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

— WESAAC 2016 —

Organizado por

Baldoíno Fonseca dos Santos Neto

Diana Francisca Adamatti

Maceió, 23 a 25 de Maio de 2016.

Anais do X Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCações — X WESAAC / dos Santos Neto, B. F.; Adamatti, D. F. (Org) — Maceió, 2016.

192p. :il.
ISSN 2177-2096

1. Agentes. 2. Sistemas de Agentes. 3. Ambientes para Agentes. 4. Aplicações de Agentes. I. dos Santos Neto, B. F. II. Adamatti, D. F.

CDD

PREFÁCIO

Este documento contém os trabalhos apresentados na Décima Edição do WESAAC (Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações). O WESAAC 2016 foi realizado na cidade de Maceió/AL, junto a Universidade Federal de Alagoas (UFAL), entre os dias 23 e 25 de Maio de 2016.

O principal objetivo desta décima edição do WESAAC é integrar pesquisadores e estudantes de todos os níveis na área de Agentes e Sistemas de Agentes, divulgar as atividades dos diversos grupos de pesquisa do Brasil, possibilitando o intercâmbio de conhecimentos e experiências. Para isso, o evento é constituído de uma combinação de Oficinas e Palestras (a parte "escola"), proferidas por pesquisadores experientes, e apresentações de Trabalhos Completos e Resumos Estendidos (a parte "workshop").

Essa será a primeira edição do evento realizada em um estado no nordeste do Brasil, mostrando que o evento tem abrangência nacional.
Também foram recebidos trabalhos de diversos grupos de pesquisa, de diferentes estados do Brasil (RS, SC, PR, SP, RJ, AL, CE), fazendo com que a integração entre pesquisadores da área possa ser ampliada.

O WESAAC 2016 teve suporte financeiro da CAPES (nº do processo: 23038.004338/2016-84) e do IFAAMAS (http://www.ifaamas.org). Para viabilidade do evento, esses apoios foram fundamentais.

Gostaríamos de agradecer a todos os pesquisadores que submeteram os seus artigos, assim como aos membros do comitê de programa, aos revisores adicionais pelo criterioso trabalho desenvolvido, aos nossos colegas da UFAL e FURG que tornaram possível o WESAAC 2016.

Maceió/AL, 23 de Maio de 2016.

Baldoino Fonseca dos Santos Neto (Universidade Federal de Alagoas)
Coordenador Geral
Diana Francisca Adamatti (Universidade Federal do Rio Grande – FURG)
Coordenadora Comitê de Programa

APOIO

ORGANIZAÇÃO

Organização Geral

Baldoino Fonseca dos Santos Neto (UFAL)

Coordenação do Comitê de Programa

Diana Francisca Adamatti (FURG)

Organização Local

Evandro Costa (UFAL)

Comitê Consultivo

Diana Francisca Adamatti (FURG)

Anarosa Brandão (USP)

Rejane Frozza (UNISC)

Gustavo Alberto Giménez Lugo (UTFPR)

Jomi Fred Hubner (UFSC)

Comitê de Programa

Adamatti, Diana - FURG (Brasil)

Aguiar, Marilton Sanchotene de - UFPel (Brasil)

Alencar, Fernanda - UFPE (Brasil)

Araújo, Ricardo - UFPel (Brasil)

Balsa, Joao - Univ. Lisboa (Portugal)

Barbosa, Raquel - FURG (Brasil)

Bazzan, Ana L. C. - UFRGS (Brasil)

Billa, Cleo - FURG (Brasil)

Boissier, Olivier - EMSE (França)

Brandão, Anarosa Alves Franco - USP (Brasil)

Campos, André - UFRN (Brasil)

Carine Webber - UCS, (Brasil)

Casare, Sara - LIP6 (França)

Castro, Paulo André L. - ITA (Brasil)

Choren, Ricardo - IME/RJ (Brasil)

Cortés, Mariela Inés - UECE (Brasil)

Costa, Antonio Carlos Rocha - FURG (Brasil)

Coutinho, Luciano - UFMA (Brasil)

Dimuro, Graçaliz - FURG (Brasil)

Ferreira Jr., Paulo R. - UFPel (Brasil)

Giménez-Lugo, Gustavo - UTFPR (Brasil)

Gonçalves, Eder - FURG (Brasil)

Hubner, Jomi Fred - UFSC (Brasil)

Koch, Fernando - IBM Research (Brasil)

Lemke, Ana Paula - IFRS (Brasil)

Lorenzi, Fabiana - UFRGS (Brasil)

Marchi, Jerusa - UFSC (Brasil)

Moreira, Alvaro - UFRGS (Brasil)

Nardin, Luis Gustavo - University of Idaho (USA)

Nunes, Ingrid - UFRGS (Brasil)

Rabelo, Ricardo J. - UFSC (Brasil)

Ricci, Alessandro - University of Bologna (Italia)

Sichman, Jaime Simão - USP (Brasil)

Silva, Joao Luis - UCS (Brasil)

Silva, Viviane - UFF (Brasil)

Silveira, Ricardo Azambuja - UFSC (Brasil)

Simari, Guillermo Ricardo - Universidad Nacional del Sur (Argentina)

Tacla, Cesar A. - UTFPR (Brasil)

Trigo, Paulo - ISEL (Portugal)

Vasconcelos, Wamberto - University of Aberdeen (UK)

SUMÁRIO

ARTIGOS COMPLETOS

A Multi-Agent Extension of Hierarchical Task Network, de Rafael C. Cardoso e Rafael H. Bordini 1-12

Uma Abordagem de Transferência de Calor Utilizando Teoria Constructal e Modelagem Baseada em Agentes, de Paola Andrea Avendaño Montoya, Newton Nyamasege Marube, Diana Francisca Adamatti e Jeferson Souza 13-22

Uma Abordagem Baseada em Agentes para um Sistema de Classificação de Timbres, de Eduardo Porto Teixeira, Diana F. Adamatti and Eder M. Gonçalves 23-33

Processo de desenvolvimento de uma ferramenta gráfica de apoio a metodologia PrometheusAEOlus, de Rafhael Rodrigues Cunha, Diana F. Adamatti e Cléo Zanella Billa 34-45

Utilização de grau de confiança entre agentes para alocação de vagas em um Smart Parking, de Lucas Fernando Souza de Castro, Gleifer Vaz Alves e André Pinz Borges 46-57

Retrospective, Relevance, and Trends of Software Agents, Environments and Applications School (WESAAC), de Enyo Gonçalves, Mariela Cortés, Marcos Oliveira, Nécio Veras, Mário Falcão e Jaelson Castro 58-69

Failure Prediction based on Monitoring Sequences of Actions and Action Duration, de Giovani Farias, Ramon Fraga Pereira, Lucas Hilgert, Felipe Meneguzzi, Renata Vieira e Rafael H. Bordini 70-81

Uma análise comparativa da especificação formal em sistemas multi-agente: os desafios e as exigências uma década depois, de Newton Nyamasege Marube, Narúsci Bastos, Eder Mateus Gonçalves, Jader Saldanha e Carlos Quadros 82-93

Um Jogo Dramático baseado na Teoria do Drama para Autorregulação de Processos de Trocas Sociais, de Renata G. Wotter, Diana F. Adamatti, Graçaliz Pereira Dimuro, Lucas Tubino Bonifácio Costa e Nelson de Faria Traversi 94-105

Um sistema multiagente com leilões para a seleção de pacientes numa clínica odontológica, de Rodrigo Rabenhorst e Cleo Billa 106-116

Tomada de Decisão em Sistemas Multiagente utilizando personalidade e emoções, de Gerson Antonio Urban Filho e Diana F. Adamatti 117-128

Modelo de Reputação Fuzzy Aplicado a um Sistema Multiagente, de Henrique Rodrigues, Diana F. Adamatti, Graçaliz Dimuro, Glenda Dimuro e Esteban Jerez 129-140

RESUMOS EXTENDIDOS

A Conceptual Middleware for Adaptive Sanctioning in Normative Multi-Agent Systems, Igor Conrado Alves de Lima, Luis Gustavo Nardin e Jaime Simão Sichman 141-146

Integrando Requisitos Organizacionais à Modelagem de Sistemas Multiagente Normativos, de Emmanuel Sávio Silva Freire e Mariela Cortés 147-150

Rede Bayesiana de Emoções e sua Implementação em um Jogo Computacional baseado em Agentes, de Gustavo Fleck, Diana Adamatti e Adriano Werhli 151-156

Modelando a Curva de Crescimento do Mycobacterium tuberculosis com a utilização de simulação multiagente: um estudo de caso para a variável quorum sensing, de Marcilene Moraes, Albano Borba, Diana F. Adamatti e Adriano Werhli 157-162

An Analysis of Javino Middleware for Robotic Platforms Using Jason and JADE Frameworks, de Dayana Da Silva Junger, João Victor Guinelli e Carlos Eduardo Pantoja 163-168

Protocolo para diálogos argumentativos no auxílio da decisão consensual em sistemas multiagentes, de Ayslan Possebom, Mariela Morveli Espinoza e Cesar A. Tacla 169-174

Implementação de Recursos em um Smart Parking baseado em Sistemas Multiagente, de Felipe Felix Ducheiko, Lucas Fernando Souza de Castro e Gleifer Vaz Alves 175-180

Checagem de Consistência de Modelos de Sistemas Multiagentes Normativos Utilizando MAS-ML Tool, de Igor Nogueira, Mariela Cortés e Enyo Gonçalves 181-186

Modelagem da Teoria da Identidade Social Utilizando Sistemas Multiagentes, de Jader Saldanha, Narúsci Bastos, Cleo Billa, Graçaliz Dimuro e Diana F. Adamatti 187-192

A Multi-Agent Extension of Hierarchical Task Network

Rafael C. Cardoso[1] **and Rafael H. Bordini**[1]

[1]School of Informatics – FACIN-PPGCC
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brazil

rafael.caue@acad.pucrs.br, rafael.bordini@pucrs.br

Abstract. *Describing planning domains using a common formalism promotes greater reuse of research, allowing a fairer comparison between different approaches. Common planning formalisms for single-agent planning are already well established (e.g., PDDL, STRIPS, and HTN), but currently there is a shortage of multi-agent planning formalisms with clear semantics. In this paper, we propose a multi-agent extension of the Hierarchical Task Network (HTN) planning formalism. Our formalism, the Multi-Agent Hierarchical Task Network (MA-HTN), can be used in the representation of multi-agent planning domains and problems. We provide a grammar for the domain and problem representation, and show a case study with the translation from a JaCaMo system, a multi-agent system development platform, to our MA-HTN formalism.*

1. Introduction

Multi-Agent Systems (MAS) are often situated in dynamic environments where new plans of action need to be constantly devised in order to successfully achieve the system goals. Therefore, employing planning techniques during run-time of a MAS can be used to improve agent's plans using knowledge that was not previously available, or even to create new plans to achieve some goal for which there was no known course of action at design time.

Finding a solution to a planning problem consists of the following process: given a description of the initial states of the world (e.g., agents, environment), a description of the desired goals, and a description of a set of possible actions, the problem consists in finding a set of plans (i.e., sequence of actions) that when executed from any of the initial states will lead to the achievement of a goal. Therefore, it is beneficial to have a planning formalism in order to formally represent these problems, defining the syntax of the languages that are used to describe all of these descriptions.

The choice of a planning formalism is usually dependent on which planner is being used. The reason behind this is that most planners have their own formalism, or at least a variation of a well-defined one previously developed and accepted by the planning community. Because multi-agent planning research has been just recently getting more attention from the planning community, there is no de-facto standard to represent multi-agent planning domains yet.

In this paper we propose the Multi-Agent Hierarchical Task Network (MA-HTN) planning formalism, allowing the representation of multi-agent planning domains and problems for HTN planning. Because we are dealing with dynamic MAS, it means that we need to keep the description of the domain and problem constantly updated, in order

to provide the online planning with the best information possible, thus increasing the chances of it finding a viable solution. We show a case study where we developed a JaCaMo MAS of the Rover planning domain, and use a MA-HTN translator to parse information about the world currently available to the JaCaMo system into MA-HTN domain and problem representations.

The rest of this paper is structured as follows. In the next section we cover the necessary background on multi-agent systems and automated planning. Section 3 introduces our Multi-Agent extension to Hierarchical Task Network (MA-HTN), along with the grammar for the representation of domain and problems. Next, in Section 4, we use the Rover domain as our case study, to show the translation process from a MAS into MA-HTN. In Section 5, we discuss related work, and we end the paper with a description of future work and some concluding remarks.

2. Background

In this section we provide a brief background on multi-agent systems and automated planning. We start by describing the different abstraction levels that can be considered for programming multi-agent systems, and give an overview of JaCaMo, the multi-agent system development platform that was used in our case study. Next, we give a succinct description of the Hierarchical Task Network (HTN) planning formalism, discuss multi-agent planning and its different phases, and contextualise where our approach was designed to be used.

2.1. Programming MAS with Multiple Abstraction Levels

According to Bordini and Dix in [Bordini and Dix 2013], "Originally agent programming languages were mostly concerned with programming individual agents, and very little was available in terms of programming abstractions covering the social and environmental dimensions of multi-agent systems as well as the agent dimension". These multiple abstraction layers are what make multi-agent oriented programming especially suited for solving complex problems that require highly social, autonomous software. We now describe the social and environmental dimensions.

Organisations in a multi-agent system are complex entities in which agents interact in order to achieve some global purpose [Dignum and Padget 2013]. They provide scope for these interactions, reduce or manage uncertainty, and coordinate agents to improve efficiency. This is especially relevant to MAS in complex, dynamic, and distributed domains. These domains are very similar to those that can be found in multi-agent planning.

Environments in agent-based systems can be virtual or physical, or even in some cases both, as it can be beneficial to simulate parts of a physical environment as virtual elements. There are two main different views on the concept of environments in MAS. In classical AI, environment represent the external world that is perceived and acted upon by the agents so as achieve their goals [Russell and Norvig 2009]. A more recent view classify the environment as a first-class abstraction that encapsulates functionalities to support agent activities [Weyns et al. 2007].

2.1.1. JaCaMo

JaCaMo[1] [Boissier et al. 2011] combines three separate technologies into a platform for MAS programming that makes use of multiple levels of abstractions. Each technology (Jason, CArtAgO, and Moise) was developed separately for a number of years and are fairly established on their own when dealing with their respective abstraction level (agent, environment, and organisation). The overview of JaCaMo can be observed in Figure 1.

Moise [Hübner et al. 2007] handles the organisation level. Agents can adopt roles in the organisation, forming groups and sub-groups. Missions are defined to achieve the organisation goals. The behaviour of the agents that adopt roles to execute these missions is guided by norms.

Jason [Bordini et al. 2007] is responsible for the agent level, it is an extension of the AgentSpeak language. Based on the Belief-Desire-Intention architecture, agents in Jason react to events in the system by executing actions on the environment, according to the plans available in each agent's plan library.

CArtAgO [Ricci et al. 2009] is based on the A&A (Agents and Artefacts) model, and deals with the environment level. Artefacts are used to represent the environment, storing information about the environment in observable properties and providing actions that can be executed through operations. When an agent focuses on an artefact, it receives the observable properties as beliefs, and it is able to execute the artefact's operations. Artefacts are grouped in workspaces, which can be distributed across multiple network nodes, providing a natural distribution to the MAS.

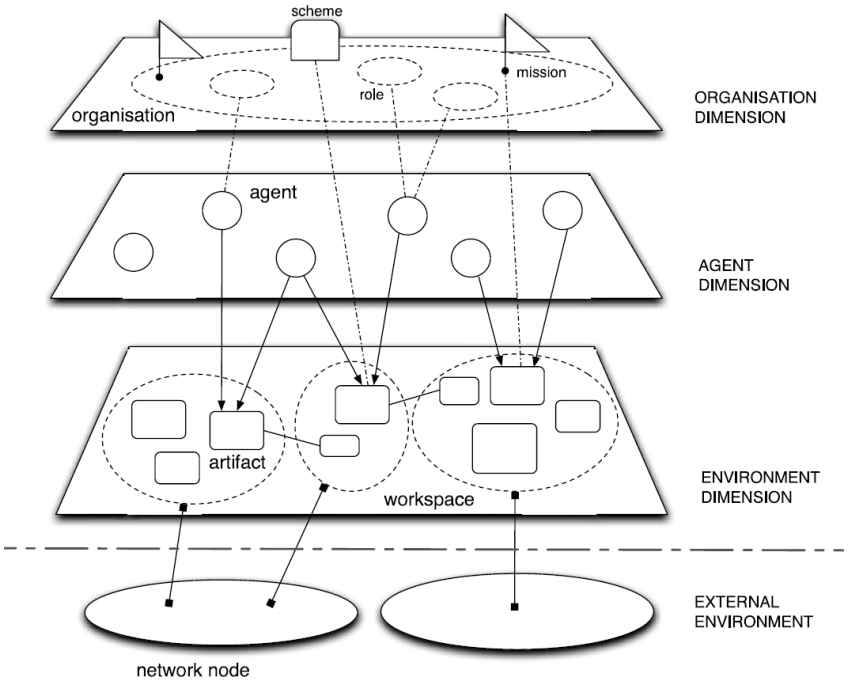


Figure 1. JaCaMo overview [Boissier et al. 2011].

[1] http://jacamo.sourceforge.net/.

2.2. Automated Planning

Automated planning is the computational study of planning, which is an abstract deliberation process on choosing and ordering actions in order to achieve some goals in the best way possible [Nau et al. 2004]. This is done by anticipating the outcome of these actions, but not every goal needs to be planned out. The deliberation process can take some time to find the best possible solution, thus sometimes merely reacting is the best approach, and then adapting to the consequences.

2.2.1. Hierarchical Task Network

HTN planning [Nau et al. 2004] is one of the techniques for automated planning. It works by decomposing tasks into subtasks, until arriving at primitive tasks that can solve the planning problem. This convenient way of writing recipes is more closely related to how a human expert would think about solving a planning problem, thus making HTN planning more suited for practical applications. Besides, the extra domain information contained in non-primitive tasks usually results in better performance than with the other types of planners.

An HTN planning domain representation contains a set of operators and a set of methods. Operators are action descriptors that can be executed given some preconditions, causing a list of postconditions to become true. They can cause a state transition to occur in the system, while methods can only decompose tasks into smaller subtasks, which can eventually lead to primitive tasks wherein an operator can be applied. An HTN planning problem representation contains a list of atoms that are true during the initial state of the system, as well as the goals of the system.

2.2.2. Multi-Agent Planning

Multi-Agent Planning (MAP) has often been interpreted as two different things. Either the planning process is centralised and produces distributed plans that can be acted upon by multiple agents, or the planning process itself is multi-agent. Recently, the planning community has been favouring the concept that MAP is actually both of these things, that is, the planning process is done *by* multiple agents, and the solution is *for* multiple agents.

When considering multiple agents the planning process gets increasingly more complicated, giving rise to several problems [Durfee and Zilberstein 2013]. Actions that agents choose to make may cause an impact in future actions that the other agents could take. Likewise, if an agent knows what actions the other agents plan to take, it could change its own current choices. When dealing with multiple agents, concurrent actions are also a possibility and have to be dealt with. These are some of the problems that drive the research on MAP.

