em enviar um arquivo na extensão .asl (arquivo-fonte de um agente Jason) de um computador para outro através do *middleware* ContextNet.

Os testes executados neste trabalho foram aplicados em dois protótipos de carros autônomos com *hardware*s idênticos — com mesmos sensores, atuadores e controladores

— controlados por agentes ARGO situados em um SMA embarcado em uma Raspberry PI posicionada nos protótipos. Cada um dos protótipos possuem em seus respectivos SMA agentes dedicados ao controle do *hardware*, pois é mais eficiente dividir as tarefas e os controles de *hardware* entre diferentes agentes [Pantoja et al. 2016a]. Na Figura 2, podem ser vistos os protótipos dos carros.

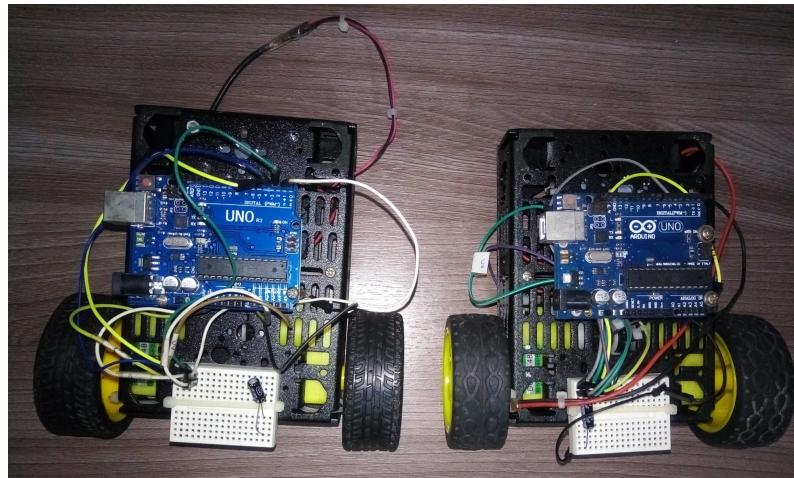


Figure 2. Protótipo dos carros autônomos.

O teste é constituído de dois veículos autônomos com SMA distintos, que são encarregados de uma missão hipotética, onde o SMA líder captura informações sigilosas para o sucesso da missão, todavia, este é atingido por um adversário de forma que seu *hardware* está irrecuperável. Para proteger a integridade dos seus conhecimentos e concluir a missão, o líder atingido aciona o protocolo de transferência de agentes utilizando a relação de predatismo, por conta da situação de risco iminente de parada total de funcionamento do seu *hardware* e, assim, transferir os seus conhecimentos, dominando o *hardware* de um outro SMA participante da missão.

Os protótipos dos carros são idênticos para que não haja problemas de compatibilidades, por exemplo, um SMA aplicado em um carro tem conhecimento para controlar um carro, agora se este SMA se transferir para um avião, não conseguirá controlar o seu *hardware* e, consequentemente, cairá. Neste caso, o protocolo de transferência de agentes com a relação de inquilinismo poderia ser aplicado, já que o SMA do carro pode se mover para o avião e fazer parte deste novo SMA até aprender a controlá-lo até que possa dominar ou auxiliar no controle do avião.

Ao se conectar ao ContextNet, todo SMA terá um código de identificação, representado pela identificação do seu agente Comunicador na rede IoT, para que o SMA de origem possa enviar agentes para o SMA de destino. Porém, como o teste proposto possui somente 2 dispositivos e sendo feito com *hardwares* idênticos, o SMA de origem não precisa atentar-se para questões de rastreabilidade e compatibilidade.

A construção dos SMA dos protótipos dos carros é similar, um agente ARGO para fazer a comunicação com o *hardware*, já, o controle das funcionalidades será feita por agentes Jason tradicionais. Desta maneira, o agente ARGO somente recebe as informações de leitura dos sensores e aguarda para atuar ou não no meio através dos seus atuadores. Porém, são os agentes Jason que através das informações recebidas do agente

ARGO e baseado em seus conhecimentos (crenças, desejos e intenções) tomam decisões para atuar ou não no meio físico enviando mensagens para o agente ARGO acionar ou não os atuadores.

5. Resultados e discussões

Nesta seção, serão abordados os resultados e discussões visando os detalhes da implementação dos protocolos de transferência de agentes, limitações e resultados.

Todos os agentes em Jason têm um arquivo onde fica armazenado o seu código-fonte, por exemplo, se houver um agente chamado bob no SMA, no projeto do Jason terá um arquivo bob.asl. Todavia, este arquivo armazena somente o código-fonte de criação do agente, com as crenças, desejos e intenções iniciais do agente, assim, os conhecimentos adquiridos em tempo de execução não estão presentes neste arquivo. Então, como o protocolo de transferência de agentes envia o arquivo asl de agentes, antes do envio, o protocolo adiciona todos os conhecimentos adquiridos em tempo de execução pelos agentes aos seus respectivos arquivos asl.

O momento da chegada ou a saída de um SMA ou agente também deve ser considerado, pois, quando um SMA é iniciado pelo *framework* Jason, este inicia o processo de criação de todos os agentes do sistema através dos arquivos asl de um projeto Jason. Todavia, o protocolo de transferência de agentes é executado pelo SMA já em tempo de execução, então é preciso integrar os agentes enviados aos SMA de destino e destruir as suas cópias no SMA de origem manualmente, além de apagar os arquivos asl no SMA de origem.

Atualmente, a biblioteca do Jason utilizada neste projeto conta com três tipos de arquitetura de agentes, a arquitetura tradicional do Jason, a ARGO e a *Communicator*, protocolo de transferência de agentes utiliza a *Communicator* para a transferência de agentes, porém, há uma discussão aberta para decidir se é preciso ou não criar uma nova arquitetura.

Além disso, o protocolo de transferência de agentes possui limitações na tomada de decisão para a escolha do SMA de destino, pois, no atual estágio de desenvolvimento, para utilizar o protocolo, o programador precisa conhecer os códigos de identificação dos SMA no ContextNet e definir o SMA de destino.

Por fim, os resultados dos testes com os protótipos dos carros foram satisfatórios e a transferência dos agentes e a execução do protocolo com a relação de predatismo funcionou corretamente. Vale ressaltar também que a velocidade de transmissão dos dados depende da rede que o ContextNet estará utilizando. Porém, como os arquivos-fontes dos agentes são de extensão asl (texto) e, portanto, pequenos, não necessitam de uma rede robusta.

6. Trabalhos Relacionados

Nesta seção serão apresentados trabalhos teóricos sobre SMA abertos e agentes móveis, que não possuem implementação e tem foco na segurança e no comportamento de um SMA ao receber um agente externo. Estes trabalhos não utilizam o transporte de agentes propriamente dito e nenhuma relação biológica; por sua vez, este trabalho propõe relações biológicas baseados na movimentação de agentes de um SMA para outro SMA em situações onde o *hardware* esteja danificado ou em perigo eminente.

O trabalho [Ossowski et al. 2007] propõe uma arquitetura abstrata chamada THOMAS, onde esta fornece a solução de problemas e limitações na programação de agentes, com a possibilidade de programar SMA abertos. Porém, não aborda a transferência de agentes para outros SMA, somente a relação entre agentes em um ambiente aberto e o tratamento de negociação entre eles. Em [Sanchez-Anguix et al. 2012], não é levado em consideração à origem dos agentes que entram no ambiente aberto proposto, e nem situações como um agente se movendo para outro SMA. Em FIRE, [Dong-Huynha et al. 2004], são demonstrados, matematicamente, modelos de confiança que são projetados para garantir que apenas agentes bem intencionados serão usados dentro do contexto. Em [Calvo 2012], foi aplicado o sistema bio-inspirado na colônia de formigas como estratégia de agrupamento de sistemas robóticos que exploram e vigilam ambientes desconhecidos. Já os protocolos propostos neste artigo visam o transporte de SMAs ou agentes para outros SMA, abordando a recepção dos mesmos dentro dos sistemas de destino e os seus objetivos finais (conhecer, ensinar, sobreviver ou explorar).

7. Considerações Finais

Este trabalho apresentou o desenvolvimento dos protocolos de transferências de agentes, a aplicação de conceitos provenientes da biologia aplicados na relação entre SMA de origem e SMA de destino, e os testes realizados em protótipos para verificar a aplicabilidade dos protocolos no meio físico. Além disso, o foco foi no protocolo com a relação de predatismo, uma vez que o atual estágio de desenvolvimento está concentrado no mesmo.

Para trabalhos futuros pretende-se realizar alguns testes em meio físico com os demais protocolos, desenvolver uma extensão para o *framework* Jason para permitir a transferência de SMA inteiros ou agentes possibilitando o controle de acionamento dos protocolos de transferência. Além disso, existe um projeto de um laboratório inteligente feito em escala real no CEFET Maria da Graça pelos autores que será aplicado os protocolos de transferência de agentes.

Referências

- Augusto Wrede, J., Nakashima, H., and Aghajan, H. (2010). Ambient intelligence and smart environments: A state of the art. pages 3–31.
- Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing multi-agent systems with JADE*, volume 7. John Wiley & Sons.
- 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.
- Calvo, R. (2012). *Sistemas bio-inspirados para coordenação de múltiplos robôs móveis*. PhD thesis, Universidade de São Paulo.
- Chebout, M. S., Mokhati, F., Badri, M., and Babahenini, M. C. (2016). Towards preventive control for open mas - an aspect-based approach. In *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO*, pages 269–274. INSTICC, SciTePress.
- 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.
- Dong-Huynha, T., Jennings, N., and Shadbolt, N. (2004). Fire: An integrated trust and reputation model for open multi-agent systems. In *ECAI 2004: 16th European Conference on Artificial Intelligence, August 22-27, 2004, Valencia, Spain: including Prestigious Applicants [sic] of Intelligent Systems (PAIS 2004): proceedings*, volume 110, page 18.
- 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.
- 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.
- Lazarin, N. M. and Pantoja, C. E. (2015). A robotic-agent platform for embedding software agents using raspberry pi and arduino boards. In *9th Software Agents, Environments and Applications School*.
- Matarić, M. J. (2007). *The robotics primer*. Mit Press.
- Ossowski, S., Julián, V., Bajo, J., Billhardt, H., Botti, V., and Corchado, J. (2007). Open mas for real world applications: An abstract architecture proposal. In *Proc. XII Conference of the Spanish Association for Artificial Intelligence (CAEPIA)*, volume 2, pages 151–160.
- Pantoja, C. E., de Jesus, V. S., and Filho, J. V. (2016a). Aplicando sistemas multi-agentes ubíquos em um modelo de smart home usando o framework jason. In *II Workshop de Pesquisa e Desenvolvimento em Inteligência Artificial, Inteligência Coletiva e Ciência de Dados (Workpedia)*. Universidade Federal Fluminense.
- Pantoja, C. E., Stabile Jr, M. F., Lazarin, N. M., and Sichman, J. S. (2016b). Argo: A customized jason architecture for programming embedded robotic agents. *Fourth International Workshop on Engineering Multi-Agent Systems (EMAS 2016)*.
- Sanchez-Anguix, V., Aydogan, R., Julian, V., and Jonker, C. M. (2012). Analysis of intra-team strategies for teams negotiating against competitor, matchers, and conceders. In *The 5th International Workshop on Agent-based Complex Automated Negotiations (ACAN 2012)*, pages 1–8.
- Stabile Jr., M. F. and Sichman, J. S. (2015). Evaluating perception filters in BDI Jason agents. In *4th Brazilian Conference on Intelligent Systems (BRACIS)*.
- Tissot-Squalli, M. (2009). *Interações ecológicas & biodiversidade*. Unijuí.
- Vasconcelos, I., Vasconcelos, R., Baptista, G., Seguin, C., and Endler, M. (2013). Desenvolvendo aplicações de rastreamento e comunicação móvel usando o middleware sddl. In *Salão de Ferramentas, Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2013)*.
- Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley.
- Zhang, D., Ning, H., Xu, K. S., Lin, F., and Yang, L. T. (2012). Internet of things. *J. UCS*, 18:1069–1071.

Um modelo para simulação de formação de membranas e micelas

Nilzair Barreto Agostinho¹, Diana Francisca Adamatti¹

¹Centro de Ciências Computacionais - FURG)
Campus Carreiros: Av. Itália km 8 Bairro Carreiros

{nilzairmb, dianaada}@gmail.com

Abstract. This paper describes a model developed in NetLogo software, based on the membrane formation model, which simulates micellar formation with amphiphilic phospholipids in aqueous media with and without addition of salt (NaCl) molecules. This model shows how forces of attraction and repulsion between different types of molecules can result in the formation of structures called micelles and how the temperature change and addition of salt can cause changes in the micellar formation process. As a result of the simulations it can be observed that the addition of salt shake system, preventing the formation of micelles. It has also been observed that the temperature variation practically does not interfere in the system and the variation of the water-oil forces and "water-water" can prevent the formation of micelles or accelerate the formation process.

Resumo. Este artigo descreve um modelo desenvolvido no software NetLogo, baseado no modelo de formação de membrana, que simula a formação micelar com fosfolipídeos anfifílicos em meio aquoso com e sem adição de moléculas de sal (NaCl). Este modelo mostra como forças de atração e repulsão entre diferentes tipos de moléculas pode resultar na formação de estruturas denominadas micelas e como a mudança de temperatura e adição de sal pode causar mudanças no processo de formação micelar. Como resultados das simulações pode-se observar que a adição de sal agita o sistema, impedindo a formação das micelas. Observou-se também que a variação de temperatura praticamente não interfere no sistema e a variação das forças "water-oil" e "water-water" podem impedir a formação de micelas ou acelerar o processo de formação.

1 Introdução

A bicamada lipídica (ou bicamada de fosfolipídeos) é uma fina membrana polar composta por duas camadas de moléculas lipídicas. Estas membranas são folhas planas que formam uma barreira das células. As membranas celulares de quase todos os organismos vivos e muitos vírus são feitas de uma bicamada lipídica, assim como as membranas que cercam o núcleo da célula e outras estruturas sub-celulares [Andersen and Koeppe 2007].

As bicamadas biológicas são geralmente compostas por fosfolipídeos anfifílicos (moléculas possuidoras de regiões distintas e características como hidrofóbicas e hidrofílicas) que possuem uma cabeça de fosfato hidrófilo e uma cauda hidrofóbica constituída por duas cadeias de ácidos gordos. Os fosfolipídeos com certos grupos de cabeças

podem alterar a química superficial de uma bicamada e podem, por exemplo, servir como sinais e "âncoras" para outras moléculas nas membranas das células. Assim como as cabeças, as caudas de lipídios também podem afetar as propriedades da membrana, por exemplo, determinando a fase da bicamada [Andersen and Koepp 2007].

Muitas destas substâncias anfifílicas possuem a característica de modificar as interações interfaciais mediante a promoção dos fenômenos de absorção. Estas são conhecidas como agentes de superfície ou tensoativos. Assim, todos os tensoativos são compostos anfifílicos, mas nem todos compostos anfifílicos podem ser considerados tensoativos [Arteaga 2006].

Para que um composto possa ser considerado tensoativo é necessário que a molécula possua propriedades relativamente equilibradas, ou seja, que não seja demasiado hidrofílica nem demasiado hidrofóbica, portanto, que possua uma cadeia hidrofóbica de no mínimo 8 átomos de carbono e possuam uma polaridade mínima. Por outro lado estes compostos anfifílicos devem possuir a possibilidade de formarem agregados micelares para serem considerados tensoativos [Arteaga 2006].

Em geral, o termo tensoativo se refere a uma propriedade da substância, enquanto que os anfifílicos têm muitas outras propriedades e são qualificadas de acordo com as aplicações: sabão, detergente, dispersante, emulsionante, espumante, bactericida, inibidores de corrosão, antiestático e outros. No entanto, há certa tendência em confundir os dois termos. É importante destacar que os tensoativos estão entre os produtos mais usuais e utilizados na tecnologia química moderna [Arteaga 2006].

As membranas biológicas geralmente incluem vários tipos de moléculas diferentes dos fosfolipídios. Um exemplo particularmente importante nas células animais é o colesterol, que ajuda a fortalecer a bicamada e a diminuir sua permeabilidade. O colesterol também ajuda a regular a atividade de certas proteínas da membrana integral [Andersen and Koepp 2007]. Quando os fosfolipídeos são expostos à água, eles se automontam em uma folha de duas camadas com as caudas hidrofóbicas apontando para o centro da folha. Porém, se um lípido particular tiver um desvio muito grande de zero curvatura intrínseca, não formará uma bicamada e, em vez disso, formará outras fases, como micelas ou micelas invertidas [Andersen and Koepp 2007].

Pode-se verificar que modelos deterministas não proporcionam uma representação adequada dos componentes biológicos, químicos e físicos devido ao grande número de variáveis e fatores envolvidos no sistema em estudo. Modelos estocásticos, como as simulações baseadas em multi-agentes (MABS) que são ferramentas muito flexíveis que permitem a representação de cada indivíduo presente em um sistema, facilitam assim a avaliação de novas hipóteses no modelo. Normalmente, determinísticos apresentam como resultados o comportamento médio de seus componentes. Ao contrário, os modelos MABS podem modelar comportamento individual produzindo muito mais resultados detalhados [Werlang et al. 2014].

O modelo original simula a formação de membranas na água. Ele mostra como simples forças atrativas e repulsivas entre diferentes tipos de moléculas podem resultar em uma estrutura de nível superior. Este é composto pela representação de moléculas de água e fosfolipídeos, e simula a interação entre os mesmos utilizando os valores das forças já descritas anteriormente. Este trabalho propõe simular a formação de micelas

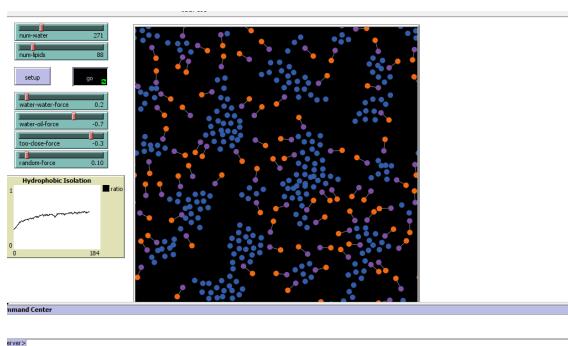


Figura 1. Modelo original de formação de membranas: As moléculas de água são representadas em azul e os lipídeos são representados em laranja

com fosfolipídeos. Esta simulação inclui forças de repulsão e atração, possibilidade do controle da temperatura, e possibilidade de adição de moléculas de sal.

O artigo contém uma sessão de introdução (1), a sessão de metodologia (2) onde é descrito o modelo original e quais métodos foram utilizados para desenvolver o modelo proposto. Logo após temos a sessão de resultados (3) e a conclusão (4).

2 Metodologia

O modelo utilizado como base para desenvolvimento deste trabalho simula a formação de membranas na água. Ele mostra como simples forças atrativas e repulsivas entre diferentes tipos de moléculas podem resultar em uma estrutura de nível superior. Este é composto pela representação de moléculas de água e fosfolipídeos, e simula a interação entre os mesmos utilizando os valores das forças já descritas anteriormente.

O presente trabalho foi baseado no modelo "Membrane Formation" do NetLogo e em conceitos físicos e bioquímicos do processo de formação de membranas e micelas [Bedau et al. 2005]. Os parâmetros que configuram as forças, porcentagens de estabilidade, distância e número de moléculas são calibrados e então a simulação pode ser iniciada. O modelo original está representado na figura 1 mostra como configurar o número de moléculas de água, número de moléculas de lipídeos, a intensidade da força "water-water-force" (Quanto uma molécula deve se mover quando está interagindo com outra molécula do mesmo tipo), a intensidade da força "water-oil-force" (Quanto uma molécula deve se mover quando está interagindo com uma molécula de um tipo diferente), intensidade da força "too-close-force" (Quanto uma molécula deve se mover quando está "muito perto" de outra molécula) e a intensidade da força "random-force" (Cada molécula mover-se-á em uma direção aleatória. Aumentar esse valor agita o sistema).

A simulação é parada se o número de lipídeos e água estáveis alcançou um percentual configurado. Uma molécula de lipídeo alcança a estabilidade se estiver há uma distância "radius" de uma molécula de água, bem como uma molécula de água encontra-se estável se estiver a um raio "radius" de outra molécula de água.

A adição de sal na solução pode impedir a formação micelar e a formação de membranas. Ocorre variações de acordo com o tipo e quantidade de sal adicionado e também de acordo com a quantidade de água presente na solução. Neste trabalho define-se o NaCl como sal que interage na solução. Considerando que 100ml de água dissolve 36g de sal

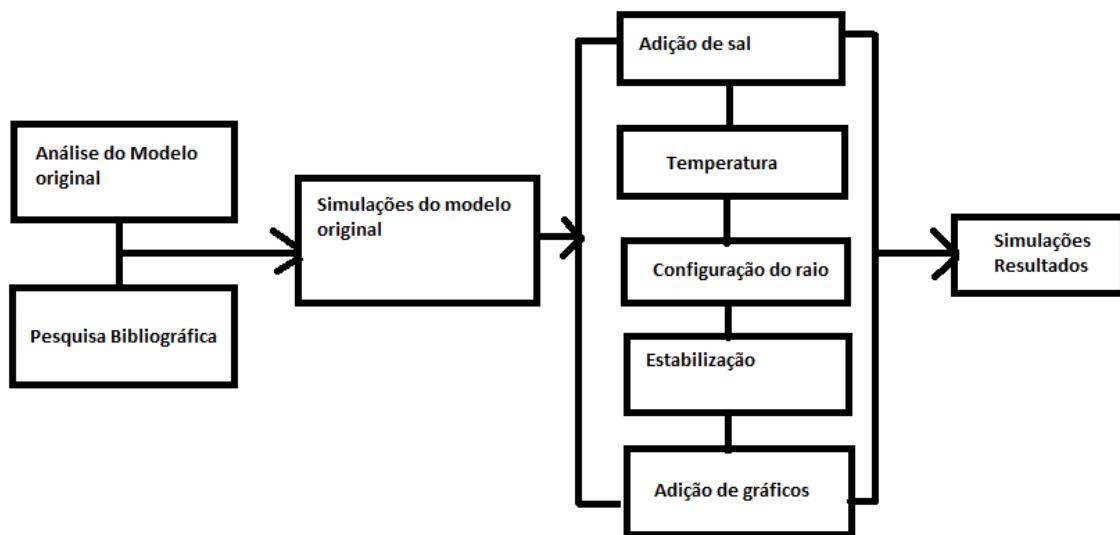


Figura 2. Metodologia: Para o desenvolvimento deste trabalho foram realizadas análises do modelo original e pesquisas bibliográficas, após foi desenvolvido o modelo de formação de micelas de acordo com a síntese indicada no terceiro bloco deste esquema e por fim foram realizadas simulações no modelo novo.

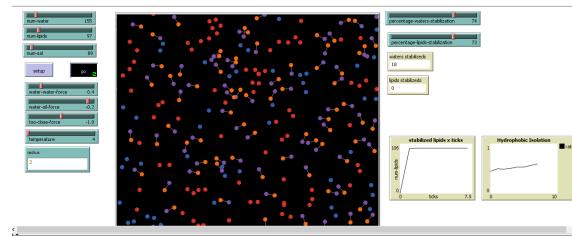


Figura 3. Modelo de formação de micelas

(NaCl) e que um mol qualquer contém $6,02 \times 1023$ moléculas, chega-se a que 100ml de água contém 33×1023 moléculas de água e que 36g de NaCl contém $3,7 \times 1023$ moléculas de água. Assim sendo, obtém-se a proporção 3.7/33. Esta proporção foi utilizada como limite de solubilidade e ponto de origem para interferência do sal na formação de micelas. O modelo de formação de micelas está representado na figura 3. O esquema representado na figura 2 mostra uma síntese de como foi estruturada a metodologia.

2.1 Características do simulador

- **Fosfolipídios:** São os principais constituintes das membranas biológicas. Seu caráter anfifílico é responsável pelo típico arranjo "bilayer" como base estrutural das membranas biológicas. Além do seu papel estrutural, alguns fosfolipídios têm uma importância funcional significativa [Schiller et al. 2007]. O "slider" numérico permite controlar o número de fosfolipídeos adicionados à simulação.
- **Moléculas de água:** São moléculas cuja composição química é de duas moléculas

de hidrogênio e de oxigênio formando um solvente que compõem parte do ambiente formador de moléculas e consequentemente também de micelas.

- **Sal:** A adição de sal em uma solução com micelas faz com que as mesmas assumam uma forma cilíndrica, o que faz com que elas tenham um movimento mais restrito, aumentando a viscosidade do sistema. Se muito sal for adicionado, ocorrerá a quebra do sistema, reduzindo drasticamente a viscosidade (o efeito do sal nas propriedades do xampú e sua relação com a estrutura micelar) . A agregação de monômeros (monômero é uma pequena molécula que pode ligar-se a outros monômeros formando moléculas maiores denominadas polímeros), em solução aquosa, é denominada de micelização. Este processo espontâneo e altamente cooperativo, por isso adequado para ser simulado em um ambiente multiagentes. A concentração na qual as micelas são formadas é chamada de concentração micelar crítica (CMC). Experimentalmente, a CMC é determinada pelo ponto de inflexão de um gráfico de alguma propriedade física da solução em função da concentração do surfatante. Uma ampla variedade de técnicas envolve medidas de tais propriedades físicas como condutividade elétrica, tensão superficial, fluorescência, viscosidade, espalhamento de luz, etc [Santos 2001]. Neste trabalho utiliza-se a temperatura e adição de sal como modificadores do CMC.
- **”Temperature”:** A variação da temperatura que pode influenciar na formação de agregados micelares, isto é, um aumento da temperatura favorece uma maior solubilidade do surfatante em água e isso faz aumentar a cmc [Santos 2001]. O ”slidertemperature” permite a seleção de um valor de temperature entre 0 e 100.
- **”water-water-force”:** Esta força representa o quanto uma molécula de água deve se mover quando está interagindo com outra molécula do mesmo tipo.
- **”water-oil-force”:** Este valor representa o quanto uma molécula deve se mover quando está interagindo com uma molécula de um tipo diferente.
- **”too-close-force”:** Já este valor indica o quanto uma molécula deve se mover quando está ”muito perto” de outra molécula.
- **”random-force”:** Ao selecionar um valor para este ”slider”, Cada molécula se moverá em uma direção aleatória, esta configuração agita o sistema.
- **”Radius”:** Este campo permite especificar o raio entre as moléculas.
- **”percentage-waters-stabilization”:** Esta opção especifica a percentagem de moléculas de água estabilizadas, ou seja, a porcentagem de moléculas agrupadas entre elas mesmo.
- **”percentage-lipids-stabilization”:** Esta opção especifica a percentagem de moléculas de lipídeos estabilizadas, ou seja, a porcentagem de moléculas de lipídeos que estão a uma raio ”radius” de pelo menos uma molécula de água.
- **”waters stabilizeds”:** Fornece o número de moléculas de água que alcançou a percentagem de estabilidade especificada.
- **”lipids stabilizeds”:** Fornece o número de moléculas de lipídeos que alcançou a percentagem de estabilidade especificada.

- **"stabilized lipids x ticks"**: Este gráfico informa o número de lipídeos estabilizados versus os "ticks"(tempo da simulação). Para efetuar a contagem dos lipídeos estabilizados é medido quantos lipídeos estão a um raio "radius"da molécula de água e após compara-se esse valor a percentagem definida no campo "number stabilized lipidis".
- **"Hydrophobic Isolation"**: O gráfico de isolamento hidrofóbico mostra a porcentagem média de vizinhos de cada molécula hidrofóbica que também são hidrofóbicos. Assim, quanto maior, mais moléculas hidrofóbicas são isoladas da água e moléculas hidrofílicas.

2.2 Formação micelar

Micelas não são estáticas, elas existem dentro de uma dinâmica de equilíbrio, simplesmente como um agregado dinâmico. Cada micela é composta por um certo número de moléculas de tensoativo, denominado como número de agregação, que rege geralmente o tamanho e a geometria do sistema micelar. O termo "micela normal"é utilizado para se referir a agregados de tensoativos em meio aquoso [Maniasso 2001]. Quando se adiciona à água pequenas quantidades de um tensoativo (substância de caráter anfifílico), uma parte é dissolvida como monômero e outra parte forma uma monocamada na interface ar-água. As moléculas da monocamada permanecem em equilíbrio com os monômeros que se formam na solução, e a cada concentração de monômero corresponde uma tensão superficial característica. Quando a concentração de monômero atinge um valor crítico que determina a saturação na interface, desencadeia-se o processo de formação espontânea de agregados moleculares (micelas) [Rossi et al. 2006].

Tensoativos são compostos anfifílicos, orgânicos ou organometálicos que formam coloides ou micelas em solução. Substâncias anfifílicas ou anfifílicas são moléculas possuidoras de regiões distintas e características como hidrofóbicas e hidrofílicas. Como nestas substâncias apenas a polaridade das diferentes regiões varia enormemente, as mesmas são também denominadas de moléculas anfipáticas, heteropolares ou polar não polares. Os tensoativos naturais incluem lipídeos simples, por exemplo, ésteres de ácido carboxílico), lipídeos complexos (esteres de ácidos graxos contendo fósforo, base nitrogenadas, e/ou açúcar) e ácidos bílicos tais como ácido cólico e deoxicólico [Maniasso 2001].

Tensoativos catiônicos, não iônicos e anfóteros, que são classificações de acordo com o grupo polar [Daltin 2011], quando empregados em quantidades acima da concentração micelar crítica e aquecidos a uma determinada temperatura, podem separar-se em duas fases isotrópicas, fenômeno este denominado "cloud point". A definição de "cloud point"está sujeita a várias interpretações muito semelhantes, que se completam. É a separação de duas fases, que parece estar associada à existência de micelas compostas por moléculas gigantes na solução. Em soluções aquosas, o emprego de tensoativos não iônicos e anfóteros formam uma fase complexa e particularmente propensa a separar- se sob uma determinada temperatura. Certos tensoativos anfóteros e alguns não iônicos em presença de altas concentrações de eletrólitos (soluções salinas) podem também apresentar separação de fase [Maniasso 2001].

Devido ao fato de a temperatura adequada depender de muitas propriedades, inclusive do tipo de tensoativos, foi criado um campo chamado "temperature"que pode ser

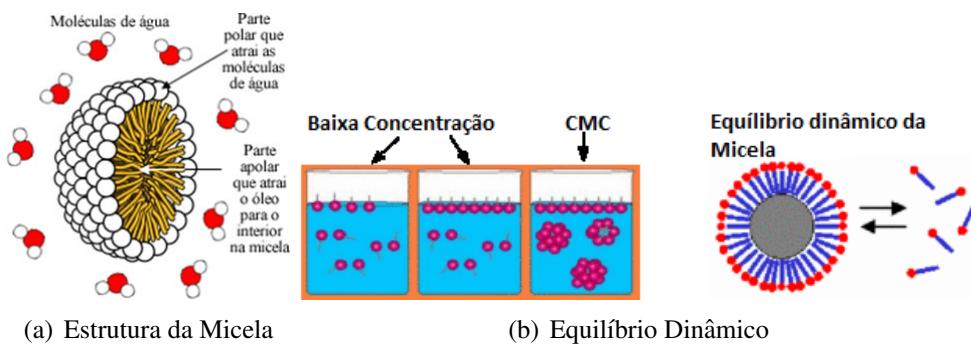


Figura 4. Estrutura e equilíbrio micelar

alterado pelo usuário. A princípio, devido a essa grande possibilidade de variação da temperatura, definiu-se que o fator temperatura só influencia na simulação a partir do valor 40, podendo esta definição ser modificada em trabalhos futuros de acordo com a obtenção de informações de um especialista da área bioquímica. Na figura 4(a), mostra-se a estrutura esférica de um agregado micelar. Já a figura 4(b) mostra como fica a estrutura e o arranjo das moléculas ao encontrar o equilíbrio dinâmico.