Durfee also establishes some useful phases for multi-agent planning in [Durfee 1999], further extended in [Weerdt and Clement 2009]: 1) *global goal refinement*, decomposition of the global goal into subgoals; 2) *task allocation*, use of task-sharing protocols to allocate tasks (goals); 3) *coordination before planning*, coordination mechanisms that prevent conflicts before planning; 4) *individual planning*,

planning algorithms that search for solutions for the problem; 5) *coordination after planning* coordination mechanisms that prevent conflicts after planning; and 6) *plan execution*, the agents carry out the solution found.

3. The MA-HTN Formalism

A planning problem consists of the following process: given a description of the initial states of the world (e.g., agents, environment), a description of the desired goals, and a description of a set of possible actions, the problem consists of finding a set of plans (i.e., sequence of actions) that when executed from any of the initial states will lead to the achievement of a goal. Therefore, it is beneficial to have a planning formalism in order to formally represent these problems, defining the syntax of the languages that are used to describe all of these representations.

We propose the Multi-Agent Hierarchical Task Network (MA-HTN) formalism, which is an extension of the centralised single-agent HTN formalism used in the SHOP2 planner [Nau et al. 2003]. MA-HTN is intended to represent *online* multi-agent planning problems, since domain and problem information are collected during execution. Thus, unlike in offline planning where these two can be specified before execution (e.g., by a system designer or a computer script), here there is a need for a mechanism to collect all of the necessary data and translate it to an input that can be useful to a planner.

We call this mechanism the *translator*, and agents use it to translate their information about the world into domain and problem specifications that can then be passed to a planner. The translator obtains information about the current state of the world from the MAS in execution, using it to generate the problem representation. The domain representation is generated from the possible actions and plans that the agents have access to.

Each agent has their own problem and domain specification. This provides a decent level of privacy on its own, since each planner only has access to their respective agent problem and domain specifications. This means that, unlike some of the other multi-agent planning formalisms, MA-HTN does not need to have privacy or public blocks. Although at some point it might be interesting to add the capability to include private goals into the formalism, for now we are interested only on representing organisational goals.

Actions from other agents can cause conflicts, either at the moment that action is executed (e.g., concurrent actions) or in the future (e.g., durative actions). For this reason, MA-HTN supports the characterisation of actions that can cause *conflict*. The recognition of these actions is not automated, though a mechanism for that purpose could be used. These actions that can cause conflicts have to be annotated by the MAS developer, in order for the translator to be able to identify them. Actions are always translated to operators in HTN, thus, only operators can cause conflicts, methods cannot.

Likewise, dependencies between actions can also exist, either as a concurrent action that requires another agent or actions that depend on actions of other agents to happen first. Similarly to conflicts, MA-HTN also supports the use of *dependency* blocks to identify actions that depend on actions from other agents. These dependency relations also have to be annotated by the MAS developer, so that the translator can add them to the specification. They also cannot be used in methods, only in operations.

The notation usually preferred to specify context-free grammars is the Backus–Naur Form (BNF). We use BNF grammars to define the specifications of MA-HTN domain and problem representations. We provide a simplified grammar to improve readability, where each single quote pair that encloses a symbol is considered a string that is expected by the planner, symbols enclosed by brackets are optional, and symbols preceded by $ are variables obtained from the MAS and represent terminal symbols. Symbols that end with the * signal, represent that zero or more instances are possible. Symbols that end with the + signal, represent that one or more instances are possible. Because MA-HTN is based on the SHOP2 HTN formalism [Nau et al. 2003], the language itself is LISP-like, though we omitted many of the necessary parenthesis in order to improve readability.

3.1. Domain Representation

In Listing 1, we show our simplified BNF grammar for multi-agent domains. The *$domain-name* variable is defined dynamically by the agent during execution of the MAS, and since there could be multiple calls to the same domain and the specification can be different from the previous call (e.g., a method could be deleted, added, or modified), agents use a counter id that increments each time planning is invoked. The name of the agent, *$agent-name* is included in the specification to represent which agent this domain belongs to. The *conflict-list* and *dependency-list* are added. Conflicts represent actions that can cause negative interactions between agents, while dependencies are actions that need other agents to succeed. The rest of the definitions are similar to SHOP2 HTN, *operators* are *primitive-tasks* while *methods* are *non-primitive-tasks* that can eventually be decomposed into *operators*.

Listing 1. MA-HTN BNF grammar for representing domains.

```
1  def-domain          ::= 'defdomain' $domain-name ;
2  agent               ::= 'agent' $agent-name ;
3
4  task                ::= primitive-task | non-primitive-task ;
5  primitive-task      ::= '!'$primitive-task-name '?'$variable* ;
6  non-primitive-task  ::= $non-primitive-task-name '?'$variable* ;
7
8  def-operator        ::= ':operator' primitive-task precondition-list delete-list add-list conflict-list
       dependency-list ;
9  precondition-list   ::= precondition* ;
10 precondition        ::= ('not' $precondition-name '?'$variable*) |
       ($precondition-name '?'$variable*) ;
11 delete-list         ::= delete* ;
12 delete              ::= $delete-name '?'$variable* ;
13 add-list            ::= add* ;
14 add                 ::= $add-name '?'$variable* ;
15 conflict-list       ::= conflict* ;
16 conflict            ::= $action-name $agent-name ;
17 dependency-list     ::= dependency* ;
18 dependency          ::= $action-name $agent-name ;
19
20 def-method          ::= ':method'" non-primitive-task (precondition-list task-list+) ;
21 task-list           ::= task+ ;
```

3.2. Problem Representation

Listing 2 shows our simplified BNF grammar for multi-agent problems. Similarly to the domain grammar, the *$problem-name* is specified, along with a reference to its respective *$domain-name*. We also have the explicit agent symbol on line 2. Remember that in our formalism each agent has its own domain and problem representations, which gives some sense of natural privacy to the system. *Facts* can be used to establish types and characteristics of things in the world, for example, *location kitchen* establishes that *kitchen* is a *location* in the world. While *initial-states* are used to represent what is true in the world at that specific moment in time, for example, *kitchen dusty* represents that the *kitchen* is *dusty*. *Goals* can be listed as either *ordered* – when the order in which the goals have to be achieved needs to be strictly followed; or *unordered* – when the order is unknown and the planner is free to find any order between goals.

Listing 2. MA-HTN BNF grammar for representing problems.

```
1  def-problem          ::= 'defproblem' $problem-name $domain-name ;
2  agent                ::= 'agent' $agent-name ;
3
4  def-facts            ::= fact-list ;
5  fact-list            ::= fact* ;
6  fact                 ::= $fact-name $fact-parameter+ ;
7
8  def-initial-states   ::= initial-state-list ;
9  initial-state-list   ::= initial-state+ ;
10 initial-state        ::= $initial-state-name $initial-state-parameter+ ;
11
12 def-goals            ::= (':ordered' | ':unordered') goal-list ;
13 goal-list            ::= goal+ ;
14 goal                 ::= $goal-name $goal-parameter+ ;
```

4. Case Study

As our case study we use the Rover domain, which has been used in several past IPCs. The Rovers domain was constructed as a simplified representation of the NASA Mars Exploration Rover missions launched in 2003 and other similar missions. This domain involves planning for several rovers, equipped with different, but possibly overlapping, sets of equipment to traverse a planet surface. The rovers must travel between waypoints gathering data and transmitting it back to a lander. The traversal is complicated by the fact that certain rovers are restricted to travelling over certain terrain types and this makes particular routes impassable to some of the rovers. Data collection involves the collection of rock and soil samples located in waypoints, as well as taking images (three different modes are available: high_res, low_res, colour) of certain objectives that are visible from waypoints. Data transmission is also constrained by the visibility of the lander from the waypoints.

The Rover domain was initially conceived as a domain for single-agent planning. Thus, we made a few extensions to turn it into a suitable multi-agent planning domain, such as further specialising rover vehicles into: vehicles for rock analysis, vehicles for soil analysis, and photographer vehicles. We also establish that the action to transmit data

to the lander is a possible point of conflict, since the channel might be busy (e.g., another agent is currently transmitting its data). A dependency in this domain is that collections of soil and rock that are located in a waypoint from which an active objective (e.g., a goal for taking images of the objective exists) is visible from, can only be collected after the image of the active objective has been taken.

We implemented this domain as a MAS using the JaCaMo MAS development platform. The *.jcm* configuration file contain all of the information necessary for starting the MAS, and we used it to set up the problem for the Rover domain. We added a *ground_team* agent, which is represented as an agent here just to simulate the input provided by humans, such as creating new dynamic goals for the robots during execution of the MAS. There are three rovers in this instance, one for each specialisation (rock analysis, soil analysis, and photographer). Then, we created four waypoint artefacts, one for each of the four waypoints in this problem, two artefacts for the objectives, and one for the lander. Each agent also has its own personal artefact, containing its starting parameters and the actions that it is able to perform. The goals in this problem are to take images of *objective1* and *objective2*, to collect soil data of *waypoint3*, and to collect rock data of *waypoint4*.

The MA-HTN translator generates problem and domain representations for each agent, as follows:

- **Problem representation:** The name of the problem and the name of the domain are obtained dynamically. The name of the agent is gathered by using a Java function that returns the name of the agent who started the translator. The information collected from the CArtAgO artefacts are parsed into initial states that form the agent's fact list and initial state list. The goal list is created from the organisational goals that were assigned to this particular agent during the goal allocation phase.
- **Domain representation:** The name of the name of the domain is obtained dynamically. The name of the agent is gathered by using a Java function that returns the name of the agent who started the translator. Operators are parsed from all of the artefacts operations that the agent has access to. The precondition list is obtained from any conditional tests in an operation, the delete and add list from the deletion and addition of observable properties, and the conflict and dependency lists need to be previously annotated into the operation in order for them to be able to be parsed. The methods are parsed from all of the plans in the agent's plan library, with the precondition list parsed from the context of the plan and the task list parsed from the body of the plan.

In Listing 3, we show the *.jcm* code with some of the configuration of our case study. For example, agent *rover1* contains the name, code filename of the agent, and all of the actions that agents should take when the system starts. For instance: which workspace they should *join*; what artefacts they should *focus*; and the roles that they should *adopt*.

Listing 3 also shows some of the environment artefacts, such as the *waypoint1* artefact that sets its three observable properties: *visible*, which determines the list of waypoints that can be reached to transmit data, in this case waypoints 2, 3, and 4; *rock_sample*, which determines if there are rock samples available in this waypoint, in this case there is; and *soil_sample*, which determines if there are soil samples available in this waypoint, in this case there is not. We also show as an example the workspace with the artefact for

rover1, with its two observable properties: *at*, which waypoint the agent is currently at, in this case the agent is at *waypoint2*; and *can_traverse*, which contains the list of terrains that the vehicle is capable of traversing.

Listing 3. A snippet of the .jcm file for the Rover case study.

```
agent rover1 : rover.asl {
    join: w1, r1
    focus: w1.lander, w1.way1, w1.way2, w1.way3, w1.way4, w1.obj1, w1.obj2, r1.rover1
    roles: rock_analysis in o1.g1
}
workspace w1 {
    artifact way1: rovers.Waypoint(["way2","way3","way4"],"true","false")
    artifact obj1: rovers.Objective(["way1","way2","way3","way4"])
    artifact lander: rovers.Lander("way1","free","")
    ...
}
workspace r1 {
    artifact rover1: rovers.Rover("way2",[par(way2,way1),par(way1,way2),par(way2,way4),par(
    way4,way2)],"obj2")
}
```

In the Moise structural specification, Listing 4 of the organisation, five roles are defined, with three of them being specialisations of the *rover* role. The group specification (omitted from the Listing) defines the cardinality for each role that is required to fully form the group, which reflects the number of agents in the system in our case, 1 ground team, 1 rover for rock analysis, 1 rover for soil analysis, and 1 photographer rover. Group specifications also contain the designation of links. In this case there are two links, an authority link from the *ground_team* to *rover* roles, and an acquaintance link from *rover* to *rover*. The authority link means that the *ground_team* can send goals and command directives to *rovers*, while the acquaintance link allows *rovers* to communicate with each other.

Listing 4. The structural specification of the rover organisation.

```
<structural-specification>
    <role-definitions>
     <role id="ground_team" />
     <role id="rover" />
        <role id="rock_analysis" >          <extends role="rover"/> </role>
        <role id="soil_analysis" >          <extends role="rover"/> </role>
        <role id="photographer" >           <extends role="rover"/> </role>
    </role-definitions>
    <links>
        <link from="ground_team" to="rover" type="authority" scope="
    intra-group" extends-subgroups="false" bi-dir="false"/>
        <link from="rover" to="rover" type="acquaintance" scope="
    intra-group" extends-subgroups="false" bi-dir="false"/>
    </links>
</structural-specification>
```

We show an example of a plan in Jason in Listing 5 of the photographer agent, and an operation from the photographer agent artefact in Listing 6. In the problem representation, CArtAgO observable properties (i.e., the current information about the state of the world) become the initial states. The goals of the organisation become the goal list. Eventually we would like to directly extract these goals from the Moise specification, but we are still investigating the best way to do this. In the domain representation, CArtAgO operations (which are the actions that can be executed in the environment) become operators, and the plans from the Jason agents plan library become methods.

Listing 5. Example of a Jason plan.

```
+!get_image_data(Objective, Mode)
    : visible_from(Objective,Area) & supports(Mode) & .my_name(Name)
<-
    !navigate(Area);
    take_image(Area,Objective)[artifact_id(Name)].
```

Listing 6. Example of a CArtAgO operation.

```
@OPERATION void take_image(String objective, String visible_from) {
    ObsProperty cond1 = getObsProperty("at");
    if ( visible_from.stringValue().contains(cond1.toString()) )
    {
        defineObsProperty("have_image", objective);
    } else {
        failed("Action take_image has failed.");
    }
}
```

5. Related Work

STRIPS is an early automated planning system from 1971 [Fikes and Nilsson 1971], that still provides the basis for many classical planners with its action theory and formalism. In STRIPS, each operator has a precondition list, add list, and delete list. These lists were allowed to contain arbitrary well-formed formulas in first-order logic. However, there were a number of problems with this formulation, such as the difficulty of providing a well-defined semantics for it [Nau et al. 2004]. The PDDL (Planning Domain Definition Language) contains STRIPS-like operators, and has been the formalism of choice in several past IPCs. The latest version of PDDL is 3.1[2].

In HTN planning, a well-established formalism is the one accepted by the SHOP2 planner [Nau et al. 2003]. It differs from PDDL in the sense that it does not necessarily involve state variables. In this formalism, planning domains contain a set of operators, methods, and axioms. Planning problems contain a set of logical atoms (i.e., facts that represent the initial state of the world), and tasks lists (i.e., high-level goals to achieve).

[2] http://ipc.informatik.uni-freiburg.de/PddlExtension

Although it is possible to represent multiple agents using those formalisms, there is no distinction between agents and the other objects of the world. This makes it difficult to represent important features of MAP, such as conflicts, dependencies, privacy, and distribution. Thus, many single-agent formalisms have been expanded to allow for the explicit description of agents. For example, in MA-STRIPS [Brafman and Domshlak 2008] the authors propose a multi-agent extension of STRIPS formalism for cooperative multi-agent systems. Besides adding the notion of agents containing their own set of actions, dependencies can be identified to classify an agent's actions into internal or public.

A multi-agent extension of PDDL 3.1 [Kovacs 2012] was designed to cope with the agents' different abilities and the constructive and destructive nature of concurrent actions. Its design is based on several requirements for multi-agent planning formalisms, those of note are: modelling concurrent actions with interacting effects; agents can have different actions/goals/utilities; straightforward association of agents and actions; distinction between agents and non-agent objects; and inheritance/polymorphism of actions/-goals/utilities. The extension can also be used to represent these problems in temporal, numeric domains.

6. Conclusions

In this paper we described the MA-HTN formalism, a multi-agent variation of the traditional single-agent HTN formalism. We also described a translator that can parse current information about the world available to a JaCaMo system into MA-HTN domain and problem representations. A case study with the Rover domain was used to illustrate the translation process and helped in comparing with the traditional (single-agent) HTN formalism.

By using agents as first-class abstractions in our formalism, we are free of the use of predicates that are usually used to assign tasks between different objects. In turn, this also lowers the number of expansions and inferences required during the planning process, which should improve planning time and performance. At any point during the execution of the MAS, the parameters of the problem may change, new dynamic goals may be created, paths can become obstructed, new agents might join the system, etc. Thus, our formalism serves as a bridge to online planners, allowing access to up-to-date information about the current state of the MAS.

Future work consists of extending the MA-HTN formalism to allow for other types of possible agent interactions, besides conflicts and dependencies as we presented here, such as privacy and distribution. It is also important to make experiments in other domains, as well as to check the suitability of translating MAS into MA-HTN domains (instead of developing MAS from planning domains). Finally, experiments with the use of the MA-HTN formalism in a planner also have to be conducted in order to provide a practical evaluation of the formalism. Another useful extension to MA-HTN is to be able to translate the solution found by a planner into plans that can be added to the respective agent's plan library.

Acknowledgements

We are grateful for the support given by CAPES and by CNPq (grant number 308095/2012-0).

References

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

Bordini, R. H. and Dix, J. (2013). Programming multiagent systems. In Weiss, G., editor, *Multiagent Systems 2nd Edition*, chapter 11, pages 5870–639. MIT Press.

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

Brafman, R. I. and Domshlak, C. (2008). From One to Many: Planning for Loosely Coupled Multi-Agent Systems. In *ICAPS*, pages 28–35.

Dignum, V. and Padget, J. (2013). Multiagent organizations. In Weiss, G., editor, *Multiagent Systems 2nd Edition*, chapter 2, pages 51–98. MIT Press.

Durfee, E. H. (1999). Distributed problem solving and planning. In *Mutliagent systems*, pages 121–164. MIT Press.

Durfee, E. H. and Zilberstein, S. (2013). Multiagent planning, control, and execution. In Weiss, G., editor, *Multiagent Systems 2nd Edition*, chapter 11, pages 485–545. MIT Press.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: a new approach to the application of theorem proving to problem solving. In *Proceedings of the 2nd international joint conference on Artificial intelligence*, IJCAI'71, pages 608–620, San Francisco, CA, USA.

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

Kovacs, D. L. (2012). A multi-agent extension of pddl3.1. In *Proceedings of the 3rd Workshop on the International Planning Competition (IPC)*, ICAPS-2012, pages 19–27, Atibaia, SÃ£o Paulo, Brazil.

Nau, D., Ghallab, M., and Traverso, P. (2004). *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Nau, D., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F. (2003). Shop2: An htn planning system. *Journal of Artificial Intelligence Research*, 20:379–404.

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

Russell, S. J. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition.

Weerdt, M. d. and Clement, B. (2009). Introduction to Planning in Multiagent Systems. *Multiagent Grid Syst.*, 5(4):345–355.

Weyns, D., Omicini, A., and Odell, J. (2007). Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):5–30.