3 Resultados

Nesta seção são descritos os resultados obtidos após serem realizadas simulações com diferentes configurações. Foram realizadas simulações com a adição e ausência de sal. Para cada simulação foram estabelecidos parâmetros diferentes incluindo a variação da temperatura, "water-water-force", "too-close-force" e "water-oil-force". Nos itens posteriores serão apresentados os resultados através de gráficos que contém o valor da isolação hidrofóbica (mostra porcentagem média de cada "vizinho" da molécula hidrofóbica que também é hidrofóbico, sendo vizinhos moléculas que encontram-se a uma distância "radius") versus número de "ticks". As configurações referentes às simulações realizadas permanecem iguais exceto por somente um parâmetro variável a cada simulação. Os valores dos parâmetros variáveis são descritos no decorrer desta sessão. A tabela 1 é tomada como padrão para os valores dos parâmetros.

Tabela 1. Variação do valor "too-close-force"

Parâmetro	S1 - fig. 5(a)	S2 - fig. 5(b)	S3 - fig. 5(c)
num-water	155	155	155
num-lipids	97	97	97
num-sal	0	0	0
water-water-force	0.2	0.2	0.2
water-oil-force	-0.2	-0.2	-0.2
too-close-force	-2	-0.2	0
temperature	4	4	4
radius	2	2	2
percentage-waters-stabilization	75	75	75
percentage-lipids-stabilization	75	75	75

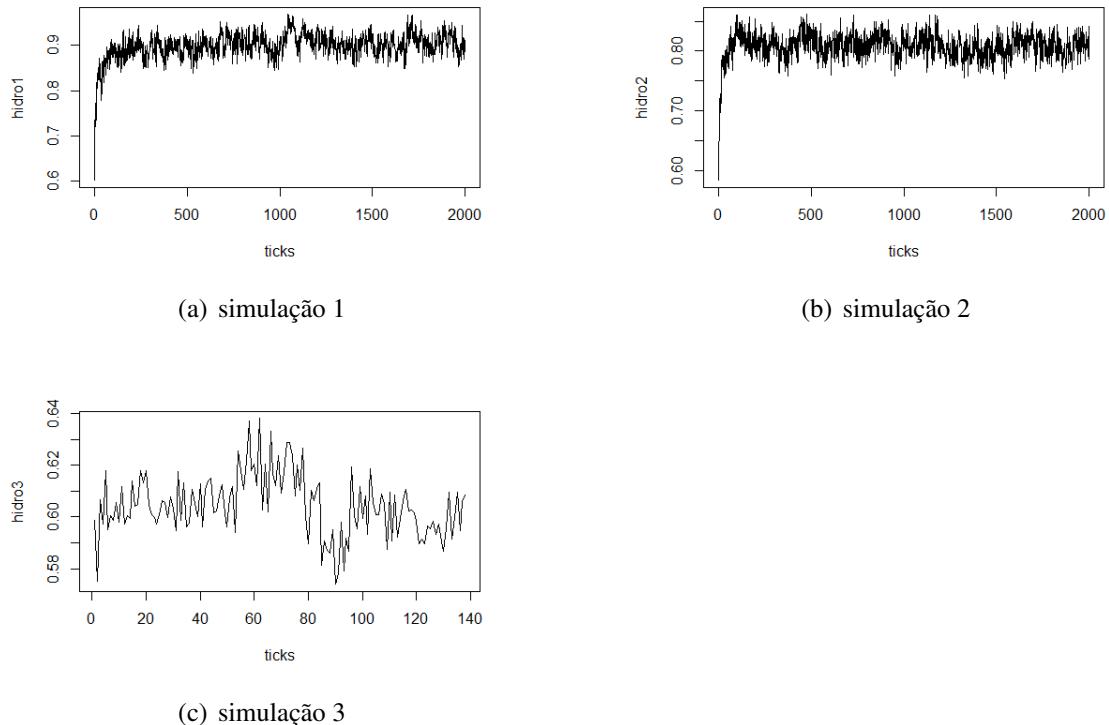


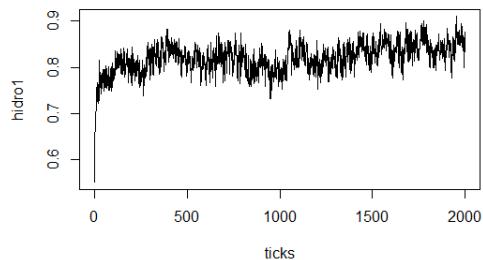
Figura 5. Variação do too-close-force

Os gráficos contidos na figura 5 mostram o resultado das simulações realizadas variando a força "too-close-force". Na figura 5(a) observa-se que mesmo ao chegar em 2000 ticks, o valor da isolação hidrofóbica não estabilizou, indicando que ao configurar o valor -2" para "too-close-force" as moléculas ficam em grande movimentação permitindo uma pequena quantidade de formação de micelas. Na figura 5(b) pode-se ver o resultado da simulação com as mesmas configurações da anterior, porém com o valor -0.2" para "too-close-force". Já na figura 5(c), visualiza-se a estabilização da isolação hidrofóbica em 140 ticks.

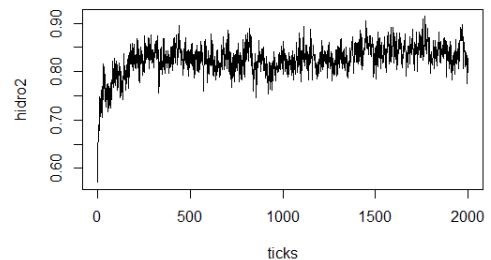
Nas simulações em que ocorre a variação da temperatura e adição de 20 moléculas de sal, utilizou-se os valores da tabela 1 e adicionou-se 20 moléculas de água e a temperatura teve seus valores definidos em 4,11,40 nas respectivas simulações representadas nas figuras 6(a), 6(b) e 6(c). Já na simulação representada na figura 7(a) adiciona-se 60 moléculas de sal e não é variada a temperatura, os demais parâmetros permanecem com o valor padrão definido inicialmente.

As simulações representadas nas figuras 8(a), 8(b) e 8(c) foram removidas as moléculas de sal e os valores da temperatura foram definidos em 4, 11 e 40 respectivamente.

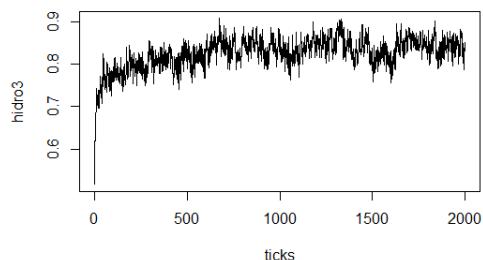
Nos resultados mostrados na figura 6 percebe-se que mesmo ao variar a temperatura, a isolação hidrofóbica não alcança a estabilidade devido a adição de moléculas de sal. Na figura 7 pode-se visualizar que a isolação hidrofóbica não estabiliza devido a adição de sal. Percebe-se que na figura 8 que sem adição de sal e com as configurações



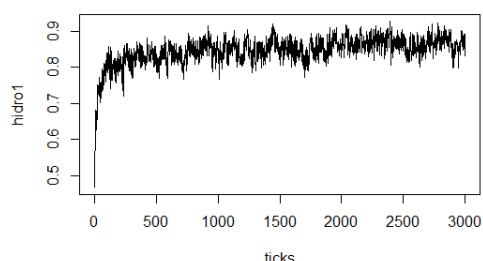
(a) simulação 1



(b) simulação 2



(c) simulação 3

Figura 6. Variação da temperatura com adição de 20 moléculas de sal

(a) simulação 1

Figura 7. Adição de 60 moléculas de sal

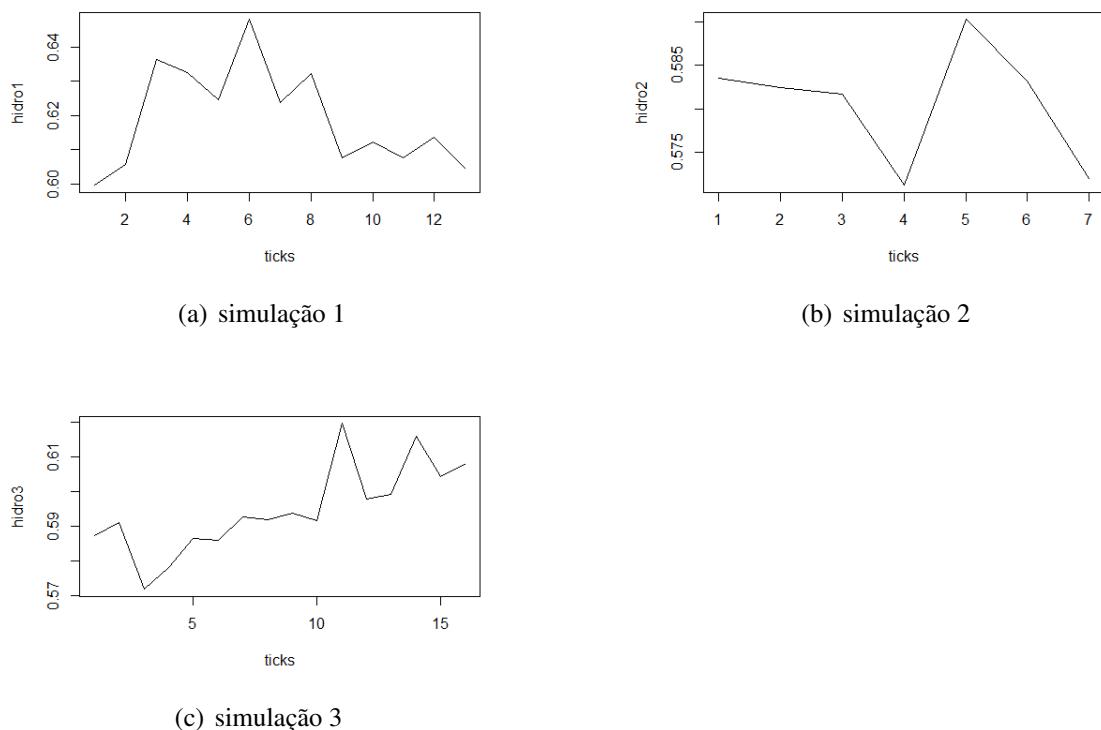


Figura 8. Sem adição de sal e variando a temperatura

descritas nessa simulação alcança-se a isolação hidrofóbica rapidamente, já que é o sal que agita o sistema.

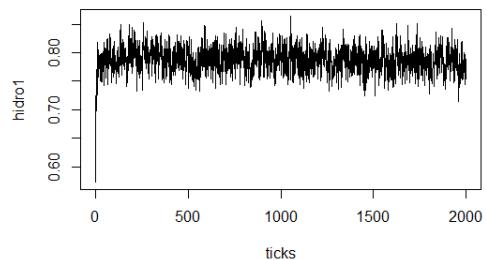
Os resultados representados nas figuras 9(a), 9(b) e 9(c) utilizam os valores padrão exceto pelo valor de "water-oil-force" que é definido como -2, -0.2 e 0 respectivamente. E nas figuras 10(a), 10(b) e 10(c) temos os resultados das simulações realizadas com os parâmetros padrão exceto a variação do valor de "water-water-force" entre 0, 0.2 e 2.

Como resultados da variação da força "water-oil", pode-se verificar na figura 9 que ao ser utilizado o valor -2" não é alcançada a estabilidade da isolação hidrofóbica, porém com os valores -0.2" e "0.2" alcança-se a estabilidade rapidamente. Nos resultados mostrados na figura 10 verifica-se que quanto maior o valor da força "water-water", mais rapidamente se alcança a estabilidade da isolação hidrofóbica.

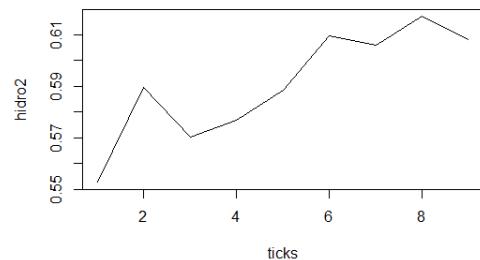
4 Conclusão

Ao analisar os resultados descritos, pode-se verificar que a adição de sal agita o sistema, impedindo a formação das micelas, pois nas figuras 6(a), 6(b) e 6(c) percebe-se que mesmo ao variar a temperatura, o sistema não alcança a estabilidade da isolação hidrofóbica. Ao observar os resultados contidos na figura 6 verifica-se que a isolação hidrofóbica também não é alcançada devido a adição de sal.

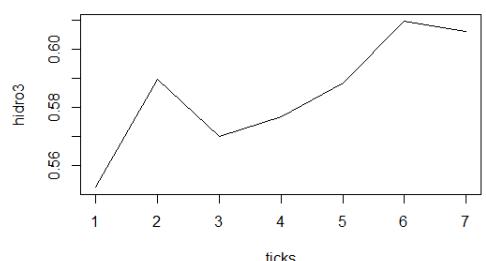
A figura 7 mostra resultados que corroboram com os resultados já analisados e discutidos anteriormente, pois com ou sem a variação da temperatura, a adição de sal mostrou-se como um fator preponderante para impedir a formação de micelas. Nos



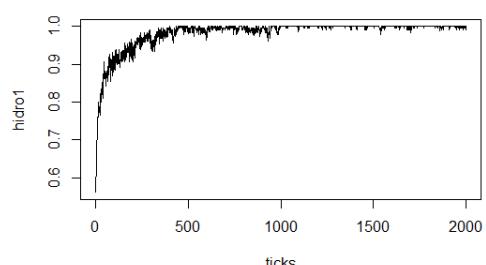
(a) simulação 1



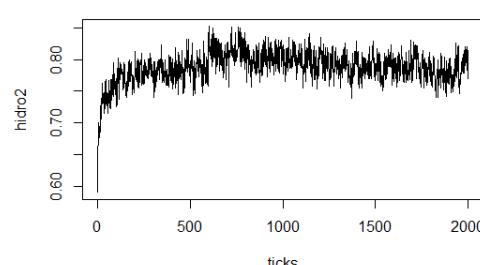
(b) simulação 1



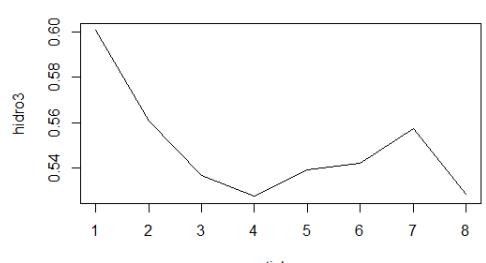
(c) simulação 3

Figura 9. Variação da força water-oil

(a) simulação 1



(b) simulação 2



(c) simulação 3

Figura 10. Variação da força water-water

gráficos contidos na figura 8 os resultados mostram que a variação da temperatura pouco influencia na formação das micelas quando não há adição de sal.

Finalmente ao analisar os resultados contidos na figura 9 e 10 é verificado que a variação das forças de interação "water-oil" e "water-water" interfere no processo de formação de micelas e identifica-se que quanto menor a força de interação "water-oil" menos estabilidade de isolação hidrofóbica o sistema alcança, bem como que quanto menor o valor da força "water-water" menor a estabilidade da isolação da hidrofóbica.

Portanto, os resultados apresentaram-se satisfatórios, pois ao simular a formação de micelas com fosfolipídeos obteve-se informações que são de grande relevância para corroboração do processo de configuração dos sistemas formadores de micelas encontrados em muitos produtos e processos de suma importância para sociedade. Verificou-se que é relevante realizar como trabalhos futuros a introdução de outros tipos de moléculas que interagem no processo de formação de micelas, assim como permitir a configuração das forças de interação num nível mais aprofundado quimicamente.

Referências

- Andersen, O. and Koeppen, R. E. (2007). Bilayer thickness and membrane protein function: An energetic perspective. *Annual Review of Biophysics and Biomolecular Structure*, 36:107–130.
- Arteaga, A. F. (2006). *Preparación, caracterización y estabilidad de emulsiones y microemulsiones O/W*. PhD thesis, Universidad de Granada - Facultad de Ciencias - Departamento de Ingeniería Química.
- Bedau, M., Buchanan, A., Gazzola, G., Hanczyc, M. M., Maeke, T., McCaskill, J. S., Poli, I., and Packard, N. H. (2005). Evolutionary design of a ddpd model of ligation. *Proceedings of the 7th International Conference on Artificial Evolution EA05. Lecture Notes in Computer Science*, 3871:201–212.
- Dalton, D. (2011). *Tensoativos: química, propriedades e aplicações*. Blucher.
- Maniasso, N. (2001). Ambientes micelares em química analítica. *Quim. Nova*, 24:87–93.
- Rossi, C., Dantas, T., Neto, A., and Maciel, M. (2006). Tensoativos: uma abordagem básica e perspectivas para aplicabilidade industrial. *Revista Universidade Rural, Série Ciências Exatas e da Terra, Seropédica, RJ: EDUR*, 25:73–85.
- Santos, S. (2001). *Hidrólise ácida de aceitais em misturas formadas por albumina do soro bovino e dodecilsulfato de sódio (sds) e polioxietileno e sds: medidas dos parâmetros de ligação por tensão superficial, condutividade elétrica e espalhamento de raio-x a baixo ângulo*. PhD thesis, Universidade Federal de Santa Catarina- UFSC.
- Schiller, J., Muller, M., Fuchs, B., Arnold, K., and Huster, D. (2007). ³¹p nmr spectroscopy of phospholipids: From micelles to membranes. *Current Analytical Chemistry*, 3.
- Werlang, P., Fagundes, M., Adamatti, D., Machado, K. S., Groll, A. V., da Silva, P., and Werhli, A. V. (2014). Multi-agent-based simulation of mycobacterium tuberculosis growth. *MBAS 2013*, 8325:131–142.

Um Agente Autônomo Concorrente para o Manipulador Robótico Jaco Kinova

Edi Moreira M. de Araujo¹, Augusto Loureiro da Costa²

¹Programa de Pós-graduação em Engenharia Elétrica
Universidade Federal da Bahia (UFBA)
Salvador – BA – Brazil

edimoreira13@yahoo.com.br, augusto.loureiro@ufba.br

Abstract. This work presents the use of the Concurrent Autonomous Agent (CAA) in an JACO Kinova robot arm, enabling it to perform complex tasks in a completely autonomous way. The communication between the CAA and the manipulator will be made through the ROS (Robot Operating System), as well as the performance of the behaviors present in the reactive level. The Concurrent Autonomous Agent is an implementation of a cognitive agent architecture based on the Generic Cognitive Model for Autonomous Agents. This model over the years proved to be very effective, initially being used for the implementation of a distributed control system for multi-robot systems, called Mecateam, obtaining significant results in RoboCup's Latin America and Brazilians. The CAA has already been implemented in several successful applications, such as the NAO humanoid robot and the AxeBot omnidirectional robot. These results point to CAA as a model of well-known cognition for the training of robots to perform tasks that require a certain degree of cognition.

Resumo. Este trabalho tem como objetivo apresentar a implementação do Agente Autônomo Concorrente (AAC) para o manipulador robótico JACO Kinova, habilitando-o à execução de tarefas complexas de uma maneira completamente autônoma. A comunicação entre o AAC e o manipulador foi feita através do ROS (Robot Operating System), bem como a realização dos comportamentos presentes no nível reativo. O Agente Autônomo Concorrente é a implementação de uma arquitetura de agente cognitivo baseada no modelo cognitivo genérico para agentes autônomos. Este modelo ao longo dos anos mostrou-se bastante eficaz, sendo inicialmente utilizado para a implementação de um sistema de controle distribuído para sistemas multi-robôs, chamado Mecateam, obtendo resultados significantes em RoboCup's Latin América e Brasileiras. O AAC já foi implementado em diversas aplicações com êxito, como exemplo, o robô humanoide NAO e o robô omnidirecional AxéBot. Tais resultados apontam o AAC como um modelo de cognição bem indicado para a capacitação de robôs a execução de tarefas que solicitam um alto grau de cognição.

1. Introdução

Manipuladores robóticos podem ser utilizados em diversas aplicações, porém muitos desses manipuladores estão limitados a movimentos sequenciais predefinidos ou necessitam de controle humano, isto é, não possuem a capacidade de identificar mudanças no ambiente ou de tomar decisões de uma maneira autônoma. O grau de autonomia presente nos

robôs está associado com a tarefa que o mesmo venha a executar. Em algumas linhas de montagem industrial, os comportamentos dos robôs podem ser supervisionados por uma máquina de estados discretos, que determina exatamente quais ações estão disponíveis em um dado estado, porém, em tarefas mais complexas, especialmente aquelas que exigem planejamento, o mecanismo de tomada de decisão deve ser mais robusto, de maneira que o robô venha adquirir a capacidade de realizar estas tarefas em um tempo hábil, com um alto grau de precisão e de uma forma completamente autônoma. De forma a lograr desta completa autonomia, estas máquinas devem ser dotadas de algum tipo de inteligência, ou seja, devem ser capazes de tomar decisões por conta própria. Para atender a estas necessidades, este trabalho propõe a implementação de um sistema inteligente, baseado em agentes cognitivos, no manipulador robótico JACO Kinova, capacitando-o desta forma a realização de tarefas complexas. A tarefa escolhida para realização de experimentos, foi a de habilitar o manipulador a montar uma Torre de Hanói com três discos (Figura 8).

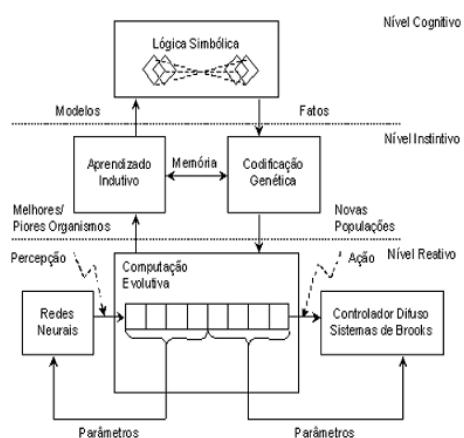


Figura 1. Modelo genérico para agentes cognitivos.

Em [Bittencourt 1997] um modelo genérico de agentes cognitivos é proposto (Figura 1). Este modelo consiste em uma arquitetura cognitiva utilizada para modelar agentes de qualquer natureza. O modelo genérico de agentes cognitivos é utilizado como base em [Costa and Bittencourt 1999] e [Bittencourt and Costa 2001] para a proposta de uma arquitetura de agentes autônomos inteligentes chamada de Agente Autônomo Concorrente (AAC). Este agente utiliza uma arquitetura de três camadas: cognitiva, instintiva e reativa (Figura 2). Cada uma delas é implementada como um processo e representa um nível decisório distinto que complementa as demais para a construção de um agente cognitivo. A complexidade do comportamento do agente é incrementada a cada camada. Resultados experimentais, utilizando o AAC, foram realizados com êxito como podemos verificar em [Costa et al. 2011], onde o mesmo teve o nível reativo embarcado no robô omnidirecional AxéBot. Neste caso o nível reativo teve como função rastrear as trajetórias geradas pelo nível instintivo do agente. Já em [Costa et al. 2015] o nível reativo foi implementado no modulo *Unifield Humanoid Robotics Software Platform* (UHRSP) do robô Humanoide NAO. O UHRSP disponibiliza uma plataforma de controle de caminhada chamado de *Zero Moment Point* (ZMP), que recebe do nível instintivo do AAC qual o comportamento será executado pelo controlador ZMP. Em [Ferreira 2014] uma rede de três microcontroladores foi projetada para embarcar o AAC no robô omnidirecional AxéBot. A rede foi concebida mimetizando a estrutura funcional do AAC. Os três níveis deste

último (a saber, o nível reativo, o nível instintivo e o nível cognitivo) foram embarcados em cada nó da rede.

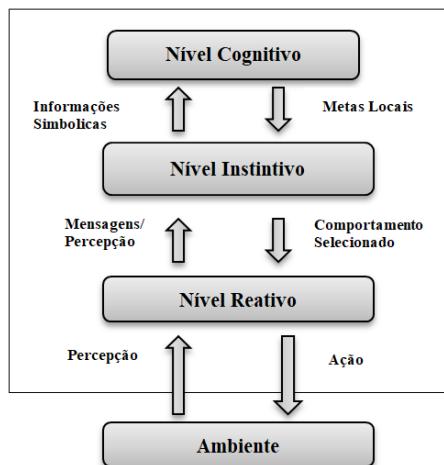


Figura 2. Fluxo de informação do Agente Autônomo Concorrente.

Com tais resultados, podemos chegar a conclusão de que o AAC pode ser implementado em qualquer sistema robótico, modificando-se apenas sua camada reativa, levando em conta a característica de hardware de cada robô. Faz-se também necessário a criação de bases de regras, para os níveis cognitivo e instintivo.

Neste presente trabalho, o nível reativo foi implementado utilizando o ROS. O ROS possui um conjunto de ferramentas e bibliotecas que permite ter o controle do manipulador JACO (Figura 14) Kinova tanto no espaço cartesiano, quanto no espaço de juntas.

Os principais objetivos e contribuições deste trabalho são:

- Implementação da camada reativa do AAC utilizando o ROS, algo nunca utilizado anteriormente e de extrema importância no auxílio a futuras implementações do AAC;
- Capacitação do manipulador robótico JACO Kinova a execução de tarefas complexas;
- Implementação do AAC em uma nova categoria de robôs (manipuladores robóticos).

O presente artigo está estruturado da seguinte forma: na seção 2 são apresentados os principais conceitos acerca do JACO Kinova e ROS. Em seguida, na seção 3 é apresentado o AAC; os experimentos e os resultados são apresentados na seção 4 e as conclusões e perspectivas futuras delineadas na seção 5.

2. JACO Kinova e ROS

2.1. Manipulador JACO Kinova

O robô JACO é um manipulador criado pela empresa canadense *Kinova Robotics*. Ele é dotado de seis juntas rotacionais, 6 elos (*links*) e um efetuador com 3 dedos (*fingers*). As seis juntas são movidas individualmente por seis motores *brushless DC*, onde dois tipos de módulos motor são usados. Nas juntas 1-3, onde a junta 1 é a mais próxima da base,

usa motores com torque de até 1 Nm, enquanto as articulações 4-6 usam pequenos motores com torque de no máximo 0.5 Nm. As seis juntas possuem *ranges* distintos, sendo que as juntas 1, 4, 5 e 6 possuem um *range* de [-10000°, 10000°] e as juntas 2 e 3 possuem o *range* de [42°, 318°] e [17°, 343°] respectivamente. Os elos são constituídos de fibra de carbono que abrigam os módulos motor. Cada dedo do efetuador possui um motor individual, totalizando assim 9 motores. A base do JACO abriga um processador de sinal digital (DSP) que envia informações para cada motor com base na entrada do usuário. Dispositivos que possuem o poder de comando são conectados a base do manipulador através de uma porta USB, configuração essa reconhecida pelo sistema de controle.

2.2. ROS

O *Robotic Operationg System* (ROS) é um software de código aberto, desenvolvido em 2007, que disponibiliza bibliotecas e ferramentas para ajudar desenvolvedores de programas robóticos a criarem suas aplicações, permitindo que diferentes grupos de pesquisa possam trocar informações e experiências, reutilizando códigos e ferramentas criadas com o seu auxílio [Barros 2014]. Ele fornece diversos serviços, incluindo abstração de hardware, implementação de bibliotecas, controle de baixo nível de dispositivos (*drivers*), gerenciamento de pacotes e troca de mensagens entre processos [Oliveira et al. 2013]. A empresa Kinova disponibiliza um conjunto de códigos e bibliotecas que proporcionam a comunicação e controle do JACO através do ROS [Kinova 2017]. Com isso, torna-se possível ter o controle tanto no espaço de juntas como o controle no espaço cartesiano do manipulador.

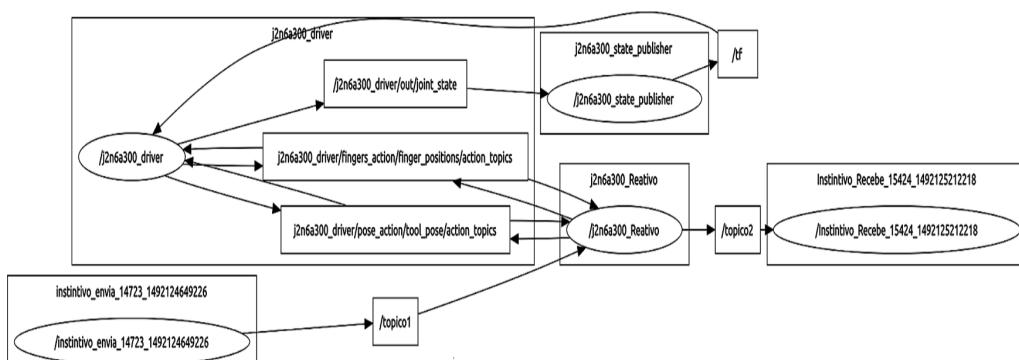


Figura 3. Fluxograma grafo do ROS (nível reativo).

Na Figura 3 podemos verificar o grafo de execução do ROS. Cada bloco representa um "nó" em execução. "Nós" são processos que executam instruções, geralmente utilizando bibliotecas do ROS. Cada nó comunica-se uns com os outros através de *streaming* de tópicos, serviços via RPC (*Remote Procedure Call*) e o servidor de parâmetros.

3. Agente Autônomo Concorrente

A arquitetura do AAC foi inspirada no modelo genérico para agentes cognitivos proposto em [Bittencourt 1997]. A ideia principal desta arquitetura era implementar percepção, ação, comunicação, cooperação, planejamento e tomada de decisão, utilizando a programação concorrente. Neste modelo concorrente, a arquitetura de tomada de decisão

era completamente centralizada, ocasionando alguns problemas de sincronismo entre o agente e o ambiente [Costa and Bittencourt 1998].

Com o intuito de solucionar esses problemas de sincronização, a arquitetura do agente cognitivo concorrente migrou de uma abordagem centralizada, para uma arquitetura descentralizada, através da sua implementação no *framework* Expert-Coop++ [Costa and Bittencourt 1999]. Esse modelo baseia-se na hipótese que as atividades cognitivas possuem três características principais: auto-organização, natureza evolutiva e dependência [Costa et al. 2011]. De acordo com esse modelo, um agente cognitivo é composto por três níveis decisórios: nível reativo, instintivo e cognitivo (Figura 2). Cada um desses níveis implementa um processo decisório distinto, complementando os demais níveis para a construção de um agente cognitivo.

Um componente importante presente em todos os níveis decisórios do AAC, o *mailbox*, tem um papel fundamental no funcionamento do agente. O *mailbox* consiste de um objeto instanciado em cada processo do Expert-Coop++ que oferece uma interface de comunicação baseada em soquetes UNIX e uma estrutura de dados que funciona como um *buffer*, armazenando as mensagens em uma fila de tamanho finito. Quando uma mensagem é lida, a mesma é removida da fila [Ferreira 2014].

3.1. Nível Cognitivo

A função do nível Cognitivo (Figura 4) é de possuir um modelo simbólico do ambiente, um conjunto de planos, uma base de conhecimento codificado em regras de produção com a qual é possível estabelecer metas e escolher o plano mais adequado para alcançá-las. Estes planos são executados pelo nível instintivo. O nível cognitivo recebe informações simbólicas do nível instintivo e as mensagens dos outros agentes (quando inserido em um sistema multiagente). Um importante aspecto do AAC é que enquanto os níveis instintivo e reativo trabalham no alcance de uma meta local, o nível cognitivo pode, concomitantemente, se dedicar a tarefas de planejamento e criação de metas [Ferreira 2014].

Nos experimentos realizados na seção 4, o nível cognitivo possui uma representação do ambiente $E(k)$, uma base de regras de produção Bi contendo um total de 25 regras (Figura 5), a qual contém o conhecimento necessário para associar ao estado corrente do ambiente $E(k)$ e escolher o plano a ser executado pelo nível instintivo. Um motor de inferência, presente no nível cognitivo, de um único ciclo, utiliza o conhecimento armazenado em Bi , para escolher o plano a ser executado pelo nível instintivo.

3.2. Nível Instintivo

O nível instintivo (Figura 4) é o responsável pelo reconhecimento das mudanças do estado do ambiente, atualização das informações simbólicas utilizadas na base de conhecimento do nível cognitivo e pela escolha do comportamento a ser utilizado pelo nível reativo. O reconhecimento das mudanças do estado do ambiente é realizado após cada ciclo de percepção-ação executado pelo nível reativo. Este nível executa planos, que se bem sucedidos, levam a satisfação de metas locais, criadas pelo nível cognitivo.

O arquivo de regras, implementado pelo usuário, fornece as regras necessárias para classificar os estados do ambiente, escolher o comportamento mais adequado a este estado e gerar informações a ser enviadas ao nível cognitivo. Lógica e quadros são os formalismos admitidos para representação de conhecimento do agente. O nível instintivo

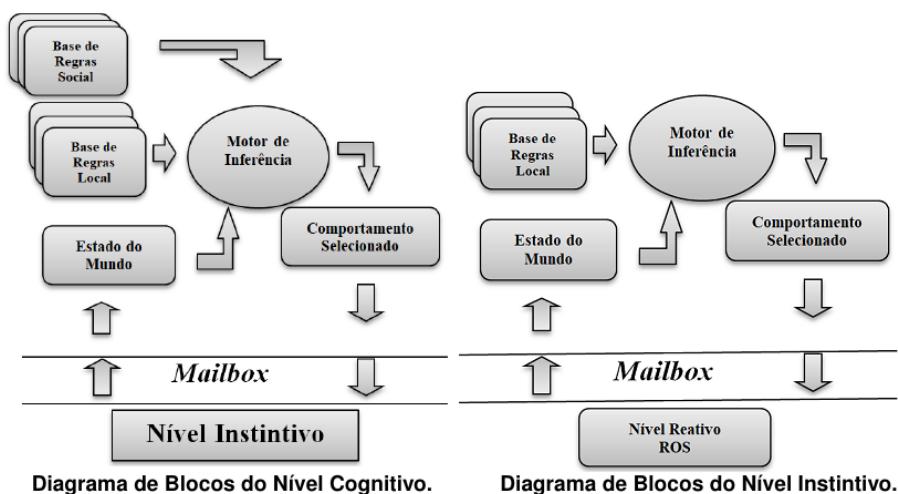


Figura 4. Diagrama de Blocos dos Níveis Cognitivo e Instintivo.

(Figura 4) utilizado neste trabalho tem como principal função executar o plano escolhido pelo nível cognitivo, reconhecer as mudanças do estado do ambiente e escolher o comportamento que será executado pelo nível reativo implementado no ROS. O reconhecimento das mudanças do estado do ambiente é realizado após cada ciclo de percepção-ação executado pelo nível reativo. Foram criados três arquivos de regras, sendo cada um destes arquivos um plano a ser escolhido pelo nível cognitivo.

Um Sistema Baseado em Conhecimento (SBC), composto por um Motor de Inferência (MI), Base de Regras (BR) e uma Base de Fatos (Estado do Mundo) são implementados neste nível.

3.3. Representação de conhecimento do AAC

Segundo Ferreira (2014), o AAC pode utilizar a Lógica de Primeira Ordem ou quadros como método de representação de conhecimento, sendo o LPO o método escolhido de representação neste trabalho. Nas Figuras 5 e 6, temos o exemplo de algumas regras implementadas nos níveis cognitivo e instintivo utilizando o conceito de objeto, atributo e valor. As representações do estado do ambiente são identificados pelo token “if”, ou seja, condições necessárias para a execução dos termo identificado pelo token “then”. Todas as regras possuem uma estrutura semelhante as regras apresentadas nas Figuras 5 e 6, possuindo três condições necessárias e uma consequência para estas condições.

Como podemos observar na Figura 5, temos exemplos de três regras implementadas no nível cognitivo, sendo todas elas responsáveis por escolher o plano a ser executado no nível instintivo. Os termos delimitados pelo token “then”, isto é, plano escolhido para ser executado pelo nível instintivo tem como estrutura: o termo “robo”, o termo “acao” e por último o plano escolhido dentre as três opções possíveis (plano_1, plano_2 ou plano_3).

A Figura 5 mostra o exemplo de uma regra implementada no nível instintivo. As condições delimitadas pelo token “if” são: “disco_menor (amarelo)”, “disco_medio (verde)” e “disco_maior (vermelho)” como os “objetos”, o termo “posicao” como o estado e os termos “p1n3”, “p1n2” e “p1n1” como os atributos. Estes atributos determinam a localização dos discos na Torre de Hanói, conforme a Tabela 1. Os termos delimitados

```

(regra_21
  (if  (logic ( disco_menor posicao p3n2 ))
        (logic ( disco_medio posicao p3n1 ))
        (logic ( disco_maior posicao p2n1 ))))
  (then (logic ( robot acao plano_3 ))))

(regra_7
  (if  (logic ( disco_menor posicao p1n1 ))
        (logic ( disco_medio posicao p3n2 ))
        (logic ( disco_maior posicao p3n1 ))))
  (then (logic ( robot acao plano_1 ))))

(regra_8
  (if  (logic ( disco_menor posicao p2n3 ))
        (logic ( disco_medio posicao p2n2 ))
        (logic ( disco_maior posicao p2n1 ))))
  (then (logic ( robot acao plano_2 ))))

```

Figura 5. Exemplo de três regras implementadas no nível cognitivo.

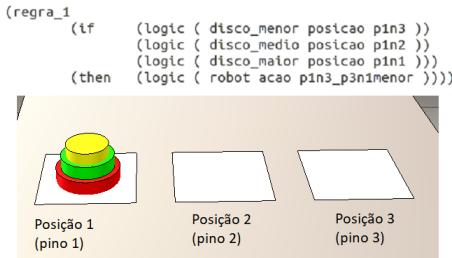


Figura 6. Exemplo de uma regra implementada no nível instintivo com o modelo do mundo.

pelo *token* “then”, isto é, o comportamento escolhido para ser executado pelo nível reativo tem como estrutura: o termo “robot”, o termo “acao” e o termo “p1n3_p3n1”, informa que o manipulador tem que pegar o disco na posição “p1n3” e leva-lo a posição “p3n1”, seguindo fluxograma de ações conforme a Figura 7

Tabela 1. Posições dos discos da Torre de Hanói utilizadas nos experimentos em relação a base do manipulador.

Localizações possíveis	Localização no espaço cartesiano	Localização e orientação desejada do manipulador em metros (x, y, z, α , β , γ)
p1n1	x = -0.5 , y = 0 e z = 0.0150	(-0.5 , 0, 0.670, -180, 0, 0)
p1n2	x = -0.5 , y = 0 e z = 0.0450	(-0.5 , 0 , 0.710, -180, 0, 0)
p1n3	x = -0.5 , y = 0 e z = 0.0750	(-0.5, 0, 0.730, -180, 0, 0)
p2n1	x = -0.5 , y = -0.25 e z = 0.0150	(-0.5 , -0.25, 0.670, -180, 0, 0)
p2n2	x = -0.5 , y = -0.25 e z = 0.0450	(-0.5 ,-0.25 , 0.710, -180, 0, 0)
p2n3	x = -0.5 , y = -0.25 e z = 0.0750	(-0.5, -0.25, 0.730, -180, 0, 0)
p3n1	x = -0.5 , y = -0.5 e z = 0.0150	(-0.5 , -0.5, 0.670, -180, 0, 0)
p3n2	x = -0.5 , y = -0.5 e z = 0.0450	(-0.5 ,-0.5 , 0.710, -180, 0, 0)
p3n3	x = -0.5 , y = -0.5 e z = 0.0750	(-0.5, -0.5, 0.730, -180, 0, 0)

Fonte: Criado pelo próprio autor.

3.4. Nível Reativo

O nível reativo é o responsável pela interação do agente com o ambiente. Ele recebe a percepção do ambiente e determina as ações que serão tomadas sobre ele. Tais ações são determinadas por um conjunto de comportamentos que serão implementados no nível

reativo do agente. A cada ciclo de ação-percepção apenas um comportamento está ativo. Os comportamentos podem ser entendidos como um conjunto de ações que devem ser executadas afim de alcançar determinados objetivos.

Neste trabalho o nível reativo foi implementado no ROS, tendo como objetivo realizar os comportamentos escolhidos e por enviar as mensagens que contém a nova configuração da Torre de Hanói (novo estado do mundo) ao nível instintivo, após cada comportamento realizado.

Para a implementação do nível reativo, criou-se um total de 3 arquivos executáveis (*Nós*) no ROS: */J2n6a300_Reativo*, */Instintivo_envia* e */Instintivo_recebe* (Figura 3). O nó */J2n6a300_Reativo*, contém os 25 comportamentos criados, sendo ele o responsável por realizar a comunicação com o nó */J2n6a300_Driver* através dos tópicos *J2n6a300_driver/pose_action/tool_pose/action_topics* ou *J2n6a300_driver/pose_action/finger_position/action_topics*. Os nós */Instintivo_envia* e */Instintivo_recebe*, são responsáveis por enviar a mensagem correspondente ao comportamento que será executado no nível reativo e por receber do nível reativo as modificações do estado do mundo $E(k)$. Estes blocos trocam mensagens com o *mailbox* presente no nível instintivo do AAC através de *sockets UDP*.

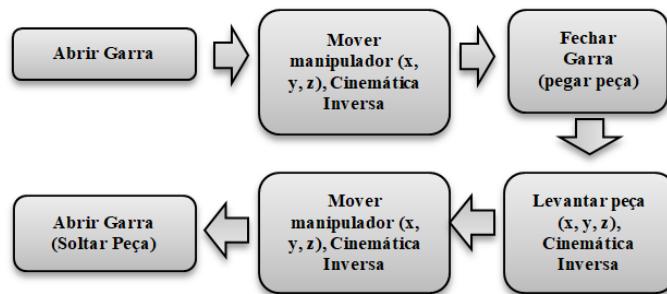


Figura 7. Fluxograma de ações de cada comportamento.

O Nó */Instintivo_envia* (Figura 3) recebe do nível instintivo a mensagem contendo o nome do comportamento a ser executado em formato de *string* e a publica no tópico */tópico1*, que por sua vez, é subscrito por */J2n6a300_Reativo*, executando o comportamento escolhido. Após a execução de cada comportamento uma mensagem contendo a nova configuração da Torre de Hanói (estado do ambiente $E(k)$) é publicada no tópico */tópico2*, que é subscrito pelo nó */Instintivo_recebe*, este responsável por enviar a mensagem ao nível instintivo do AAC.

4. Implementação das regras e Experimentos

A tarefa escolhida para a realização dos experimentos foi a de capacitar o manipulador JACO kinova a montar uma Torre de Hanói com 3 objetos, a partir de qualquer configuração inicial possível, dentre os 25 estados iniciais aceitos para uma Torre com 3 discos. O problema consiste em passar todos os discos de um pino para o último pino, usando um dos pinos como auxiliar, sendo considerado o final do jogo quando os três discos ficam montados, conforme a última imagem da Figura 8. Existem duas regras que devem ser seguidas na montagem da Torre:

- Só poderá movimentar um disco por vez;
- Nunca um disco maior pode estar sobre um disco menor.

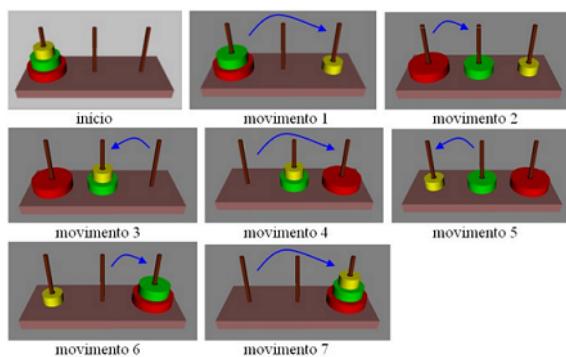


Figura 8. Exemplo e uma Torre de Hanói com três discos sendo montada.

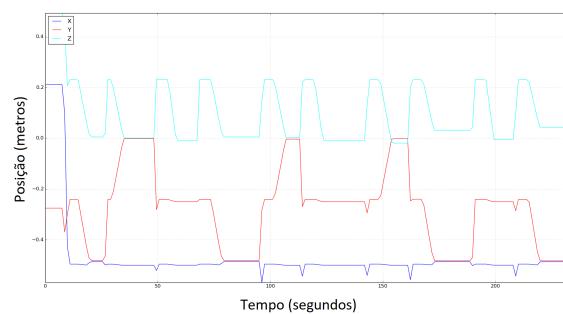


Figura 9. Posição do efetuador vs Tempo durante a montagem da Torre de Hanói do experimento II.

Foram feitos simulações e experimentos com todos os estados iniciais possíveis, onde em todos os casos o manipulador montou a Torre de Hanói com o menor número de movimentos. A seguir pode ser visto o resultado de dois experimentos com o robô JACO Kinova presente no Laboratório de Robótica na Universidade Federal da Bahia. Para o estado inicial da Torre de Hanói do primeiro experimento, o nível cognitivo escolheu o *plano_1* para ser executado no nível instintivo, podendo ser observado todas as regras utilizadas na Figura 12. A evolução no tempo da posição do efetuador durante este experimento pode ser observado na Figura 13.

O estado inicial do segundo experimento pode ser visto na Figura 15. Para este estado inicial as premissas da regra_16 presente no nível cognitivo foram aceitas, logo o *plano_2* foi escolhido para ser executado pelo nível instintivo. Todas as regras para a execução deste experimento pode ser visto na Figura 10 e na Figura 14 podemos ver todos os comportamentos sendo executados pelo manipulador JACO.

A evolução no tempo da posição do efetuador durante este experimento pode ser observado na Figura 13.

5. Conclusão

Este trabalho teve como principal objetivo habilitar o manipulador JACO Kinova a execução de tarefas complexas de uma maneira completamente autônoma. Foi escolhido uma arquitetura de agente inteligente (Agente Autônomo Concorrente) para ser utilizado juntamente com o ROS para capacitar o manipulador a execução de tais tarefas. Os três níveis do AAC foram empregados, sendo que no nível cognitivo um Sistema Baseado em

```

Aguardando... (0)
Aguardando... (1)
(3 1) (1 1) (2 1)
p3n1
p1n1
p2n1
-- - Facts Base - - -
(logic (disco_maior posicao p2n1))
(logic (disco_medio posicao p1n1))
(logic (disco_menor posicao p3n1))
(logic (robot acao p3n1_p1n2menor))

----->robot acao p3n1_p1n2menor

Aguardando... (1)
(1 2) (1 1) (2 1)
p1n2
p1n1
p2n1
-- - Facts Base - - -
(logic (disco_maior posicao p2n1))
(logic (disco_medio posicao p1n1))
(logic (disco_menor posicao p2n2))
(logic (robot acao p2n1_p3n1maior))

----->robot acao p2n1_p3n1maior

Aguardando... (1)
(1 2) (1 1) (3 1)
p1n2
p1n1
p3n1
-- - Facts Base - - -
(logic (disco_maior posicao p3n1))
(logic (disco_medio posicao p3n2))
(logic (disco_menor posicao p2n1))
(logic (robot acao p1n2_p2n1menor))

----->robot acao p1n2_p2n1menor

Aguardando... (1)
(2 1) (1 1) (3 1)
p2n1
p1n1
p3n1
-- - Facts Base - - -
(logic (disco_maior posicao p3n1))
(logic (disco_medio posicao p1n1))
(logic (disco_menor posicao p2n1))
(logic (robot acao p1n1_p3n2medio))

----->robot acao p1n1_p3n2medio

Aguardando... (1)
(2 1) (3 2) (3 1)
p2n1
p3n2
p3n1
-- - Facts Base - - -
(logic (disco_maior posicao p3n1))
(logic (disco_medio posicao p3n2))
(logic (disco_menor posicao p2n1))
(logic (robot acao p2n1_p3n3menor))

----->robot acao p2n1_p3n3menor

Aguardando... (1)
(3 3) (3 2) (3 1)
p3n3
p3n2
p3n1
-- - Facts Base - - -
(logic (disco_maior posicao p3n1))
(logic (disco_medio posicao p3n2))
(logic (disco_menor posicao p3n3))
(logic (robot acao pare_sucesso))

----->robot acao pare_sucesso

```

Figura 10. Regras do plano_2, utilizadas para a realização do experimento II presente no nível instintivo.



Figura 11. Estado Inicial e estado objetivo do experimento I.

```

::regra_1
(if      (logic ( disco_menor posicao p1n3 ))
(logic ( disco_medio posicao p1n2 ))
(logic ( disco_maior posicao p1n1 )))
(then   (logic ( robot acao p1n3_p3n1menor )))

(regra_2
(if      (logic ( disco_menor posicao p3n1 ))
(logic ( disco_medio posicao p1n2 ))
(logic ( disco_maior posicao p1n1 )))
(then   (logic ( robot acao p1n2_p2n1medio ))))

(regra_3
(if      (logic ( disco_menor posicao p3n1 ))
(logic ( disco_medio posicao p2n1 ))
(logic ( disco_maior posicao p1n1 )))
(then   (logic ( robot acao p3n1_p2n2menor ))))

(regra_4
(if      (logic ( disco_menor posicao p2n2 ))
(logic ( disco_medio posicao p2n1 ))
(logic ( disco_maior posicao p1n1 )))
(then   (logic ( robot acao p1n1_p3n1maior ))))

(regra_5
(if      (logic ( disco_menor posicao p2n2 ))
(logic ( disco_medio posicao p2n1 ))
(logic ( disco_maior posicao p3n1 )))
(then   (logic ( robot acao p2n2_p1n1menor ))))

(regra_6
(if      (logic ( disco_menor posicao p1n1 ))
(logic ( disco_medio posicao p2n1 ))
(logic ( disco_maior posicao p3n1 )))
(then   (logic ( robot acao p2n1_p3n2medio ))))

(regra_7
(if      (logic ( disco_menor posicao p1n1 ))
(logic ( disco_medio posicao p3n2 ))
(logic ( disco_maior posicao p3n1 )))
(then   (logic ( robot acao p1n1_p3n3menor ))))

(regra_25
(if      (logic ( disco_menor posicao p3n3 ))
(logic ( disco_medio posicao p3n2 ))
(logic ( disco_maior posicao p1n1 )))
(then   (logic ( robot acao pare_sucesso ))))

```

Figura 12. Regras do plano_1, utilizadas para a realização do experimento I presente no nível instintivo.

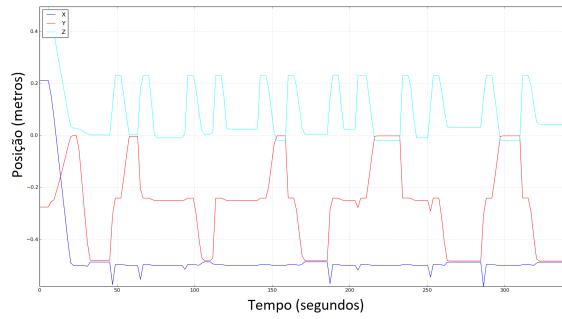


Figura 13. Posição do efetuador vs Tempo durante a montagem da Torre de Hanói do experimento I.

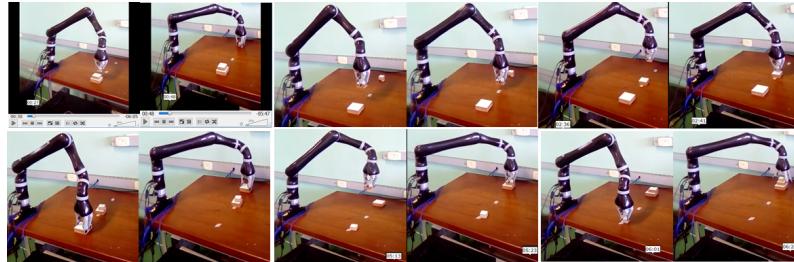


Figura 14. Manipulador JACO Kinova executando a montagem da Torre de Hanói do experimento I.

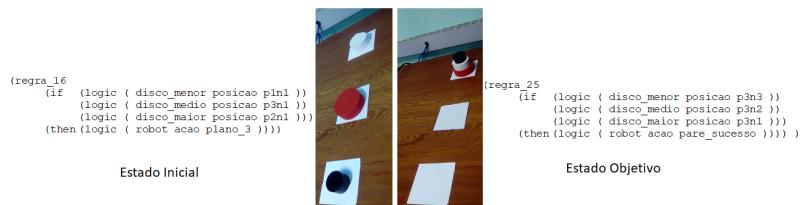


Figura 15. Estado Inicial e estado objetivo do experimento II.

Conhecimento (SBC) com uma base de regras em LPO, foi utilizado com o intuito de selecionar os planos a serem executados no nível instintivo. No nível instintivo um SBC foi implementado com três bases regras, sendo cada uma dessas um plano a ser executado. Como foi visto na seção 1, o AAC até então nunca havia sido utilizado juntamente com o ROS, sendo está uma das principais contribuições deste trabalho. O nível reativo responsável pela resposta em tempo real do agente, foi implementado então no ROS, utilizando o *metapackage* Kinova-ROS [Kinova 2017], possibilitando desta forma o controle total do manipulador, tanto no espaço cartesiano quanto no espaço de juntas.

Nos experimentos, conseguimos encontrar resultados satisfatórios, visto que, em todos os casos o manipulador executou as tarefas com o número mínimo de movimentos possíveis.

Este trabalho trouxe como principais contribuições:

- Utilização do AAC juntamente com o ROS, algo nunca feito antes e de extrema importância para futuras implementações do AAC, visto o grande número de bibliotecas e ferramentas que o ROS disponibiliza para ajudar desenvolvedores de programas robóticos a criarem suas aplicações. Outro ponto a ser considerado, é

que o código do nível reativo foi escrito totalmente em *Python* (linguagem de programação), enquanto o Expert-coop++ é escrito inteiramente em C++. O ROS possui a característica de multi-linguagem, ou seja, ele foi projetado para suportar diversos idiomas de programação como o C++, Python, Octave e Lisp. Desta forma, o número de possibilidades de utilização do AAC aumenta, visto que, sua utilização não fica restrita a apenas pessoas que programem em C++;

- Capacitou o manipulador JACO a execução de tarefas complexas de uma maneira autônoma. Com o nível reativo implementado no ROS, basta apenas rescrever as regras dos níveis cognitivos e instintivos, para capacitar o JACO a realização de novas tarefas em ambientes 3D, sem a necessidade de implementar um novo nível reativo. Como exemplo o manipulador poderia realizar tarefas como: "jogar" xadrez, simular situações em ambientes industriais (soldagem de peças, movimentar peças em estoques), manipulação de ampolas (amostras) em análises biomédicas, etc.;
- Implementou o AAC em um nova categoria de robôs (manipuladores robóticos);

Futuros trabalhos visam a utilização de cenários mais robustos, com a inserção de sistemas de visão computacional e a inserção do manipulador em uma comunidade multi-agente.

Referências

- Barros, T. T. T. (2014). Modelagem e implementação no ros de um controlador para manipuladores móveis.
- Bittencourt, G. (1997). In the quest of the missing link. *International Joint Conference of Artificial Intelligence*, pages 310–315.
- Bittencourt, G. and Costa, A. d. (2001). Hybrid cognitive model.
- Costa, A. C. P. L. and Bittencourt, G. (1998). Ufsc-team: A cognitive multi-agent approach to the robocup 98 simulator league. pages 371–376.
- Costa, A. L. and Bittencourt, G. (1999). From a concurrent architecture to a concurrent autonomous agents architecture. pages 274–285.
- Costa, A. L., Ferreira, D. S. F., Conceição, A. G. S., and Pappas, G. (2015). Embedding the concurrent autonomous agent into a humanoid robot. *IFAC-PapersOnLine*, 48(19):203–208.
- Costa, A. L. d., Scolari, A. G., Ribeiro, T. T., and Junior, J. s. (2011). Embarcando o agente autônomo concorrente no robô móvel omnidirecional axébot. nível reativo. *X Simpósio Brasileiro de Automação Inteligente (SBAI)*.
- Ferreira, D. S. F. (2014). Embarcando o agente autônomo concorrente em uma rede de microcontroladores de um robô móvel omnidirecional. Master's thesis, Dissertação de Mestrado-Universidade Federal da Bahia, Programa em Pós Graduação em Engenharia Elétrica, Salvador - BA.
- Kinova (2017). *Ros Parameter Server*.
- Oliveira, L. L. R. d. et al. (2013). Controle de trajetória baseado em visão computacional utilizando o framework ros. Master's thesis.

Especificação de Casos de Uso para a Modelagem de Requisitos de Sistemas Multiagente Normativos[†]

Emmanuel Sávio Silva Freire¹, Patrícia Maria Barbosa Ferreira²

¹Departamento de Ensino – Instituto Federal do Ceará (IFCE/Campus Morada Nova)
Av. Prefeito Raimundo José Rabelo, 2717 – 62940-000 – Morada Nova – CE – Brasil

²Departamento de Ensino – Instituto Federal do Ceará (IFCE/Campus Aracati)
Rodovia CE-040, Km 137, s/n – Aeroporto – 62800-000 - Aracati – CE – Brasil
savio.freire@ifce.edu.br, {savio.essf, pati270297}@gmail.com

Abstract. *Normative multiagent systems (NMASs) has stood out in complex system development. Considering that gathering and analysis phase is essential in development process of these systems, an extension of use case UML diagram was proposed to allow the modelling of functionalities provided by agents. Nevertheless, it was not proposed a template to specify use cases. Therefore, this paper discusses the necessity to define a template of use case capable to specify the functionalities for NMASs along with the formalization of this template using VDM++ method. Because it is a preliminary work, it does not have results to compare with related work.*

Resumo. *Os sistemas multiagente normativos (SMANs) tem se destacado no desenvolvimento de sistemas complexos. Considerando que a fase de levantamento e análise de requisitos é essencial no processo de desenvolvimento desses sistemas, uma extensão do diagrama UML de casos de uso foi proposta para permitir a modelagem das funcionalidades providas por agentes. Entretanto, não foi proposto um template para a especificação dos casos de uso. Com isso, este artigo discute a necessidade da definição de um template de casos de uso capaz de especificar as funcionalidades para os SMANs juntamente com a formalização desse template utilizando o método VDM++. Pelo fato de ser um trabalho preliminar, não se dispõem de resultados para comparar com trabalhos relacionados.*

1. Introdução

Os sistemas multiagente normativos (SMANs) [Boella, van der Torre e Verhagen 2006] têm sido utilizados para o desenvolvimento de sistemas complexos. Esses sistemas são formados por um conjunto de agentes inteligentes que interagem entre si para alcançar seus objetivos, considerando as normas que regulam o comportamento desses agentes. Logo, uma norma define as ações que podem, que devem e que não devem ser executadas [Silva, Braga e Figueiredo 2010]. Neste contexto, a Engenharia de Software tem definido técnicas, métodos e ferramentas para auxiliar o processo de desenvolvimento desses

[†]Este trabalho faz parte do Projeto número 5924 intitulado “Suporte para a Modelagem de Requisitos para a Modelagem de Casos de Uso para Projetos de Sistemas Multiagente Normativos” do Edital IFCE/PIBIC/2017. A autora Patrícia Ferreira agradece pelo suporte financeiro recebido pelo IFCE para a realização deste trabalho.

sistemas. Entretanto, o maior foco é em linguagens de modelagem e de implementação [Freire 2017]. Assim, a fase de levantamento e análise de requisitos tem sido pouco abordada. Segundo Sommerville (2011), a fase de levantamento e análise de requisitos é essencial para que os usuários do sistema a ser desenvolvido possam identificar e fornecer as suas necessidades para os analistas de requisitos.

Considerando a importância dos requisitos, a Linguagem de Modelagem Unificada (do inglês, *Unified Modeling Language - UML*) [UML 2017] possui o diagrama de casos de uso que dá suporte a fase de análise de requisitos, porém não considera os agentes nem as normas presentes nos SMANs. Assim, Freire (2017) propôs a extensão do diagrama UML de casos de uso, baseado na primeira extensão realizada por Guedes (2012), possibilitando a representação visual das normas e da interação delas com as outras entidades previstas no diagrama.

No entanto, não foi proposto um *template* para casos de uso para permitir a especificação das funcionalidades providas pelos agentes nem das normas que regulam o comportamento dessas entidades. Vale ressaltar que a especificação de requisitos engloba a descrição dos requisitos de usuário e de sistema em um documento utilizado durante a fase de implementação do sistema. Além disso, esse documento pode ser especificado por meio de linguagem natural, notações gráficas ou especificações matemáticas [Sommerville 2011].

Assim, esse artigo teve como objetivo discutir a necessidade da definição de um *template* para a especificação de casos de uso no contexto de desenvolvimento de SMANs. A definição desse *template* estaria vinculada a versão do diagrama de casos de uso proposto por Freire [2017] para auxiliar a especificação e a validação dos requisitos junto aos *stakeholders*. Além disso, pretende-se formalizar o *template* definido por meio do *Vienna Development Method ++* (VDM++) [Larsen et al. 2018], possibilitando a redução da ambiguidade inerente à especificação de requisitos. Vale ressaltar que esse método foi escolhido pois (i) apresenta consolidação em relação a sua semântica e sintaxe, (ii) apresenta conceitos relacionados à orientação a objetos, podendo ser extensível para o paradigma orientado a agentes, e (iii) tem sido amplamente utilizado na indústria [Overture 2017].

O presente artigo está estruturado como segue: a Seção 2 apresenta o referencial teórico acerca do diagrama de casos de uso para SMANs proposto por Freire (2017), dos *templates* utilizados para a especificação de casos de uso e do método VDM++. A Seção 3 discute a necessidade da definição de um *template* e o seu mapeamento para VDM++ considerando o diagrama de casos de uso proposto por Freire (2017). Finalmente, a Seção 4 apresenta as conclusões e os trabalhos futuros.

2. Referencial Teórico

2.1. Diagrama de Casos de Uso para Sistemas Multiagente Normativos

O diagrama de casos de uso proposto por Freire (2017) foi gerado por meio da extensão do metamodelo de Guedes (2012) por meio da inclusão dos elementos normativos definidos por Figueiredo e Silva (2010). Assim, a nova versão do diagrama permite a modelagem de requisitos para SMANs, considerando seus atores internos, suas normas e suas organizações. Mais especificamente, a modelagem de normas permite a identificação do conceito deôntrico (o que é permitido, obrigado ou proibido), do contexto (organização) no

qual a norma está relacionada, do recurso (caso de uso) que está sendo regulamentado pela norma e da entidade (ator) que está sendo restringida pela norma.

A extensão considerou alterações nas sintaxes abstrata e concreta. Na sintaxe abstrata, foram incluídas novas metaclasses para representar os elementos normativos com as entidades (ator e caso de uso) previamente existentes no diagrama de casos de uso proposto por Guedes (2012). Para a sintaxe concreta, foram utilizados elementos gráficos para representar os novos elementos (*Organization, Norm, Context, Restrict, Resource* e *Ownership*). Um exemplo (veja a Figura 1) baseado no sistema “Ambiente Multiagente de Ensino-Aprendizagem” (AME-A) foi modelado por Freire (2017) utilizando a nova versão do diagrama. Vale ressaltar que não foi desenvolvida uma ferramenta de suporte à nova versão nem foi proposto um *template* para a especificação de casos de uso. Diversos autores, como Sommerville (2011), sugerem a utilização de documentação textual para essa especificação.