Uma Abordagem de Transferência de Calor Utilizando Teoria Construtal e Modelagem Baseada em Agentes

Paola A. Avendaño[1], Newton N. Marube[2], Diana F. Adamatti[1], Jeferson Ávila Souza[1]

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

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

pao.andrea9030@gmail.com, nyamasege@gmail.com, dianaada@gmail.com, jasouza@furg.br

Abstract. *Most of heat transfer problems are focused on understanding how the geometries inside the solid surfaces under influence the performance of heat transfer. The present study consists of a heat transfer plate built from two types of materials and aims to find the best geometry in the plate, such that the sum of the temperatures of its elements is the smallest possible. An agent-based model developed in NetLogo tool using the constructal theory is presented. The results obtained were compared with those obtained from randomly arranging the materials and it was found that using the constructal theory it is possible to obtain optimal geometries.*

Resumo. *A maioria dos problemas de transferência de calor estão focados em compreender como as geometrias dentro de superfícies sólidas influenciam o desempenho da transferência de calor. O problema abordado no presente trabalho consiste em uma placa com transferência de calor construída com dois tipos de materiais e o objetivo consiste em procurar a melhor geometria dentro da placa de forma que, a soma das temperaturas dos seus elementos seja a menor possível. É apresentado um algoritmo desenvolvido no ambiente NetLogo e com base na Teoria Construtal. Os resultados fornecidos foram comparados com os obtidos dispondo os materiais de maneira aleatória e se concluiu que ao usar a Teoria Construtal é possível obter geometrias ótimas.*

1. Introdução

Nos últimos anos, problemas de engenharia relacionados à área de transferência de calor receberam grande atenção. O objetivo de muitos desses trabalhos, centra-se em explicar como a geometria dentro de superfícies sólidas estudadas influencia no comportamento da transferência de calor e buscam, além de entender, melhorar seus desempenhos e buscar novas geometrias. O estudo da transferência de calor é de grande importância para diversas aplicações como por exemplo, para o aumento da eficiência de motores de turbina a gás e de trocadores de calor, para estudar a eficiência de motores de combustão interna, motores elétricos e condutores térmicos, para garantir o controle preciso de temperaturas em sistemas de microescala como as mídias de armazenamento

e a redução das temperaturas de operação de peças no interior de computadores pessoais.

Por outro lado, a Teoria Construtal está se tornando uma metodologia poderosa por ser entendida como uma geração da tendência de todos os objetos da natureza em fluir através de caminhos que ofereçam menor "resistência". Sendo um princípio físico que une o animado com o inanimado [Bejan 2000]. Por ser uma Teoria geral, esta pode ser aplicada em todos os domínios, onde existe algum tipo de evolução. Assim, a mesma vem sendo utilizada na solução de problemas de biologia, física, organizações sociais, evolução tecnológica, sustentabilidade e engenharia, entre outros [Bejan e Zane 2012].

Apesar de que existem inúmeros trabalhos que abordam à otimização em placas submetidas a alguma fonte de calor, não foram encontrados trabalhos relacionados ao assunto que tenham utilização modelagem e simulação baseada em agentes. Assim, o objetivo desse trabalho é apresentar uma modelagem e simulação deste problema no escopo de agentes.

No capítulo 2 é feita uma revisão sobre a transferência de calor e a teoria construtal, apresentando alguns trabalhos relacionados. No capítulo 3 é descrito o algoritmo desenvolvido usando modelagem baseada em agentes em conjunto com a teoria construtal. No capítulo 4 encontram-se resultados obtidos após simulações feitas utilizando o algoritmo implementado. Finalmente, no capítulo 5 são discutidos os resultados, feitas as conclusões e apresentadas propostas para trabalhos futuros.

2. Referencial Teórico

2.1. Difusão de Calor (Difusão Térmica)

A condução de calor é a transferência de energia térmica que ocorre devido a iteração entre as moléculas (colisões) a qual é motivada por um gradiente de temperatura.

Um dos objetivos principais da análise da condução de calor é determinar o campo de temperaturas em um meio resultante das condições impostas em suas fronteiras. Ou seja, deseja-se conhecer a distribuição de temperaturas, que representa a variação de temperatura em relação à posição no meio.

A equação (1) é a forma geral, em coordenadas cartesianas, da equação da condução de calor em um sólido. Essa equação, frequentemente chamada de equação do calor, fornece a ferramenta básica para a análise da condução do calor. A partir de sua solução, pode-se obter a distribuição de temperaturas *T(x,y)* como uma função do tempo [Incropera, et al. 2000].

$$\rho c_p \frac{\delta T}{\delta t} = k \frac{\delta^2 T}{\delta x^2} + k \frac{\delta^2 T}{\delta y^2} + q''' \qquad (1)$$

onde: *T* é a temperatura *[K]*, *t* é o tempo *(s)*, *x* e *y* são as coordenadas cartesianas *[m]*, *k* é a condutividade térmica *[W/m K]*, ρ é a massa específica *[kg/m³]*, c_p é o calor específico *[W/kg K]* e *q'''* é a taxa de geração de calor *[kW/m³]*.

Os métodos tradicionais para a solução numérica de equações diferenciais são os Métodos de Diferenças finitas (MDF), de Volumes Finitos (MVF) e de Elementos Finitos (MEF).

A natureza da solução por diferenças finitas dependerá do instante de tempo específico no qual as temperaturas estão sendo determinadas. A formulação explicita permite expressar a temperatura de um elemento da malha em função da temperatura de todos os vizinhos no instante anterior, as quais devem ser conhecidas. Essa formulação origina um conjunto de equações algébricas que podem ser resolvidas uma a uma, obtendo-se a temperatura em cada ponto do espaço para o novo nível de tempo.

A forma explícita da equação de diferenças finitas para o ponto *(m,n)* no instante de tempo *p* e a discretização mostrada na figura 1 é:

$$\frac{1}{\propto}\frac{T_{m,n}^{p}-T_{m,n}^{p-1}}{\Delta t}=\frac{T_{m+1,n}^{p-1}-T_{m-1,n}^{p-1}-2T_{m,n}^{p-1}}{(\Delta x)^2}+\frac{T_{m,n+1}^{p-1}-T_{m,n-1}^{p-1}-2T_{m,n}^{p-1}}{(\Delta y)^2}+q''' \qquad (2)$$

onde: $\propto=\frac{k}{\rho c_p}$

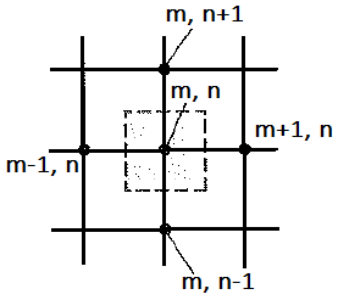


Fig.1 Esquema de discretização de diferenças finitas [Incropera, et al., 2000]

2.2. Teoria Construtal

A teoria construtal responde pela tendência universal de sistemas de fluxo de se transformar em configurações em evolução que proporciona maior e mais fácil acesso ao fluxo ao longo do tempo. A teoria construtal resolve as muitas e contraditórias declarações ad hoc de otimização, design final, e seu destino na natureza, tais como a máxima e mínima de entropia de produção e máxima e mínima de resistência de fluxo, e também explica os designs que são observados e copiados na biomimética [Bejan 2000].

Muitos estudos relacionados a transferência de calor já foram desenvolvidos usando a lógica proposta pela teoria construtal. Souza e Ordonez (2013), por exemplo, propõem um algoritmo de otimização baseado na teoria construtal para procurar uma geometria ótima, objetivando minimizar o calor dentro da placa. Neste trabalho, foram calculados os gradientes de temperatura dos elementos de baixa condutividade e estes elementos foram trocados por material de alta condutividade (um grupo de elementos que tenha apresentado o maior gradiente). O método utilizado pelos autores para calcular os gradientes de temperatura dentro do domínio computacional (uma região retangular cujos pontos interiores estão igualmente espaçados) foi o Control Volume

Finite Element Method (CVFES). Na figura 2 pode-se observar as geometrias que foram encontradas nas simulações e suas semelhanças com árvores.

De acordo com a quantidade de material inserido durante cada iteração (par ou impar), as árvores podem ser simétricas ou assimétricas. Além disso, os autores concluem que os resultados obtidos dependem tanto da quantidade de elementos substituídos a cada passo do tempo, quanto da relação da condutividade dos materiais (quantas vezes um é mais condutivo que o outro).

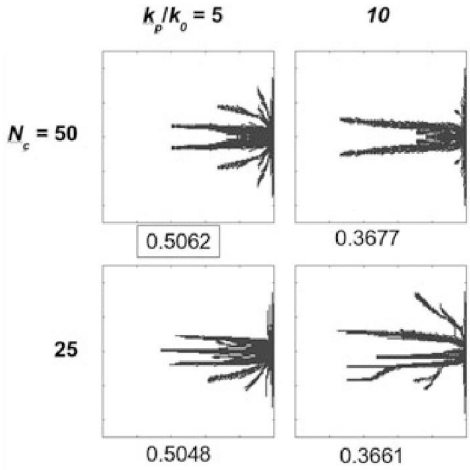


Fig.2. Formação de padrões e temperatura mínima [Sousa e Ordonez 2013]

3. Modelo Baseado em Agentes

Neste trabalho é apresentado um problema de transferência de calor por condução em regime permanente, no qual se tem uma placa sólida bidimensional quadrada, construída com dois materiais diferentes, um de alta e um de baixa condutividade térmica. Existe uma temperatura prescrita em todas as bordas da placa e uma temperatura inicial no interior.

O objetivo principal do trabalho é determinar posições apropriadas para os materiais de alta e baixa condutividade com os quais a placa é construída de maneira que possa se minimizar a soma das temperaturas de todos os elementos. A figura 3 mostra as condições de contorno utilizadas e uma possível configuração para o problema, onde a região cinza representa o material de alta condutividade e a região branca o material de baixa condutividade dentro da placa.

A geração de calor ocorre só nas regiões de material de baixa condutividade, assim, a equação para a região branca da figura 3 está dada por:

$$\rho c_p \frac{\delta T}{\delta t} = k_0 \frac{\delta^2 T}{\delta x^2} + k_0 \frac{\delta^2 T}{\delta y^2} \qquad (3)$$

onde: k_0 representa a condutividade do material de baixa condutividade *[W/m K]*.

Modelagem baseada em agentes é usualmente usada em sistemas complexos. Considera-se que os fenômenos simples e complexos podem ser o resultado de iterações

entre indivíduos autônomos e independentes, os quais operam dentro dos sistemas de acordo com diferentes formas de iteração [Bandini et al. 2009].

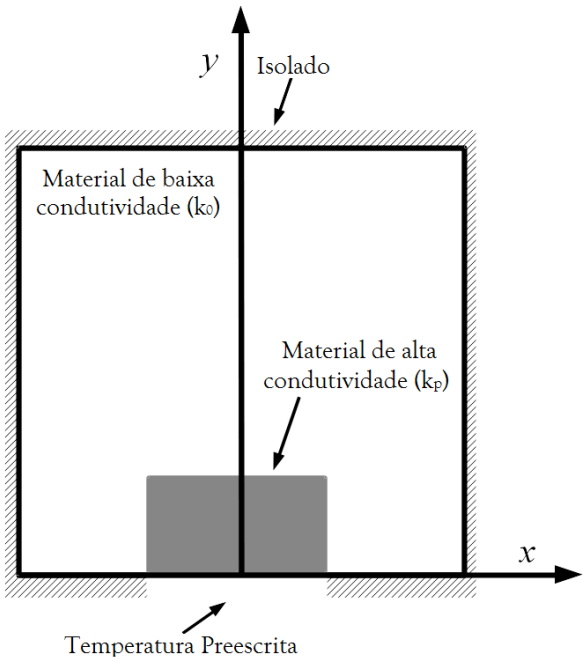


Fig.3. Descrição do problema

O modelo apresentado estende o modelo de difusão de calor de NetLogo que simula a distribuição de temperatura em estado transiente e estacionário, em função do tempo e localização de uma placa fina. A modelagem baseada em agentes é utilizada nesse modelo de difusão de calor para ilustrar a dependência do tempo que é observada durante a simulação. À medida que a simulação é executada, o calor é transmitido desde as partes mais quentes para as partes mais frias da placa, como pode se ver na cor que varia dentro da placa. Portanto, a temperatura da placa começa a mudar imediatamente e, possivelmente, de forma diferente em distintos lugares, convergindo gradualmente para um estado estacionario [Tisue e Wilensky, 2004].

NetLogo é uma linguagem de programação útil para a simulação de diversos fenômenos sociais, particularmente para a simulação de problemas complexos que evoluem no tempo.

Dentro do ambiente de programação existem centenas ou milhares de elementos que são chamados de “agentes”, que de maneira independente podem receber instruções ou guardar informação que é oferecida pelo modelador, o que faz com que seja possível subtrair a cada iteração diferentes informações do tipo global ou pontual [Tisue e Wilensky, 2004].

Desta forma, desenvolveu-se um algoritmo baseado em agentes que utiliza a Teoria Construtal para a busca das posições ótimas do material de alta condutividade dentro da placa. A ferramenta utilizada para desenvolver o algoritmo foi NetLogo.

Para atingir o objetivo do trabalho, primeiramente é necessário dividir o domínio computacional em pequenos elementos, todos com condutividade térmica baixa e calcular o campo de temperaturas usando o método de Diferenças finitas e uma

formulação explicita. Após chegar à solução de regime permanente (quando a mudança das temperaturas de uma iteração para outra se torna desprezível) deve-se trocar um grupo de elementos de baixa condutividade k_0 que apresenta a maior temperatura por material de alta condutividade k_p. O processo de substituição deve se repetir até usar a quantidade de material de alta condutividade disponibilizado para o problema. A figura 3 é um exemplo para mostrar como ocorre o processo de substituição sequencial. Para o caso, trata-se de uma placa com 441 elementos, na qual é substituído um elemento de baixa condutividade térmica por um de alta a cada iteração até ter uma placa com 5% de material com k_p.

A figura 4 (a) mostra domínio computacional com todos os elementos tendo condutividade baixa. As figuras 4 (b) e (c) mostram as duas primeiras iterações em que ocorre a substituição do material que começaram só após se atingiu o regime permanente. A figura 4 (d) mostra a configuração obtida depois de substituir 22 elementos que correspondem ao 5% do material.

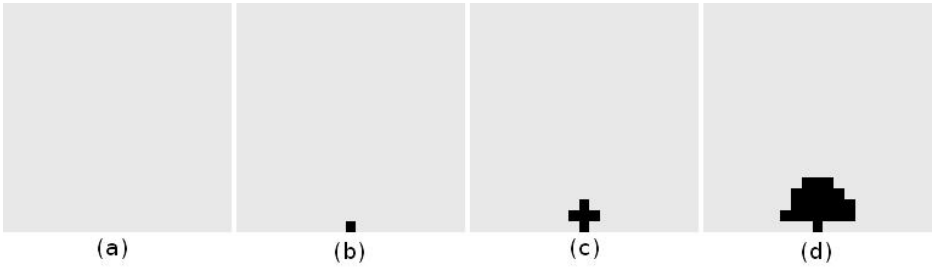


Fig.4. Processo de substituição sequencial

A seguir é apresentado o pseudo-algoritmo proposto:

Etapa 1: fazer a discretização do domínio computacional, isto é, dividir a placa em pequenos elementos.

Etapa 2: definir as condições de contorno do problema, a relação (k_p/k_0) entre as condutividades dos materiais, a porcentagem total de elementos com condutividade k_p que serão usados e a quantidade (N_c) de elementos que serão substituídos a cada iteração.

Etapa 3: atribuir baixa condutividade para todo o domínio computacional.

Etapa 4: calcular o campo de temperatura da placa até atingir o regime permanente utilizando o método de diferenças finitas e depois ordenar de maior a menor as temperaturas dos elementos com condutividade térmica baixa.

Etapa 5: determinar, entre as regiões com material de baixa condutividade, a localização dos N_c elementos que apresentaram a maior temperatura.

Etapa 6: substituir os N_c elementos identificados na etapa anterior por elementos de alta condutividade.

Etapa 7: verificar se a quantidade de material de alta condutividade disponibilizado foi inserido na placa. Se a quantidade total foi inserida, então a troca de material deve se suspender, caso contrario, deve-se retornar à etapa 4..

A figura 5, apresenta a interface gráfica do modelo desenvolvido e na qual pode-se observar umas das simulações em andamento.

Segundo o pseudo-algoritmo proposto, deve-se fazer o refinamento da malha e cada "Patch" (unidade gráfica do NetLogo) dentro da interface representa um elemento da malha.

A etapa 2 é o momento para definir as condições de contorno do problema, isto é, a temperatura prescrita para as bordas e a temperatura inicial no interior da placa. Para isso, são usados os botões de controle "Initial-temp" e "Heat-flux" que podem ter valores entre 0 e 100 graus Celsius. Além disso, nesta etapa devem ser definidas as seguintes variáveis: a quantidade de elementos que será substituída a cada iteração (botão "N_c"); a porcentagem total de material de alta condutividade que será colocado na placa (botão "%-High-cond-material"); e a relação entre as condutividades dos materiais (botão k_p/k_0). Depois de escolher a relação entre os dois materiais, basta clicar no botão "Update Alpha" e o valor da condutividade dos materiais será reiniciado (outra opção é alterar o valor da relação entre as condutividades diretamente pelo valor desejado).

Durante a terceira etapa é atribuída uma condutividade baixa (k_0) para todos os elementos da malha e a partir da etapa 4 é calculado o campo de temperatura no domínio computacional usando o Método de Diferenças Finitas. Depois se chegar no regime permanente, deve-se substituir por material de alta condutividade, a quantidade "N_c" de elementos de baixa condutividade que apresente a maior temperatura. O procedimento desde a etapa 4 é repetido até que a quantidade total de material seja inserido.

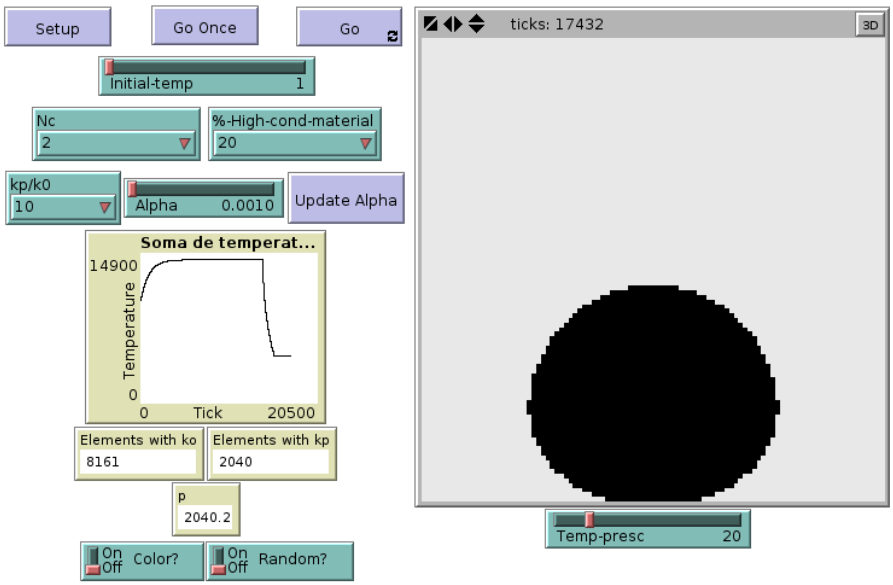


Fig.5. GUI NetLogo

4. Resultados

Foram realizadas simulações comparando os resultados obtidos quando o material é substituído em posições aleatórias e quando é inserido baseados nas máximas

temperaturas. A figura 6 permite ver o comportamento da soma da temperatura dos elementos da placa antes durante e após o processo de substituição para o problema com a seguinte configuração:

- Malha com 10201 elementos;
- temperatura inicial=1°C;
- Temperatura no ponto central do limite inferior=20°C;
- Temperatura prescrita nas outras fronteiras=0°C;
- Porcentagem de material de alta condutividade que será substituído dentro da placa=20 (2040 elementos);
- Relação k_p/k_0 =10.

As simulações foram feitas para diferentes valores de N_c, sendo possível concluir que para os casos analisados, a variação da quantidade de material inserido a cada iteração, influencia no tempo que o campo de temperatura tarda para chegar ao regime permanente, ou seja, quanto maior a quantidade de material a inserir a cada unidade do tempo, mais rapidamente chega-se ao regime permanente.

A soma das temperaturas mostrou ser menor sempre que a inserção do material de alta condutividade for baseada na Teoria Construtal. Na figura 6 pode ser visualizado que quando o campo de temperaturas atinge o regime permanente, a soma das temperaturas utilizando Teoria Construtal é próxima de 4674.03°C independente do N_c (que aparece entre parêntesis ao lado do tipo de simulação – aleatório ou construtal). Já para a substituição de material em posições aleatórias, a estabilidade é próxima de 11400°C. Assim, é possível afirmar que as simulações que utilizam Teoria Construtal, auxiliam na construção de geometrias melhores, e visam atingir o objetivo do trabalho.

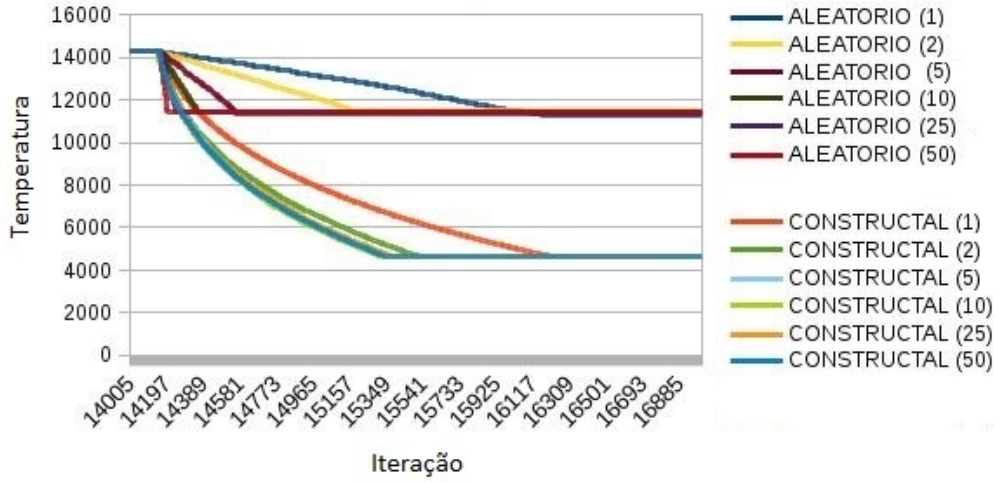


Fig. 6. Soma das temperaturas

A figura 7, (a) e (b) mostra uma das geometrias achadas e o campo de temperaturas respetivamente. Na parte (a), a região cinza representa o material de baixa condutividade (k_0) e a preta o material de alta condutividade (k_p). Na parte (b), o campo

de temperaturas é representado usando uma escala de cor vermelho, onde a zona mais clara representa a maior temperatura e a zona mais escura a menor temperatura.

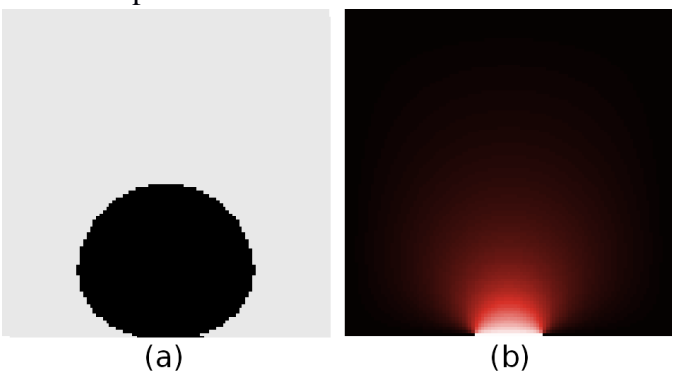


Fig.7. Geometria e campo de temperatura.

5. Conclusões e Trabalhos Futuros

Neste trabalho desenvolveu-se um algoritmo baseado em agentes que utiliza a Teoria Construtal para a busca das posições ótimas de um material de alta condutividade dentro de uma placa com geração de calor. O objetivo foi minimizar a soma das temperaturas dos elementos que formam a placa. O ambiente de simulação NetLogo foi utilizado para a modelagem do problema, que mostrou ser um ambiente de desenvolvimento poderoso para a simulação de fenômenos naturais.

Os resultados obtidos usando o algoritmo desenvolvido foram analisados e comparados com resultados obtidos dispondo o material de alta condutividade em posições aleatórias dentro da placa o que permite concluir que a teoria construtal garante resultados ótimos para o problema de optimização apresentado.

Propõe-se para futuros trabalhos mudar as condições de contorno da placa usando geração uniforme de calor e fluxo constante, basear a troca de material nos gradientes de temperatura e não na temperatura máxima, e mudar o objetivo da optimização do problema, por exemplo, pela busca da minimização das maiores temperaturas que se apresentam dentro da placa e comparar com resultados achados na literatura.

References

Bandini, Stefania, Manzoni, Sara and Vizzari, Giuseppe. (2009) “Agent Based Modeling and Simulation: An Informatics Perspective”, Journal of Artificial Societies and Social Simulation 12 (4) 4 <http://jasss.soc.surrey.ac.uk/12/4/4.html>.

Bejan, A., Zane, J. P. (2012) “Design in Nature: How the Constructal Law Governs Evolution in Biology, Physics, Technologic and Social Organization”, 1st Ed, New York, Doubleday.

Incropera, F. P., DeWitt, D. P., Bergman, T. L., Lavine, A. S. (2002) “Fundamentals of Heat and Mass Transfer”, 6th Ed, New York: J.Wiley.

Rocha, L. A., Lorente, S., Bejan, A. (2013) “Understanding Complex Systems”, 1st Ed, New York: Springer.

Souza, J.A., Ordonez, J. C. (2013) “Constructal Design of High-Conductivity inserts”, Livro: L. A.Rocha; S. Lorente; A. Bejan. (Org.). “Understanding Complex Systems”, 1ed, New York: Springer, v. 1, p. 91-1112013,

Tisue, S., Wilensky, U. NetLogo. (2004) “A Simple Enviroment for Modeling Complexity”, International Conference on Complex Systems, Boston, Maio 16-21.

Uma Abordagem Baseada em Agentes Para Um Sistema de Classificação de Timbres

Eduardo P. Teixeira[1], **Eder M. N. Gonçalves**[1], **Diana F. Adamatti**[1]

[1]Programa de Pós-Graduação em Computação (PPGComp)
Centro de Ciências Computacionais (C3)
Universidade Federal do Rio Grande (FURG)
Rio Grande – RS – Brasil

{eduardoteixeira, edergoncalves, diananaadamatti}@furg.br

Abstract. *This paper proposes a agent-based approach to timbre recognition, focusing on the parallelization of the classification model. For this, we assign a method of recognition of timbres to different agents, where each agent is a specialized entity in a particular timbre, characteristic of a specific instrument, seeking a distributed solution for solving the timbre recognition problem.*

Resumo. *Este trabalho propõe uma abordagem baseada em agentes para o reconhecimento de timbres, com enfoque na autonomia dos agentes ao modelo de classificação de timbres. Para isto, atribui-se um método de reconhecimento de timbres a diferentes agentes, onde cada agente é uma entidade especialista em um determinado timbre, característico de um instrumento específico, visando uma solução ao problema de reconhecimento de timbres de forma distribuída.*

1. Introdução

Existem quatro principais dimensões em sons: altura, intensidade, duração e timbre. A quarta dimensão, o timbre, é a mais vaga e complexa das dimensões [Eronen et al. 2001, Casey et al. 2008]. Até mesmo para a percepção humana, o reconhecimento de timbres é uma tarefa difícil, bem como a definição desta característica. O *American National Standards Institute* define timbre de uma maneira puramente excludente:

> "...atributo de sensação em que um ouvinte pode julgar que dois sons com a mesma intensidade e altura são diferentes" [Klingbeil 2009].

São muitos os trabalhos cujo tema é o reconhecimento e identificação de timbres, dentre os quais podem ser citados [Helmholtz and Ellis 2009, Strong 1963, Luce and Clark Jr 1967, Benade 2012, Nordqvist 2004, Klapuri 2004, Kitahara 2007]. Esta característica abstrata é de grande interesse no campo de MIR (*Music Information Retrieval*) [Casey et al. 2008].

A tecnologia de Sistemas Multiagentes é uma nova forma promissora para a performance musical interativa, em outros termos, execução colaborativa, onde agentes instrumentistas se unem na produção de uma performance musical, como visto em [Sampaio et al. 2005, Sampaio et al. 2008].

Em trabalhos recentes, essa tecnologia foi adaptada para resolver problemas específicos em um escopo musical limitado, como detecção de pulso, simulação de instrumentos ou acompanhamento automático. Sendo uma área bastante consolidada, adequada

para solucionar problemas que exijam distribuição, seja de natureza lógica ou geográfica, e em que a complexidade do problema seja minimizada por esta abordagem. Sendo assim, Sistemas Multiagentes são úteis em várias subáreas da Computação Musical.

Neste contexto, esse artigo apresenta uma abordagem baseada em agentes para a solução do problema de reconhecimento de timbres, visando um sistema escalável e paralelizável, onde cada agente atua como um especialista em um determinado instrumento, sendo o responsável por sua classificação.

2. Fundamentos Musicais

O conceito mais básico por trás de qualquer área de estudo musical é a definição de som. Ele é produzido quando um objeto (a fonte sonora) vibra e faz o ar ao seu redor se mover [Rumsey and McCormick 2012]. Este efeito pode ser representado como uma esfera que pulsa regularmente, com centro na fonte sonora, e seu tamanho oscila levemente entre maior e menor que o normal. Assim que pulsa, o som irá comprimir e rarefazer o ar ao seu redor, resultando em uma série de compressões e rarefações viajando para longe da esfera, similar a uma versão tridimensional de uma pedra que cai sobre um lago. Se a pressão varia de acordo com um padrão repetitivo, diz-se que o som tem uma forma de onda periódica. Se não houver nenhum padrão discernível, é chamado de ruído. Entre esses dois extremos existe um vasto domínio de sons quase periódicos e quase ruidosos [Roads 1996].

As ondas sonoras são ondas de compressão causadas por vibrações, mas a música de uma sinfonia varia consideravelmente do choro de um bebê ou do sussurro de um confidente.

Todas as ondas sonoras podem ser caracterizadas por sua *altura*, pela sua *intensidade*, pela sua *duração*, e pela sua qualidade sonora ou *timbre* [Lapp 2003]:

- A altura é uma característica do som que faz referência a nossa percepção de agudos e graves. Fisicamente, sons agudos possuem maiores frequências, e sons graves menores. O ser humano é capaz de ouvir em um intervalo entre 20Hz e 20.000Hz. Baleias e golfinhos escutam frequências ainda maiores [Lapp 2003].
- A intensidade do som está relacionada com a amplitude da onda de som. A maioria das pessoas tem algum reconhecimento da escala de decibéis (dB). Eles podem ser capazes de dizer que 0dB é o limiar de audição e de que o som na pista ao lado de um jato acelerando é de cerca de 140dB.
- A duração diz respeito ao tempo da onda sonora, o seu período, e o tempo que o som leva até cessar. É uma importante característica ao se estudar aspectos rítmicos dos sons.
- O timbre é a mais vaga e complexa das quatro dimensões dos sons [Eronen et al. 2001]. Considerável energia e esforços foram aplicados para promover o entendimento do timbre, uma das características mais abstratas da música.

2.1. Representações dos Sons

Além da frequência fundamental de um som, que representa o seu "tom", ou a sua altura, podem haver muitas frequências presentes em uma forma de onda. Uma representação

no domínio da frequência, ou espectrograma, mostra as frequências principais contidas em um som. Os componentes individuais de frequência do espectro podem ser referidos como harmônicas e parciais. Frequências harmônicas são múltiplos inteiros da frequência fundamental [Roads 1996], e podem ser facilmente identificadas em um analisador de espectro como visto na Figura 1.

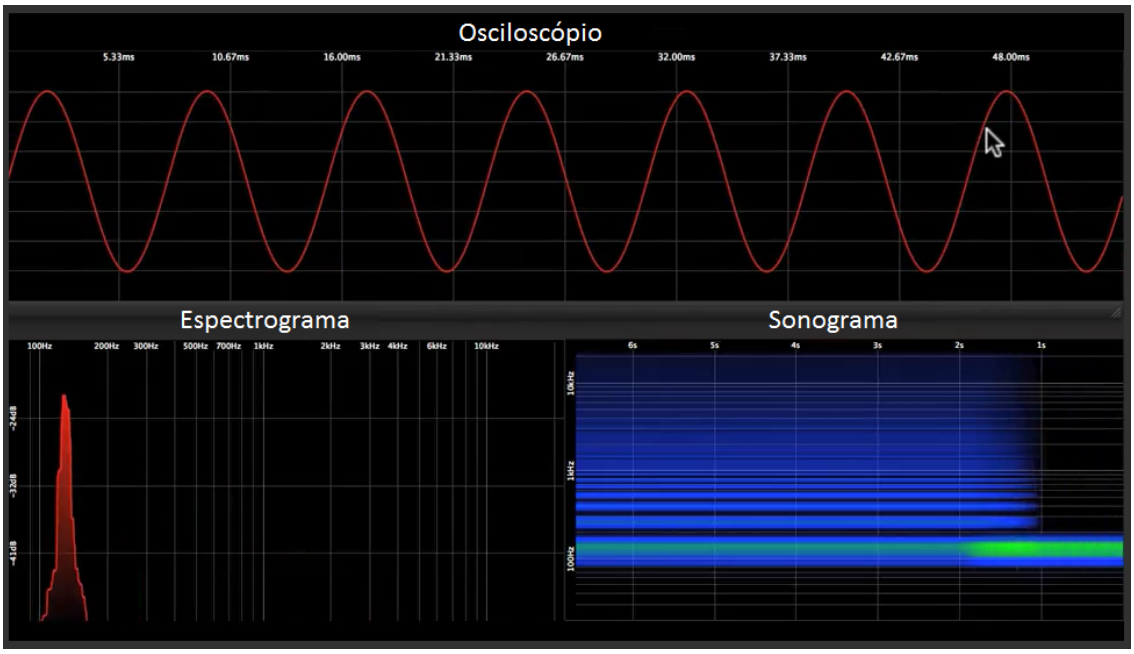


Figure 1. Diferentes representações de uma entrada sonora. Acima, um sinal criado por um osciloscópio. Abaixo, a esquerda, seu respectivo espectro no domínio da frequência. Abaixo, a direita, a visualização de um sonograma.

2.1.1. Osciloscópio

Um osciloscópio é usado para indicar a forma de onda de um som. Ele aceita sinais sonoros em forma elétrica e exibe suas análises em tela. O osciloscópio exibe um ponto em movimento que varre horizontalmente um número de velocidades fixas da esquerda para a direita e cuja deflexão vertical é controlada pela tensão do sinal de som (positivo para cima, negativo para baixo). Deste modo, representa graficamente a forma de onda do som, uma vez que varia com o tempo. Muitos osciloscópios tem duas entradas e podem traçar duas formas de onda ao mesmo tempo. Isto pode ser particularmente útil para a comparação das fases relativas de dois sinais [Rumsey and McCormick 2012].

2.1.2. Analisador de Espectro

O analisador de espectro funciona de diferentes formas, dependendo do método de análise de espectro. Um analisador em tempo real mostra um espectro de linha em constante atualização, e mostra os componentes do sinal de entrada na escala horizontal, juntamente com as suas amplitudes na escala vertical de frequência [Rumsey and McCormick 2012].

Em um analisador de espectro é possível observar as harmônicas (ou parciais), que são características da fonte sonora utilizadas no reconhecimento de timbre.

No domínio da frequência é possível recuperar informação musical com os mais diversos objetivos.

2.1.3. Sonograma

Em um sonograma é representado um espectrograma que varia com o tempo. Este tipo de representação é útil no reconhecimento de fala entre outras aplicações. Auxilia a visualizar as alterações nas frequências harmônicas em sons sendo executados ao longo do tempo. Ele representa três dimensões: a frequência (em Hertz, na vertical), o tempo (em segundos, na horizontal) e a intensidade (em decibéis, representado com diferentes colorações).

2.2. O Reconhecimento de Timbres

Como exemplo de trabalho estado da arte em MIR, que trabalha com o timbre como característica principal, pode-se citar [Devi et al. 2012], que trata o timbre como um conjunto de características, como a envoltória sonora (a forma como o som se inicia, se mantém e termina ao longo do tempo) para a amplitude e a frequência; tempo de ataque (início de cada nota musical); decaimento (em alguns instrumentos o som sofre um decaimento após o ataque até se estabilizar); sustentação (corresponde ao tempo de duração da nota musical) e intensidade [Devi et al. 2012]. Essas características podem ser observadas na Figura 2.

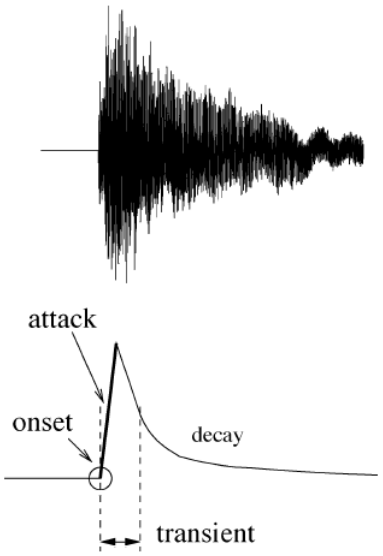


Figure 2. Formato do espectro de uma única nota sendo executada [Bello et al. 2005].

Historicamente, os primeiros estudos datam dos anos cinquenta, onde pode-se citar o trabalho de [Helmholtz and Ellis 1954], que evidenciou que as amplitudes relativas das parciais harmônicas de um som, muito mais que suas fases relativas, são determinantes primários do timbre.

Já [Strong 1963] interpretou o espectro de vários instrumentos de orquestra e demonstrou que o oboé, clarinete e fagote são identificados primeiramente na base no seu espectro de magnitude.