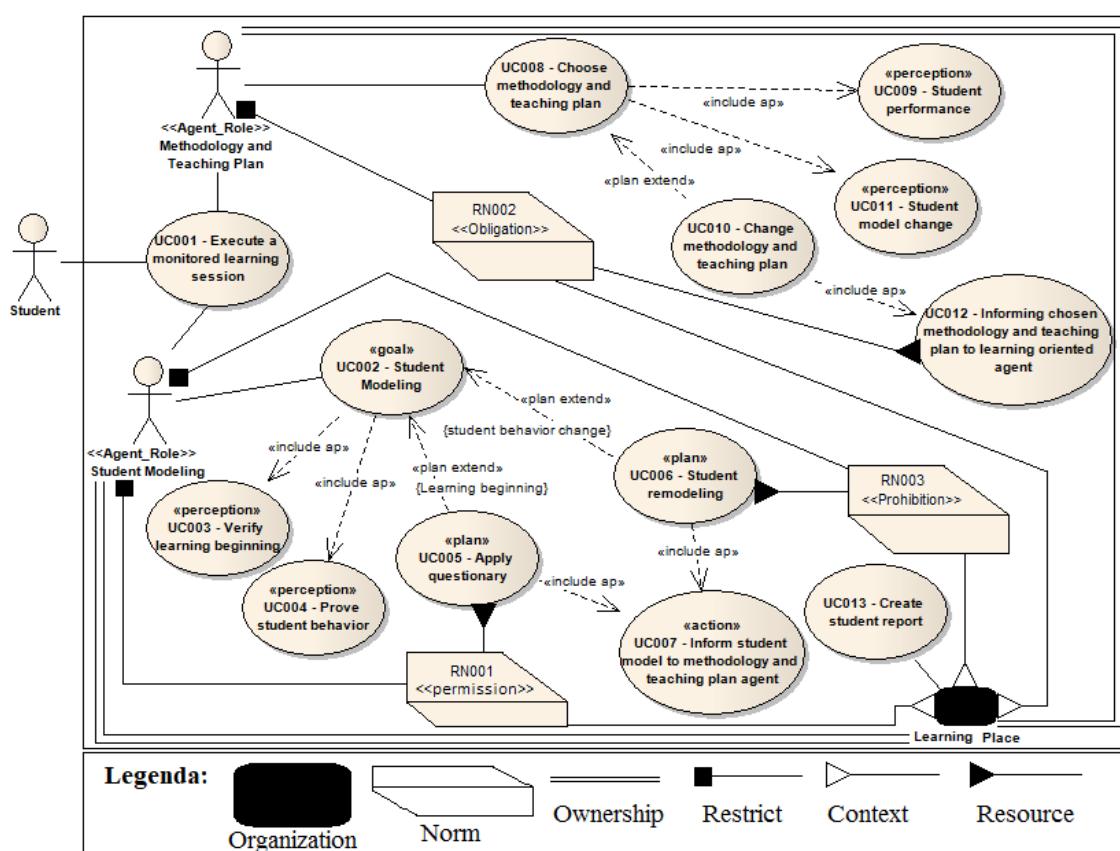


Figura 1. Um exemplo do diagrama de casos de uso proposto por Freire (2017)

2.2. Templates para Especificação de Casos de Uso

A especificação de requisitos engloba a descrição dos requisitos de usuário e de sistemas em um documento textual [Sommerville 2011]. Por um lado, os requisitos de usuário correspondem aos requisitos funcionais e não funcionais do sistema, descrevendo apenas o comportamento externo do sistema. Por outro lado, os requisitos de sistema são baseados nos requisitos de usuário e têm o intuito de serem utilizados para auxiliar no projeto do sistema, pois possuem uma especificação completa e detalhada do sistema. Vale ressaltar que um requisito funcional está relacionado com a funcionalidade que o sistema deve prover, enquanto o requisito não funcional representa as restrições aos serviços ou funções oferecidos do sistema.

Normalmente, a especificação de requisitos é realizada por meio de linguagem natural, notações gráficas (diagramas UML de caso de uso ou de sequência) em conjunto com anotações de texto e especificações matemáticas. Além disso, a especificação de um *template* auxilia na escrita dos requisitos. Mais especificamente, a linguagem UML [UML 2018] é utilizada para a notação gráfica de requisitos [Sommerville 2011], porém não possui uma estruturação específica para a documentação de um caso de uso. Por conta disso, existem diversas propostas de documentação por diferentes autores [Cockbum 2001] [Larman 2004] [Some 2010] [Sommerville 2011]. Entretanto, nenhuma delas dá suporte à especificação de casos de uso para sistemas multiagente (SMAs).

Consequentemente, Guedes (2012), considerando a sua versão do diagrama de casos de uso para SMAs, definiu um *template* para documentar os casos de uso internos, contendo os seguintes campos: (i) nome do caso de uso interno, (ii) seu estereótipo, (iii) o principal *AgentRole_Actor* que o utiliza, (iv) os prováveis *AgentRole_Actors* secundários, (v) outros atores que possam estar envolvidos no processo, (vi) as possíveis percepções associadas ao caso de uso interno, (vii) as possíveis ações associadas, (viii) os possíveis planos associados, (ix) pontos de extensão de plano (se existirem), (x) um resumo da função do caso de uso interno, (xi) as possíveis pré e pós-condições que devem ser satisfeitas antes e depois da execução do caso de uso interno, (xii) o fluxo principal, e (xiii) os possíveis fluxos alternativos de cada caso de uso interno. Vale ressaltar que esse *template* dá suporte parcial ao diagrama de casos de uso definido por Freire (2017), pois não contempla os elementos normativos incluídos na extensão desse autor.

2.3. Vienna Development Method++ (VDM++)

O *Vienna Development Method* ++ (VDM++) é uma linguagem de especificação formal que permite a especificação de sistemas orientados a objetos com comportamento paralelo e *real-time* [Larsen et al. 2018]. Essa linguagem foi originada por meio da extensão de VDM-SL [Larsen et al. 1996]. A sintaxe concreta é descrita no Formalismo de Backus-Naur (BNF, do inglês *Backus-Naur Form*) e possui tipos de dados definidos. Além disso, é possível definir funções estáticas, ou seja, não é necessário a instância de objetos para a execução das funções. Os algoritmos definidos nesse método podem conter operações e funções. A diferença entre elas é em relação à utilização de variáveis locais e globais.

O trabalho de Wong, Mit e Sidi (2016) utilizou o método VDM++ para formalizar um *template* de caso de uso. Esse *template* era baseado nos elementos contidos no diagrama UML de casos de uso. As pré-condições e pós-condições presentes no *template* foram definidas para pré-condições e pós-condições contidas em VDM++. Com o objetivo de demonstrar a sintaxe do método, é apresentado o mapeamento de casos de uso para VDM++ proposto por Wong, Mit e Sidi (2016):

```

UcNameUML = {verb, noun}
UcNameUML. noun |→ classNameVDM++
if (genUcNameUML <> null) then
  genUcNameUML |→ subClsOfVDM++. className2
  
```

3. Discussão

Por meio dos *templates* de casos de uso, o analista de requisitos pode especificar detalhadamente as funcionalidades do sistema. Assim, é possível descrever as requisições dos usuários e as respostas correspondentes do sistema. Entretanto, essa especificação não

é retratada no diagrama de casos de uso, pois é possível modelar apenas as funcionalidades (casos de uso) em conjunto com os atores que irão interagir com o sistema. Mais especificamente, a versão do diagrama de casos de uso proposta por Freire (2017) permite a modelagem de normas, organizações, papéis de agente e atores. Os atores são externos ao sistema. Esse conceito já existia no diagrama UML de casos de uso. As entidades normas, organizações e papéis de agente foram incluídas no diagrama permitindo uma melhor visão do sistema e a relação entre as normas e as entidades do sistema. No entanto, não é possível detalhar as funcionalidades de cada caso de uso.

Neste sentido, a necessidade da elaboração de um *template* é requerida, visto que os *templates* existentes na literatura não são suficientes para o detalhamento das entidades contidas no diagrama de casos de uso proposto por Freire (2017). Mesmo o *template* definido por Guedes (2012) para SMAs, não possui campos específicos para o detalhamento dos elementos normativos, focando apenas nas percepções, nas ações e nos planos de agentes (representados por meio de papéis de agente) contidos em tais sistemas.

Portanto, a evolução do *template* de Guedes (2012) para contemplar os elementos normativos seria uma boa alternativa para a especificação de casos de uso e das normas que formam um SMAN. Logo, esse *template* passaria a ter um ou mais campos específicos para as normas e as relações entre elas e as outras entidades do diagrama de casos de uso. Neste caso, o *template* colaboraria para a especificação em linguagem natural das funcionalidades e das normas que as regulam. Os benefícios associados a esse novo *template* seriam: (i) um melhor detalhamento das normas e da organização, (ii) auxílio na especificação e na validação dos requisitos junto aos *stakeholders*, e (iii) uma melhor compreensão das funcionalidades que serão executadas pelos agentes e a relação delas com as normas.

Por se tratar de um *template* em linguagem natural, podem ocorrer erros oriundos da ambiguidade em relação à escrita e aos termos utilizados na especificação. Esses erros podem ser reduzidos pela formalização do *template* utilizando o método VDM++. Esse método foi utilizado por Wong, Mit e Sidi (2016) para formalizar o *template* utilizado para a especificação de casos de uso oriundos do diagrama UML de casos de uso. O resultado encontrado por esses autores foram as regras de mapeamento entre o *template* utilizado e o método VDM++, seguindo a sua sintaxe e a sua semântica. Esse mapeamento foi possível pois o método VDM++ possibilita a especificação de sistemas orientados a objetos.

Considerando que o paradigma orientado a agentes estende linguagens de modelagem e implementação, e *frameworks* criados especificamente para o paradigma orientado a objetos, a análise da semântica e da sintaxe de VDM++ é requerida com o intuito de identificar os pontos convergentes entre esse método e as entidades previstas no diagrama de casos de uso proposto por Freire (2017). Assim, o *template* definido para essa versão do diagrama seria transformado para o método formal VDM++. Finalmente, após a definição do *template* e do mapeamento para VDM++, será possível modelar um exemplo para validar o *template* e as regras de mapeamento.

4. Conclusão e Trabalhos Futuros

Este artigo apresentou uma discussão acerca da definição de um *template* para a especificação de casos de uso para SMANs. Para tanto, é proposta a reutilização do *template* definido por Guedes (2012) com o intuito de incluir campos para as entidades previstas no diagrama de casos de uso proposto por Freire (2017). Além disso, após a definição do *template*, é proposta a

formalização do *template* utilizando o método VDM++. Esse método foi sugerido, pois já foi utilizado na formalização de *templates* de casos de uso para o diagrama UML de casos de uso. Espera-se como trabalhos futuros: (i) a definição do *template* de casos de uso, (ii) a análise do método VDM++ em relação às entidades dos SMANs, (iii) a formalização do *template* definido no item (i), e (iv) a verificação da consistência entre o *template* e as regras oriundas da formalização por meio de um exemplo de modelagem.

Referências

- Boella, G., van der Torre, L. e Verhagen, H. (2006) “Introduction to normative multiagent systems”, Computational & Mathematical Organization Theory, vol. 12, no. 2, pp. 71–79.
- Cockburn, A. (2001) Writing Effective Use Cases. CA: Addison-Wesley, 2001.
- Freire, E. S. S. (2017) Extensão do Metamodelo do Diagrama de Casos de Uso para a Modelagem de Requisitos em Projetos de Sistemas Multiagente Normativos. In: Diana F. Adamatti; Mariela I. Cortés; Anarosa A. Brandão (Org.). E-book do WESAAC: 11th workshop-escola de sistemas de agentes, seus ambientes e aplicações. 1 ed. Rio Grande: Editora da FURG, 2017, v.1, p. 137-148.
- Guedes, G. T. A. (2012) Um Metamodelo UML para a Modelagem de Requisitos em Projetos de Sistemas Multiagentes, Tese de doutorado. Porto Alegre: UFRGS, Instituto de Informática.
- Larman, C. (2004) Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd ed. USA: Prentice Hall, 2004.
- Larsen, P. G., Hansen B. S., Brunn, H., Plat, N., Toetenel, H., Andrews, D. J., Dawes, J., Parkin G. e others (1996) Information technology – Programming languages, their environments and system software interfaces – Vienna Development Method – Specification Language – Part 1: Base language. December 1996.
- Larsen, P. G., Lausdahl, K., Battle, N., Fitzgerald, J., Wolff, S., Sahara, S., Verhoef, M., Tran-Jørgensen, P.W. V., Oda, T. and Chisholm, P. (2018) Overture Techinal Report Series – VDM-10 Language Manual. Disponível em: https://raw.githubusercontent.com/overturetool/documentation/master/documentation/VDM10LangMan/VDM10_lang_man.pdf. Acessado em 20 de dezembro de 2017.
- Overture (2017) The Vienna Development Method. Disponível em: <http://overturetool.org/>. Acessado em 20 de dezembro de 2017.
- Silva, V., Braga, C. e Figueiredo, K. (2010) A Modeling Language to Model Norms. In: Workshop on Coordination, Organization, Institutions and Norms in agent systems at International Conference on Autonomous Agents and Multi-Agent Systems, Toronto.
- Some, S. S. (2010) “Formalization of Textual Use Case Based on Petri Nets”, International Journal of Software Engineering and Knowledge Engineering, vol. 20, no. 5, pp. 695-737.
- Sommerville, I. (2011) Engenharia de Software. 9 ed. São Paulo: Pearson Addison-Wesley.
- UML (2017) Unified Modeling Language Specification, versão 2.5, OMG. Disponível em: <http://www.omg.org/spec/UML/2.5/PDF/>. Acessado em: 20 de dezembro de 2017.
- Wong, S. Y., Mit, E. and Sidi, J. (2016) Integration of Use Case Formal Template using Mapping Rules. In: Third International Conference on Information Retrieval and Knowledge Management, Malasya.

Epidemiologia: Compreensão da Dinâmica das Doenças e Emergência através da Modelagem

Bianca Parulla Marques¹, Douglas Furtado Félix¹,
Diana Francisca Adamatti¹

¹Programa de Pós Graduação em Modelagem Computacional (PPGMC)
Universidade Federal do Rio Grande (FURG)
Caixa Postal 474 – 96.203.900 – Rio Grande – RS – Brasil

{biancaparullam, douglasfurtadof, dianaada}@gmail.com

Abstract. *Epidemiology studies the spread of infectious diseases among individuals of the same species, and the reasons for the occurrence of this fact and how to contain its spread. This study describes how the spread of an epidemic works in a closed population, using multiagent simulation and how to solve the problem, using a prototype built on a preexisting model. To this end, the results showed the importance of vaccination to prevent the spread of the disease, which in turn leads to fewer health problems for society and, as a consequence, fewer expenses for the economy.*

Resumo. *A epidemiologia estuda a disseminação de doenças infecciosas entre indivíduos de mesma espécie, e as razões para o acontecimento deste fato e como conter a sua propagação. Neste estudo é descrito como funciona a disseminação de uma epidemia em uma população fechada, utilizando simulação multiagente e como se pode ajudar a solucionar o problema, tendo como aporte um protótipo construído com base em um modelo pré-existente. Com este intuito, os resultados obtidos revelaram a importância da vacinação para evitar a disseminação da doença, que por sua vez gera menos problemas de saúde à sociedade e como consequência menos gastos para a economia.*

1. Introdução

A epidemia indica um impacto calamitoso para a sociedade, e pode ser melhor entendida se for analisada conforme o alcance de propagação de uma doença [Cruz 2013].

Neste âmbito de doenças infecciosas que se disseminam rapidamente, a Modelagem Matemática tornou-se útil para compreender a propagação, e assim obter um controle e avaliação sobre a epidemia [De Barros 2015].

Os sistemas multiagentes são utilizados para criar arquiteturas de agentes que são entidades computacionais inteligentes e autônomas, que interagem entre si para solucionar um problema que não pode ser resolvido individualmente, como no caso da epidemia [Wooldridge 1995].

Neste trabalho será apresentado a extensão de um modelo pré-existente na ferramenta NetLogo, para a simulação de uma epidemia baseada no modelo de Kermack e McKendrick, que representa a prática sistêmica de um fenômeno, que dá início no momento em que uma pessoa infectada é introduzida em um ambiente predisposto, onde

a população é fechada, ou seja, não haverá nascimentos, mortes ou viagens para dentro ou fora do grupo. A atividade ocorre de modo que os indivíduos andam pelo espaço de forma aleatória, e quando entram em contato com um indivíduo infectado, este contrairá a doença. Como um outro passo, os infectados terão a possibilidade de se reabilitar, e uma vez curado, o indivíduo se faz imune ao vírus.

Desta forma, o objetivo deste trabalho é fazer algumas alterações no modelo para ver como o modelo se comporta perante personalizações definidas pelo usuário, tais como:

- Definir inicialmente o número de infectados;
- Definir a velocidade em que os agentes vão se movimentar no ambiente;

Além de inserir no ambiente um agente "vaccinated", no qual se torna de grande importância para o trabalho onde este incentiva de alguma forma a população para a vacinação e acabar por não contrair a doença, e extinguir as epidemias.

Este artigo está estruturado em cinco seções. A primeira seção apresenta uma introdução do trabalho, na segunda seção contém o referencial teórico para conceituar a epidemiologia, epidemia, o modelo matemático, sistemas multiagente e o ambiente de simulação NetLogo. A terceira seção retrata sobre o modelo de simulação utilizado para este trabalho, que tem por continuação na quarta seção onde são apresentadas as alterações da simulação, e por fim, a última seção aborda as conclusões e considerações finais deste trabalho.

2. Referencial teórico

Nesta seção são apresentados os temas necessários para entendimento do trabalho proposto.

2.1. Epidemiologia

A epidemiologia estuda a disseminação de uma doença infecciosa entre indivíduos de mesma espécie, e também as razões para o surgimento do fenômeno e como conter a propagação. A epidemiologia pode ser classificada de três formas: endemia, epidemia e pandemia, sendo classificadas de acordo com o alcance da doença. Denomina-se endemia quando a população é atingida de acordo com as condições ambientais e não passa de uma área restrita, já é chamada de epidemia se atinge além dos limites de uma determinada área, e pandemia é classificado quando a doença se espalha pelo mundo, alcançando até mesmo outros continentes [Cruz 2013].

2.2. Epidemia

Segundo Forattini [Forattini 2005], a definição para epidemia é "*o nome dado ao estado de incidência ou agravo à saúde além do normalmente esperado dentro da faixa de variação da prevalência da doença, em determinada área ou grupo populacional*".

No âmbito da epidemiologia de doenças infecciosas, tornou-se útil a Modelagem Matemática para o entendimento da propagação, controle e avaliação da epidemia. A epidemiologia matemática é de caráter interdisciplinar, abrangendo a área da matemática, biologia, física entre outras [De Barros 2015].

2.3. Modelo Matemático SIR

O modelo SIR (Suscetível Infectado Recuperado) foi proposto por Kermack e McKendrick em 1927, e é constituído pelo processo de dividir a população em três classes, a classe suscetível que são os indivíduos que ainda não foram contaminados pela doença, a classe dos infectados pela doença e a classe dos recuperados que são os indivíduos curados ou que vieram a morrer com o contágio [De Barros 2015].

Este modelo simula uma população constante, ou seja, não há nascimentos e migrações, obtém-se então:

$$N = S + I + R = cte$$

onde:

N = total da população;

S = número de indivíduos suscetíveis;

I = número de indivíduos infectados;

R = número de indivíduos recuperados;

cte = constante.

Este modelo matemático também fundamenta-se em duas hipóteses complementares:

- A razão de variação da população suscetível é proporcional ao número de encontros entre a população suscetível e a infectada;
- A razão de variação da população recuperada é proporcional à população infectada.

No modelo SIR é possível incorporar ações de controle como a vacinação ou isolamento, entre outras.

2.4. Vacinação

Segundo Nepomuceno [Nepomuceno 2005], de modo geral, as doenças infecciosas podem ser controladas por vacinação e por isolamento dos indivíduos infectados.

Neste trabalho foi abordada a importância da vacinação para a não disseminação de um vírus contagioso. Este tipo de precaução é eficaz, conferindo proteção em alguns casos para toda a vida, ou por um determinado período de tempo [Duarte 2016]. Como apresentado anteriormente, este estudo envolve um modelo matemático onde o indivíduo uma vez vacinado, não será contaminado pelo vírus novamente, diminuindo assim a quantidade de indivíduos infectados.

2.5. Simulação Baseada em Multiagente - MABS

Segundo Rebonatto [Rebonatto 2000] a simulação computacional é aplicada como elemento auxiliar na tomada de decisões, principalmente no planejamento a médio e longo prazos e em situações que envolvem custos e riscos elevados. Os modelos de decisão permitem o exame de detalhes em grande precisão.

Um Sistema Multiagente (SMA) é definido como um sistema baseado em um número de entidades autônomas que interagem em um ambiente, onde cooperam entre si para solucionar um problema que não pode ser resolvido individualmente [Wooldridge 1995].

A Simulação Baseada em Multiagente (MABS), surgiu com a união das tecnologias de Simulação e Sistemas Multiagente. Muitas aplicações em SMAs são desenvolvidas para simular alguma situação da realidade. Para tanto, o fenômeno real é decomposto em um conjunto de elementos e em suas interações. Cada elemento é modelado como um agente e o modelo geral é o resultado das interações entre estes agentes [Adamatti 2011].

2.6. Ferramenta Netlogo

O Netlogo¹ é um *software* gratuito para modelagem de ambientes multiagente. Os programadores podem instruir os agentes, todos operando de forma independente. É um *software* simples, e permite fácil manuseio e execução, possui uma extensa documentação e tutoriais, e um amplo conjunto de modelos para simulações pré-escritas que podem ser utilizadas e modificadas. Estas simulações abordam conteúdos de áreas das ciências naturais e sociais [de Lima et al. 2009]. São algumas das funcionalidades do NetLogo:

- Estrutura de linguagem simples;
- Agentes móveis (*turtles*) caminham sobre uma grade de agentes estacionários (*patches*);
- Criação de links entre *turtles* para construir agregados, redes e grafos de agentes;
- Visualização 2D e 3D do modelo;
- Monitores que permitem inspecionar e controlar os agentes.

3. Modelo Utilizado: EpiDEM Basic

Este método representa a irradiação de uma doença infecciosa dentre uma população fechada. Tal protótipo é construído com base em um modelo matemático, denominado SIR, desenvolvido por Kermack e McKendrick, que descreve a prática sistêmica de um fenômeno que inicia-se no momento em que uma pessoa infectada é introduzida em um grupo populacional totalmente predisposto, como já especificado na seção 2.3. Ele tem por principal característica a atribuição de uma população fechada, ou seja, não haverá nascimentos, mortes ou viagens para dentro ou fora da população. Também pressupõe que há mistura única, na qual cada pessoa, no mundo, terá a mesma possibilidade de interagir com qualquer outro indivíduo da população. Já em relação a termos virais, o paradigma supõe que não existem períodos ocultos ou inativos.

Seu funcionamento acontece de maneira em que, os indivíduos andam pelo espaço denominado de forma aleatória. Ao entrar na área de contato de uma pessoa infectada, em qualquer um dos oito vizinhos, ou também no mesmo local, o indivíduo não infectado terá maior probabilidade de contrair a doença. Outra característica é o usuário ter a liberdade de definir o número de pessoas no seu universo, bem como as chances de contrair a doença.

O próximo passo é a pessoa infectada ter a possibilidade de se recuperar. Isso acontece depois da mesma cumprir o tempo de reabilitação, que pode ser estipulado o pelo

¹<https://ccl.northwestern.edu/netlogo/>

usuário. O tempo de recuperação de cada indivíduo é definido através de uma distribuição aproximadamente normal em relação à média do tempo de recuperação já definido. Uma vez recuperado, o indivíduo é permanentemente imune ao vírus.

Em relação às cores dos indivíduos, estas relacionam-se com o estado de saúde, e são divididas em três: para os indivíduos não infectados é usada a cor branca, indivíduos infectados caracterizam-se pela cor vermelha, já os indivíduos verdes serão os recuperados.

Na figura 1, pode-se observar na tela inicial, à direita do processo de simulação do Epidem Basic, no qual foi iniciado com o botão denominado de “setup”. Também v a configuração inicial dos “sliders”, já pré-definidos pelo sistema e podendo ser alterados pelo usuário. Logo após, estão os botões, “setup”, utilizados para montar a simulação, ou seja, executar um procedimento, que, por sua vez, é uma sequência de comandos, e o “go”, para iniciar o processo, usado para que os comandos funcionem repetidamente até o fim da simulação. Percebe-se, ainda, três gráficos, o “Cumulative Infected and Recovered”, que representa a porcentagem total de indivíduos infectados e recuperados ao longo da difusão da doença, o “Infection and Recovery Rates” mostra graficamente as taxas estimadas em que a doença está se espalhando e o “Populations”, que demonstra o número de pessoas infectadas, ou não, durante todo processo.

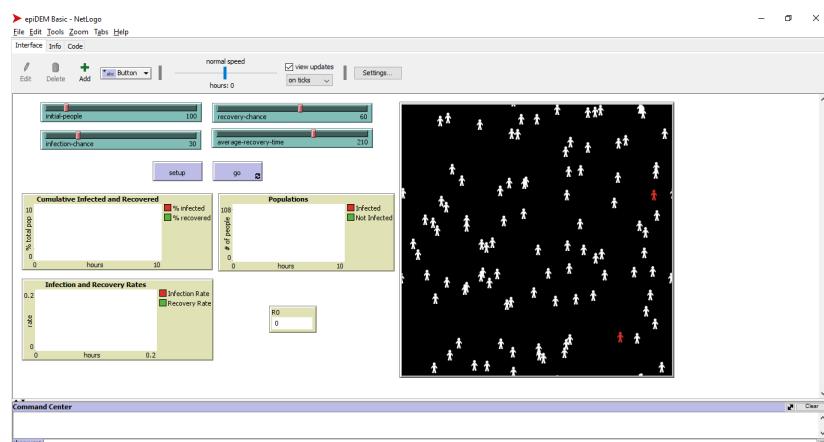


Figura 1. Tela inicial aplicação Epidem Basic

Já na figura 2, é possível perceber o fim da simulação e seus resultados. Na tela à direta, observa-se que todos os indivíduos que estão na cor verde foram curados, concluindo-se que todos os indivíduos foram, antes disso, contaminados. Os gráficos plotados demonstram que o número de infectados e o de recuperados ficaram iguais, também percebe-se que o crescimento da população infectada é inversamente proporcional ao crescimento da população curada.

4. Simulação do EpiDEM Basic com alterações

Nesta seção aborda-se as alterações feitas no programa e suas funcionalidades. Preliminarmente, foi adicionado um “slider” para que o usuário escolhesse o número inicial de infectados, por este motivo não utilizou-se o número inicial calculado através de uma porcentagem, característica do Epidem Basic original. Nesta situação, quanto maior a quantidade inicial, menor é o tempo para infecção dos outros indivíduos.

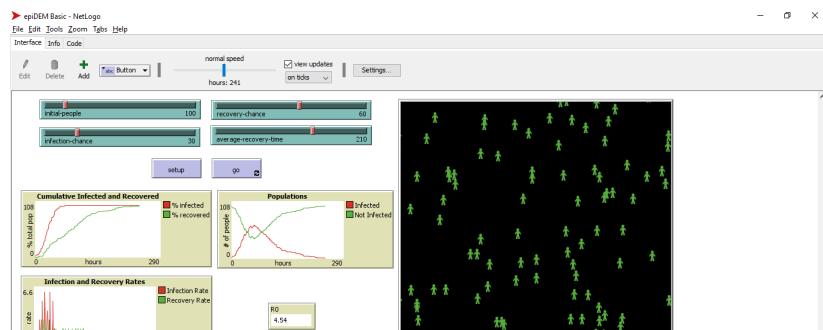


Figura 2. Tela final da aplicação Epidem Basic

Já na etapa posterior, foram adicionados três outros “sliders”, que serão utilizados para diminuir ou aumentar a velocidade dos diferentes agentes, que podem ser manipulados com diferentes configurações. Um exemplo que pode ser descrito, é quando um indivíduo está infectado (vermelho), se torna mais lento que os demais, assim diminuindo a sua propagação e, por sua vez, aumentando o tempo de proliferação do vírus. Contudo, outras configurações podem ser utilizadas, dependendo da simulação a ser realizada.

Outra característica para a personalização foi adicionar um agente denominado “vaccinated”. Esse indivíduo é de fundamental importância para esta etapa do trabalho, pois mostra funcionalidade das vacinas para a população, conscientizando-a a partir disso a não ficar infectado, bem como a estimular outros ao seu redor a se vacinar. Isto acontece através da proximidade dos ”vaccinated”, com os demais sujeitos não infectados ao seu redor. Desse modo, gera-se um multiplicador para novos “vaccinated”.

Por fim, foi adicionado o agente “vaccinated” ao gráfico Populations. Essa nova variável foi acrescentada ao gráfico para que possamos acompanhar e analisar a evolução dos processos.

A figura 3, representa o início da simulação já alterada, e suas novas características, as quais são totalmente personalizáveis, originando inúmeras possibilidades de configurações.

A figura 4, ilustra as alterações e suas aplicações já executadas e finalizadas, essenciais para os estudos.

5. Conclusões

A simulação multiagente permite a facilidade de encontrar a solução para um problema real, podendo assim colocar em prática no ambiente externo, e de alguma forma ajudar em diversas áreas de estudo.

Neste trabalho propõe-se acrescentar algumas funcionalidades em um modelo pré-existente, com o intuito de perceber a capacidade de propagação de uma epidemia. Com este objetivo, ratifica-se perceber a importância da vacinação para evitar esta disseminação, gerando menos problemas de saúde à sociedade e por consequência menos gastos para a economia.

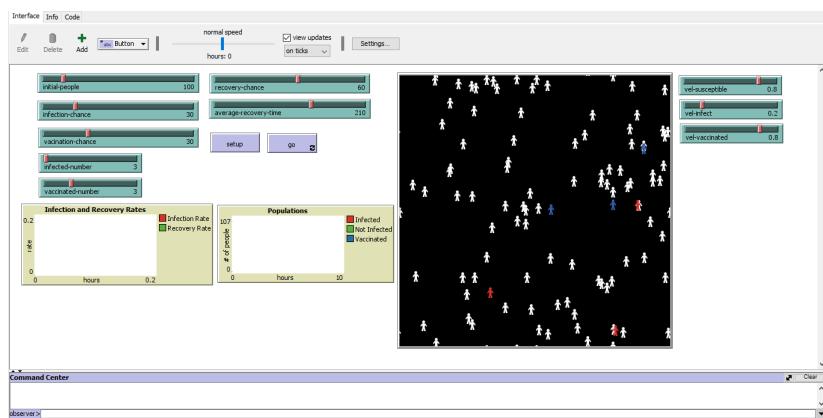


Figura 3. Tela inicial da aplicação Epidem Basic já alterada

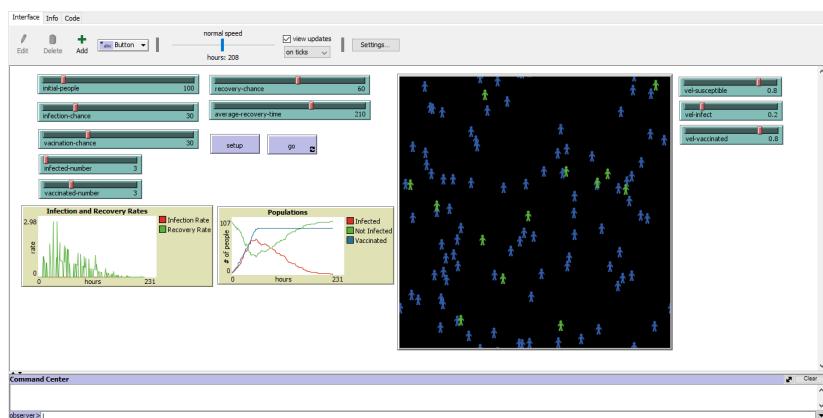


Figura 4. Tela final da aplicação Epidem Basic alterada

1

Estudos ainda devem ser realizados para que o conhecimento possa ser, ainda mais aprofundado, mas já é possível perceber que a simulação multiagente é de grande relevância e importância para ajudar a resolver problemas de diversas áreas de estudos.