Com estes trabalhos, começam a surgir um maior número de estudos que buscam determinadas características no formato espectral dos sons para o reconhecimento de timbre, como o trabalho proposto em [Luce and Clark Jr 1967], que demonstrou que a família de metais (instrumentos de sopro como trombone, saxofone, trompete, entre outros), é caracterizada por um único *cutoff* na frequência, e que esta característica se correlaciona fortemente com a identificação desta família de instrumentos. Posteriormente, [Benade 2012] foi além, e mostrou que o corte na frequência é um dos principais determinantes do timbre nos instrumentos de sopro de um modo geral.

Outra característica que vem sendo fortemente correlacionada com o timbre é o centróide espectral, que em termos gerais pode ser definido como o "ponto de balanço" do espectro, se mostra diretamente ligado ao "brilho" do instrumento, uma dimensão primária e subjetiva do timbre, como verificado em [Grey 1977, Lichte 1941, von Bismarck 1974]. Nos trabalhos [Beauchamp 1982a, Beauchamp 1982b], o autor demostra que o centróide varia em muitos instrumentos com a intensidade do som.

No estudo realizado em [Strong 1963], foi demonstrado que, no reconhecimento de muitos instrumentos, a identificação ocorre principalmente pela envoltória temporal, em parte porque suas envoltórias espectrais não são únicas, lembrando que a definição de timbre é excludente, sendo este responsável por diferenciar sons que possuem mesma altura e intensidade [Klingbeil 2009].

Neste contexto, é possível observar que o reconhecimento do timbre está fortemente ligado à forma do espectro sonoro, que usualmente é dividido em *onset*, *attack*, *transient* e *decay*. Este formato é associado ao caso ideal de uma única nota sendo executada, como ilustrado na Figura 2 por [Bello et al. 2005].

Outra característica que passa a ser analisada em relação à influência de alteração do timbre são os aspectos temporais do som, cuja importância foi reconhecida primeiramente por [Risset and Wessel 1982]. Em sequência a esta descoberta, pode-se citar o trabalho de [Handel 1995], que apresenta a unificação da utilização das características temporais e espectrais no reconhecimento de timbre.

Já [Schoenberg 1978] considerou o timbre como uma segunda dimensão do tom, e hoje sabe-se que o timbre pode ser considerado de alguma forma em uma característica multidimensional, como descrito primeiramente em [Grey 1977], e inclusive pode ser representado visualmente, como foi apresentado no trabalho proposto por [Soraghan 2014].

Um dos primeiros trabalhos completos com enfoque de reconhecimento de timbre, com a utilização de várias das características apresentadas, foi o proposto por [Martin and Kim 1998], que considera que existem duas categorias de características para reconhecimento de timbre: temporal e espectral, ambas com grande importância no reconhecimento de timbre. Neste trabalho algumas características são utilizadas, como:

- Altura: sinais produzem uma estrutura identificável em relação à altura no correlograma, num frame bidimensional, com defasagem no eixo horizontal e a frequência em relação à vertical, sulcos verticais indicam o período do sinal, e

por inversão, a altura.

- Envoltória espectral: uma vez que a altura tenha sido detectada, a altura de crista vertical do correlograma pode ser mensurada como uma função da frequência, para se obter uma estimativa da forma da envoltória espectral. O centroide espectral é simplesmente o centroide da envoltória espectral.
- Intensidade: a soma da energia na envoltória espectral aproxima a intensidade sonora instantânea do sinal. Acompanhar esta ao longo do tempo leva a medidas simples de modulação de amplitude, o que pode revelar o *Tremolo* e, por correlação com modulações de frequência, ressonâncias. Como sugerido em [Beauchamp 1982a, Beauchamp 1982b], a relação entre a intensidade e o centroide espectral pode ser uma importante correlação perceptual do timbre.
- Ataque assíncrono: ao rastrear a envoltória espectral ao longo do tempo, com estimativas de altura concorrentes, é possível medir as características de ataque de um tom harmônico musical de uma forma psicofisicamente adequada.
- Enarmonia: os desvios harmônicos no sinal vão ser refletidos como desvios da estrutura vertical no correlograma instantâneo.

Ao todo, o trabalho de [Martin and Kim 1998] utilizou trinta e uma características diferentes, a maioria sendo variações das citadas acima. Dentre as estratégias de reconhecimento adotadas, utilizou-se o método k-NN, hierárquico e não-hierárquico para identificação de timbres.

Como descrito por [Eronen et al. 2001], *Formants* são protuberâncias criadas por uma ou mais ressonâncias na fonte sonora. Elas representam a informação essencial para o reconhecimento de voz e fala, e também para reconhecimento de instrumentos musicais. Um recurso robusto para medir a informação de *Formants*, ou a envoltória espectral suavizada, são os coeficientes cepstrais. Dentre as técnicas utilizadas no trabalho de [Eronen et al. 2001], está *Mel-frequency Cepstral Coefficients* (MFCC) [Davis and Mermelstein 1980].

O método de MFCC se tornou uma das técnicas mais populares na extração de características em sistemas automáticos de reconhecimento de fala [Eronen et al. 2001], pois oferece uma descrição da forma espectral do som bastante eficiente, posteriormente sendo amplamente utilizada no reconhecimento de timbres. Um *cepstro*[1] é o resultado de aplicar a transformada inversa de Fourier (IFFT) ao logaritmo do espectro estimado de um sinal. O MFCC se baseia no procedimento de cepstro. Nele, as bandas de frequência estão posicionadas de forma logarítmica utilizando a escala de Mel[2]. A transformada de Fourier é substituída por uma transformada de cosseno discreta (DCT)[3]. Ela tem uma propriedade de "compactação de energia" eficiente: a maior parte da informação do sinal tende a se concentrar em alguns componentes do DCT de baixa frequência. É por isso que, por padrão, apenas os 13 primeiros componentes são devolvidos. Por convenção, o coeficiente zero indica simplesmente a energia média do sinal [Lartillot et al. 2014].

[1]A denominação "cepstro" vem da palavra "spectro" com as primeiras letras em ordem inversa, fazendo referência ao uso da transformada inversa de Fourier.

[2]A escala de Mel, nomeada por [Stevens et al. 1937], é uma escala de percepção de alturas julgada por ouvintes a ter espaçamentos iguais entre os sons. Esta escala de frequência se aproxima mais a resposta do sistema auditivo humano do que as faixas de frequências linearmente espaçadas.

[3]Uma transformada de cosseno discreta (DCT) é uma transformada de Fourier semelhante à transformada de Fourier discreta (DFT), mas utilizando apenas números reais.

3. Abordagem Baseada em Agentes para a Classificação de Timbres

Como cada instrumento possui determinadas características de timbre que são identificadas por diferentes descritores, é possível imaginar que uma solução distribuída e paralela seja adequada para reconhecimento polifônico, aumentando a eficiência na solução de problemas.

Segundo [Thomaz 2009], a tecnologia de agentes se torna particularmente adequada para aplicações musicais, devido à possibilidade de associar um agente computacional com o papel de um cantor ou instrumentista. Ele destaca algumas vantagens dessas associações, como mapear características como desempenho, percepção, adaptação e improvisação de um lado, e processos artificiais no outro. Além disso, é possível definir formas de inter-relação social entre os agentes, que traz esta tecnologia ainda mais perto de performance musical colaborativa.

Para a resolução do problema de reconhecimento de timbres, valendo-se de uma abordagem baseada em agentes, este trabalho utiliza um conjunto de agentes especialistas em determinados timbres, onde cada um é responsável pela classificação de um instrumento, como exemplifica a Figura 3.

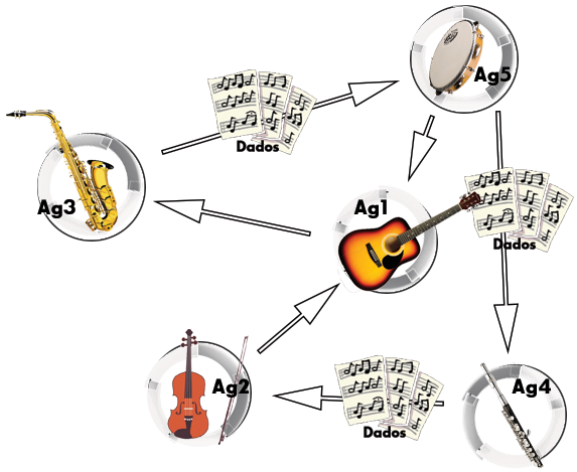


Figure 3. Representação simplificada do sistema proposto. São apresentados cinco agentes especialistas, cada um representado pelo instrumento que ele é encarregado de reconhecer. Dentro do ambiente, os agentes trocam informações entre eles, bem como dados musicais (conjunto de características extraídas) que precisam ser classificados.

Tratam-se de agentes cognitivos, que podem receber um conjunto de características do ambiente ou de outros agentes. Quando um novo sinal de áudio é previamente processado, e suas características são extraídas, esta nova entrada é enviada para o ambiente, onde um agente se encarrega de começar o processo de classificação. Caso ele reconheça as características como sendo ele o reconhecedor correto, com uma tolerância percentual empírica, este retorna ao ambiente notificando que o som foi devidamente classificado. Caso o agente especialista não atinja o percentual esperado, ele encaminha o conjunto de características para um novo agente. Este processo está representado na arquitetura apresentada na Figura 4.

Dentre as vantagens esperadas em adotar um sistema baseado em agentes para a resolução do problema de classificação de timbres pode-se citar:

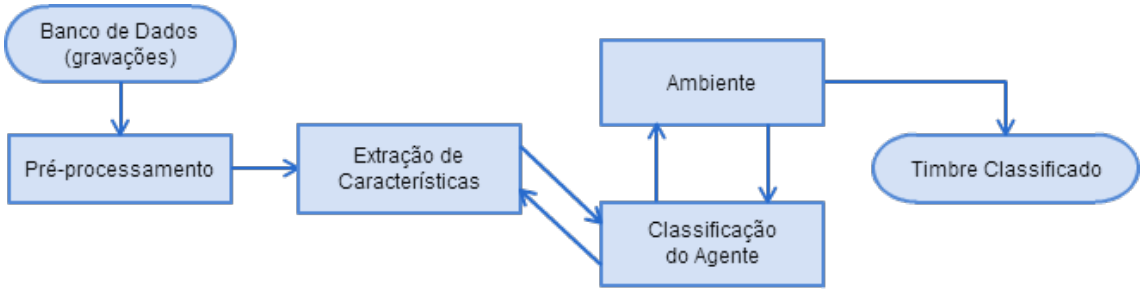


Figure 4. Arquitetura do sistema desenvolvido.

- Maior escalabilidade do sistema, pois é possível adicionar um novo agente, responsável por classificar um novo instrumento, sem realizar um novo treinamento em todo o sistema.
- Paralelização da classificação, pois quando existir uma tarefa de classificação de muitas entradas, a classificação delas poderá ser distribuída entre vários agentes.
- Aperfeiçoamento dos resultados, já que cada agente será específico para cada timbre é possível que eles sejam melhores especialistas do que uma única entidade classificadora, ou que, dependendo da implementação, troquem informação entre si para obter melhores resultados.

4. Desenvolvimento

O sistema baseado em agentes foi desenvolvido no NetLogo [Wilensky 1999], as operações de treino e classificação dos agentes foram desenvolvidas no MatLab [Guide 1998] com a utilização da MIRtoolbox [Lartillot and Toiviainen 2007] para a leitura de áudio e a extração de características musicais. A integração entre estas duas ferramentas foi realizada através de uma extensão não oficial do NetLogo, denominada MatNet[4].

O banco de dados utilizado consiste em uma coleção de arquivos com gravações de notas únicas de diversos instrumentos. Os arquivos de áudio utilizados foram obtidos através da OLPC (*One laptop per child - free sound samples*[5]).

A Figura 5 apresenta a interface gráfica do sistema desenvolvido, onde são representados os agentes e a interação entre eles.

Para o treinamento do agente, através do seu ID, são selecionados arquivos de áudio do banco de dados e aplicado o método de MFCC (*Mel-frequency cepstral coefficients*) apresentado em [Davis and Mermelstein 1980], que consiste em um conjunto de 13 coeficientes para cada arquivo. Desta forma é criada uma matriz de coeficientes para cada agente especialista de tamanho Nx13, onde N representa o número de arquivos utilizados no treinamento. O método de classificação utilizado consiste em aplicar o MFCC para a entrada e comparar com a média da matriz Nx13 do agente, onde as diferenças entre cada um dos 13 coeficientes são somadas e normalizadas. O funcionamento dos agentes, após serem treinados, consiste em aguardar uma atribuição do ambiente ou de outro agente. Quando ele é atribuído de uma classificação, ele realiza o método de

[4]http://github.com/mbi2gs/netlogo-matlab-extension/wiki
[5]http://wiki.laptop.org/go/Free_sound_samples

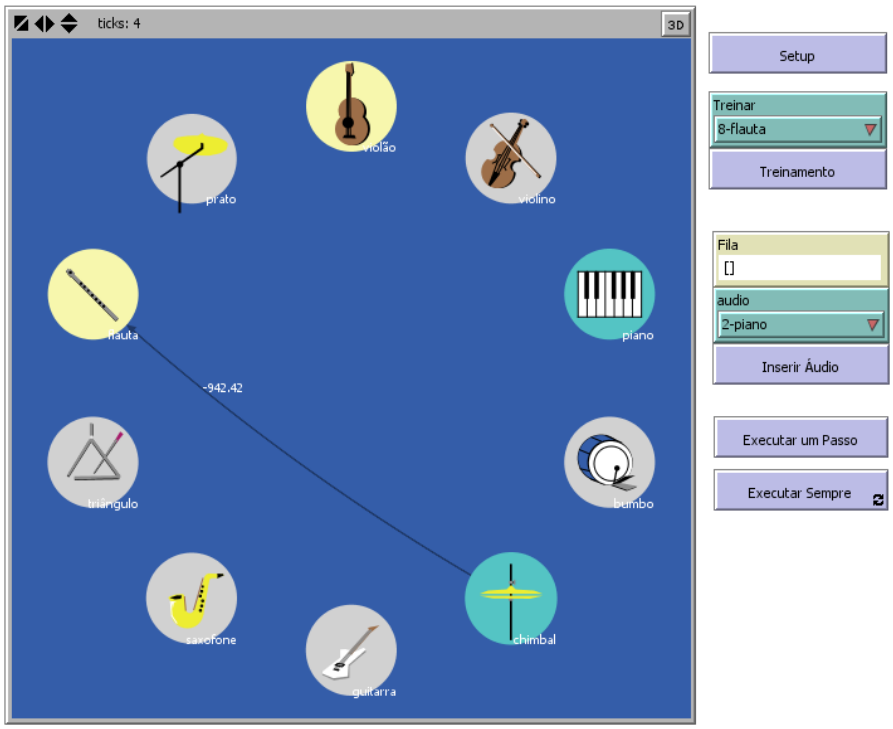


Figure 5. Dentro da visualização de mundo estão representados 10 agentes especialistas, onde o agente chimbal após não conseguir classificar sua entrada a encaminha para o agente flauta.

classificação da entrada: caso a resposta for maior que um limiar empírico, ele reporta ao ambiente que ele é o correto classificador; caso contrário encaminha a entrada para outro agente aleatoriamente.

5. Conclusão

Neste trabalho foi apresentado um sistema baseado em agentes para a solução do problema de classificação de timbres musicais, em que para cada instrumento seria associado um agente especialista capaz de reconhecer apenas um único instrumento. Em um ambiente multiagente, onde vários agentes cognitivos podem realizar trocas de informação, a classificação de timbres seria realizada de forma distribuída, paralela e escalável.

O método de classificação implementado se provou bastante rápido, conforme a sua simplicidade, mas foram observados erros de classificação em instrumentos da mesma família, como o violão e a guitarra, possivelmente pela grande semelhança dos coeficientes de MFCC destes arquivos. Para a solução deste problema, uma provável solução seria utilizar um conjunto de características maior, que explorassem outros comportamentos no domínio do tempo e da frequência. Outra limitação apresentada pelo sistema diz respeito ao método de encaminhamento dos agentes. Em alguns testes de execução, a mesma entrada chegou a passar quatro vezes pelo mesmo agente antes de alcançar o agente correto. Para solucionar este problema, poderia ser implementado um vetor associado a cada entrada, do tamanho do número de agentes no ambiente, servindo como um histórico por onde a entrada circulou, que deverá ser verificado antes de realizar uma atribuição.

References

Beauchamp, J. W. (1982a). Data reduction and resynthesis of connected solo passages using frequency, amplitude, and "brightness" detection and the nonlinear synthesis technique. *The Journal of the Acoustical Society of America*, 71(S1):S101–S101.

Beauchamp, J. W. (1982b). Synthesis by spectral amplitude and" brightness" matching of analyzed musical instrument tones. *Journal of the Audio Engineering Society*, 30(6):396–406.

Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., and Sandler, M. B. (2005). A tutorial on onset detection in music signals. *Speech and Audio Processing, IEEE Transactions on*, 13(5):1035–1047.

Benade, A. H. (2012). *Fundamentals of Musical Acoustics: Second*. Courier Corporation.

Casey, M., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., Slaney, M., et al. (2008). Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696.

Davis, S. B. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366.

Devi, J. S., Srinivas, Y., and Krishna, N. M. (2012). A study: Analysis of music features for musical instrument recognition and music similarity search. *IJCSI*.

Eronen, A. et al. (2001). Automatic musical instrument recognition. *Mémoire de DEA, Tempere University of Technology*, page 178.

Grey, J. M. (1977). Multidimensional perceptual scaling of musical timbres. *The Journal of the Acoustical Society of America*, 61(5):1270–1277.

Guide, M. U. (1998). The mathworks. *Inc., Natick, MA*, 5:333.

Handel, S. (1995). Timbre perception and auditory object identification. *Hearing*, pages 425–461.

Helmholtz, H. L. and Ellis, A. J. (1954). *On the sensations of tone as a physiological basis for the theory of music (AJ Ellis, Trans.)*. New York: Dover.(Original work published in 1885).

Helmholtz, H. L. and Ellis, A. J. (2009). *On the Sensations of Tone as a Physiological Basis for the Theory of Music*. Cambridge University Press.

Kitahara, T. (2007). Computational musical instrument recognition and its application to content-based music information retrieval. *Unpublished PhD Thesis, Kyoto University, Kyoto, Japan. Retrieved*, 10(31):07.

Klapuri, A. (2004). *Signal processing methods for the automatic transcription of music*. Tampere University of Technology Finland.

Klingbeil, M. K. (2009). *Spectral Analysis, Editing, and Resynthesis: Methods and Applications*. PhD thesis, Columbia University.

Lapp, D. R. (2003). *The physics of music and musical instruments*. Wright Center for Innovative Science Education, Tufts University.

Lartillot, O. and Toiviainen, P. (2007). A matlab toolbox for musical feature extraction from audio. In *International Conference on Digital Audio Effects*, pages 237–244.

Lartillot, O., Toiviainen, P., and Eerola, T. (2014). *MIRtoolbox 1.6 User's Manual*.

Lichte, W. H. (1941). Attributes of complex tones. *Journal of Experimental Psychology*, 28(6):455.

Luce, D. and Clark Jr, M. (1967). Physical correlates of brass-instrument tones. *The Journal of the Acoustical Society of America*, 42(6):1232–1243.