Como trabalhos futuros, pode-se realizar um estudo envolvendo o problema matemático SIRS (Suscetível, Infectado, Recuperado, Suscetível) onde simula a disseminação de uma doença contagiosa em que o vírus sofre mutação, podendo assim perder a validade da vacina, e deixar o sujeito vulnerável novamente à doença, conforme ilustra a figura 5.

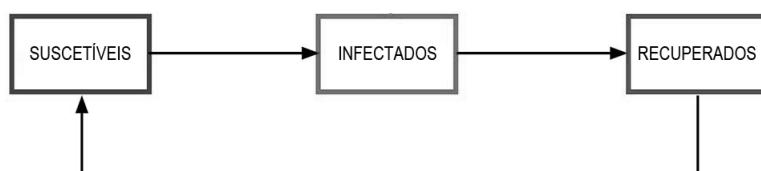


Figura 5. Modelo Matemático SIRS

Agradecimentos

Agradecemos a Universidade Federal do Rio Grande (FURG), ao Programa de Pós Graduação em Modelagem Computacional (PPGMC) e a Coordenação de Aperfeiçoamento de Nível Superior (CAPES) pelo espaço e apoio cedidos.

Referências

- Adamatti, D. F. (2011). Simulação baseada em multiagentes como ferramenta em estudos interdisciplinares.
- Cruz, V. S. (2013). Análise computacional da disseminação de epidemias considerando a diluição e a mobilidade dos agentes.
- De Barros, A. M. R. (2015). Modelos matemáticos de equações diferenciais ordinárias aplicados à epidemiologia. *Revista de Ciências Exatas e Tecnologia*, 2(2):62–67.
- de Lima, T. F. M., Faria, S. D., Soares Filho, B. S., and de Senna Carneiro, T. G. (2009). Modelagem de sistemas baseada em agentes: Alguns conceitos e ferramentas. *Anais XIV Simpósio Brasileiro de Sensoriamento Remoto, Natal, Brasil*, pages 25–30.
- Duarte, C. I. S. (2016). *Vacinas e plantas, relação em larga escala*. PhD thesis.
- Forattini, O. P. (2005). *Conceitos Básicos de Epidemiologia Molecular Vol. 64*. EdUSP.
- Nepomuceno, E. G. (2005). Dinâmica, modelagem e controle de epidemias. *UFMG. Tese de Doutorado*. http://www.cpdee.ufmg.br/defesas/D_534.
- Rebonatto, M. T. (2000). Simulação paralela de eventos discretos com uso de memória compartilhada distribuída.
- Wooldridge, Michael, J. N. R. (1995). Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152.

Developing a smart parking solution based on a Holonic Multiagent System using JaCaMo Framework

Lucas Fernando Souza de Castro¹, André Pinz Borges¹, Gleifer Vaz Alves¹

¹Programa de Pós-Graduação em Ciência da Computação

Universidade Tecnológica Federal do Paraná

Campus Ponta Grossa (UTFPR)

Av Monteiro Lobato, s/n - Km 04 - Ponta Grossa - PR - Brazil

lcastro@alunos.utfpr.edu.br, {gleifer, apborges}@utfpr.edu.br

Abstract. A smart city is described as an intelligent city composed of technological services to improve the human life in a way to make it easier and practical. One of the most costly things to optimize in a smart city is the traffic, particularly, the process of looking for a parking space. Therefore, some computer solutions have been developed to provide a parking space an autonomous way in a smart city, and one of them is using Multiagent System (MAS). This paper proposes a Holonic Multiagent System (HMAS) to assign and manage parking spaces in a smart parking system called Holonic Multiagent Parking System (MAPS-HOLO) developed through JaCaMo Framework which comprises three layers of MAS programming: agent, environment and normative programming. Besides the assign parking space process, the system will be able to handle run-time agents failures in different levels: driver agent, sector agent, and manager agent failure.

1. Introduction

A smart city is a city with technological systems able to make easier the daily routine of the population, such as: pay the municipal property tax through a mobile app, use a public wireless network walking on a park or even find a parking space before leaving home. Considering that a smart city there are many things could be optimized, [Caragliu et al. 2011] introduces six categories which compose a smart city: smart economy, smart mobility, smart environment, smart people, smart living, and smart governance.

The smart mobility comprises systems able to further easier access to move around the city (e.g., walking or driving). In New York, a motorist spends almost 40% of its driving time looking for a parking space [Koster et al. 2014]. Hence, some technological solutions have become common to find a parking space in some cities around the world. For example, cities like San Francisco (USA) [ParkingEdge 2013], Naples (Italy) [Napoli et al. 2014], and São Paulo [Dias and Este 2016] are examples of places where computer solutions provide an easier access to find/reserve a parking space.

The vast majority of research concerning smart parking technology is towards smart cities. Besides the smart city, a parking lot should have some te to be classified as a smart parking [Revathi and Dhulipala 2012]. Most of the technologies used by smart parking employ some intelligent systems, such as fuzzy logic, computer vision, and multiagent systems. These technologies have been used by different scenarios, however, in

[Fraifer and Fernström 2016] the authors presented the main trends in smart parking context. Figure 1 shows a chart of the trends in technology.

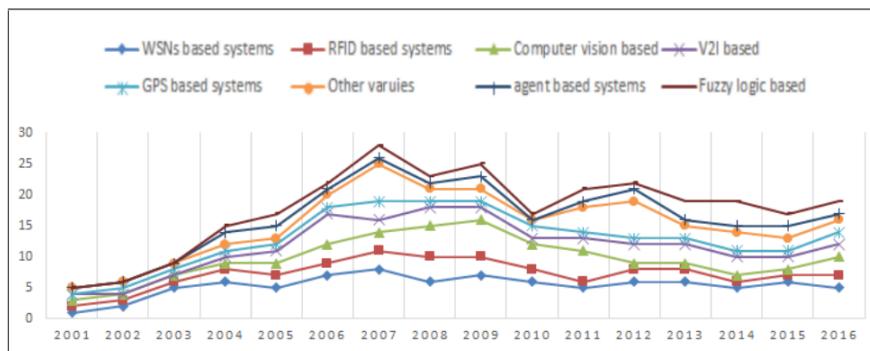


Figure 1. Smart parking technologies trends [Fraifer and Fernström 2016]

As seen in the Figure 1, the most used technology in the smart parking context is the fuzzy logic due to the usage of sensors and their requirement to calibrate and detect errors. Likewise, in the second place is the MAS due to the characteristics of a parking lot (dynamicity of drivers, usage/monitoring parking spaces, sensors, and payments [Idris et al. 2009]). The MAS is being used to build solutions to organize and manage smart parking. Moreover, it could be applied in monitoring a parking space through sensor or a camera [Basavaraju 2015], assign parking spaces and even simulate the environment.

Allocating parking spaces may be a complex task due to the limited amount of them. Hence, it is necessary to evaluate each parking request based on some pattern. One pattern which could be applied is the trust degree value, which is related to agent's commitment to the rules and usage of the system. The trust degree was applied in a MAS through JaCaMo Framework¹, which is a framework with Jason² (agent programming), Cartago³ (artifact development), and Moise⁴ to the normative programming [Boissier et al. 2013] to allocate parking spaces, where the driver agents compete with each other to win a parking space given by a manager agent based on their trust degree value [Castro et al. 2017].

In this paper, we propose a new MAS able to assign and manage smart parking, which indeed is based on a Holonic Multiagent System (HMAS), i.e., a MAS with holons or holonic agents. The term HOLONIC was designed by Arthur Koester in [Koestler 1967] where he showed how the universe and human body could be observed holons. The HOLON is a Greek word which means whole and part simultaneously. Then, this concept applied to Koster's idea results that universe is composed of a whole (universe itself) and its parts (planets, galaxies, etc.) and so on. A holonic agent is an agent capable of having nested agents and to create holarchies (grouping of holonic agents) with other holonic agents [Giret and Botti 2004]. Inside of a holarchy, it is necessary an organizational model govern the agents. A conventional model is the head-body, where a head agent is acting as a leader and the body agents working as cooperators [Fischer et al. 2003].

¹JaCaMo Website: <http://jacamo.sourceforge.net/>

²Jason: <http://jason.sourceforge.net/wp>

³Cartago: <http://cartago.sourceforge.net/>

⁴Moise: <http://moise.sourceforge.net/>

Our HMAS is divided into four categories of agents: *manager, sector, parking space, and driver Agent*. The primary goal of this paper is to propose the development of a system able to assign parking spaces to *Driver Agents* and deal with runtime agent failures. The failures might happen in each agent. However, every failure has to be handled differently according to the agent (*manager, sector, parking space, and driver*). The development of the HMAS will be on JaCaMo Framework. Moreover, one of the goals of MAPS-HOLO is to evaluate the usage of JaCaMo to develop HMAS.

Furthermore, the usage of holonic agents in the MAS applied in a smart parking scenario may have the following advantages: (i) a higher capacity to deal with the failures due to the usage of holarchies; (ii) higher scalability due to the concept of HOLOS, where the HMAS can run as a whole (a group of holarchies) or/and run as part (single holon).

The remainder of this paper is organized as follows. In Section 2 is described the architecture of the HMAS and how we intend to handle the runtime agent failures. Finally, Section 3 presents the final considerations.

2. Proposal of Holonic Multiagent System

The proposed paper is part of a research initiative named MultiAgent Parking System (MAPS), where some systems have been conducted through JaCaMo Framework to provide smart parking. In this section, we present the architecture designed for the MAPS-HOLO as well as discuss some possible agent's failures and how we intend to solve them.

2.1. MAPS-HOLO Architecture: General Overview

The architecture of the MAPS-HOLO is divided into four types of agents: *manager, sector, parking space, and driver agent*. The usage of different types of agents to manage the smart parking provides a lighter workload to the agents due to the task sharing among agents (e.g., parking space monitoring). As it follows, Figure 2 shows a holonic perspective of the system compared with the JaCaMo Framework layers (agent, artifacts, and organizational layer). The diagram A (on the left) presents the holonic representation with the holons and their interactions, whereas the diagram B shows the holons mapped to the JaCaMo perspective.

As seen in the Figure 2, every agent has its own features and duties as follows:

- **Manager Agent:** It receives the parking spaces requests from drivers and validates the inquiries based on their preferences (parking space location, and price). After the validation process, it sends to the *sector agent* the parking space request. Finally, after the answer from *sector*, the agent sends to the driver the result of the request (Success if the driver received a parking space according to its preferences, and failure if the driver did not receive the parking space);
- **Sector Agent:** The *sector agent* is in charge to manage the parking spaces in its sector. After it receives the request from the *manager*, the agent chooses an available parking space and send the request to the *parking space agent*. If the *parking space agent* refuses the request, the *sector agent* has to choose another agent and repeat the process;
- **Parking Space Agent:** It is in charge to observe the physical parking space through sensors. The agent may have three different status: busy (if there is a

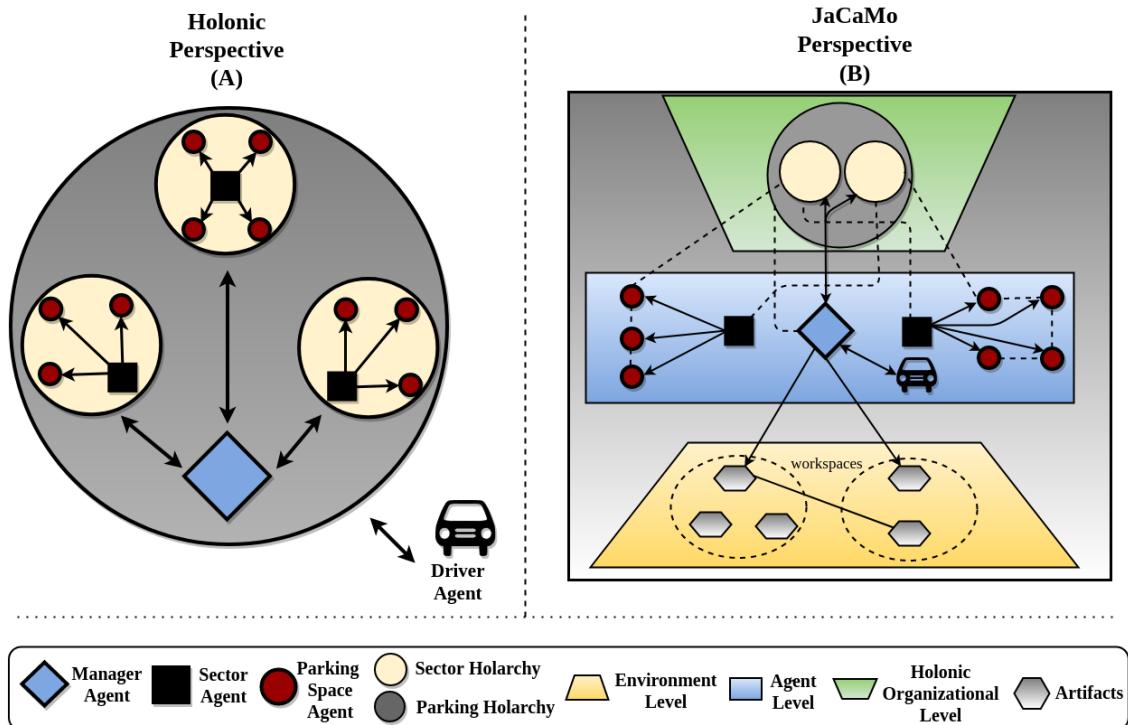


Figure 2. MAPS-HOLO Holonic/JaCaMo - Perspective

driver agent using it), free (available to be assigned) or reserved (when the parking space was assigned to a *driver agent*, but it did not park yet);

- **Driver Agent:** It represents the actual vehicle which needs a parking space. The drivers need to set its preferences (location and price) to request a space. Then, the driver will receive the answer to its request, and it may accept or refuse it.

Although the holarchies of the HMAS could be seen as agent organizations, the holarchy in the MAPS-HOLO will provide more features than a simple organization of agents. Every holarchy owns a contract, which a group of commitments that the agents have to follow [Fischer et al. 2003]. This contract prescribes how the agents interact (e.g., direct messages or broadcasting), rules of resource sharing, and the strategies about how the holarchy handles the failures. The JaCaMo through Moise provides development of organizations with roles and norms to the agents which make viable to create standard holarchies. There are two holarchies (as shown in Figure 2) in the HMAS: *Parking Holarchy*, and *Sector Holarchy*.

- **Parking Holarchy:** It is composed by three groups of agents: *manager*, *sector* and *parking space*. Its head agent is the *manager* who is responsible to coordinate the smart parking environment, and the body agents are the *sector* and *parking space agents*. It has a contract, which is a set of commitments that regulates the holarchy such as: i) Communication among *manager* and *sectors agents*; ii) Range of *sector agents*; iii) Order and priority among *sectors*;
- **Sector Holarchy:** The *Sector Agent* is its head agent, and the body agents are the parking spaces. Its contract specifies how the *parking space agents* perceive the physical parking space (e.g., interval to check sensors) and how the body agents interact with each other.

2.2. Agent Failures

One advantage of using holons is how the agents get connected with the others through holarchies providing a higher capacity to face runtime agent failures that could finish a whole MAS execution. However, the simple usage of holonic agents does not solve the agent failures automatically. Hence, it is necessary to program how the HMAS will handle the failures and check if there is an agent failure. To check if a failure occurred, periodically the agents must inform an artifact with their state. If an agent reached the timeout, the *observer agent* is notified and will handle the failure. The following topics describe how the failures will be dealt with:

- **Parking Space Agent Failure:** The agent notifies the *Sector Agent* of its holarchy. The head agent of its holarchy picks another *parking space agent* to manage the resources from the failed agent;
- **Sector Agent Failure:** Due to the *sector agent* being the head agent of *sector holarchy*, the following steps will be executed: i) All the *parking space agents* of its holarchy will be notified; ii) Another *sector agent* will be notified by the *manager agent* to manage these *parking space agents* notified in the previous step by the *manager*; iii) The notified *sector agent* manages now the new *parking spaces agents*;
- **Manager Agent Failure:** As one may expect, the manager failure will represent many changes in the system in order to properly recover it. All the *sector agents* will be notified about the *manager's* failure and the *Parking Holarchy* will have the head agents changed to multiple head agents (Sector Agents). Hence, the process of assigning/managing parking spaces will be changed to a self-organization and coordination of *sector agents*. Moreover, the *manager* has an important role when the *parking space* and/or *sector agent* fail. Then, to handle these failures a new type of agent will be used: *Observer*. Besides the failures, the *observer agent* will inspect the self-coordination among the *sector agents* to check how they are working. The *observer agent* does not operate when the system starts or when the *manager* runs, the *observer* is employed only when the *manager* fails.

Given, the attempts to handle the failures, we expect that the *driver agent* can use the system without interruptions even if the agents fail. Moreover, our proposal of using an HMAS is to provide a higher capacity to deal with failures and scalability to build a stable system to face the dynamicity of smart parking.

3. Final Considerations

The primary goal of our paper is to propose a smart parking solution based on Holonic Agents. The system will be able to assign and manage parking spaces in smart parking through holonic agents and their holarchies. The usage of holarchies will provide a higher capacity to deal with different types of failures (Manager, Sector, and Parking Space). Finally, whereas the MAPS-HOLO will be finished some contributions could be highlighted: i) HMAS to manage/assign parking spaces in a smart parking; ii) The usage of holonic agents through JaCaMo Framework; iii) Usage of MAPS-HOLO not only for cars, but also different sorts of resources, such as bike, computer time, etc.); iv) Plans to handle runtime agent failures. The next steps to build the MAPS-HOLO are: i) Agent and artifact programming; ii) Mapping holarchies through Moise; iii) Developing holarchies

artifacts; iv) Submitting the HMAS to programmed and random failures; v) Implementing the MAPS-HOLO in a real scenario.

Acknowledgments

Thanks for the support of CAPES by means of the masters scholarship.

References

- Basavaraju, S. R. (2015). Automatic smart parking system using internet of things (iot. *International Journal of Scientific and Research Publications*, 5(12):629–632.
- 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.
- Caragliu, A., Bo, C. D., and Nijkamp, P. (2011). Smart cities in europe. *Journal of Urban Technology*, 18(2):65–82.
- Castro, L. F. S., Alves, G. V., and Borges, A. P. (2017). Using trust degree for agents in order to assign spots in a smart parking. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 6(2):45.
- Dias, E. M. and Este, W. (2016). Sistema Inteligente de Estacionamento Público Integrado-SIEPI. *Journal of Progressive Research in Mathematics*, 7(4):1173–1182.
- Fischer, K., Schillo, M., and Siekmann, J. (2003). Holonic multiagent systems: A foundation for the organisation of multiagent systems. In *HoloMAS*, pages 71–80. Springer.
- Fraifer, M. and Fernström, M. (2016). Investigation of Smart Parking Systems and their technologies. In *Thirty Seventh International Conference on Information Systems. IoT Smart City Challenges Applications (ISCA 2016), Dublin, Ireland*, pages 1–14.
- Giret, A. and Botti, V. (2004). Holons and agents. *Journal of Intelligent Manufacturing*, 15(5):645–659.
- Idris, M., Leng, Y., Tamil, E., Noor, N., and Razak, Z. (2009). Car Park System: A Review of Smart Parking System and its Technology. *Information Technology Journal*, 8(2):101–113.
- Koestler, A. (1967). The ghost in the machine. 1967. *London: Hutchinson*.
- Koster, A., Koch, F., and Bazzan, A. L. (2014). Incentivising crowdsourced parking solutions. In Nin, J. and Villatoro, D., editors, *Citizen in Sensor Networks*, volume 8313 of *Lecture Notes in Computer Science*, pages 36–43. Springer International Publishing.
- Napoli, C. D., Nocera, D. D., and Rossi, S. (2014). Agent negotiation for different needs in smart parking allocation. In *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection - 12th International Conference, PAAMS 2014, Salamanca, Spain, June 4-6, 2014. Proceedings*, pages 98–109.
- ParkingEdge (2013). San francisco - best parking. available in: <http://sanfrancisco.bestparking.com> (accessed in 1st march 2018).
- Revathi, G. and Dhulipala, V. (2012). Smart parking systems and sensors: A survey. In *Computing, Communication and Applications (ICCCA), 2012 International Conference on*, pages 1–5.

Agente inteligente Aplicado em Jogos Digitais

Patrick P. Rodrigues¹, João L.T. da Silva¹

¹Centro Universitário e Faculdade de Tecnologia (UNIFTEC)
95012-669 – Caxias do Sul – RS – Brasil

patrickprodriques@hotmail.com, joaoluis.tavares@gmail.com

Abstract. *Currently the area of Digital Games tends to offer more and more realism and immersiveness, promoting a better experience to the player. Along with this the player expects an increasingly natural behavior of NPCs (Non-Player Character), looking for a more natural adaptation of the NPC's interactions in the context of the game. In this sense, Artificial Intelligence, in its area of Multiagent Systems, offers a promising technology to implement intelligent cognitive NPCs. However, specific features in game engines such as real-time, synchronization and communication aspects are not easily supported in terms of multiagent platforms, making it difficult to fully integrate AI techniques in this context. On the other hand, the agent autonomy feature may offer benefits for more convincing gameplay, and even for serious games, for instance. In this article, we focus on modeling a decisional component in an NPC agent to define an intelligent behavior of the NPC, which is adaptive according to the specific playing of each player. We intend, with these initial ideas and prototypes, to experiment the concept of intelligent agents applied to the area of digital games, using an artificial neural network as a decisional component.*

Resumo. *Atualmente a área de Jogos Digitais procura oferecer cada vez mais realismo e imersividade, promovendo uma melhor experiência ao jogador. E o jogador espera um comportamento cada vez mais natural dos NPCs (Non-Player Character), procurando uma adaptação mais natural das interações com o contexto do jogo. Neste sentido, a Inteligência Artificial, na sua área de Sistemas Multiagentes, oferece uma tecnologia promissora para implementar NPCs cognitivos inteligentes. No entanto, características específicas em motores de jogos, como aspectos de tempo real, sincronização e comunicação não são facilmente compatíveis em termos de plataformas multiagentes, dificultando a completa integração de técnicas de IA neste contexto. Por outro lado, a característica de autonomia dos agentes pode oferecer benefícios para uma jogabilidade mais convincente, sendo até necessária para jogos sérios, por exemplo. Neste artigo, vamos nos concentrar na modelagem de um componente decisional em um agente NPC para definir um comportamento inteligente do NPC, que seja adaptativo de acordo com a maneira de jogar específica de cada jogador. Esperamos, com estas ideias e protótipo iniciais, experimentar o conceito dos agentes inteligentes aplicado à área de jogos digitais, utilizando uma rede neural artificial como componente decisional.*

1. Introdução

Atualmente, os jogos digitais estão presentes na vida de muitas pessoas, sendo excelentes formas de entretenimento, possuindo as mais variadas temáticas e mecânicas. Independ-

dente do estilo, normalmente a Inteligência Artificial (IA) é um componente presente em todos os gêneros. Seja ela aplicada no controle dos NPCs (*Non-Player Character*) ou nas dinâmicas contidas no cenário, seu uso possui uma alta relevância na experiência final proporcionada pelo jogo.

Desenvolver um componente de IA para um jogo pode representar um desafio, visto que a uma certa complexidade para que o produto final proporcione o desafio correto aos jogadores, pois cada um tem seu próprio estilo de jogo e a IA deve conseguir lidar com pelo menos a maioria desses perfis. Se a IA não for desenvolvida de uma forma satisfatória, afim de proporcionar o equilíbrio ideal, os jogadores podem acabar perdendo o interesse no jogo. Existem algumas vertentes que buscam maneiras de solucionar esse tipo de problema, como é o caso do trabalho em [Fogel et al. 2004] o qual visa desenvolver comportamentos adaptativos de vários NPCs através de abordagem evolutiva, aproximando de um comportamento mais humano. [Miikkulainen 2006] também usa algoritmos genéticos para a geração dos parâmetros e pesos de uma rede neural, para a adaptação sem um alvo explícito.

Este trabalho apresenta um conceito de como melhorar a experiência do jogador ao interagir com NPCs, utilizando agente inteligente adaptativo, com uma Rede Neural Artificial (RNA) como componente decisional do agente. Consideramos aqui, que um agente inteligente é uma entidade contínua e autônoma capaz de atuar em um determinado ambiente, tomando suas próprias decisões [Leyton-Brown and Shoham 2008]. Com o uso da RNA esse agente pode adquirir a capacidade de aprender com a sua experiência de jogo, assim como um jogador humano faz e dessa forma oferecer ao *player* uma experiência diversificada e mais imersiva.

2. Projeto de RNA para um Agente adaptativo de jogo

Para demonstração desse projeto foi desenvolvido um jogo onde o jogador possui dois comandos. Os comandos são representados por *0* e *1*, em um deles o jogador dispara um míssil na posição *0* e no outro na posição *1*, o papel da RNA é o de fazer o agente inteligente (*bot*) prever qual será a próxima jogada realizada pelo jogador, a partir dos comandos utilizados anteriormente. A previsão é exibida ao jogador de forma visual, com um objeto aparecendo na posição que o *bot* previu para a jogada seguinte. Os componentes utilizados no jogo podem ser vistos na Figura 1.



Figura 1. Protótipo do jogo

3. Protótipo

Para realização do projeto, foi desenvolvido um protótipo com três agentes com funções específicas: *Engine*, *API* e *RNA*. Essa divisão possibilita a comunicação entre as diferentes abordagens. A divisão é necessária pois os diferentes agentes possuem diferentes tecnologias e diferentes características, cada uma com seu propósito, necessitando assim de uma *API* para comunicação.

3.1. Decisão Adaptativa

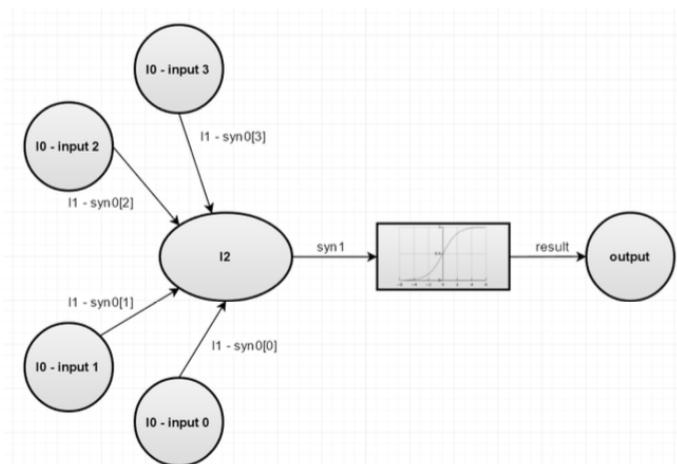
O módulo de rede neural utiliza dois modos: o modo de treinamento (*training*), onde recebe o log em *JSON* gerado pela *engine* através da *API*. O treinamento realiza todas as interações necessárias visando minimizar os erros e ajustar seus pesos sinápticos. O modo de previsão (*prediction*), recebe apenas as entradas e tenta prever a saída esperada com os conhecimentos adquiridos no modo de treinamento. A estrutura do *JSON* (figura 2) contém quatro entradas para uma saída onde o arquivo de treinamento é a lista de golpes que o jogador executou durante o jogo, separados em grupos de cinco. Os primeiros quatro ataques são considerados como entradas e o último, a saída. Usando esses dados a RNA é capaz de treinar para prever o quinto ataque do próximo grupo de ataques que será executado pelo jogador.

```
[  
  {  
    "inputs": [[0, 0, 1, 1],  
               [0, 1, 1, 0],  
               [1, 0, 1, 0],  
               [1, 1, 1, 1]],  
    "outputs": [[0],  
                [1],  
                [1],  
                [0]]  
  }  
]
```

Figura 2. Exemplo de JSON de treinamento

A RNA do agente é dividida em três camadas: a camada *l0*, contendo as representações dos comandos, que podem ser *0* ou *1*. A segunda camada, *l1*, onde ficam os pesos sinápticos da primeira camada. Após cada interação, a camada *l2* recebe as entradas *l0* multiplicadas pelos pesos sinápticos *syn0*. Após, o resultado em *l2* é multiplicado pelo valor correspondente a sua sinapse de saída *syn1*, e esses valores passam pela função de ativação sigmoide, gerando o *output*. Na etapa de treinamento é calculado o erro dessa saída em comparação com as saídas fornecidas pelo usuário. A estrutura básica da RNA, pode ser vista na figura 3

Após todo ciclo e suas repetições, o agente é capaz de atingir um valor próximo ao ideal para seus pesos sinápticos. A Figura 4 ilustra as sinapses (*syn0* e *syn1*) sendo inicializadas com valores aleatórios, a multiplicação das entradas pelos valores contidos nas sinapses e os cálculos visando minimizar o erro. No final de todo processo os valores contidos nas sinapses são salvos em *JSON* e estão prontos para serem utilizadas pelo método de previsão.

**Figura 3. Estrutura da RNA**

```
[
  {
    "syn0": [[2.1527945143164984, 1.3328690694820238, -2.4662028383481283, -2.824410357116697, -0.7805288111175995],
              [-2.85059313369365, -2.14888078981325, -0.13428335506776606, 3.2653073976903024, 0.14077593354945456],
              [-0.8757825358301803, -0.3943343828932792, 0.9457808716655521, 1.3036759132368, -1.0350996747507961],
              [5.136615479835274, 3.505712793504951, -2.4098894988745876, -6.210780982518584, -0.4371286576833272]],
    "syn1": [[[6.595451483743302],
               [4.2945636133835166],
               [-3.8599390791562476],
               [-8.676074066650129],
               [-0.5614033160011409]]
    ]
  }
]
```

Figura 4. Exemplo de JSON das sinapses

No modo de previsão, o agente captura apenas quatro ataques feitos pelo usuário e envia-os para a API que realiza a chamada do método de previsão na RNA. O método de previsão recebe as quatro entradas do usuário e as sinapses que foram calculadas anteriormente no método de treinamento, onde são multiplicados pelos pesos sinápticos e o resultado obtido será a previsão da quinta jogada do usuário. Esse resultado será então enviado para o agente executor (*engine*), que será responsável por executar a ação correspondente de defesa do *bot*. Tudo isso acontece em tempo de jogo.

3.2. Testes e resultados

Os experimentos dividem-se em teste de conceito e teste de ambiguidade, considerando a ordem de execução dos padrões, suas entradas e saídas, valor previsto pela RNA e erro de treinamento do padrão da jogada. A previsão pode ser considerada correta quando o valor da saída coincide com o valor previsto pela RNA.

3.2.1. Teste de conceito

Para o teste de conceito do sistema, foram executadas dez rodadas, com diversos padrões sem ambiguidade, onde a RNA apresentou um bom desempenho, mesmo com um número relativamente baixo de execuções. Na Tabela 5, observa-se que 80% dos padrões foram identificados corretamente. Quanto mais acontece a repetição de um padrão, mais

o agente infere a previsão da saída do padrão, fazendo assim com que seu erro de treinamento diminua, mesmo que esse padrão passe por uma futura ambiguidade seu valor de previsão será mantido, até que o padrão com a nova saída ultrapasse as repetições do padrão antigo.

Execução	Entradas	Saídas	Previsto	Erro	Resultado
1	0,1,0,1	1	0	1,44%	Erro
2	0,1,0,0	0	1	2,42%	Erro
3	0,1,0,1	1	1	1,89%	Acerto
4	0,1,1,1	1	1	1,53%	Acerto
5	0,0,0,0	0	0	1,96%	Acerto
6	0,1,1,1	1	1	1,71%	Acerto
7	0,1,0,0	0	0	1,52%	Acerto
8	0,1,1,1	1	1	1,40%	Acerto
9	0,0,0,0	0	0	1,38%	Acerto
10	1,0,0,0	0	0	1,27%	Acerto

Figura 5. Teste de conceito

3.2.2. Teste de ambiguidade

Com o intuito de demonstrar a capacidade de adaptação do agente inteligente, realizou-se um teste com padrões ambíguos, onde na primeira e segunda execução foram executados o padrão $(0,1,0,1)$ com a saída sendo definida como (0) , para as demais jogadas o mesmo padrão $(0,1,0,1)$ foi utilizado, porém com a saída alterada para (1) . A Tabela 6 ilustra a assimilação da modificação do padrão em três execuções. Mesmo com erro de 44,44%, devido as ambiguidades causadas pelas saídas anteriores, o agente conseguiu assimilar a mudança, prevendo corretamente a nova saída esperada na sexta execução.

Execução	Entradas	Saídas	Previsto	Erro	Resultado
1	0,1,0,1	0	0	1,87%	Acerto
2	0,1,0,1	0	0	1,56%	Acerto
3	0,1,0,1	1	0	44,44%	Erro
4	0,1,0,1	1	0	50%	Erro
5	0,1,0,1	1	0	48%	Erro
6	0,1,0,1	1	1	44,44%	Acerto

Figura 6. Teste de ambiguidade

4. Discussão e Conclusão

Existem diversas técnicas para o desenvolvimento de agentes inteligentes capazes de atuar em um ambiente dinâmico. Na área de jogos uma das técnicas é a *dynamic scripting*, que utiliza um conjunto de regras e aprendizagem não supervisionada para adaptar o comportamento do *bot* ao contexto do jogo de forma rápida e eficiente. Esse tipo de abordagem facilita o contexto da implementação, pois é um método mais ágil de processamento, entretanto a ação da IA, embora adaptativa fica restrita a um conjunto de regras de base

[Spronck et al. 2003]. Métodos supervisionados de aprendizagem como as RNAs, normalmente não são utilizados em jogos, devido a demanda na performance de processamento, pois os requisitos de tempo são difíceis de atingir. Entretanto, o uso desse tipo de técnica sob certos critérios, como quantidade controlada de variáveis, pode ser utilizado em jogos, adequando-se ao requisito de tempo de processamento rápido necessário para uma jogabilidade fluida. Também pode proporcionar um certo nível de adaptação ao agente, já que não necessita seguir um conjunto pré concebido de regras, como o *dynamic scripting*, podendo ser baseada exclusivamente nos resultados gerados através da interação dos jogadores.

Técnicas de aprendizagem supervisionada podem ser utilizadas também em jogos tradicionais, sendo capazes também de criar um jogador “perfeito”, como no caso do *Google Go*, uma IA capaz de jogar Go vencendo grandes jogadores humanos. Para isso a equipe de desenvolvimento do Google utilizou redes neurais, responsáveis por avaliar as posições e a possível eficácia dos movimentos realizados no tabuleiro. Além disso foi utilizado o método de Monte Carlo, para realizar as simulações necessárias das milhares de jogadas possíveis de ocorrem no jogo [Silver et al. 2016].

A utilização de uma RNA como componente decisional no agente inteligente, no presente trabalho, apresentou resultados que demonstram a adaptabilidade do agente, conseguindo aprender a maneira com que o jogador interage com o sistema. Isso faz com que o jogo ofereça uma experiência diferenciada, capaz de reconhecer a maneira de jogar de cada um. Porém as técnicas utilizadas demandam um certo poder de processamento, algo que não é muito evidente em um jogo simples, como no caso do desenvolvido para o projeto. Por isso essa metodologia deve ser utilizada preferencialmente em jogos com um número controlado de possibilidades, dessa forma pode ser apresentado um diferencial ao jogo, melhorando a jogabilidade e a diversão proporcionada, conseguindo assim despertar maior interesse por parte dos jogadores.

Referências

- Fogel, D. B., Hays, T. J., and Johnson, D. R. (2004). A platform for evolving characters in competitive games. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 2, pages 1420–1426 Vol.2.
- Leyton-Brown, K. and Shoham, Y. (2008). *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Morgan and Claypool Publishers, 1st edition.
- Miikkulainen, R. (2006). Creating intelligent agents in games. *The Bridge*, pages 5–13.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503.
- Spronck, P., Sprinkhuizen-kuyper, I., and Postma, E. (2003). Online adaptation of game opponent ai in simulation and in practice. In *In Proceedings of the Fourth International Conference on Intelligent Games and Simulation*, pages 93–100.

Explorando a Comunicação entre Sistemas Multi-Agentes Embarcados em Ambientes Inteligentes para IoT: Uma Proposta de Laboratório

Palloma da S. M. Nunes^{1,2}, Igor M. de Almeida¹, Thiago C. Picanço¹, Carlos E. Pantoja^{1,2}
Leandro M. Samyn¹, Vinicius S. de Jesus¹, Fabian C. P. B. Manoel¹

¹Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET-RJ)
Campus Maracanã – 20271-110 – Rio de Janeiro – RJ – Brasil

²Departamento de Ciência da Computação
Universidade Federal Fluminense (UFF)
Campus Praia Vermelha – 24210-346 – Niterói – RJ – Brasil

{pallomapit,igor.m.almeida,thiagoc.picanco}@hotmail.com

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

Abstract. This work aims to present an Intelligent Laboratory of Autonomous Systems (LISA), which consists in an Ambient Intelligence (AmI) system with devices connected to an Internet of Things (IoT) architecture and supported by the agents approach. Embedded Multi-Agent Systems are able to manage a laboratory in a cognitive way, controlling electronic devices and intelligent systems connected to each other through the IoT. The LISA uses the ContextNet middleware for the interconnection of embedded devices with SMA and other systems of the laboratory. The embedded devices uses Javino as the connection between the embedded multi-agent systems programmed in Jason with the electronic components of the device.

Resumo. Este trabalho tem como objetivo apresentar o Laboratório Inteligente de Sistemas Autônomos (LISA), que consiste de um Ambiente Inteligente (AmI) com dispositivos conectados à uma arquitetura da Internet das Coisas (IoT) e apoiado pela abordagem de agentes. Sistemas Multi-Agentes embarcados são capazes de gerenciar um laboratório de forma cognitiva, controlando dispositivos eletrônicos e sistemas inteligentes conectados entre si através da IoT. O LISA utiliza o middleware ContextNet para interconexão dos dispositivos embarcados com SMA e os demais sistemas do laboratório. Os dispositivos embarcados utilizam o Javino como conexão entre os sistemas multi-agentes embarcados programados em Jason com os componentes eletrônicos do dispositivo.

1. Introdução

A Internet das Coisas — ou *Internet of Things* (IoT) [Zhang et al. 2012] — é um conceito relacionado à conexão de dispositivos eletrônicos utilizados no dia-a-dia à Internet e com capacidade computacional e autonomia para auxiliar pessoas. Os Ambientes Inteligentes — ou *Ambient Intelligence* (AmI) [Augusto Wrede et al. 2010] — são sistemas que utilizam técnicas computacionais para permitir o gerenciamento de um ambiente de forma pervasiva e pro-ativa. O AmI faz parte da área da computação Ubíqua [de Araujo 2003],

onde os usuários podem estar inseridos no ambiente sem que a tecnologia utilizada seja percebida por eles. Tais ambientes também podem ser construídos utilizando tecnologias para a IoT.

Diversos ambientes inteligentes utilizam técnicas de inteligência artificial em sua arquitetura, e dentro dessas técnicas, a abordagem de agentes tem sido explorada. Os agentes são entidades cognitivas capazes de se comunicar e agir pro-ativamente e estão inseridas em um ambiente que pode ser tanto simulado quanto real. Já, os Sistemas Multi-Agente (SMA) são um conjunto de agentes organizados de forma a interagirem para atingirem objetivos comuns ou conflitantes [Wooldridge 2009]. Dentre as diversas linguagens existentes para o desenvolvimento de SMA, é possível destacar o *framework* Jason [Bordini et al. 2007], que possui um interpretador para o *AgentSpeak* [Rao 1996] baseados na arquitetura *Belief-Desire-Intention* (BDI).

O objetivo deste trabalho é propor o desenvolvimento de um ambiente inteligente explorando a utilização das tecnologias de SMA e da IoT para prover um ambiente aberto onde possam existir dispositivos inteligentes embarcados com SMA, além de dispositivos móveis, como *smartphones*, não inteligentes ou sem SMA, os quais estarão interconectados através da infraestrutura de IoT (um servidor *gateway*). No caso da infraestrutura não estar disponível, os SMA embarcados poderão funcionar autonomamente. A infraestrutura suportará diferentes SMA, os quais são capazes de controlar um laboratório de ensino e pesquisa visando auxiliar seus frequentadores, fazendo com que estes possuam mais conforto e segurança ao utilizar o laboratório.

Para o desenvolvimento dos SMA será utilizado o Jason, por já existir soluções que permitem a integração com *hardware* [Pantoja et al. 2016b]. Contudo essa integração não permite a comunicação entre *hardwares* diferentes. Para isso, o *middleware* para IoT ContextNet [Endler et al. 2011], é utilizado pois é capaz de realizar a comunicação dos dispositivos inteligentes através da rede *wireless* provendo um ambiente aberto para os SMA. Será apresentada uma prova de conceito simples com os agentes para provar a comunicação em um ambiente com IoT, tendo-os como uma alternativa a dispositivos que são apenas repetidores de informações, buscando possibilitar uma maior autonomia aos mesmos.

Na próxima seção serão mostrados os trabalhos relacionados; na terceira seção, a metodologia do projeto; na quarta seção, informações sobre o Laboratório Inteligente de Sistemas Autônomos; na quinta seção, os testes e resultados realizados no laboratório; já na sexta seção será mostrada a conclusão do trabalho e; em seguida, as referências utilizadas neste artigo.

2. Trabalhos Relacionados

Existem trabalhos na literatura que aplicam SMA em ambientes ubíquos e pervasivos, porém não focam na possibilidade de existir diversos dispositivos embarcados gerenciados cada um por um SMA e que são capazes de se comunicar entre si. Uma abordagem de interação entre o meio físico (*hardware*) e o meio cognitivo (*software*) explorando o pior caso no que diz respeito a disposição de uma capacidade limitada de controladores (apenas um) em um banheiro inteligente foi proposta, porém, o SMA é limitado ao protótipo desenvolvido e não é escalável [Branda et al. 2017], ou seja, não permitindo a entrada de novos dispositivos sem ter que reprogramar o sistema.

Igualmente, a utilização de SMA em AmI pode ser visto em um modelo de *Smart Home* [Pantoja et al. 2016a], que é gerenciado por um SMA fechado, portanto, sem uso de meios de comunicação entre eventuais SMA ou dispositivos que venham a estar presentes neste ambiente. Também existem trabalhos voltados *Smart Homes* como em [Andrade et al. 2016], porém este utiliza um ambiente simulado e não faz menção ao uso e emprego de *middlewares* específicos para IoT. Além disso, a solução é fechada, visto que novos dispositivos não podem entrar ou sair do ambiente sem que haja uma alteração no projeto do SMA.

3. O Laboratório Proposto

O Laboratório Inteligente de Sistemas Autônomos (LISA) é um laboratório inteligente construído com base em AmI e IoT que pretende possibilitar a recepção de pessoas de forma autônoma, de modo que, os usuários (previamente cadastrados) tenham seu acesso permitido através de um cartão o qual contém um identificador. Uma vez consentida a entrada, o ambiente será capaz de se adaptar ao usuário liberando algumas funcionalidades para que ele possa interagir com os dispositivos ali presentes como: porta, luz e ar condicionado, bem como os sensores de luminosidade, temperatura e RFID, que estarão interligados no laboratório. Em vista disso, cada dispositivo inteligente será embarcado com um SMA autônomo e serão capazes de trocar informações necessárias com outros SMA através de um *middleware* para IoT.

Estas ações serão escolhidas de acordo com as preferências dos usuários que tiveram o acesso permitido à sala. Por exemplo, se o usuário tem preferência por temperaturas baixas e o sensor presente no ambiente detecta uma alta temperatura, o ar condicionado, se desligado, deve ligar para resfriar o ambiente. Vale lembrar que situações conflituosas de preferência podem ocorrer e, para isso, poderão ser adotadas diferentes soluções como a hierarquia (professor e aluno) dando prioridade ao desejo do professor; já a segunda utiliza o quantitativo como prioridade, ou seja, se mais usuários desejam determinada faixa de temperatura, por exemplo, o ambiente reagirá para atender tal demanda.

O LISA funcionará com o uso de SMA embarcados em dispositivos para controle pró-ativo das partes eletro-eletrônicas. O uso dos SMA embarcados será relacionada aos dispositivos eletro-eletrônicos presentes no ambiente, como circuitos controladores contendo dispositivos como relé, contadora e microcontrolador ATMEGA que serão montados para possibilitar os acionamentos; e a que utiliza um banco de dados relacional, que armazenará as informações de cadastro dos usuários e suas preferências.

Vale ressaltar que os SMA embarcados receberão dados dos sensores do laboratório (1 responsável por um RFID — que controla a entrada de pessoas, outro responsável pela luminosidade e temperatura) e existirão SMA que serão responsáveis por enviar comandos aos atuadores (1 SMA para a porta, 1 SMA para lâmpadas e ar condicionado). Além disso, os SMA que recebem informações dos sensores avaliarão os dados recebidos da camada física do ambiente e as informações cadastradas no banco de dados, para tomar decisões e enviar os comandos aos SMA ligados aos atuadores, através da comunicação entre diferentes SMA.

3.1. Comunicação entre SMA Embarcados

O fluxo de comunicação entre os SMA embarcados se dará da seguinte maneira: os SMA ligados aos sensores serão os responsáveis por enviar as mensagens de acionamento aos

SMA ligados aos atuadores. Para a comunicação entre os controladores e o SMA será utilizada uma arquitetura customizada chamada ARGO [Pantoja et al. 2016b], que faz uso do Javino [Lazarin and Pantoja 2015] para possibilitar a comunicação entre uma linguagem de alto nível a um controlador com linguagem de baixo nível utilizando a porta serial; essa arquitetura consiste em uma arquitetura customizada de agentes. Já para a comunicação entre diferentes SMA, é proposta uma arquitetura customizada responsável pela criação de agentes Comunicadores, que utilizam o *middleware* ContextNet para via- bilizar a troca de dados entre diferentes SMA embarcados em um mesmo ambiente. Na Figura 1, é exemplificado o uso dos agentes ARGOS e de agentes Comunicadores na arquitetura de um ambiente inteligente.

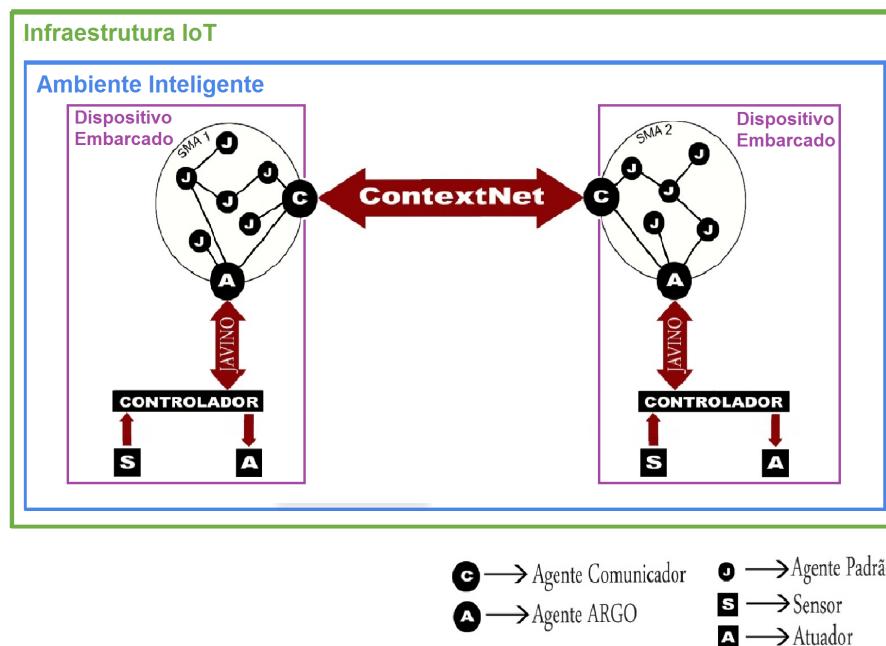


Figura 1. A arquitetura conceitual da comunicação entre dois SMA distintos em um ambiente inteligente utilizando a proposta deste trabalho.

3.2. Testes e Resultados

Para avaliar a possibilidade da criação do LISA, alguns testes foram realizados, como a tentativa de realizar a comunicação entre dois SMA ligados a computadores distintos, sem nenhuma conexão física entre eles e com *hardware* (sensores e atuadores) dedicados a estes SMA. Para efetivar a comunicação entre um SMA remetente e um SMA destinatário, em cada um existiam dois agentes criados através das arquiteturas *ARGO* e *Communicator*. O *agenteARGO* do SMA remetente possuía planos para verificar as percepções recebidas dos sensores e, de acordo com estas, enviar mensagens para o *agenteRemetente* presente no mesmo SMA. Já este *agenteRemetente* (arquitetura *Communicator*), por sua vez, enviava - através da ação interna customizada *sendOut* - uma mensagem para o *agenteDestinatario*, localizado em um SMA distinto e embarcado em outro dispositivo.

O *agenteDestinatario*, também criado com a arquitetura *Communicator*, possui planos feitos de acordo com as possíveis mensagens que poderiam ser recebidas. A cada plano ativado, este agente comunicador envia uma determinada mensagem para um *agenteARGO* do SMA destinatário, que é responsável por enviar os comandos aos atuadores

(LEDs) de seu SMA. Vale lembrar que entre os sensores e o SMA Remetente e entre os atuadores e o SMA Destinatário existem microcontroladores ATMEGA como intermediário para essa comunicação e que nas arquiteturas *ARGO* e *Communicator* foram usados, respectivamente, o Javino e o ContextNet.

Como resultados iniciais, pode-se observar que, quando o sensor de temperatura detectava uma temperatura pré-definida como alta, o LED representativo do atuador ar condicionado acendia. O exemplo anterior foi também aplicado aos demais sensores (luminosidade e botões de acionamentos), como mostra a Figura 2, que apresenta (na parte esquerda) o botão sendo acionado e (na parte direita) o LED correspondente à porta sendo aceso.

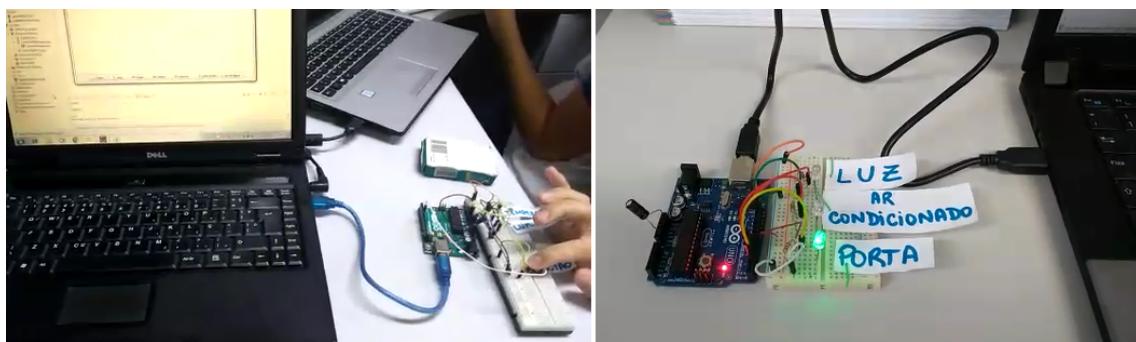


Figura 2. Exemplo de funcionamento utilizando a conexão entre SMA distintos localizados em notebooks também distintos

Dados os resultados, pretende-se fazer outros testes: o primeiro, realizando interações entre os SMA e o banco de dados, fazendo uso de uma ação interna customizada para acesso às preferências dos usuários. Já o segundo, realizando interações entre os SMA e a camada física do laboratório, através da confecção de circuitos eletro-eletrônicos, que possibilitará o acionamento dos dispositivos mais robustos (como ar condicionado) da sala. Além destes, propõe-se testes de performance mais específicos: como testes relacionados à tomada de decisões avaliando mais cuidadosamente o que deve ser acionado, para evitar que dispositivos eletro-eletrônicos não danifiquem pelo ligamento e desligamento contínuo. Para tais testes, será preciso tratar o envio de comandos dos agentes aos microcontroladores de modo a tornar o agente mais cognitivo do que reativo.

4. Conclusão

Este trabalho apresentou uma proposta de ambiente inteligente, que utiliza diferentes SMA embarcados se comunicando entre si, utilizando o *middleware* ContextNet para IoT. A solução proposta neste trabalho permite que sistemas embarcados com SMA utilizando o Jason, ARGO e agentes Comunicadores possam interagir com outros sistemas embarcados presentes em Ambientes Inteligentes distintos. A maioria dos *frameworks* para desenvolvimento de soluções para agentes não contam com uma abordagem preparada para ser escalável e direcionada para a Internet das Coisas, e as que possuem alguma forma de comunicação, utilizam a atual infraestrutura de rede para comunicação e não oferecem uma interface nativa para controladores com sensores e atuadores.

Pode-se destacar também que esta abordagem é aplicável a diversos domínios de aplicação pois se baseiam em arquiteturas customizáveis e não em soluções específicas

e direcionadas. Utilizando a arquitetura customizada proposta, é possível fazer com que diferentes *Smart Homes* e Ambientes Inteligentes consigam interagir entre si. Como trabalhos futuros pretende-se criar modelos para organizações sociais de agentes físicos, visando a possibilidade de obtenção de sistemas organizacionais de dispositivos inteligentes em uma rede IoT.

Referências

- Andrade, J. P. B., Oliveira, M., Gonçalves, E. J. T., and Maia, M. E. F. (2016). Uma abordagem com sistemas multiagentes para controle autônomo de casas inteligentes. *XIII Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*.
- Augusto Wrede, J., Nakashima, H., and Aghajan, H. (2010). Ambient intelligence and smart environments: A state of the art. pages 3–31.
- Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. John Wiley & Sons Ltd.
- Brandao, F., Nunes, P., de Jesus, V. S., Pantoja, C. E., and Viterbo, J. (2017). Managing natural resources in a smart bathroom using a ubiquitous multi-agent system.
- de Araujo, R. B. (2003). Computação ubíqua: Princípios, tecnologias e desafios. In *XXI Simpósio Brasileiro de Redes de Computadores*, volume 8, pages 11–13.
- 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.
- Lazarin, N. M. and Pantoja, C. E. (2015). A Robotic-Agent Platform for Embedding Software Agents using Raspberry Pi and Arduino Boards. In *9th Software Agents, Environments and Applications School*.
- Pantoja, C. E., de Jesus, V. S., and Filho, J. V. (2016a). Aplicando sistemas multi-agentes ubíquos em um modelo de smart home usando o framework jason.
- Pantoja, C. E., Stabile, M. F., Lazarin, N. M., and Sichman, J. S. (2016b). ARGO: An extended jason architecture that facilitates embedded robotic agents programming. In Baldoni, M., Müller, J. P., Nunes, I., and Zalila-Wenkstern, R., editors, *Engineering Multi-Agent Systems: 4th International Workshop, EMAS 2016*, pages 136–155. Springer.
- 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 Multi-Agent Systems*. Wiley.
- Zhang, D., Ning, H., Xu, K. S., Lin, F., and Yang, L. T. (2012). Internet of things. *J. UCS*, 18:1069–1071.

Modelo de Raciocínio e Protocolo de Negociação para um Estacionamento Inteligente com Mecanismo de Negociação Descentralizado

Felipe Felix Ducheiko¹, Gleifer Vaz Alves¹

¹Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná (UTFPR)
Caixa Postal 84.016 – 210 – Ponta Grossa – PR – Brasil

feliipeducheiko@alunos.utfpr.edu.br, gleifer@utfpr.edu.br

Abstract. *In large cities usually a driver will waste time or will drive a long way until find a free parking spot. Therefore, wasting fuel and generating traffic jams. Thinking about it the MAPS project (MultiAgent Parking System) is designed to develop a smart parking using multi-agent system techniques. This multiagent system to allocate parking spots has a centralized negotiation mechanism. The present work proposes a reasoning model and negotiation protocol to implement a decentralized negotiation mechanism in the MAPS project.*

Resumo. *Em grandes cidades normalmente um motorista gasta uma considerável parcela de tempo ou percorre longas distâncias para encontrar uma vaga de estacionamento livre, ocasionando desperdício de combustível e gerando engarrafamentos. Pensando nisto o projeto MAPS (MultiAgent Parking System) é idealizado com objetivo de desenvolver um smart parking (estacionamento inteligente) utilizando técnicas de sistema multiagente. Este sistema multiagente para alocação de vagas de estacionamento possui um mecanismo de negociação centralizado. O presente trabalho propõe um modelo de raciocínio e protocolo de negociação para implementar um mecanismo de negociação descentralizado no projeto MAPS.*

1. Introdução

O conceito *Smart City* surgiu durante a última década com a fusão de várias ideias, a essência do conceito é integrar as tecnologias que até agora têm sido desenvolvidas separadamente, mas que tem ligações claras em seu funcionamento e podem ser desenvolvidas de forma integrada [Batty et al. 2012]. Dentre os desafios a serem enfrentados pelas cidades destacam-se os de mobilidade urbana. Estima-se que em Nova York 40% dos congestionamentos são ocasionados por motoristas buscando vagas de estacionamento [Koster et al. 2014].

Quando se percebe que a demanda de vagas de estacionamentos não está sendo satisfeita a solução adotada normalmente é um aumento quantitativo do número de vagas. Porém, a utilização das mesmas vagas de modo mais inteligente pode amenizar ou até solucionar o problema. *Smart Parkings* são sistemas compostos por dispositivos de *hardware*, capazes de detectar o nível de ocupação do estacionamento e *softwares* integrados, para gerir a atribuição desses espaços de estacionamento [Nocera et al. 2014]. Tais sistemas são concebidos para auxiliar os motoristas na localização de vagas disponíveis,

colaborando com a solução de problemas relacionados à mobilidade urbana. Dentre os vários modelos computacionais que podem ser utilizados para implementar um *Smart Parking* destacam-se os Sistemas Multiagentes (SMAs).

SMAs são sistemas compostos de vários elementos computacionais que realizam interações entre si, de modo a atingirem seus objetivos, sendo tais elementos conhecidos como agentes. Esses agentes possuem duas características importantes: primeiramente são capazes de ações autônomas e em segundo lugar têm a capacidade de interagir uns com os outros pela interação análoga às interações sociais humanas [Wooldridge 2009]. SMAs tem raízes na Inteligência Artificial e possuem características excepcionais para simular e testar cenários para apoiar a tomada de decisão [Mensonides et al. 2008].

Por meio das habilidades sociais agentes devem ser capazes de negociar uns com os outros, afim de solucionarem problemas de forma distribuída, como ocorre em sociedades. Negociação é um processo complexo de tomada de decisão em que cada parte representa de forma autônoma seus pontos de vista e interage com as outras para resolver conflitos e chegar a acordos, maximizando os ganhos de todas as partes [Choi et al. 2001].

Para desenvolver um mecanismo de negociação é possível utilizar três configurações: i) *one-to-one*: onde um agente negocia somente com um agente, neste caso os agentes possuem preferências simétricas; ii) *many-to-one* (centralizado): nesta configuração, um único agente negocia com vários agentes, como acontece em um leilão; e iii) *many-to-many* (descentralizado): neste caso, muitos agentes negociam com muitos agentes simultaneamente, como acontece em um mercado [Wooldridge 2009].

Com o objetivo de aplicar métodos e técnicas de Sistema Multiagente na criação de soluções para alocação de vagas e gerenciamento de um *Smart Parking* foi concebido o projeto MAPS (*MultiAgent Parking System*) [Castro et al. 2017]. O SMA desenvolvido no MAPS possui um mecanismo de negociação centralizado, onde todos os agentes motoristas negociam exclusivamente com o agente administrador. Esta abordagem possui vantagens, como o fato dos agentes motoristas trocarem mensagem apenas com o agente administrador, diminuindo o custo computacional. Mas também possui desvantagens, visto que o agente administrador pode falhar, deste modo comprometendo todo o SMA. Outro ponto negativo é a questão da autonomia dos agentes, visto que os agentes motoristas em um sistema com mecanismo de negociação centralizado ficam sujeitos as imposições do agente administrador.

Para o desenvolvimento de agentes autônomos com capacidades sofisticadas e flexíveis de negociação três itens devem ser definidos: i) protocolo de negociação, o qual define as ações que os agentes podem tomar em uma negociação; ii) o problema que a negociação quer solucionar; e iii) Modelo de raciocínio, o qual define as ofertas iniciais, a gama de ofertas aceitáveis, as contraofertas, quando a negociação deve ser abandonada e quando um acordo deve ser fechado [O'Hare and Jennings 1996]. O objetivo do presente trabalho é estender o projeto MAPS, propondo um Modelo de Raciocínio e o Protocolo de Negociação que servirá como base para implementar um SMA para alocação de vagas de estacionamento com um mecanismo de negociação descentralizado.

2. Modelo de Raciocínio

Modelagem computacional pode facilitar processos de negociação, computando uma ampla gama de alternativas e examinando seus resultados em busca de tendências

[Oliver 1996]. Para modelar computacionalmente problemas utilizando técnicas de negociação em SMAs os agentes devem ser capazes de gerar, por meio de táticas, ofertas e contraofertas. Táticas são o conjunto de funções que determinam como calcular o quanto valioso é um determinado recurso, por meio de características como preço, volume, duração, qualidade, entre outros e considerando um critério, como tempo, recursos, distância, entre outros [Faratin et al. 1998]. Táticas são geradas por combinações lineares de funções simples, sendo que o conjunto de valores utilizados para avaliar o quanto valioso é um recurso é a Imagem da função e o critério utilizado é o Domínio [Faratin et al. 1998]. No decorrer desta seção serão apresentadas algumas funções geradoras de táticas para o contexto de *Smart Parking* com mecanismo de negociação distribuído.

O modelo de raciocínio aqui proposto assume a existência de apenas um tipo de agente no SMA, o agente motorista, o qual pode assumir dois papéis distintos: vendedor, quando o agente está deixando uma vaga e comprador, quando o agente está procurando uma vaga. Neste modelo a negociação é iniciada pelo agente motorista vendedor, quando ele anuncia para todos os agentes próximos que está deixando uma vaga e deseja vendê-la.

Os motoristas que recebem a informação que a vaga está disponível para negociação e estão procurando uma vaga de estacionamento assumem o papel de comprador. Considerando que o agente no papel de comprador receba esta notificação de vaga disponível nas proximidades, então o agente comprador irá fazer uma proposta com base na função 1 (apresentada abaixo e desenvolvida pelos autores deste artigo) e envia para o motorista no papel de vendedor que ofertou a vaga. Os motoristas que recebem a oferta de vaga do vendedor e não estão procurando uma vaga ignoram a mesma.

$$Pc = 10 - \frac{Dist(PD, PV)}{25} \quad (1)$$

Onde: Pc = proposta do agente no papel de comprador, em unidade monetária geral (UMOG); $Dist()$ = distância entre dois pontos, em unidade de medida geral (UMEG); PD = ponto da localização onde o motorista comprador deseja obter a vaga; PV = ponto onde se localiza a vaga que o motorista vendedor está deixando.

A Função 1 gera um valor em UMOGs, valor este que o agente motorista comprador está disposto a pagar por esta vaga ofertada pelo agente vendedor e varia proporcionalmente à proximidade entre a vaga que está sendo ofertada e o local onde o motorista comprador deseja estacionar, em outras palavras, quanto mais próxima a vaga é do local onde ele deseja estacionar maior é o valor que ele estará disposto a pagar. Neste artigo utilizam-se as siglas UMOG e UMEG para representar unidades genéricas.