Martin, K. D. and Kim, Y. E. (1998). 2pmu9. musical instrument identification: A pattern-recognition approach. In *Presented at the 136th meeting of the Acoustical Society of America*. Citeseer.

Nordqvist, P. (2004). *Sound classification in hearing instruments*. PhD thesis, KTH-S3.

Risset, J.-C. and Wessel, D. L. (1982). Exploration of timbre by analysis and synthesis. *The psychology of music*, 28.

Roads, C. (1996). *The computer music tutorial*. MIT press.

Rumsey, F. and McCormick, T. (2012). *Sound and recording: an introduction*. CRC Press.

Sampaio, P., Tedesco, P., and Ramalho, G. (2005). Cinbalada: um laboratório multiagente de geração de ritmos de percussao. In *Proceedings of the X Brazilian Symposium on Computer Music*.

Sampaio, P. A., Ramalho, G., and Tedesco, P. A. (2008). Cinbalada: Multiagent rhythm factory. *J. Braz. Comp. Soc*, 14(3):31–49.

Schoenberg, A. (1978). *Theory of harmony*. Univ of California Press.

Soraghan, S. (2014). Animating timbre-a user study. *The 2014 IEEE International Conference on Systems, Man, and Cybernetics*.

Stevens, S. S., Volkmann, J., and Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190.

Strong, W. J. (1963). *Synthesis and recognition characteristics of wind instrument tones*. Massachusetts Institute of Technology.

Thomaz, L. F. (2009). A framework for implementing musical multiagent systems. *6th Sound and Music Computing Conference*, pages 119–124.

von Bismarck, G. (1974). Timbre of steady sounds: A factorial investigation of its verbal attributes. *Acta Acustica united with Acustica*, 30(3):146–159.

Wilensky, U. (1999). Netlogo.

Processo de Desenvolvimento de uma Ferramenta Gráfica de Apoio a Metodologia Prometheus AEOlus

Rafhael R. Cunha [1], Diana F. Adamatti[1], Cléo Z. Billa[1]

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

{rcrafhaelrc,dianaada,cleo.billa}@gmail.com

Resumo. *A Engenharia de Software (ES) é uma área de engenharia que busca a construção de softwares com qualidade, utilizando métodos e respeitando prazos. Contudo, suas técnicas tradicionais não suportam completamente a demanda no desenvolvimento de Sistemas Multiagente, originando uma subárea, denominada Agent Oriented Software Engineering (AOSE). Ainda não existe uma padronização para esta subárea, resultando em diversas metodologias desenvolvidas por motivos distintos. Outro fator predominante para a instabilidade dessa subárea, consiste em suas ferramentas de apoio serem limitadas no processo de geração automática de código para plataformas específicas de desenvolvimento multiagente. O intuito principal deste trabalho é desenvolver uma ferramenta para apoiar a metodologia Prometheus AEOlus, permitindo que o usuário desenvolva os diagramas presentes na especificação da metodologia. Adicionalmente, como objetivo secundário, foi elaborado um mecanismo capaz de percorrer todas as informações definidas pelo usuário e realizar a geração automática de código para a linguagem agentspeak, que é aderente a plataforma de desenvolvimento Jason. A ferramenta proposta apresentou resultados satisfatórios, o que a valida como uma nova alternativa para o desenvolvimento de sistemas multiagente.*

1. Introdução

Na área de inteligência artificial, o paradigma orientado a agentes tem sido pesquisado e utilizado para minimizar a complexidade e aumentar a eficiência de softwares distribuídos. Esta prática tem-se mostrado eficiente para a construção de softwares com essas características, viabilizando um aumento no desenvolvimento de Sistemas Multiagente (SMA).

Uma das formas de se programar sistemas multiagente é utilizando a arquitetura *Beliefs, Desires e Intentions* (BDI). Essa arquitetura é uma forma de modelar o agente pensando em seu estado mental. A fundamentação filosófica para esta concepção de agentes vem dos trabalhos de [Dennett 1989] sobre sistemas intencionais e [Bratman 1987] sobre raciocínio prático. A metodologia Prometheus AEOlus surgiu como forma para modelar agentes BDI, suportando as dimensões de organização e ambiente, que também compõem o desenvolvimento de um SMA. Como propósito final, a metodologia Prometheus AEOlus promete ser aderente ao *framework* JaCaMo. Entretanto, até o presente momento, esta metodologia não possuía nenhuma ferramenta para apoiar o seu utilizador.

Desta forma, o objetivo deste trabalho é desenvolver uma ferramenta gráfica que suporte a metodologia Prometheus AEOlus, facilitando sua utilização. Além disso,

desenvolveu-se um mecanismo de geração de código automático para a linguagem *agentspeak*, parte da plataforma Jason, suprindo uma das dimensões presentes no *framework* JaCaMo.

2. Fundamentação Teórica-Tecnológica

Esta seção apresenta conceitos teóricos e tecnológicos para a compreensão dos tópicos abordados ao longo deste trabalho. Na subseção 2.1 é descrita a origem da engenharia de software orientada a agentes. A subseção 2.2 descreve em linhas gerais a metodologia Prometheus AEOlus, a qual é a base deste trabalho. Na subseção 2.3 é retratado o processo de desenvolvimento orientado a modelos. Por último, na subseção 2.4 é explicado o processo de desenvolvimento de *plug-ins* para a IDE Eclipse.

2.1. Engenharia de Software Orientada a Agentes

A (*Agent Oriented Software Engineering*) (AOSE) foi desenvolvida para suprir as necessidades encontradas no desenvolvimento de sistemas complexos. Esta subárea mescla as áreas de Inteligência Artificial e Engenharia de Software para oferecer suporte ao desenvolvimento de sistemas orientados a agentes [Guedes 2012].

Para [Gleizes and Gomez-Sanz 2009], a AOSE está ganhando relevância por duas principais razões: primeiro, as estruturas conceituais atingiram um nível de maturidade que torna razoável dedicar esforços para encontrar um consenso entre linguagens de modelagem já propostas e também suporte a ferramentas; segundo, a influência da engenharia orientada a modelos enfatiza o valor potencial de ter modelos no centro do processo de desenvolvimento.

2.2. Prometheus AEOlus

Segundo [Uez 2013], a metodologia Prometheus AEOlus foi desenvolvida baseando-se em duas tecnologias: a metodologia Prometheus e o *framework* JaCaMo. A autora complementa afirmando que o propósito da metodologia é fazer uma extensão da metodologia Prometheus de modo a incluir a especificação de Ambiente e Organização.

A metodologia possui notações gráficas definidas para cada conceito, as quais são utilizadas para representar agentes, ações, cenários, mensagens, percepções, crenças, entre outros. Esses conceitos são utilizados nos doze diagramas que a metodologia oferece para suporte ao desenvolvimento de SMA [Uez 2013].

O processo de desenvolvimento definido na metodologia é dividido em quatro fases: especificação do sistema, projeto arquitetural, projeto detalhado e implementação [Uez 2013]. A primeira fase tem como objetivo principal especificar os cenários e os objetivos do sistema. Quanto ao projeto arquitetural, nessa fase são definidos os elementos que formam o sistema. Soma-se a essas fases, o projeto detalhado, o qual tem por objetivo definir a estrutura interna dos agentes, através de suas crenças, planos e capacidades [Uez 2013]. A última fase é a implementação, na qual o objetivo é gerar código para o *framework* JaCaMo.

2.3. Model-Driven Engineering

Segundo [Rube 2013], a *Model-Driven-Engineering* (MDE) ou Engenharia Dirigida a Modelos é um novo enfoque na área de engenharia de software. MDE utiliza modelos

como artefatos de software com o objetivo de facilitar o trabalho e reduzir o tempo de desenvolvimento e o número de erros no software gerado. Modelos são um conjunto de elementos que descrevem um sistema [Mellor 2004] com grau de abstração maior que o próprio sistema [Kleppe et al. 2003]. Para [Miller et al. 2001], um modelo pode ser definido como uma especificação formal de uma função, estrutura e/ou comportamento de um sistema.

Segundo [Rube 2013], a MDE tem algumas aplicações, sendo: *Model-Driven Development* (MDD), Engenharia Reversa, *Software Process Engineering* (SPE), *Domain Specific Language* (DSL) e *Model-Driven Integration* (MDI). Para este trabalho, foi utilizada a DSL, que conforme [Van Deursen and Klint 2002], é uma linguagem específica de domínio que fornece uma notação adaptada para um domínio de aplicação e baseia-se seus conceitos e características em um domínio relevante.

2.4. Plataforma Eclipse para Desenvolvimento de *Plug-ins*

A plataforma Eclipse é baseada em *plug-ins* que são utilizados para ampliar as funcionalidades da IDE [Foundation 2014a]. Esses *plug-ins* são codificados na linguagem de programação Java e podem oferecer diversas modalidades de serviço como biblioteca de códigos, guias de documentação ou uma extensão da própria plataforma [Rivieres and Wiegand 2004].

As bibliotecas de código auxiliam os programadores com funcionalidades já implementadas e podem ser oferecidas pelo *plug-in* em formato de *Application Programming Interface* (API). A documentação auxilia os desenvolvedores nos aspectos técnicos da linguagem de programação, bibliotecas de código, demonstrando os recursos disponíveis e sua aplicações. Na extensão da plataforma, os desenvolvedores utilizam os componentes e recursos do Eclipse como interface gráfica, mecanismos de interpretação textual e de compilação para desenvolverem soluções tecnológicas para outras finalidades como por exemplo uma ferramenta de apoio a uma nova linguagem de programação.

2.5. Processo de Desenvolvimento de um Editor Gráfico utilizando o *plug-in Graphical Modelling Framework*

Segundo [Foundation 2014b], o *Graphical Modelling Framework* (GMF) é um arcabouço para desenvolvimento de editores gráficos para modelos de domínio dentro da plataforma Eclipse. Ele foi baseado em outros dois arcabouços denominados *Graphical Editing Framework* (GEF), utilizado para a criação de editores gráficos genéricos e, *Eclipse Modelling Framework* (EMF), que permite ao desenvolvedor construir metamodelos e gerar código Java referidos ao mesmo.

O processo de desenvolvimento do *plug-in* deste trabalho é mostrado na Figura 1. Esta figura demonstra o fluxo de trabalho necessário para gerar um editor gráfico utilizando o *plug-in* GMF do eclipse.

O processo de desenvolvimento consiste na concepção de seis arquivos. O primeiro chama-se Ecore e representa o *Domain Model*. É neste arquivo que é especificado o metamodelo do editor. O metamodelo serve para regulamentar todas as ações posteriores que o editor gráfico possa ter, por exemplo, quais entidades terão relacionamento e quais não terão.

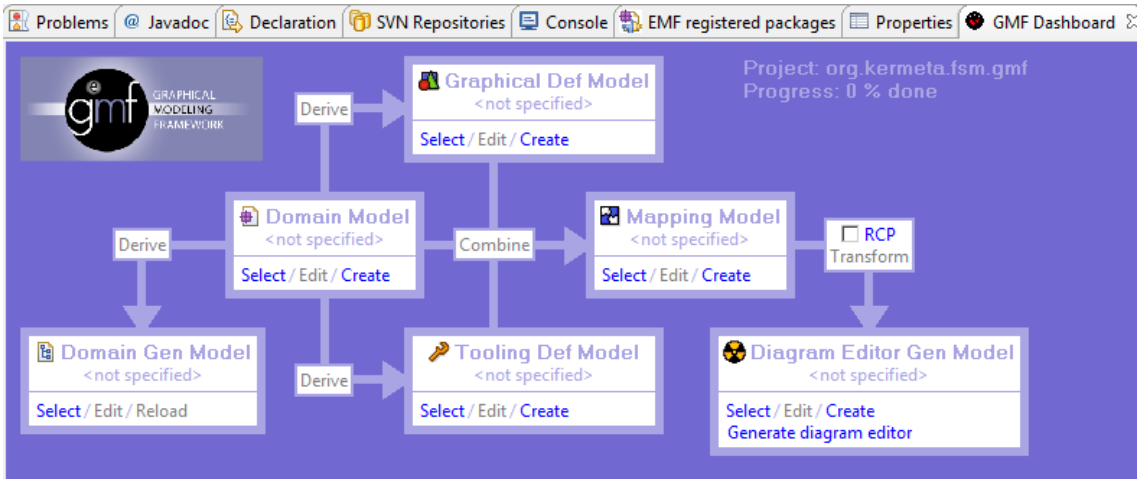


Figura 1. Dashboard do GMF Eclipse.

O próximo passo na elaboração do editor gráfico é derivar o *Domain Gen Model* do editor. Por esse motivo que existe o botão *Derive* ao lado esquerdo da caixa *Domain Model* no *dashboard*. O processo de derivação consiste em realizar uma transformação do metamodelo gerado no Ecore para um metamodelo específico com extensão **.genmodel**, para que a partir deste, sejam gerados códigos na linguagem de programação Java. Além disso, o arquivo com extensão **.genmodel**, é a base para a geração do restante de todos os artefatos do projeto de geração de um editor gráfico utilizando o GMF eclipse. Os arquivos **.ecore** e **.genmodel** correspondem ao *plug-in* EMF Eclipse, que correspondem a primeira parte da junção que resultou na concepção do GMF Eclipse.

Sobre o campo de desenho utilizado para modelar os diagramas, este é derivado do *Domain Model* e corresponde a caixa chamada *Graphical Del Model*. O arquivo gerado nesta transformação tem a extensão **.gmfgraph**. Além disso, através da plataforma eclipse pode-se editar este arquivo, adicionando pontos provenientes de figuras geométricas, ou outras customizações sobre cada entidade derivada do arquivo **.ecore**. É neste arquivo que se configura toda a caixa de desenho que será utilizada para modelar o diagrama que está sendo desenvolvimento através do editor gráfico.

Em relação a paleta de componentes que irá do lado do campo de desenho do editor gráfico desenvolvido, esta é concebida também por meio do *Domain Model*. Após a solicitação de derivação do *Domain Model* em relação a caixa chamada *Tooling Del Model* é gerado um arquivo com a extensão **.gmftool** [Gronback 2009]. Este arquivo contém a respectiva configuração de como irá ser apresentado os componentes gráficos para serem utilizados no campo de desenho do editor gráfico [Gronback 2009].

O próximo passo da elaboração da ferramenta é fazer a combinação entre todos os arquivos gerados anteriormente. Por intermédio do botão *Combine* é feito essa combinação, gerando um arquivo na caixa chamada *Mapping Model*, ilustrada na Figura 1. O arquivo gerado nessa caixa tem a extensão **.gmfmap**. Segundo [Gronback 2009], talvez o mais importante de todos os modelos em GMF é o *Mapping Model*. Nele, elementos de definição do diagrama (nós e *links*) são mapeados para o modelo de domínio e elementos de ferramentas atribuídas. O *Mapping Model* representa o diagrama de definição real e é usado para criar um modelo de gerador.

O último passo no desenvolvimento do editor gráfico é fazer a geração do arquivo que ficará responsável em fazer a junção de todos os outros componentes abordados no decorrer do desenvolvimento. O nome da caixa equivalente a esse processo é *Diagram Editor Gen Model*, ilustrada na Figura 1. A partir do arquivo **.gmfmap** e arquivo **.ecore** é gerado o arquivo **.gmfgen**. Este arquivo é responsável por armazenar as configurações de onde será gerado os códigos fontes para a execução do editor gráfico modelado no decorrer de todo esse processo. Posterior a configuração desse arquivo, inicia-se o processo de transformação, gerando o código fonte correspondente as etapas anteriores realizadas. No momento da finalização dessa transformação, tem-se os códigos fontes que poderão ser executados para a utilização do editor gráfico desenvolvido, finalizando assim o processo.

3. Ferramenta de Suporte a Metodologia Prometheus AEOlus

Esta seção apresenta a arquitetura geral do *plug-in* desenvolvido, bem como os passos realizados até a sua finalização. A subseção 3.1 descreve a arquitetura geral do *plug-in*. Na subseção 3.2 são descritos os procedimentos realizados para desenvolvimento do diagrama de Visão Geral do Agente. Por fim, na subseção 3.3, são transmitidos os passos incrementados para a geração automática de código.

3.1. Arquitetura Geral da Ferramenta

A arquitetura geral da ferramenta Prometheus AEOlus é ilustrada na Figura 2. Cada pacote representa um *plug-in* gerado para incorporar funções à ferramenta desenvolvida. O pacote chamado *gerador*, representa o *plug-in* responsável pela geração de código da ferramenta. Dentro desse *plug-in*, são agrupadas as classes responsáveis por realizarem a função designada ao *plug-in*. Esse pacote contém uma ligação de dependência com o pacote *diagram* porque ele necessita de informações fornecidas pelo mesmo.

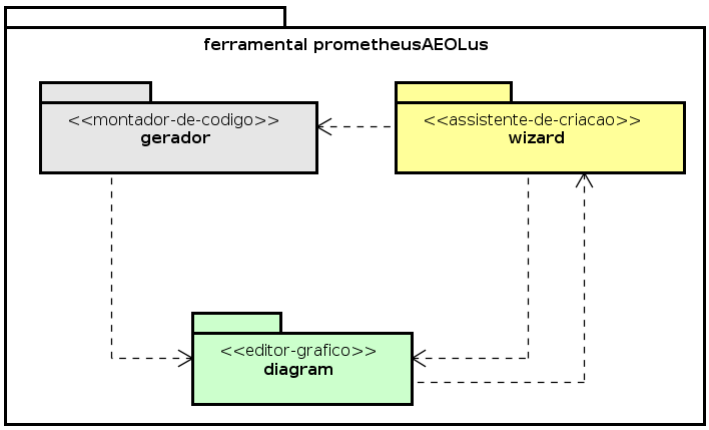


Figura 2. Arquitetura Geral do *plug-in* Prometheus AEOlus.

O pacote *diagram* comporta o *plug-in* editor gráfico. Ele contém todos os arquivos de configuração e os códigos gerados por todo o desenvolvimento proveniente do *plug-in* GMF Eclipse. Diferente dos demais *plug-ins* que compõem esta ferramenta, o editor gráfico não possui um inicializador, resultando na dependência ao *plug-in wizard* para ser inicializado.

O pacote *wizard* contém o *plug-in* responsável por fazer a junção entre os outros dois *plug-ins* que constituem a ferramenta desenvolvida. Ele reúne as funcionalidades

dos outros *plug-ins* descritos, por esse motivo possui dependência aos demais. A peculiaridade desse *plug-in* em relação aos demais consiste na inexistência de códigos-fontes, apenas pontos de extensão adicionados no arquivo *eXtensible Markup Language* (XML) de configuração do *plug-in*, onde são informados com quais *plug-ins* ele tem relação de dependência. Detalhes da implementação de um dos diagramas presentes na metodologia são descritos a seguir.

3.2. Desenvolvimento do Diagrama Visão Geral do Agente

Segundo [Uez 2014], o diagrama de visão geral do agente descreve o agente internamente, ou seja, detalhando seus planos, suas habilidades e crenças, bem como as capacidades que o agente pode ter. Os planos tem como finalidade indicar quais ações o agente deve executar para atingir um objetivo. Cada plano deve ter um evento *trigger* que iniciará a execução do plano. Eventos *triggers* podem ser desde mensagens recebidas ou até mesmo uma percepção do ambiente. Um evento é ligado ao plano através de uma seta pontilhada. Conforme [Uez 2014], os planos podem conter envio de mensagens, ações, outras percepções recebidas ou crenças. Quaisquer desses elementos são ligados ao plano através de setas simples e contínuas e são ligados entre si por setas pontilhadas que indicam a ordem em que essas ações devem ocorrer.