Destaca-se que neste artigo escolheu-se estabelecer um valor máximo que um motorista possa pagar para obter uma vaga, esse valor é igual a 10 UMOGs, isto acontecerá quando a vaga em questão estiver localizada exatamente no ponto onde o motorista desejava a vaga, isto é, a distância entre ponto desejado e o ponto da vaga é zero. Em relação à distância de vagas igualmente foi determinado um valor máximo para a distância de uma vaga, o qual é igual a 250 UMEGs. Se a vaga estiver numa distância maior que 250 UMEGs então a vaga não será considerada para negociação. A proposta enviada pelo motorista comprador para o motorista vendedor deve ser analisada, isto acontece por meio

da Função 2, que foi desenvolvida com base no trabalho de [Faratin et al. 1998].

$$CPv = \begin{cases} \text{Se } P_c \geq MVV \text{ Então ACEITAR} \\ \text{Se } P_c < MVV \text{ & } PC \geq \frac{MVV}{2} \text{ Então } CP(MVV) \\ \text{Se } P_c < \frac{MVV}{2} \text{ Então REJEITAR} \end{cases} \quad (2)$$

Onde: CPv = contraproposta do vendedor; P_c = proposta do agente comprador; MVV = Média do Valor de Venda; $CP()$ = envia uma contraproposta para o comprador.

A Função 2 será utilizada para analisar a proposta do motorista comprador com base no valor médio das vagas próximas a vaga sendo negociada. O valor MVV é a média do valor de venda das vagas próximas e será obtido por meio de um histórico contendo as últimas negociações realizadas com sucesso que cada vaga do sistema possuirá registrado. A proposta recebida é aceita e o acordo é fechado se a proposta for maior ou igual ao valor médio das vagas próximas e rejeitado se for menor que a metade do valor médio das vagas próximas, em ambos os casos a negociação entre estes dois agentes acaba. Caso contrário uma contraproposta é gerada com base no valor médio das vagas próximas e enviada para o agente comprador. Caso o agente motorista comprador receba esta contraproposta a Função 3 é utilizada para avaliar a mesma.

$$U = \begin{cases} \text{Se } (CPv - P_c) \leq (\delta * P_c) \text{ Então ACEITAR} \\ \text{Se } \text{não REJEITAR} \end{cases} \quad (3)$$

Onde: U = Ultimato; P_c = primeira proposta do comprador; CPv = contraproposta recebida do motorista vendedor; δ = variação da faixa de fechamento de acordo ($0 < \delta \leq 1$).

A Função 3 garante um fim a negociação, visto que ou a proposta é aceita e os agentes entram em um acordo ou a proposta é rejeitada sem que nenhum acordo seja fechado. A proposta é aceita quando a diferença entre a proposta inicial da negociação (P_c) e a contraproposta recebida (CPv) é inferior a $\delta\%$. Este valor de δ pode ser dinamicamente modificado, quanto mais próximo de 1 maior a gama de contraproposta que serão aceitas e mais próximo de 0 menor a gama de contraproposta que serão aceitas.

No início da negociação o agente motorista vendedor ficará por um determinado período aguardando propostas para a vaga ofertada. Depois que este tempo se esgotar o vendedor irá organizar as ofertas recebidas em ordem decrescente de seu valor e começará negociando da proposta mais alta para a mais baixa, podendo repetir este processo de negociação com vários agentes até que um acordo seja fechado. Porém, o agente vendedor poderá negociar com um comprador por vez, podendo apenas enviar uma contraproposta para um novo comprador após ter finalizado a negociação com o agente anterior.

Para que a negociação aconteça os agentes motoristas do SMA precisarão realizar transações de UMOGs, para tanto será necessário utilizar o conceito de carteira. Esta carteira consiste no número de UMOGs que o agente possui para negociar vagas, utilizando o conceito de carteira os motoristas compradores poderão transferir UMOGs para vendedores para pagar por sua vaga, e, estas UMOGs transferidas poderão ser utilizadas posteriormente para a negociação de novas vagas.

3. Protocolo de Negociação

Um protocolo de negociação é um conjunto de regras que especificam a gama de movimentos legais disponíveis para cada agente em qualquer fase de um processo de

negociação [Endriss et al. 2006]. A Figura 1 apresenta o protocolo de negociação que deverá ser utilizado para implementar o SMA para alocação de vagas de estacionamento com mecanismo de negociação descentralizado, utilizando como base o modelo de raciocínio apresentado na seção anterior. Este protocolo define que sempre a negociação

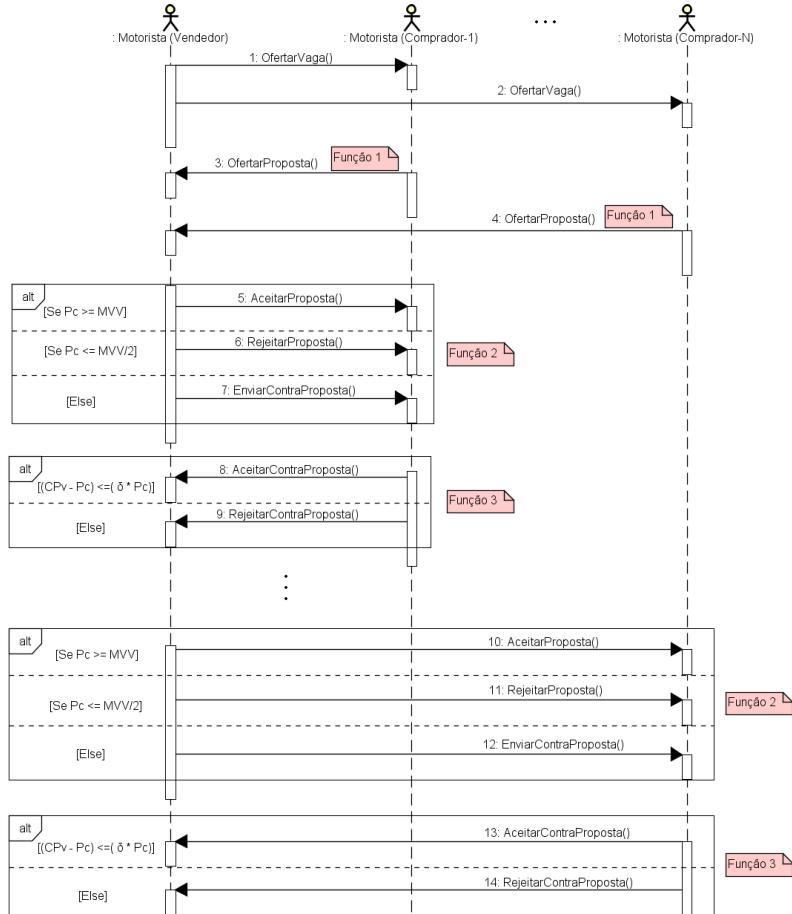


Figura 1. Protocolo de Negociação

deverá ser iniciada pelo agente motorista vendedor, informando os outros motoristas do sistema que possui uma vaga para negociar (1 e 2). Então, os agentes motoristas que estão procurando uma vaga (compradores) enviam uma proposta para este agente vendedor.

Os agentes compradores geram uma proposta, utilizando a Função 1, e a enviam para o motorista vendedor (3 e 4). Então, o motorista vendedor analisa estas propostas em ordem decrescente utilizando a Função 2, podendo tomar as seguintes decisões: (i) aceitar proposta (5), onde a negociação acaba e o acordo é fechado; (ii) rejeitar proposta (6), onde a negociação entre estes dois agentes acaba e o agente vendedor começa uma negociação com outro comprador; e (iii) enviar contraproposta (7), onde o agente vendedor gera uma contraproposta utilizando ainda a Função 2 e envia para o agente comprador.

Quando o agente comprador recebe esta contraproposta ele a analisa utilizando a Função 3 e pode tomar duas ações: (i) aceitar a contraproposta (8), onde um acordo entre os dois motoristas é fechado e a negociação acaba; e (ii) rejeitar contraproposta (9), onde a negociação entre estes dois agentes acaba e o agente vendedor inicia uma negociação com outro motorista comprador, isto se repete até que não haja mais agentes para negociar.

4. Considerações Finais

Estacionar em áreas de grande concentração urbana vem se tornado uma tarefa cada vez mais difícil, uma solução que pode ajudar a solucionar o problema é a implementação de tecnologias que auxiliem os motoristas a encontrarem vagas de estacionamentos livres. Este trabalho propõe um modelo de raciocínio e protocolo de negociação para um *Smart Parking* com mecanismo de negociação descentralizado, onde não existe a figura de um agente centralizador no sistema e todos os agentes independem de um módulo central buscando melhorar questões de mobilidade urbana.

Como sequência imediata deste trabalho, pretende-se desenvolver os seguintes passos: (i) implementar testes com as funções aqui propostas, buscando ajustar e calibrar os parâmetros e valores usados nas funções; (ii) implementar um SMA utilizando o modelo de raciocínio e protocolo de negociação aqui propostos; (iii) desenvolver modelos complementares de raciocínio e protocolo de negociação, para assim elaborar comparação entre os diferentes modelos implantados.

Referências

- Batty, M., Axhausen, K., Fosca, G., Pozdnoukhov, A., Bazzani, A., Wachowicz, M., Ouzounis, G., and Portugali, Y. (2012). Smart cities of the future.
- Castro, L. F. S. D., Alves, G. V., and Borges, A. P. (2017). Using trust degree for agents in order to assign spots in a Smart Parking. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 6(2):45–55.
- Choi, S., Liu, J., and Chan, S.-P. (2001). A genetic agent-based negotiation system. *Comput. Netw.*, 37(2):195–204.
- Endriss, U., Maudet, N., Sadri, F., and Toni, F. (2006). Journal of artificial intelligence research. *Negotiating Socially Optimal Allocations of Resources*, 25:315–348.
- Faratin, P., Sierra, C., and Jennings, N. (1998). Robotics and autonomous systems. *Negotiation decision functions for autonomous agents*, 24:159–182.
- Koster, A., Koch, F., and Bazzan, A. (2014). Incentivising crowdsourced parking solutions.
- Mensonides, M., Huisman, B., and Dignum, V. (2008). Towards agent-based scenario development for strategic decision support. In *Agent-Oriented Information Systems IV*, pages 53–72. Springer Berlin Heidelberg.
- Nocera, D. D., Napoli, C. D., and Rossi, S. (2014). A social-aware smart parking application. ceur workshop proceedings. 1260.
- O’Hare, G. M. P. and Jennings, N. R., editors (1996). *Foundations of Distributed Artificial Intelligence*. John Wiley & Sons, Inc., New York, NY, USA.
- Oliver, J. (1996). A machine-learning approach to automated negotiation and prospects for electronic commerce. *Journal of Management Information Systems*, 13(3):83–112.
- Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. J. Wiley, New York, 2nd edition.

Proposta de modelo de autorrecuperação de sistemas de distribuição de energia elétrica utilizando sistemas multiagente

Italo R. C. Campos, Filipe Saraiva

Faculdade de Computação – 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 describes the self-healing problem of a power distribution system and a proposes to approach it, based on multiagente, systems applied in an adapted model from 33-bus system. The work proposal a four steps algorithm to power restoration in a fail environment, considering some variables and constraints of electric system.

Resumo. Este artigo descreve o problema da autorrecuperação de um sistema de distribuição de energia elétrica e uma proposta para abordá-lo, baseada em sistemas multiagente, aplicada a um modelo adaptado do sistema de 33 barras. O trabalho propõe um algoritmo de quatro passos para a restauração da potência em um ambiente de falha, levando em consideração algumas variáveis e restrições do sistema elétrico.

1. Introdução

Nos anos mais recentes muitos esforços estão sendo direcionados para o desenvolvimento do setor de energia, o que tem trazido enormes avanços nas técnicas para operar, planejar e otimizar a distribuição da energia gerada pelas mais diversas fontes. Para acompanhar esse desenvolvimento o sistema de distribuição também tem ganho novas formas de implementação e, mais do que nunca, técnicas de computação têm se tornado fundamentais para o desenvolvimento desses sistemas. Uma das mais anunciadas tecnologias nesse sentido são os smart grids (ou redes elétricas inteligentes), que, em linhas gerais, são resultado da aplicação de dispositivos com capacidade de comunicação e de softwares inteligentes ao longo da rede de distribuição, com o objetivo de promover automação, conectividade e outras funcionalidades à geração, transmissão, distribuição e consumo da energia elétrica [Souza 2015].

Uma vez que os sistemas inteligentes aplicados à rede elétrica têm capacidade para decidir e configurar o funcionamento dessas redes, são necessários estudos para conceder à rede utilitária dessa tecnologia quesitos relacionados à otimização, segurança, privacidade, velocidade e comunicação. Com relação a isso a abordagem em sistemas multiagente torna-se promissora por garantir, dentro de seus parâmetros intrínsecos, muitas dessas funcionalidades [Saraiva 2015]. Um sistema multiagente – MAS – é um sistema composto por múltiplos agentes, onde estes devem exibir um comportamento autônomo e ao mesmo tempo interagir entre si através de regras de comunicação para atingir um objetivo individual ou compartilhado [Reis 2003]. Nesses termos, o presente trabalho busca apresentar um sistema multiagente distribuído para

lidar com o problema da autorrecuperação de uma rede elétrica de distribuição em um ambiente de falha. O trabalho ainda é resultado de uma pesquisa em andamento, portanto as situações propostas estão em análise e os resultados da aplicação desse modelo serão publicados em trabalhos futuros.

O presente artigo inicia com uma breve revisão, na seção 2, de outros trabalhos que utilizam variadas abordagens ao problema de autorrecuperação de redes elétricas. O trabalho segue a descrição do problema na seção 3, considerando um modelo de rede elétrica específico. Logo após, na seção 4, será discutida a modelagem de um sistema multiagente e os passos que esse sistema deverá seguir em direção à autorrecuperação dado que existe uma falha na rede que deixa nós sem alimentação. Na seção 5 serão levantadas algumas restrições do modelo e os resultados que se esperam para o sistema. A seguir são apresentadas as conclusões do trabalho.

2. Abordagens para o problema da autorrecuperação de smart grids

A autorrecuperação da rede elétrica de distribuição compreende um dos comportamentos mais importantes que uma rede deve exibir para que seja considerada uma rede inteligente [Souza 2015] e, portanto, existem muitos trabalhos da comunidade científica neste viés.

Autorrecuperação da rede elétrica, em linhas gerais, significa a automatização de tarefas que, diante de uma falha no fornecimento de energia elétrica, trabalham para a restauração da potência às cargas afetadas, sem a interação de um ser humano. Lambiase [2012] define para seu trabalho autorrecuperação – ou *self healing* – como “a capacidade do sistema de identificar, isolar e se recompor automaticamente de faltas”.

Nessa linha, Sharma, Srinivasan e Trivedi [2016] propõem um modelo semelhante ao usado neste trabalho. Os autores utilizam dois modelos de redes elétricas para realizar simulações, onde o processo de restauração da potência da rede se dá através do uso de geradores e baterias distribuídos ao longo desta. No caso de falha em algum ponto, os agentes de carga formam ilhas para que se defina uma demanda total e assim solicitem restauração da potência aos agentes que controlam as fontes de energia distribuídas na rede (geradores, baterias e veículos elétricos). Os resultados do trabalho demonstram que, ao serem levados em conta o aspecto incerto dos geradores de energia distribuídos, um maior número de nós pode ser atendido. Outro resultado interessante a se notar é que cada ilha é capaz de recuperar um número de nós proporcional à energia produzida pelos geradores distribuídos ou armazenada nas baterias. Em ambos, o aspecto distribuído dos geradores e de seus agentes gerenciadores garantem à rede melhor capacidade de processamento de informação e tomada de decisão, já que em cada nó da rede há um agente que reage ao ambiente de falha de maneira independente, porém comunicativa.

O trabalho de Souza [2015] simula dois ambientes básicos de falta, sendo um com o uso de geradores distribuídos e outro sem o uso destes. O autor propõe a técnica de corte de cargas por prioridade e ilhamento. Os resultados salientam que o fator determinante, nesse caso, foi o ilhamento, já que permitiu ao sistema restaurar cargas de maior prioridade, a custo da penalização das cargas de menor prioridade.

Ferreira, et. al. [2013] utilizam outra abordagem para o problema de autorrecuperação. Em seu trabalho, os autores se valem de algoritmo genético para chegar à autorrecuperação através de uma reconfiguração de chaveamento do sistema. Os autores propõem um problema de minimização para então encontrar a configuração ótima de chaves. O modelo proposto foi utilizado em três ambientes diferentes, sendo o primeiro com um ponto de falha, o segundo com dois e o terceiro com três pontos de falha. Em todos os casos a abordagem foi capaz de encontrar soluções, diminuindo gradualmente o número de nós restabelecidos conforme o número de falhas aumenta. Visando diminuir o número de resultados redundantes e melhorar a análise das cargas, os autores consideraram a aplicação de um outro modelo em conjunto e a verificação do algoritmo utilizado. Para essa abordagem os autores consideraram um único centro de processamento e controle.

Uma proposta diferente é abordada por Wang et. al. [2015], onde os autores utilizam o conceito de microgrids interconectadas. A principal característica do modelo é a evolução de um antigo modelo centralizado, onde cada rede elétrica passa a ser composta por redes menores e independentes, as microgrids. A ideia é que cada microgrid tem autonomia e maior controle sobre as demandas e restrições de si mesma. A abordagem segue com duas camadas de comunicação, onde a camada mais baixa diz respeito à comunicação interna da microgrid (cargas, geradores e baterias distribuídas) e a mais alta camada gere a comunicação externa entre o sistema elétrico. Os resultados da proposta atribuíram grande confiabilidade ao modelo devido ao caráter autônomo das microgrids. Os autores também concluíram que o modelo distribuído reduz custos em relação ao modelo centralizado.

É interessante notar que os modelos que utilizam a abordagem de distribuição de processamento e tomada de decisão, também utilizam agentes autônomos. Além do mais, o trabalho de Wang et. al. [2015], apesar de não usar diretamente MAS, ressalta o caráter autônomo, comunicativo e descentralizado das microgrids como fatores relevantes nos resultados. Em relação ao modelo centralizado de Ferreira et. al. [2013] a característica mais importante a ser ressaltada é a adaptabilidade do algoritmo genético ao ambiente de falhas. Apesar disso, o número de operações pode aumentar consideravelmente quando comparado aos outros modelos. Com base nesses resultados este trabalho tem por ideal propor um modelo de MAS distribuído e descentralizado, minimizando o número de operações de chaveamento.

3. Problema da autorrecuperação da rede elétrica de distribuição

Nesse ponto do trabalho vale delimitar alguns aspectos da rede que serão levados em consideração para a formulação da proposta de modelo de MAS. O modelo utilizado é baseado no sistema de 33 nós do IEEE, onde algumas adaptações foram realizadas nas linhas de reserva (tie lines). A Figura 1 mostra este modelo.

No modelo em questão, todos os nós estão sendo alimentados através da linha ativa que os ligam. Os nós no modelo são entidades que demandam energia da rede elétrica. As linhas de reserva (ou tie lines) são linhas ligadas estrategicamente de forma a servirem de caminho alternativo para o suprimento em casos de falha, condição primordial para a abordagem aqui proposta. É importante salientar que as mesmas devem permanecer abertas, de modo que não formem ciclos na rede.

A proposta de autorrecuperação considera, durante todo o seu processo, restrições que serão tratadas de maneira implícita nas próximas seções do trabalho e, portanto, serão explicitadas a seguir:

- I. A rede elétrica deverá ser sempre radial;
- II. O fluxo da corrente nas linhas não deve ultrapassar as capacidades dos fios [Sharma, Srinivasan e Trivedi 2016], isto é

$$i_n \leq i_{max} ,$$

onde i_n é a corrente da linha em questão e i_{max} é a capacidade máxima de corrente dessa linha;

- III. A tensão fornecida aos nós deverá sempre estar nos limites de tensão da rede, então

$$T_{min} \leq T_n \leq T_{max} ,$$

onde T_n é a tensão que chega ao nó e T_{min} e T_{max} são as tensões mínimas e máximas da rede;

- IV. A rede elétrica deve possuir linhas de reserva abertas de forma que permitam a restauração do suprimento de energia através de caminhos alternativos.

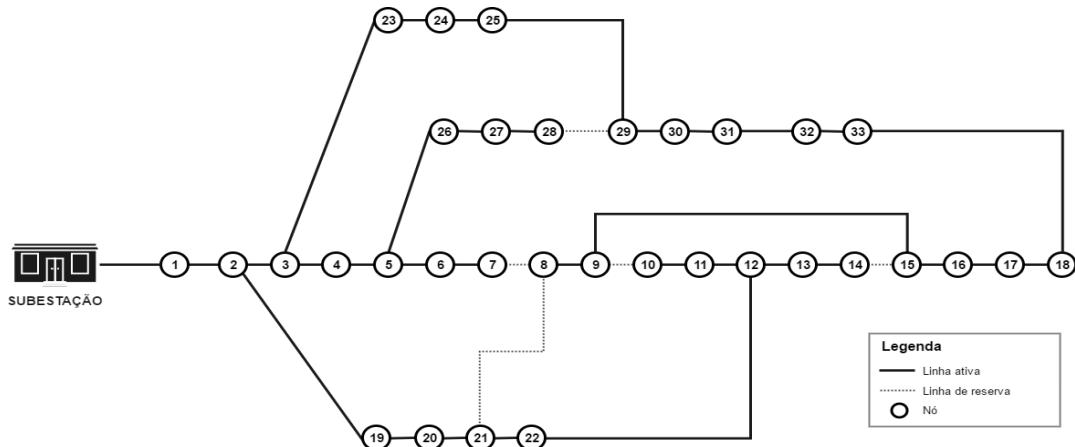


Figura 1. Modelo de 33 nós utilizado na proposta de autorrecuperação da rede elétrica (adaptado de 33-bus by IEEE).

O modelo utilizado considera apenas um alimentador, que é representado pelo primeiro nó do sistema, a saber, a subestação do sistema de distribuição. O problema apresentado aqui consiste em simular uma falha em uma das linhas de distribuição e o objetivo do sistema será a tentativa de autorrecuperação de quantos nós forem possíveis.

4. Modelagem do sistema multiagente proposto

O objetivo da presente abordagem é recuperar o fornecimento de energia nas cargas em ambientes de falta, utilizando o menor número possível de operações de chaveamento e uma solução distribuída. Para a abordagem é utilizado um sistema multiagente que será o responsável por coordenar a tomada de decisão e por gerenciar as operações de chaveamento na rede. O sistema contará com dois tipos de agente: o agente de carga (AC) e o agente de chaveamento (SW). O agente de carga será o responsável por manter unicamente a leitura das informações que chegam ao nó onde ele reside e deve conhecer

o endereço apenas dos agentes de carga adjacentes a ele. O agente de chaveamento é um agente responsável unicamente por manter o controle da chave que liga dois nós. Portanto, para o modelo proposto, serão 34 AC, um para cada um dos 33 nós da topologia e um para representar a subestação, e 38 SW, um para cada linha (ativa e de reserva). Além disso, num primeiro momento, cada AC só terá o conhecimento dos endereços dos AC adjacentes a ele, o que significa que, por exemplo, o AC do nó 8 (simplesmente AC 8) conhece e se comunica apenas com os AC 7, AC 9, e AC 21. É importante observar que para essa estrutura de comunicação as tie lines são levadas em consideração, entretanto, estas permanecem desligadas. Isso nos leva a concluir que a topologia da comunicação entre os agentes é a mesma topologia da rede elétrica.

Para que seja possível o processo de autorrecuperação, será considerada uma falha na rede de forma que alguns nós fiquem sem o fornecimento de energia. Para este exemplo, será definida uma falha na linha que liga os nós 22 e 12, provocando uma falta para um conjunto N de nós formados pelos nós 10, 11, 12, 13 e 14. Assim

$$N = \{10, 11, 12, 13, 14\}.$$

O algoritmo de autorrecuperação compreende quatro passos que serão descritos abaixo e a seguir discutidos.

Passo 1. Isolar a área em falta: uma vez ocorrida a falha, cada AC do conjunto N detecta, isoladamente, a falta de energia. A partir de então, cada AC comunica seus vizinhos sobre a falta e, ao mesmo tempo, recebe os status dos mesmos, permitindo assim que os AC detectem, entre si, quais AC adjacentes possuem fornecimento normal de energia. No exemplo, o AC 10 detectará que ele mesmo não possui abastecimento de energia bem como seu adjacente AC 11, mas que AC 9 está recebendo carga normalmente. A partir daí é possível delimitar (isolar) a área que sofreu falta, no caso, o conjunto N.

Passo 2. Definir o agente ativo: definir um agente ativo significa definir qual AC dos que sofreram a falta (no caso, de N) será o agente que tomará as decisões em nome de toda a área em falta (aqui chamada de ilha¹). Ao se definir um agente ativo (AA), automaticamente todos os demais são agentes do conjunto N serão agentes passivos (AP). A escolha do AA pode se dar de várias formas, mas para este trabalho será utilizado o seguinte critério: o AC mais externo (o que possui pelo menos um adjacente com carga) com maior ID² será o AA da ilha. Se essa escolha falhar o algoritmo é terminado e a potência não será restaurada, pois significa que não há *tie lines* para religar a ilha. Quando eleito o AA da ilha, todos os AP da mesma devem enviar os endereços e os status de seus AC adjacentes e informar ainda os endereços dos SW que residem nas linhas que cada nós conhece, informação importante pois é através dela que o AA realiza as operações de chaveamento. No exemplo proposto, o AA escolhido seria o AC 14.

Passo 3. Religar a ilha: uma vez eleito, o AA deverá exercer o papel de verificar se é possível religar a ilha à rede elétrica por outra linha e, se sim, escolher qual a melhor

¹ A nomenclatura “ilha” não significa que o algoritmo utiliza estratégia de ilhamento. A “ilha” dita aqui é uma consequência da falta na rede.

² O ID de um agente pode variar dependendo da plataforma onde ele é implementado. No presente problema, o ID está atrelado ao número da barra onde o agente reside.

forma de religá-la. Essa etapa do algoritmo pode levar em consideração diferentes variáveis da rede elétrica, como a tensão necessária para religar a ilha à rede, se a rede é capaz de suprir a demanda de tensão da ilha, se os limites de corrente da tie line comportam a demanda da ilha, entre outras. É importante ressaltar que essas informações serão pontuais, já que cada AC só possui informações de seu próprio nó e status dos nós adjacentes. No exemplo, o AC 14 decidiria se ligaria a ilha através da chave de si próprio ou do AC 10.

Passo 4. Realizar operações de chaveamento: se há possibilidade de religar a ilha e já é escolhida a melhor forma de como fazer isso, o AA comunica, primeiramente o SW que mantém controle da chave onde ocorreu a falta (no exemplo, o SW entre o AC 12 e AC 22) e este a abre, evitando assim que, ao se restaurar o abastecimento por aquela linha, haja ciclos na rede. Após isso o AA comunica o SW que mantém o controle da chave da tie line a ser ligada e este, por sua vez, realiza o fechamento da chave, restaurando assim a potência em todos os nós da ilha.

5. Resultados esperados e características

Há muitas características a serem ressaltadas e discutidas a respeito do algoritmo proposto. A mais proeminente é que todo o agente de carga é um potencial agente de carga ativo. Isso é verdade porque não é possível saber onde haverá falha, e daí todo AC no sistema precisa ter a capacidade de se tornar um AA. De modo análogo, todo o agente de carga também é um potencial agente de carga passivo. Logo, fala-se de somente dois tipos de agente no sistema, o agente de carga, que hora é AA, hora é AP e o agente SW.

Uma outra característica importante é que a forma como os agentes enxergam o sistema permite que o MAS proposto seja altamente escalável, já que, ao adicionar ou remover novas cargas no sistema elétrico, somente os agentes adjacentes deverão ser adaptados para recebê-las. Por outro lado, cada agente do sistema exige igual poder computacional, o que pode aumentar o custo para sistemas elétricos com muitos nós.

O número de operações de chaveamento durante a autorrecuperação de uma rede elétrica é um fator muito relevante e o modelo proposto tem a capacidade de operar tal feito com um número constante de dois chaveamentos nos casos de sucesso: um para abertura da chave para a linha defeituosa e outro para fechamento da chave para a tie line que fornecerá energia para a ilha.

Outro ponto para observação é a extrema dependência do sucesso da restauração com o número de tie lines e a forma como elas estão alocadas no sistema. Isso significa que quanto maior o número de tie lines, maior serão as chances de sucesso do sistema, ao passo que a recíproca também é verdadeira. Podemos chamar esse fenômeno de granularidade da rede.

Para trabalhos futuros, é interessante pensar na integração do modelo às fontes de energia renováveis e baterias distribuídas ao longo da rede. Esse fato pode aumentar as chances de sucesso da autorrecuperação, além de diminuir a dependência do modelo com o número de tie lines do sistema elétrico. Nesse sentido, o trabalho de Sharma, Srinivasan e Trivedi [2016] evidencia que a taxa de sucesso na recuperação de nós em

falta é aumentada se levadas em consideração fontes de energia distribuídas pela rede. Portanto pode ser possível aliar trabalhos desse tipo ao modelo aqui proposto.

6. Conclusão

O modelo proposto é pertinente à realidade dos smart grids e pode oferecer uma nova maneira de implementar a autorrecuperação nessas redes. O modelo aqui apresentado tem foco na natureza distribuída dos sistemas multiagente e na sua alta escalabilidade, por isso pode se tornar vantajoso no sentido de tornar o processo de autorrecuperação mais simples e rápido. O algoritmo de autorrecuperação possui quatro passos que envolve apenas a parte da rede afetada pela falta e é realizado, nos casos de sucesso, com um número constante de duas operações de chaveamento.

Para implementar o sistema aqui proposto é usada a plataforma JADE, pois esta atende à característica descentralizada dos sistemas multiagente de maneira simples e oferece uma API amigável e de largo suporte por sua comunidade [Bellifemine 2007]. Para trabalhos futuros pretende-se apresentar os resultados da simulação desse sistema proposto além de testá-lo utilizando modelos de redes elétricas com maior número de nós e de alimentadores, verificando assim sua eficácia, escalabilidade, desempenho e outros parâmetros do sistema que permitam averiguar sua viabilidade.

Referências

- Bellifemine, F., Caire, G. and Greenwood, D. (2007) “Developing Multi-Agent Systems with JADE”, Jhon Willey & Sons, Reino Unido.
- Ferreira, L. R., et. al. (2013) “Solução do Problema de Self-Healing para Redes de Distribuição Radiais Através de Otimização via Algoritmo Genético”, Universidade de Curitiba.
- Lambiase, C. B. (2012) “Aplicação de Self Healing em Sistemas Elétricos”, Departamento de Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Brasil.
- Reis, L. P. (2003) “Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico”, Tese de Doutorado, Faculdade de Engenharia da Universidade do Porto, UP, Porto, Portugal.
- Saraiva, F. O. (2015) “Aplicações Híbridas entre Sistemas Multiagentes e Técnicas de Inteligência Artificial para Redes Inteligentes de Distribuição de Energia Elétrica”, Tese de Doutorado, Departamento de Engenharia Elétrica e de Computação, USP, Brasil.
- Sharma, A., Srinivasan D. and Trivedi A. (2016) “A Descentralized Multi-Agent Approach for Service Restoration in Uncertain Environment”, IEEE publishing.
- 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 de Mestrado do Programa de Pós-Graduação em Engenharia Elétrica, UFPR, Brasil.
- Wang, Z., et. al. (2015) “Networked Microgrids for Self-Healing Power Systems”, IEEE publishing.

Modelagem do Ritmo Circadiano utilizando Sistemas Multiagente: um estudo de caso da influência da dor

Angélica T. dos Santos¹, Catia M. S. Machado¹, Diana F. Adamatti¹

¹Programa de Pós Graduação em Modelagem Computacional (PPGMC)

Universidade Federal do Rio Grande (FURG)

Caixa Postal 474 – 96.203.900 – Rio Grande – RS – Brasil

theisangelica@gmail.com, {catiamachado, dianaadamatti}@furg.br

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. Nesse sentido, o trabalho proposto tem como meta o estudo de um modelo matemático e computacional baseado em simulação de multiagente, da sincronização e dessincronização do ritmo circadiano em relação a variável dor. Resultados da simulação multiagente mostram o quanto a periodicidade do ritmo circadiano é alterada pela variável dor.

1. Introdução

Os ritmos circadianos são oscilações em processos fisiológicos e comportamentais que ocorrem num período de 24 horas. Essas oscilações estão presentes em todas as atividades dos seres vivos.

Os ritmos biológicos do ritmo circadiano são mudanças cíclicas que se repetem ao longo de um determinado período e estão relacionados com às alterações dos processos fisiológicos do corpo, sendo que a atividade de dormir é um mecanismo de reparo das células que regulam os processos físicos, intelectuais e psíquicos.

A simulação é uma técnica que envolve a construção de um modelo de uma situação real para posterior experimentação. Assim, em pesquisas desenvolvidas por [Borbely et al. 1984], foi desenvolvido um modelo matemático que descreve o comportamento do ritmo circadiano, sendo que [Borbely and Achermann 1999] aprimoraram este modelo, a qual [Skeldon 2014], implementou o modelo aprimorado utilizando sistema multiagente.

Mediante isso, a contribuição deste trabalho é implementar a variável dor no modelo desenvolvido por [Skeldon 2014], seguindo o mesmo modelo matemático de [Borbely and Achermann 1999]. Desta forma, o objetivo geral é estudar o comportamento do ritmo circadiano influenciado pela variável dor, utilizando sistemas multiagente.

2. Referencial Teórico

O ciclo vigília-sono tem ritmicidade de 24 horas, em que o período de vigília ocorre durante o dia e o sono durante a noite, dando ênfase que o sono realizado durante o dia não tem a mesma qualidade do sono noturno, bem como a vigília que ocorre a noite não é igual à do dia.

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, que pode ser expresso por uma curva sinusoidal, com valores alcançando seu nível máximo de propagação do sono no início da manhã e o seu mínimo no início da noite.

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 [Borbely and Achermann 1999].

Segundo a *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].

O relógio biológico de cada pessoa é sincronizado conforme suas atividades decorrentes do dia. Assim, a marcação horária interna eventualmente é precisa. Na regulação interna, é necessário ter os mecanismos de ajustes que permitem a sincronização. Essa sincronização é realizada pelo fenômeno de ajuste, chamado de "arrastamento". Esse fator externo que comanda o ajuste, denomina-se "zeitgeber". Os "zeitgeber" são os sincronizadores do relógio biológico. Assim, o ritmo circadiano e homeostático são sincronizados pelo "zeitgebers", de maneira que os mesmos estejam sempre interligados, para que possam estar interligados é necessário um "pacemaker" (marcapasso). Para ocorrer a sincronização e dessincronização entre os ritmos, sempre tem um modelo matemático que comanda os ritmos [Borbely et al. 1984].

A simulação computacional é uma ferramenta que vem ganhando espaço nas mídias digitais, pois a mesma tem várias funcionalidades como projetar, planejar, controlar e avaliar alternativas do mundo real. Para essa prática é necessário softwares que representam o mundo real em simulação [Rebonatto 2000].

3. Modelo proposto - Extensão do Modelo de Skeldon

Partindo do Modelo proposto por [Skeldon 2014], que analisa o Processo \tilde{S} , união do ritmo circadiano e homeostático. Neste momento será inserida a dor, a contribuição ao trabalho. A dor é uma variável que afeta a qualidade do sono, quando a pessoa tem dor a mesma pode não dormir, ou não exercer o sono ideal para ter um descanso merecido.

Mediante vários estudos realizados, procurou-se uma outra maneira de analisar a influência da dor no ritmo circadiano. Assim, considerando todo o modelo matemático, a dor é calculada inversamente proporcional, pela equação (1), que foi obtida por meio de testes com dados empíricos.

Todo o processo da inserção da variável dor está em inglês, visto que toda a interface e nome de variáveis do modelo base estão nesta língua. Também espera-se, desta forma, que sua utilização possa ser mais abrangente.¹

$$(1 - \left(\frac{\text{pain}}{10} \right) * 0.2955) \quad (1)$$

¹O valor 0.2955 foi adequado através de testes empíricos.

Somente com a equação (1) não é possível realizar o cálculo da dor. Para ser realizado o cálculo, a mesma necessita ser inserida no código em NetLogo, sempre levando em consideração que dor foi definida considerando os parâmetros da Escava Visual Analógica [Miguel 2003].

A variável da dor é inserida na “turtle 1 is the circadian” - ritmo circadiano, tartaruga 1, mostrado na figura 1. No código, a turtle 1 realiza a análise do horário de acordar, intensidade da dor, horário de dormir e intensidade da dor. O sistema multiagente, capta cada variável, inspeciona e realiza o cálculo da dor. Pois, desta maneira o agente, a “turtle”, percorre todo o dia, analisando a dor em relação ao período diurno e noturno.

```

1 ; turtle 1 is the circadian
2     ask turtle 1 [
3         pd
4         set T (T_start * tscale + ticks) / tscale
5         if Listening-to-your-body-clock = "No" [
6             if (Sat_sleep > 24) [
7                 if (T = Sat_sleep mod 24) [ set wake 0 ]
8             ]
9
10            if (T = Sun_wake) [ set wake 1
11                set pain Sun_pain]
12            if (T = Sun_sleep) [ set wake 0
13                set pain Sun_pain]

```

Figura 1. Influência da dor nos dias da semana.

4. Análise dos Resultados

Os resultados foram obtidos por meio de testes, fazendo assim, uma comparação em relação à influência da intensidade do nível da dor. A escolha do indivíduo foi de forma aleatória, sendo com idade 45 anos, acorda às 7 horas, dorme às 22 horas, wake e sleep timeconstant 45. Assim, essa pessoa tem um sono controlado e quase sempre dorme as oito horas recomendadas.

Na maioria das vezes, o indivíduo não possui sempre a mesma intensidade do nível de dor. Conforme a figura 2, o indivíduo teve níveis de dor diferente: Domingo - 0; Segunda-feira - 6; Terça-feira - 10; Quarta-feira - 8; Quinta-feira - 6; Sexta-feira - 4 e Sábado - 2.

Com essa variação da influência dos níveis de dor, é perceptível, que como na terça-feira ele teve dor nível dez, no dia seguinte quarta-feira a dor diminuiu de nível, mais ainda continuou com um pico elevado de dor. Consequentemente, na dor variável, é intuitivo que o mesmo tenha toda a sua semana de observação alterada, sendo que após a dor dez, todos os dias ficaram comprometidos, pois não tem como ter dor dez em um dia e no seguinte não ter nada de dor. As curvas em cinza são mais acentuadas, sendo que após o encontro da curva cinza com a preta, a mesma passa a ter uma leve inclinação na horizontal, tanto no inicio da noite, quanto no inicio do dia. Quanto mais forte for o nível de dor, maior serão os picos em vermelho que aparecem na curva que mostra o período diurno. A dor influência diretamente na qualidade do sono, sendo que quando o indivíduo tem dor, ele não tem o descanso merecido durante todos os dias.

Outro ponto que merece um destaque é que a má qualidade do sono, influenciado pelo nível de dor, compromete a qualidade de vida, tanto do indivíduo, quanto da

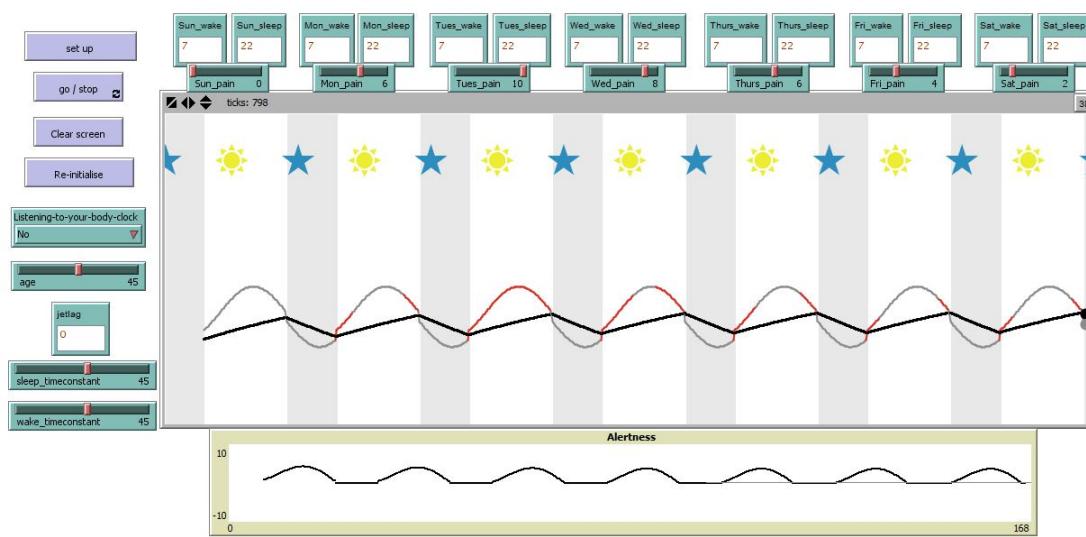


Figura 2. Cenário 1 - Influência do nível de dor variável.

população.

5. Considerações Finais

Os resultados indicam que os fatores que interferem a qualidade do sono são capazes de interferir na participação ativa do indivíduo em atividades físicas, trabalhos, assim como, no decorrer das atividades do cotidiano.

Este estudo teve por objetivo avaliar o comportamento do ritmo circadiano influenciado pela variável dor, utilizando sistemas multiagente. Através das variáveis da simulação mostrou-se uma aplicação do dia a dia, onde que o indivíduo sofreu alterações no nível de dor, durante o período de sete dias.

Referências

- Borbely et al. (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.
- Borbely, 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.
- Miguel, J. P. (2003). A dor como 5º sinal vital. registo sistemático da intensidade da dor, 09/dgcg (2003). <http://www.myos.com.pt/files/circular5sinalvital.pdf>. [Online; accessed 01-October-2017].
- Rebonatto, M. T. (2000). Simulação paralela de eventos discros com uso de memória compartilhada distribuída. Master's thesis, Programa de Pós-Graduação em Computação, UFRGS, Porto Alegre.
- Skeldon, A. (2014). Are you listening to your body clock? <http://personal.maths.surrey.ac.uk/st/A.Skeldon/sleep.html>. [Online; accessed 15-January -2017].

Sistemas Multiagente como Ferramenta para Desenvolvimento do Pensamento Computacional

Vinícius B. Martins¹, Diana F. Adamatti¹

¹Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Caixa Postal 474 – 96.203.900 – Rio Grande – RS – Brasil

Resumo. *Trazendo à tona os problemas com o aprendizado que os calouros dos cursos na área de computação enfrentam, este artigo descreve o estágio inicial de uma pesquisa que busca comprovar a melhora no aprendizado desses alunos com o uso de Sistemas Multiagente relativo ao pensamento computacional, utilizando a ferramenta NetLogo.*

1. Introdução

Devido à dificuldade de algumas disciplinas do ensino superior e ao aumento tecnológico, torna-se essencial a criação de ferramentas para o auxílio no aprendizado para alunos de diversas áreas de atuação. A interdisciplinaridade que uma ferramenta de Sistema Multiagente (SMA) atinge as tornam fortes candidatas para o uso no contexto citado [Adamatti 2011]. Para tanto, os alunos devem desenvolver o Pensamento Computacional [Wing 2006], de forma a utilizá-lo utilizá-lo em qualquer meio ou área, desde pesquisa para obter uma lista de páginas *web* até para obter uma estratégia de jogo. Seguindo essa lógica de pensamento, [Wing 2006] fala que o Pensamento Computacional não é o modo como um computador pensa, mas a forma que um ser humano deve pensar para resolver seus problemas de forma mais organizada e otimizada e que deveria ser ensinado para todos, não somente para alunos da área da computação.

Os SMAs são usados em grandes quantidades nas salas de aula, porém não foi feito estudos que comprovem que o uso de desses sistemas ajude na aprendizagem dos alunos, como [Antunes et al. 2005] e [Adamatti 2011]. Talvez o trabalho mais próximo do objetivo do estudo aqui proposto seja o trabalho de [Recchi and Martins 2013], o qual descreve o uso da ferramenta NetLogo em turmas dos cursos de química e ciências na Universidade Federal da Fronteira Sul (UFFS). Segundo os autores, o uso da ferramenta auxiliou os alunos dos cursos a aprender os assuntos em razão da implementação no software (regras, agentes, etc), tendo, antes da implementação, que estudar o funcionamento das pessoas e objetos envolvidos no assunto para poder programar isso no *software*.

Apesar da quantidade de jovens ingressantes no ensino superior em Computação, as taxas de abandono podem chegar a 50% [Palmeira and Santos 2015]. A causa dessas evasões são variadas, estando entre elas: a relação curso-aluno (não ser o que quer), relação aluno-professor (competência destes em aula), questões financeiras, não conformidade da área de trabalho com a universidade [Gessinger et al. 2013, Casartelli et al. 2013].

Dentro de tal cenário e com a alta taxa de reprovação de alunos na disciplina de Algoritmos e Estruturas de Dados I (AEDI), 71,47% dos alunos desde 2013 ¹, busca-se trabalhar com os alunos, aulas relacionadas aos aprendizados na disciplina por outra

¹Dados retirados de tabelas feitas pela coordenação do curso de Sistemas de Informação da FURG

visão, utilizando Sistemas Multiagente, além de analisar se há melhoria no aprendizado utilizando sistemas multiagente.

Segundo [Jonassen 2004], o aprendizado só ocorre quando o aluno precisa resolver um problema e, resolvendo-o, ele aprende e comprehende o que deve ser feito para resolvê-lo. Partindo desse pressuposto, após a apresentação da sintaxe do NetLogo, serão aplicados diversos exercícios para ser treinado o pensamento computacional dos alunos. Este, é definido como sendo o processo de entendimento de um problema e a execução do mesmo de forma que um computador entenda. [Wing 2006, Resnick 2012].

Espera-se verificar se um sistema multiagente ajuda no aumento do pensamento computacional e, portanto, na melhora do aprendizado dos alunos numa disciplina de programação básica num curso de Computação. Pois, segundo o estudo de [Camera and Chicon 2017], o uso de uma ferramenta gráfica para o auxílio da apresentação do conteúdo mostra melhora no aprendizado final dos alunos.

Este artigo descreve o início de um estudo que busca verificar se ferramentas Multiagente podem dar suporte ao estudo do pensamento computacional como forma a aumentar a curva de aprendizado do aluno em disciplinas introdutórias do curso de Sistemas de Informação (SI) da Universidade Federal do Rio Grande (FURG).

2. Metodologia

Tratando-se de alunos do primeiro ano no curso de SI da FURG, onde as aulas ocorrem durante o turno matutino, as aulas sobre NetLogo acontecerão no turno contrário, vespertino. A ação contará com um professor voluntário, também aluno do curso, porém mais avançado na graduação. Os alunos deverão, necessariamente, não terem tido contato com programação antes de entrarem na universidade.

O estudo terá duração média de dois meses, onde acontecerão encontros semanais para a apresentação do conteúdo e para exercícios. Nas aulas, será ensinado o básico de programação orientada a agentes no NetLogo, levando em consideração os tópicos que os alunos já aprenderam na disciplina de Algoritmos e Estruturas de Dados I (AEDI) até o momento do minicurso (variáveis, condicionais, laços).

Os alunos que decidirem participar dos testes assinarão um termo de consentimento para declarar ter concordado em serem voluntários da pesquisa.

Na primeira e na última aula do minicurso, pedir-se-á aos alunos preencherem dois questionários para testar o raciocínio lógico antes e depois, possibilitando a verificação de melhora no pensamento lógico. A outra forma de visualização de melhora no desempenho será a comparação de notas entre os alunos que fizeram parte do teste e a média dos alunos que não fizeram.

2.1. NetLogo

Quando se fala em Sistemas Multiagente, uma das ferramentas mais utilizadas de modelagem e simulação de agentes é o NetLogo [Leon et al. 2015].

NetLogo é uma ferramenta programável de modelagem multiagente [Wilensky 1999] e, como tal, possui uma interface gráfica que auxilia a utilização e visualização da execução do simulação, como mostrado na área em preto da Figura

1. Possui duas versões: a 2D e a 3D. Outra vantagem da ferramenta é, utilizando a parte gráfica disponível, a definição do valor de variáveis que pode ser selecionado rapidamente com o uso de botões ou escala de valores.

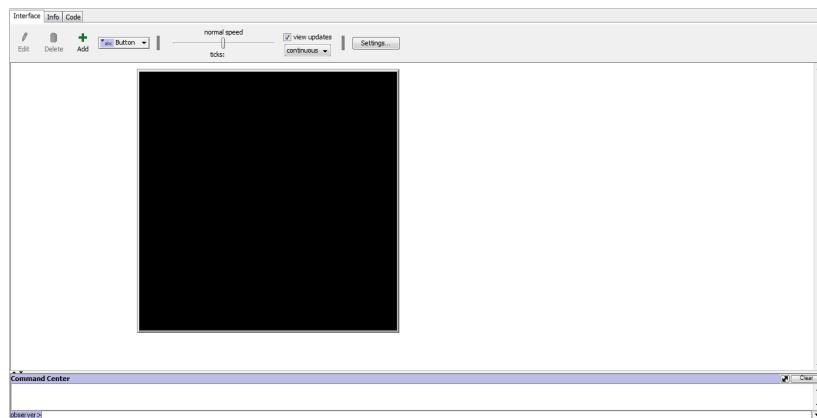


Figura 1. Área de trabalho da versão 2D da ferramenta NetLogo.

O NetLogo trabalha com um sistema de tempo interno chamado *tick*. Durante a simulação, pode-se fazer com que os *ticks* passem mais devagar ou mais rápido, dependendo da intenção do usuário. Os agentes da simulação são chamados de *turtles*. A linguagem programável do NetLogo é chamada de *Logo*. Ela foi criada por Seymour Papert e Wally Feurzeig, em 1968, e foi baseada na linguagem *Lisp*, mas com a intenção de torná-la mais fácil e programável para adultos ou crianças.

2.2. Questionários

Os questionários tem a intenção de testar as habilidades em raciocínio lógico dos alunos voluntários. As questões utilizadas serão das provas da Olimpíada Brasileira de Matemática (OBMEP) de 2015, 2016 e 2017². Os questionários contarão com 10 questões cada, retiradas do nível 1 das provas da OBMEP. Um exemplo de questão é dado a seguir:

”No final de um dia de atividades, um parque de diversões arrecadou 100 reais com os ingressos de 100 pessoas. Sabemos que cada adulto precisava pagar 3 reais para entrar, jovem 2 reais e cada criança 30 centavos. Qual é o menor número de adultos que entraram nesse dia no parque?”[Barbosa and Feitosa 2017, Questão 11]

3. Conclusões e Próximas Atividades

A ideia do estudo surgiu com o objetivo de mostrar como uma ferramenta de apoio ao estudo é importante ao desenvolvimento de um aluno, assim como a falta de assuntos relacionados na área.

O estudo ainda está em seu estágio inicial, ainda precisando ser executado e extraído os dados relevantes à pesquisa. Como resultados, espera-se que se consiga comprovar o objetivo da pesquisa, ou seja, mostrar que o uso de sistemas multiagente como ferramenta auxiliar no ensino de programação ajuda no melhor entendimento do assunto por parte do aluno e, dessa forma, reduzir a taxa de reprovação numa das mais importantes (e básicas) disciplinas de um curso da área da computação.

²Bancos de questões disponíveis no site <http://www.obmep.org.br/banco.htm>

Como próximas etapas do projeto, vislumbra-se a aplicação das aulas aos calouros de Sistemas de Informação, a extração de dados e criação de estatísticas que comprovem ou não a melhora no ensino e a publicação dos resultados.

Referências

- Adamatti, D. F. (2011). Simulação baseada em multiagentes como ferramenta em estudos interdisciplinares. *Revista Novas Tecnologias na Educação*, 9(1).
- Antunes, M. N. R., Tavares, O. L., and Azevedo Filho, J. T. d. S. (2005). Um ambiente virtual para apoiar a aprendizagem na área de direito: uma arquitetura cscw com recursos multiagente para simulação de júri. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 1, pages 158–168.
- Barbosa, R. and Feitosa, S. (2017). Obmep - banco de questões 2017. Rio de Janeiro, IMPA. Disponível em: <http://www.obmep.org.br/bq/bq2017.pdf>. Acessado em: 24 de Janeiro de 2018.
- Camera, P. E. and Chicon, P. M. M. (2017). Relato dos resultados sobre o curso de extensão introdução à programação. *I Seminário de Pesquisa Científica e Tecnológica*, 1(1).
- Casartelli, A. d. O., Benso, A. C. d. S., Morosini, M. C., and Gessinger, R. M. (2013). Um estudo sobre os motivos e fatores relacionados com o abandono estudantil na pucrs. *Libro de Actas de III CLABES, 2012, México*.
- Gessinger, R. M., do Rosário Lima, V. M., Leite, L. L., and Moraes, M. C. (2013). O uso pedagógicos de recursos tecnológicos como estratégia para qualificar o ensino e contribuir para a redução da evasão na educação superior. *Libro de Actas de III CLABES, 2013, México*.
- Jonassen, D. H. (2004). *Learning to solve problems: An instructional design guide*, volume 6. John Wiley & Sons.
- Leon, F., Paprzycki, M., and Ganzha, M. (2015). A review of agent platforms. *Technical Report, ICT COST Action IC1404, Multi-Paradigm Modelling for Cyber-Physical Systems (MPM4CPS)*.
- Palmeira, L. B. and Santos, M. P. (2015). Evasão no bacharelado em ciência da computação da universidade de brasília: análise e mineração de dados. *Dissertações Universidade de Brasília*.
- Recchi, A. M. S. and Martins, M. M. (2013). Netlogo: Linguagem de programação educacional para o ensino de química e ciências. *Encontro de Debates sobre o Ensino de Química*, 1(01).
- Resnick, M. (2012). Point of view: Reviving papert's dream. *Educational Technology*, 52(4):42.
- Wilensky, U. (1999). Netlogo (and netlogo user manual). Disponível em: <http://ccl.northwestern.edu/netlogo/>. Acessado em: 11 de Janeiro de 2018.
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35.

Interpretação de Imagens como Tarefa Cognitiva: uma Abordagem Inicial com Redes Neurais de Aprendizagem Profunda

Cleverson Carneiro¹, Nicolas Mansur Beleski¹, Gustavo Giménez Lugo¹

¹Universidade Tecnológica Federal do Paraná

Resumo. O presente trabalho aborda a funcionalidade de interpretar imagens como uma tarefa cognitiva em agentes. Para verificar alguns dos limites das abordagens consideradas mais efetivas foram efetuados experimentos com redes convolucionais. A principal contribuição, a partir dos resultados, é a constatação e explicitação de possíveis caminhos para elaborar uma arquitetura cognitiva que, embora utilize aprendizagem profunda para aquisição de entradas, o faça sob um arcabouço cognitivo.

1. Introdução

Processamento digital e reconhecimento de objetos em imagens é uma das áreas que mais beneficiou-se com os recentes avanços em inteligência artificial (IA). Este progresso deve-se principalmente aos desenvolvimentos em aprendizagem profunda e redes neurais convolucionais (RNC) [Simonyan and Zisserman, 2014, He et al., 2016].

Apesar destes avanços, ainda existem questões a serem respondidas quanto a capacidade das RNC de generalizarem os conhecimentos obtidos quando comparados, por exemplo, as capacidades humanas de aprendizagem.

Neste trabalho aborda-se a área de pesquisa de ciências cognitiva, que atua buscando compreender como acontece o aprendizado conceitual humano e oferece uma abordagem quanto a construção de modelos que permitem o desempenho humano em tarefas de reconhecimento.

1.1. Limites Teóricos

Apesar de redes neurais atingirem resultados similares a pessoas em alguns desafios de classificação de imagens, a forma como o aprendizado acontece e o conhecimento é processado ainda é bem diferente. A fim de obter nível de resultados de humanos, redes neurais dependem de um treinamento utilizando de dezenas à milhares de imagens, estas que requerem ser manualmente rotuladas para cada classe individual de objeto. Apesar da utilidade destas técnicas em problemas específicos, o extenso treinamento não parece um caminho factível para resolver o problema de classificação em imagens de forma generalizada. Por exemplo, é estimado que pessoas conseguem distinguir aproximadamente 30000 objetos diferentes [Biederman, 1987] e ainda mais categorias que podem ser criadas dinamicamente [Lampert et al., 2014]. Estas diferenças no aprendizado levantam questões quanto possíveis abordagens alternativas para analisar problemas de classificação e reconhecimento em imagens.



Figura 1. Duas implementações utilizadas nos testes.

1.2. Experimento

Para experimentar quanto a eficiência das redes neurais convolucionais no que tange a interpretação de imagem, implementa-se um sistema que busca identificar uma pessoa em uma imagem como passo inicial para uma eventual tarefa do tipo *follow-me*.

O experimento foi realizado usando-se duas redes neurais convolucionais em paralelo (Figura 1 (a) e (b)), combinando-se os resultados individuais utilizando princípios de lógica *naive* baseada em objetivos e utilizando linguagem natural. Ambas redes neurais possuem limitações quanto a detecção de pessoas na forma proposta, entretanto quando aplicadas em conjunto a probabilidade de detecção aumenta, como na Figura 2. A partir destes resultados iniciais, conjectura-se que uma arquitetura que assimile o resultado de múltiplas redes neurais e raciocine sobre estes resultados pode obter soluções mais robustas.

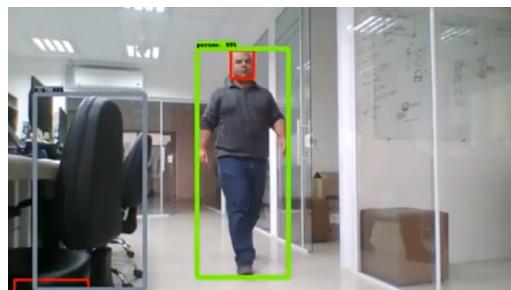


Figura 2. Combinando redes neurais: https://youtu.be/_feL5vE24TM

2. Arquitetura Cognitiva

Uma arquitetura cognitiva é composta por sub-sistemas autônomos distribuídos [Fodor, 1983], que percebem seu ambiente, aprendem com a experiência, antecipam os resultados dos eventos, agem para perseguir metas e se adaptam às mudanças das circunstâncias [Ikeuchi, 2014]. Uma arquitetura cognitiva destinada a interpretação de uma imagem pode ser composta pode ser descrita pelas camadas de percepção estruturada, modelo causal e raciocínio.

2.1. Percepção Estrutural

No âmbito do sistema cognitivo visual, [Marr and Vision, 1982] desenvolveu uma teoria computacional no estudo da visão, baseando-se em matemática, psicologia cognitiva, neurociência e estudos clínicos construindo uma hierarquia de diferentes níveis na

estruturação do conhecimento visual. Em [Prinz, 2012] é descrita uma hierarquia organizacional do sistema visual, incluindo as partes do cérebro responsável por tipos diferentes de processamento das informações visuais conforme descrita na Figura 2.1.

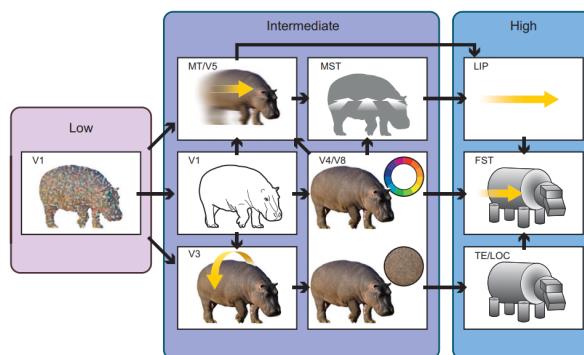


Figura 3. Diagrama do sistema visual [Prinz, 2012]

Se considerarmos as observações feitas por [Marr and Vision, 1982] que separou as regras computacionais de sua implementação e observarmos o trabalho de [Prinz, 2012] que segmentou o processo cognitivo da visão em regiões específicas do cérebro é acreditado que pode-se reproduzir o processo cognitivo da visão através de uma combinação de Redes Neurais Artificiais Especializadas.

3. Representação do Conhecimento

A interpretação de uma imagem é totalmente influenciada pelo propósito atual do agente. Por exemplo, uma cena composta por uma sequência de imagens faz com que haja uma overdose de percepções, que se o agente não tiver algumas restrições prévias definidas em sua base de conhecimento, não trabalhará de forma eficiente.

Para continuar a explorar e testar a arquitetura cognitiva como solução computacional para interpretação de imagens, dá-se ao agente a missão de seguir uma pessoa, desta forma precisar-se representar o conhecimento de "seguir uma pessoa". Segundo [Guarino, 1995] uma maneira de representar o conhecimento é através do uso da ontologia.

3.1. Ontologia

Uma ontologia é uma especificação explícita e formal de uma conceituação [Studer et al., 1998]. Na prática a ontologia é a descrição de classes de um domínio, as propriedades de cada classe descreve seus atributos e regras de restrições entre as classes e seus atributos. A ontologia, junto com suas instâncias define uma base de conhecimento. Desenvolver uma ontologia inclui [Noy et al., 2001]:

- Definir as classes da ontologia.
- Definir seus atributos e seus valores permitidos.
- Organizar as classes em uma hierarquia taxonômica (sub-classe e super-classe).
- Preencher os atributos das classes para as instâncias.

4. Conclusão

O trabalho realizado constatou que a utilização de redes neurais convolucionais de aprendizagem profunda mostraram-se eficientes em classificar imagens dentro de um cenário restrito e controlado, porém ao aplicá-las em imagens parciais, dinâmicas e com baixa nitidez as RNC apresentaram limitações e baixa confiabilidade, principalmente dado a forma de seu treinamento.

A aplicação de múltiplas redes controladas por um componente cognitivo externo, por outro lado, apresentou uma solução mais eficiente. A composição de resultados de diferentes áreas do objeto a ser identificado é inspirada na forma da cognição humana. A aplicação de uma arquitetura cognitiva mostrou-se promissora para aperfeiçoar a performance do agente inteligente.

Inicialmente para a construção da arquitetura cognitiva proposta, verificou-se a importância de formalizar o problema e sua respectiva ontologia. Baseado nos resultados iniciais obtidos, acredita-se que existe espaço e benefícios na abordagem problemas de reconhecimento em imagens utilizando-se de uma perspectiva cognitiva.

Referências

- [Biederman, 1987] Biederman, I. (1987). Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115.
- [Fodor, 1983] Fodor, J. A. (1983). *The modularity of mind: An essay on faculty psychology*. MIT press.
- [Guarino, 1995] Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *International journal of human-computer studies*, 43(5-6):625–640.
- [He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- [Ikeuchi, 2014] Ikeuchi, K. (2014). *Computer vision: A reference guide*. Springer Publishing Company, Incorporated.
- [Lampert et al., 2014] Lampert, C. H., Nickisch, H., and Harmeling, S. (2014). Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465.
- [Marr and Vision, 1982] Marr, D. and Vision, A. (1982). A computational investigation into the human representation and processing of visual information. *WH San Francisco: Freeman and Company*, 1(2).
- [Noy et al., 2001] Noy, N. F., McGuinness, D. L., et al. (2001). Ontology development 101: A guide to creating your first ontology.
- [Prinz, 2012] Prinz, J. (2012). *The conscious brain*. Oxford University Press.
- [Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [Studer et al., 1998] Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: principles and methods. *Data & knowledge engineering*, 25(1-2):161–197.