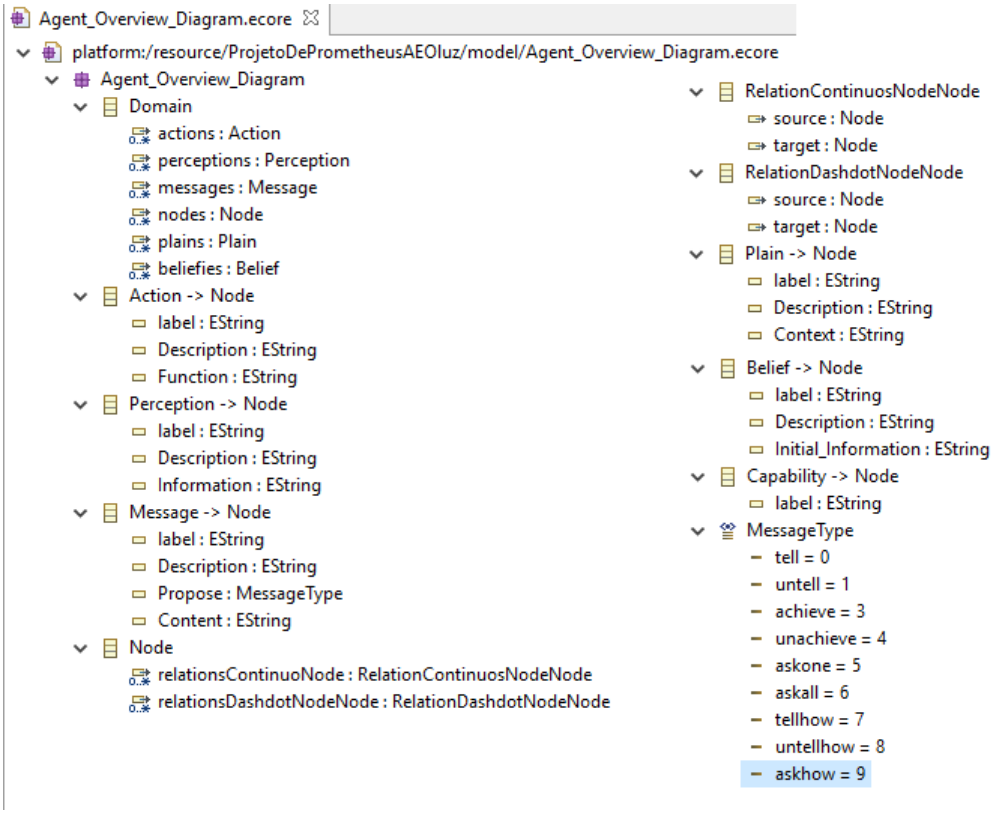


Figura 3. Metamodelo do Diagrama de Visão Geral do Agente na perspectiva do Ecore.

O metamodelo desenvolvido correspondente ao diagrama de visão geral do agente é ilustrado na Figura 3. A entidade *Node* tem como objetivo ser uma entidade genérica que isola atributos e relacionamentos comuns às demais entidades do metamodelo. Sua

diferença em relação às demais consiste no número de relacionamentos. A entidade *Node* neste metamodelo possui dois relacionamentos. O primeiro intitulado *relationsContinuoNode*, corresponde as ligações que utilizam uma seta simples, contínua. Já o outro relacionamento denominado *relationsDashdotNodeNode*, representa os relacionamentos que utilizam uma seta pontilhada entre duas entidades que herdam da entidade *Node*.

A entidade *RelationContinuosNodeNode* e *RelationDashdotNodeNode* possuem ambas os atributos *source* e *target* que representam a origem e o destino das relações. Ambos os atributos são do tipo *Node* justamente para serem reaproveitados nas entidades que herdam os relacionamentos da entidade *Node*.

A entidade *Action* representa uma ação e herda o relacionamento da entidade *Node*. Esta entidade contém os atributos *label*, *description* e *function*, ambos do tipo *EString*. A entidade *Perception* também herda o relacionamento da entidade *Node*. Ela possui os atributos *label*, *description* e *information*, ambos também do tipo *EString*. Já a entidade *Message* representa uma mensagem no metamodelo. Essa entidade também herda o relacionamento da entidade *Node*. Além disso, ela possui os atributos *label*, *description*, *propose* e *content*. Com exceção do atributo *propose*, ambos os outros são do tipo *EString*, portanto, armazenam textos. O atributo *propose* é do tipo *MessageType*, que é uma entidade do tipo *Enumeration*, ao qual representa os tipos de mensagens que podem ser enviados pelos agentes.

A entidade *Plain* representa os planos que os agentes terão. Essa entidade possui os atributos *label*, *description* e *context*, ambos do tipo *EString*. Além disso essa entidade herda os relacionamentos presentes na entidade *Node*. A entidade *Belief* representa as crenças no campo de desenho. Essa entidade possui os atributos *label*, *description* e *inicial_information*, ambos do tipo *EString*.

A entidade *Capability* não aparece na descrição deste diagrama, conforme [Uez 2013]. Entretanto, no momento de desenvolvimento desse metamodelo percebeu-se que sua única distinção em relação ao metamodelo do Diagrama de Capacidade era a entidade *Capability*, possibilitando assim que houvesse uma junção entre os metamodelos. Em seguida, modelou-se a entidade *Capability* no metamodelo deste diagrama. A entidade *Capability* possui somente o atributo *label* cujo o tipo é *EString*.

3.3. Desenvolvimento do *plug-in* gerador de código

O GMF Eclipse utiliza um arquivo derivado do XML, intitulado *XML Metadata Interchange* (XMI), para fazer sua persistência de dados, conforme Figura 4. Este arquivo contém diversas *tags* que tornam o processo de extração de informações mais acessível. No processo de geração de código para a plataforma Jason foi utilizado os arquivos XMI gerados pelo GMF. Estes arquivos continham informações referentes à modelagem de diagramas do gênero Modelo Geral da metodologia Prometheus AEOlus, ou seja, o diagrama que reúne todas as entidades e relacionamentos presentes na metodologia. Por questões de implementação, optou-se por desenvolver um diagrama geral que possibilitasse a modelagem de todas as entidades da metodologia, de modo que o processo de geração de código fosse mais acessível para este primeiro protótipo.

O *plug-in* gerador de código é dividido em 4 camadas, sendo: *modelo*, *gerador*, *visao* e *servico*. A primeira camada é responsável por representar através de objetos as entidades utilizadas pelos diagramas da metodologia Prometheus AEOlus. Além disso,

```
<?xml version="1.0" encoding="UTF-8"?>
<PrometheusAEOLus_generalModel:Domain xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:PrometheusAEOLus_generalModel="PrometheusAEOLus_generalModel">
  <nodes xsi:type="PrometheusAEOLus_generalModel:Goal"/>
  <nodes xsi:type="PrometheusAEOLus_generalModel:Role"/>
  <nodes xsi:type="PrometheusAEOLus_generalModel:Action"/>
  <nodes xsi:type="PrometheusAEOLus_generalModel:Artifact"/>
  <nodes xsi:type="PrometheusAEOLus_generalModel:Perception"/>
  <nodes xsi:type="PrometheusAEOLus_generalModel:Protocol"/>
  <nodes xsi:type="PrometheusAEOLus_generalModel:Message"/>
  <nodes xsi:type="PrometheusAEOLus_generalModel:Belief"/>
  <nodes xsi:type="PrometheusAEOLus_generalModel:Plain"/>
  <nodes xsi:type="PrometheusAEOLus_generalModel:Scenario"/>
  <nodes xsi:type="PrometheusAEOLus_generalModel:Capability"/>
  <nodes xsi:type="PrometheusAEOLus_generalModel:Agent"/>
  <missions/>
  <groups/>
  <workspaces/>
  <edgesSimple source="//@nodes.11" target="//@nodes.1"/>
</PrometheusAEOLus_generalModel:Domain>
```

Figura 4. Exemplo do Diagrama de Modelo Geral na perspectiva do *plug-in* desenvolvido.

essa camada também possui classes que ajudam o *plug-in* a encontrar informações presentes no XMI. A camada *gerador* engloba as classes que centralizam as tarefas do *plug-in* com o ambiente Eclipse e as classes responsáveis por escrever a *string* que conterá os códigos que devem ser persistidos no arquivo que o *plug-in* gerará posteriormente. A camada *servico* é responsável por ler o arquivo XMI gerado pelo *plug-in* editor gráfico, transformando-o em objetos, de modo que no final do processo seja retornado uma estrutura de dados do tipo *hashmap* com os devidos objetos transformados. A camada *visao* tem como propósito gerenciar a utilização dos serviços e a geração de código pelo *plug-in* desenvolvido. Por intermédio de métodos provenientes das classes dessa camada, é possível ativar as funcionalidades do *plug-in*. A ativação ocorre através de eventos que são disparados pelos usuários que o utilizam.

4. Estudo de Caso - *Build a House*

Em [Boissier et al. 2013], foi apresentado o estudo de caso chamado *Build a House*. O estudo de caso em questão trata-se de um personagem chamado *Giacomo* que deseja construir sua casa. Para isso, ele precisa contratar empresas que possam executar as tarefas relacionadas com a construção e coordenar o trabalho dessas empresas. A contratação será feita por meio de uma licitação, na qual a empresa que apresentar o menor preço para a tarefa será contratada.

Para comprovar a aplicabilidade do *plug-in*, optou-se nesta seção por demonstrar a modelagem de um agente deste estudo de caso, utilizando a ferramenta gráfica desenvolvida. Complementarmente, mostra-se os respectivos códigos gerados para o agente modelado, evidenciando o processo de geração de código automatizado.

A Figura 5 representa o diagrama do modelo Geral do Agente *Giacomo* modelado para o estudo de caso em questão. Além deste diagrama, o estudo de caso conta com outros diagramas, que foram suprimidos deste artigo por questão de espaço. Este dia-

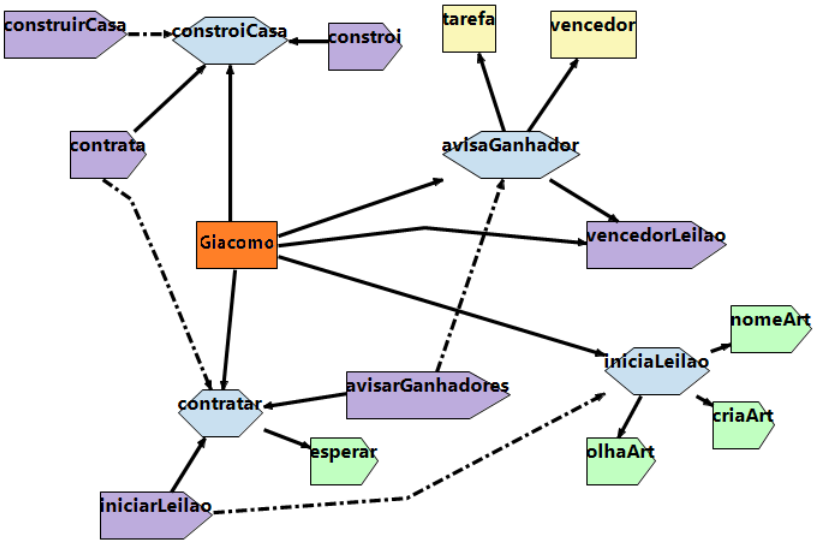


Figura 5. Diagrama de Modelo Geral do Agente Giacomo - Estudo de Caso *Build a House*

grama é oriundo do Diagrama de Visão Geral do sistema, onde os agentes são modelados de forma simplificada, sendo-os estendidos e enriquecidos através dos diagramas de visão geral do Agente. Para facilitar a geração de códigos, optou-se por criar um diagrama extra para a metodologia, intitulado Diagrama de Modelo Geral, onde possibilita-se que sejam modeladas todas as informações necessárias para realizar a geração de códigos para a ferramenta Jason. Este diagrama reúne os conceitos presentes no Diagrama de Visão Geral do Sistema, Diagrama de Visão Geral do Agente e Diagrama de Capacidade.

Na Figura 5, a entidade intitulada *Giacomo* representa o agente modelado. As entidades *constroiCasa*, *avisaGanhador*, *contratar* e *iniciaLeilao* representam os planos que compõem o agente. As entidades *construirCasa*, *constroi*, *contrata*, *vencedorLeilao*, *avisaGanhadores* e *iniciarLeilao* representam as mensagens enviadas/recebidas por esses planos ou as percepções obtidas por eles. As entidades *esperar*, *olhaArt*, *criaArt* e *nomeArt* representam as ações a serem desempenhadas na execução do plano. E por fim, as entidades *tarefa* e *vencedor* representam as crenças ligadas aos respectivos planos. Ressalta-se que este diagrama é um exemplo de uso do Diagrama de Modelo Geral, apresentando diferenças em relação aos demonstrados em [Uez 2013], sendo acrescentado a entidade que representa o agente, bem como as suas ligações com planos e mensagens. Essa alteração fez-se necessária em virtude do gerador de códigos precisar de informações precisas encontradas na concatenação de diversos diagramas, tornando o processo de geração de códigos complexo.

A Figura 6 exemplifica a geração de código ocorrida para o agente *Giacomo*. Esta geração é proveniente da modelagem do diagrama ilustrado na Figura 5. A sintaxe do Jason recomenda que primeiramente sejam informados os objetivos do agente, posteriormente as crenças que esse agente possui em relação ao ambiente e por fim, os planos que o compõem. Além disso, toda a parametrização das propriedades de cada um dos elementos foi baseado no estudo de caso de [Uez 2013].

Na Figura 6, a linha 2 refere-se a uma mensagem disponibilizada ao utilizador

```
1
2  //insert here the goals to boot your agent
3
4  @constroiCasa
5  +!construirCasa :true
6  <-!contrata;
7  !constroi.
8
9  @contratar
10 +!contrata :true
11 <-!iniciarLeilao(x);
12 !avisarGanhadores;
13 .wait(500).
14
15 @avisaGanhador
16 +!avisarGanhadores :true
17 <-.broadcast(tell, vendedor(Tarefa, Vencedor));
18 ?tarefa;
19 ?vencedor.
20
21 @iniciaLeilao
22 +!iniciarLeilao(x) :true
23 <-.concat("leilao_",Tarefa,NomeArtefato);
24 focus(id);
25 makeArtifact(NomeArtefato, "leilao", [Tarefa,ValorMaximo],Id).
26
```

Figura 6. Código gerado para o Agente Giacomo - Estudo de Caso *Build a House*

do *plug-in*, para que o mesmo possa informar os objetivos individuais de cada um dos agentes gerados. Posteriormente, são informados os planos dos agentes, conforme já falado anteriormente. Cabe lembrar que na linguagem *AgentSpeak*, a sintaxe de um plano ocorre da seguinte forma:

Trigger : context <- body

Portanto, as linhas 4, 9, 15 e 21 da Figura 6 representam a notação para representar a inicialização de um plano. A linha 5 representa a *Trigger* de um plano. A *Trigger* é um evento que dispara a execução de um plano. Conforme [Uez 2013], somente duas coisas podem representar uma *trigger* de um plano: entidades do tipo *Mensagem* ou *Percepção* ligadas ao plano através de setas pontilhadas. O *True* posterior aos ":"indica o contexto do plano, neste caso estipulado por parâmetros pelo diagramador. As linhas 6 e 7 representam respectivamente a inclusão de novos objetivos.

Ainda na Figura 6, as linhas 10, 11, 12 e 13 representam o plano *contratar*. Conforme é possível visualizar, após o <- são informadas todas as entidades que compõem a *body* de um plano. As linhas 15, 16, 17 e 18 representam o plano *avisaGanhador*. O comando *broadcast* refere-se a uma mensagem enviada em massa para todos os agentes que compõem o sistema. Além disso, através da Figura 5, nota-se que as ligações entre as entidades mensagens e planos representam as mensagens que devem ser trocadas pelo sistema, entretanto, é necessário que a mensagem também tenha a ligação com o agente remetente e os destinatários, quando houver o segundo caso. As linhas 18 e 19 representam consultas as crenças do plano e são originadas das ligações entre planos e crenças. Por último, as linhas 22, 23, 24 e 25 representam o plano *iniciaLeilao*. A linha 23 em diante ilustra uma concatenação de ações que compõem o plano em questão, findando assim a geração de código para o agente *Giacomo*.

5. Conclusões

Este trabalho propõe a criação de um *plug-in* de modelagem gráfica que tem como objetivo principal apoiar a metodologia Prometheus AEOlus. Esta metodologia foi desenvolvida por [Uez 2013] e tem como finalidade permitir a modelagem integrada das três dimensões que envolvem um SMA. Esta integração visa a interligação dos conceitos modelados nos *work products* com o *framework* de desenvolvimento de SMA chamado JaCaMo. Para colaborar com o processo de integração com o *framework* em questão, desenvolveu-se agregado ao *plug-in* gráfico, um mecanismo de varredura de informações que propicia a geração automática de código fonte para a linguagem *agentspeak*, associando o *plug-in* ao ambiente de desenvolvimento de agentes chamado Jason, o qual faz parte do *framework* JaCaMo. As dimensões do ambiente (CartAgO) e da organização (Moise+), as quais compõem o restante do *framework* JaCaMo, não foram tratadas neste trabalho.

Para avaliar o *plug-in* desenvolvido, modelou-se o estudo de caso *Build a House*. Demonstrou-se o resultado da modelagem do diagrama na ferramenta e posteriormente a sua geração de código para a plataforma de desenvolvimento Jason. Ao desenvolver o *plug-in* foi possível constatar que a metodologia possui uma série de diagramas, tornando seu uso bastante custoso. Essa diversidade é consequência de diversos deles possuírem entidades bastante semelhantes, ocorrendo uma interdependência e repetição de informações entre os diagramas desenvolvidos no decorrer do uso da metodologia.

Em suma, o desenvolvimento deste trabalho contribuiu para a potencialização da metodologia desenvolvida por [Uez 2013]. A utilização da plataforma GMF Eclipse foi uma escolha positiva para o desenvolvimento do *plug-in* gráfico, visto que apresenta uma boa estrutura de uso e possibilita que outras pessoas façam a manutenção da parte gráfica do *plug-in* posteriormente.

Outro ponto positivo no desenvolvimento deste trabalho foi a escolha de sua arquitetura. A mesma separa as responsabilidades da solução em diferentes *plug-ins*, facilitando sua manutenção e evitando a incumbência exagerada sobre as camadas do *plug-in*. Um fator a aprimorar é *plug-in* gerador de códigos, visto que o mesmo foi codificado na linguagem Java e apresenta uma solução engessada para o domínio específico da metodologia na qual se deseja apoiar. Em um trabalho futuro, pode-se desenvolver uma nova forma de geração de códigos, onde seja possível o uso de MDE, tornando o *plug-in* desenvolvido totalmente automatizado e orientado a modelos, contribuindo inclusive para a sua extensão para outras plataformas de desenvolvimento.

6. Agradecimentos

Os autores agradecem à Universidade Federal do Rio Grande - FURG e a Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul - FAPERGS pelo suporte financeiro na realização do presente trabalho.

Referências

Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., and Santi, A. (2013). Multi-agent oriented programming with jacamo. *Science of Computer Programming*, 78(6):747–761.

Bratman, M. (1987). Intention, plans, and practical reason. Harvard University Press, Cambridge, MA.

Dennett, D. C. (1989). *The intentional stance*. MIT press.

Foundation, E. (2014a). Eclipse documentation - current release. Disponível em: http://help.eclipse.org/luna/index.jsp. Acesso em 19 de janeiro de 2016.

Foundation, E. (2014b). Graphical modeling framework. Disponível em: http:/wiki.eclipse.org/Graphical_Modeling_Framework. Acesso em 19 de janeiro de 2016.

Gleizes, M.-P. and Gomez-Sanz, J. J. (2009). *Agent-Oriented Software Engineering X: 10th International Workshop, AOSE 2009, Budapest, Hungary, May 11-12, 2009, Revised Selected Papers*, volume 6038. Springer.

Gronback, R. C. (2009). *Eclipse modeling project: a domain-specific language (DSL) toolkit*. Pearson Education.

Guedes, G. T. A. (2012). *Um metamodelo UML para a modelagem de requisitos em projetos de sistemas multiagentes*. PhD thesis, Universidade Federal do Rio Grande do Sul.

Kleppe, A. G., Warmer, J. B., and Bast, W. (2003). *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional.

Mellor, S. J. (2004). *MDA distilled: principles of model-driven architecture*. Addison-Wesley Professional.

Miller, J., Mukerji, J., et al. (2001). Model driven architecture (mda). *Object Management Group, Draft Specification ormsc/2001-07-01*.

Rivieres, J. and Wiegand, J. (2004). Eclipse: A platform for integrating development tools. *IBM Systems Journal*, 43(2):371–383.

Rube, R. I. (2013). Introducción a la ingeneriería del software dirigida por modelos. Disponível em: https://ocw.uca.es/pluginfile.php/2487/mod_resource/content/0/T1%20-%20MDE.pdf. Acesso em 20 de janeiro de 2016.

Uez, D. M. (2013). Método para o desenvolvimento de software orientado a agentes considerando o ambiente e a organização. Master's thesis, Universidade Federal de Santa Catarina.

Uez, D. M. (2014). Descrição do método prometheus aeolus. Disponível em: http://www.uez.com.br/aeolus/docs/aeolus_11112014.pdf. Acesso em 23 de setembro de 2015.

Van Deursen, A. and Klint, P. (2002). Domain-specific language design requires feature descriptions. *CIT. Journal of computing and information technology*, 10(1):1–17.

Utilização de grau de confiança entre agentes para alocação de vagas em um Smart Parking

Lucas Fernando Souza de Castro, Gleifer Vaz Alves, André Pinz Borges

[1]Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná (UTFPR)
CEP 84016 – 210 – Ponta Grossa – PR – Brasil

l.castropg@gmail.com, {gleifer,apborges}@utfpr.edu.br

Abstract. *Looking for a parking spot in a big city could be a problem. There are computing solutions that are being developed to optimize this problem. One of these solutions is using multiagent systems (MAS). In this paper a MAS is developed in order to allocate spots in a smart parking using the framework JaCaMo. This paper comprises of two types of agents: manager and drivers. The manager is responsible to manage the parking spots which will be assigned for drivers according to a corresponding degree of trust. In order to verify the effectiveness of the MAS, several simulations were conducted in empirical scenarios. Experiments shows that the degree of trust impacts in the parking spot allocation process.*

Resumo. *Procurar por uma vaga de estacionamento em uma grande cidade pode ser um problema. Soluções computacionais estão sendo desenvolvidas para otimizar este problema. Uma dessas soluções é utilizando sistemas multi-agentes (SMA). Neste artigo um SMA é desenvolvido com o objetivo de alocar vagas em um estacionamento inteligente usando o framework JaCaMo. Este trabalho compreende dois tipos de agentes: gerente e motoristas. O gerente é responsável por gerenciar as vagas, as quais são atribuídas aos motoristas de acordo com um grau correspondente de confiança. Para verificar a eficácia do SMA proposto, foram conduzidas simulações em cenários empíricos que mostram que o grau de confiança impacta o processo de alocação de vagas.*

1. Introdução

Tecnologias a fim de facilitar a vida do homem moderno possuem uma grande demanda nos dias atuais, em função da necessidade do homem em realizar tarefas com cada vez menos recursos e tempo. Contudo, tais tecnologias muitas vezes não são capazes de atender tais demandas, sendo necessário otimizá-las, uma vez que estão inseridas em um mundo finito de recursos. Locais como cidades, pontos comerciais e lares deixaram de ser apenas um lugar de convívio de pessoas, mas sim um sistema social entre pessoas e objetos tecnológicos [Caragliu et al. 2011].

Cidades inteligentes têm como objetivo geral proporcionar elementos que facilitem a vida da população por meio do uso de tecnologia da informação para a minimização de custos e uso de recursos. Nestas cidades ainda há sub-elementos que podem ser otimizados, tais como: *economy*, *mobility*, *enviroment*, *people*, *living* e *governance*

[Caragliu et al. 2011]. Inseridos nas cidades inteligentes, os estacionamentos inteligentes ou *smart parking* são definidos como locais que utilizam tecnologias para automatizar e aprimorar as tarefas diárias de um estacionamento, como por exemplo, a alocação de vagas [Revathi and Dhulipala 2012]. Cidades como San Francisco nos Estados Unidos [SFPark 2015] e outras espalhadas pela América do Norte [Parkingedge 2013] desenvolveram sistemas computacionais para a automatização do processo de alocação e precificação das vagas de acordo com a sua utilização. Logo, é possível que motoristas verifiquem a disponibilidade e valores de vagas antes mesmo de saírem de suas casas.

No desenvolvimento de soluções computacionais para os *smart parkings* destacam-se o uso de sistemas multiagentes (SMA), devido a natureza do cenário dos estacionamentos. Nestes locais há um grande número de variáveis que devem ser controladas simultaneamente, no caso dos estacionamentos, os motoristas, vagas, pagamentos e outras variáveis. Além do processo de alocação de vagas, destaca-se o processo de negociação de vagas devido ao número limitado de vagas. Além de cidades americanas, em Napoles na Itália foi desenvolvido uma solução para *smart parking* que compreende o processo de negociação e precificação de vagas utilizando SMA [Di Napoli et al. 2014]. Destaca-se também que soluções envolvendo SMA possuem um fator social empregado, onde os agentes podem cooperar ou disputar um recurso, sendo no caso dos estacionamentos uma vaga [Di Nocera et al. 2014].

Sistemas multiagentes são sistemas compostos por agentes autônomos com um nível social. Os agentes possuem objetivos a serem atingidos e estão inseridos em um ambiente dinâmico [Wooldridge 2009]. A fim de tornar o desenvolvimento do SMA capaz de proporcionar características de um raciocínio humano, são utilizado modelos que abstraem tal raciocínio, como o modelo BDI (*Belief, Desire, Intention*). Além do modelo, é necessário utilizar ferramentas para desenvolver o SMA com estas características. Um exemplo de tal ferramenta é o *framework* JaCaMo .

O JaCaMo é um *framework* baseado no modelo BDI sendo composto por três módulos para o desenvolvimento de SMAs: uma linguagem de desenvolvimento para os agentes, um *framework* para o desenvolvimento dos artefatos presentes no ambiente que os agentes estão inseridos e uma ferramenta responsável pela normalização social do sistema [Boissier et al. 2013]. Para a implementação dos agentes, é utilizada a linguagem Jason, a qual foi desenvolvida baseada na linguagem AgentSpeak(L) para a programação de agentes BDI. O *framework* Cartago atua nos artefatos do ambiente que os agentes estão inseridos. Os artefatos presentes no ambiente de um modo geral fornecem funcionalidades aos agentes. E por fim, a ferramenta Moise atua na organização social dos agentes, delimitando o que os agentes podem e devem fazer e como realizam suas tarefas.

O presente trabalho destina-se a modelagem e desenvolvimento de um SMA capaz de alocar vagas em estacionamentos baseado na utilização de um agente centralizador (*manager*) para o gerenciamento destas vagas. A utilização das vagas é dada pelos agentes *drivers*, sendo que eles possuem um grau de confiança (*trust*) variável entre $(0-999)$. A confiança é proporcional a preferência de cada agente em possuir tal vaga devido ao número limitado de vagas no estacionamento. O atual trabalho está inserido em projeto do grupo de pesquisa GPAS (Grupo de Pesquisa em Agentes e Sotfware) denominado `MAPS` (***Multi**A**gent** P**arking** S**ystem***). O objetivo do `MAPS` é estender o trabalho desenvolvido em pesquisas anteriores [Gonçalves and Alves 2015] e também desta pesquisa.

Há soluções computacionais para a problemática dos *Smart parkings* [Zhao et al. 2014], inclusive com a utilização de SMA em diferentes plataformas, como por exemplo a linguagem Jade [Di Napoli et al. 2014]. O principal objetivo deste trabalho consiste na análise do grau de confiança entre os agentes presentes no estacionamento e como este grau impacta no processo de alocação de vagas do estacionamento, visto que há agentes *drivers* com graus completamente distintos de confiança, mas que compartilham o mesmo objetivo, i.e., uma vaga no *Smart Parking*.

O restante do artigo está organizado como segue: Na seção 2 é apresentado a modelagem e implementação do sistema multiagente para o `MAPS`. Na seção 3 são apresentados os resultados que o SMA apresentou em diferentes de cenários de utilização, e por fim, na seção 4 a conclusão.

2. Modelo do SMA para o projeto MAPS

O objetivo do SMA desenvolvido é alocar vagas para agentes *drivers*. O agente responsável pelo processo de alocação de vagas é o agente *manager*. Inserido no processo de alocação de vagas, há sub-processos que ocorrem para que a vaga seja devidamente alocada. A figura 1 ilustra uma diagrama de casos de uso para demonstrar as ações que os agentes podem realizar.

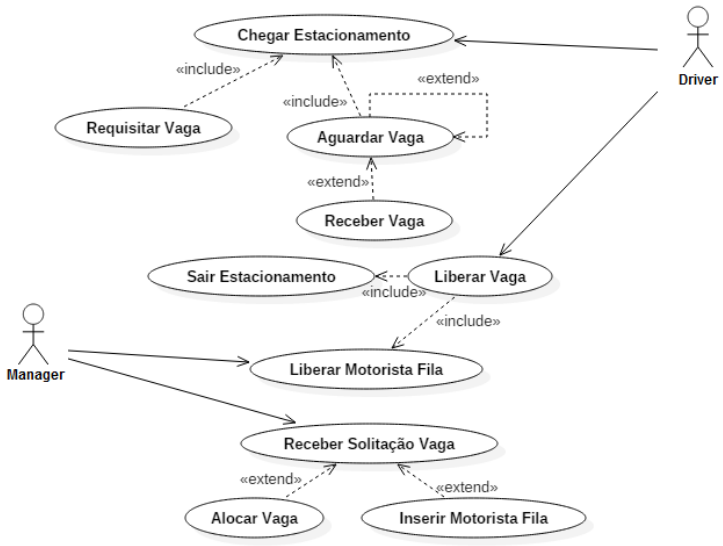


Figura 1. Digrama de casos de uso - SMA

2.1. Principais funcionalidades

O primeiro passo do desenvolvimento do SMA foi elencar os requisitos do sistema providos aos agentes para o tornar o sistema robusto e flexível a futuras extensões do projeto `MAPS`, dentre eles:

Ator: Agente *Manager*

1. **Receber solicitação de requisição de vaga:** O *manager* pode a qualquer momento receber requisições providas dos agentes *driver* para vagas no estacionamento. Nem sempre uma requisição de vaga é atendida, pois o estacionamento pode estar lotado;

2. **Decidir para qual agente alocar uma vaga:** Caso o estacionamento esteja lotado, ou próximo da sua lotação máxima, o *manager* se baseia no grau de confiança (*degree of trust*) dos agentes *drivers* e no tempo que ele está aguardando a vaga, caso for acima de 60 segundos. O tempo de 60 segundos foi estipulado devido a aplicabilidade do SMA desenvolvido a estacionamentos de médio porte, podendo este valor ser ajustável de acordo com a utilização do estacionamento. Para o cálculo do grau de confiança dos agentes, a seguinte fórmuma é utilizada:

$$degreeTrust(X) = degreeTrust(X) + \alpha$$

$\alpha = 1$ caso o *driver* utilizou a vaga selecionada pelo *manager*

$\alpha = 0$ caso contrário

3. **Possuir conhecimento e controle sobre o estacionamento:** O agente *manager* possui todo controle do estacionamento, pois é por meio dele que um agente requisita, ganha, utiliza e libera uma vaga. Além disso, o *manager* tem conhecimento de todas as vagas, o seu estado (livre, ocupada ou reservada), sua localização e quem está ocupando-a;
4. **Receber o aviso de um motorista que está saindo do estacionamento:** Ao sair do estacionamento, o *driver* deve avisar ao *manager* que está deixando a vaga livre. Após isso ocorrer, o *manager* decide qual agente *driver* irá receber a vaga, caso exista uma fila de espera.
5. **Controlar a fila do estacionamento:** Caso o estacionamento esteja cheio, os novos *drivers* que requisitarem uma vaga serão inseridos em uma fila. A fila é organizada de acordo com o grau de confiança que os agentes *drivers* possuem. Porém, pode haver um *driver* esperando a mais de 60 segundos nessa fila, sendo assim será dado prioridade a este agente.

Ator: Agente *Driver*

1. **Requisitar uma vaga ao agente *manager*:** O *driver* ao chegar no estacionamento requisita uma vaga ao agente *manager*, podendo ser respondido com uma vaga ou tendo que aguardar por uma;
2. **Receber uma vaga e estacionar:** Ao receber uma vaga, o agente deverá dirigir-se a esta vaga e estacionar o seu veículo;
3. **Aguardar uma vaga:** Caso o estacionamento esteja lotado ao requisitar uma vaga, o agente *driver* deverá aguardar até que o agente *manager* o notifique com uma liberação de vaga;
4. **Possuir características e crenças de um usuário de estacionamento:** A fim de tornar a abstração do agente *driver* próxima a um motorista real, os agentes *driver* presentes no sistema possuem as seguintes crenças: tempo para chegada no estacionamento, tempo aproximado de estadia no estacionamento, vaga que está estacionado e grau de confiança.
5. **Possuir um grau de confiança perante o agente *manager*:** Assim como descrito no item anterior, um agente *driver* é capaz de possuir um grau de confiança perante o *manager*. Este grau compreende as atitudes desse motorista no estacionamento, como por exemplo: não infringir as regras e utilizar o estacionamento de forma regular. O valor poderá diminuir caso o motorista não cumpra as regras estabelecidas.

Na figura 2 é ilustrado de uma maneira geral o funcionamento do *Smart parking* e como o grau de confiança impacta no processo de alocação de vagas.

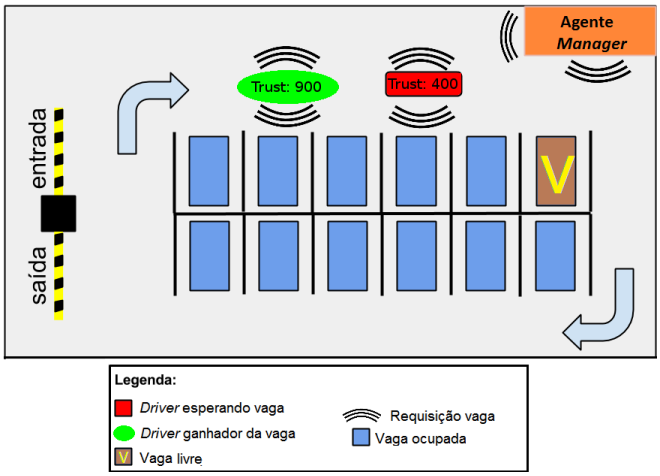


Figura 2. Estrutura básica do estacionamento - Adaptado de [Gonçalves and Alves 2015]

2.2. Desenvolvimento do SMA utilizando JaCaMo

A etapa de implementação do SMA é subdividida em três etapas. A etapa inicial da implementação é o desenvolvimento dos agentes e suas interações. A segunda etapa é a implementação dos artefatos do ambiente e suas interações com os agentes e por fim a etapa de otimização da utilização desses artefatos. Destaca-se que essa implementação do `MAPS` apresenta apenas o uso do Jason e Cartago. A programação normativa do sistema através do Moise será desenvolvida em uma etapa subsequente do projeto. A seguir é apresentado na figura 3 um diagrama geral elaborado por meio da metodologia Prometheus da arquitetura do SMA.

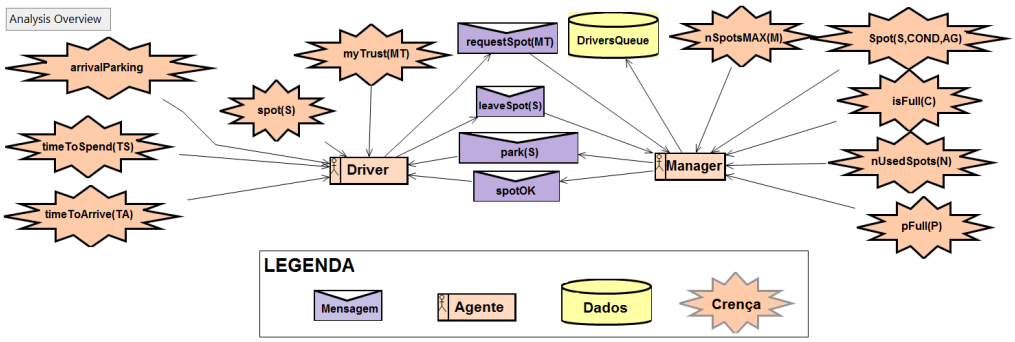


Figura 3. Visão geral do SMA

Etapa 1: Implementação dos agentes

A implementação dos agentes utilizando o *framework* JaCaMo é feita na linguagem Jason. O modelo BDI denota que as crenças são as informações a respeito do agente e do ambiente em que ele está inserido. Já os desejos são denotados como os objetivos que

o agente possui, ou o que deseja alcançar, e por fim, as intenções são como o agente irá atingir esses objetivos, o que no Jason são denotados como planos.

O JaCaMo possui uma linguagem chamada JCM a qual é possível determinar as crenças iniciais do agente, seus objetivos, artefatos do ambiente e outros parâmetros relacionados ao SMA. No código 1 é apresentado o arquivo JCM do sistema multiagente, exibindo dois agentes *drivers* (`m1` e `m2`) com suas crenças iniciais e a declaração do agente *manager*. Além das crenças iniciais, há crenças adquiridas ao decorrer da execução. Um exemplo de uma crença adquirida durante a execução é a vaga que um *driver* recebe.

```
1  mas mAPS{
2  agent manager
3          focus: maS3.Control
4          focus: maS3.Gate
5  agent m1: driver.asl {
6          beliefs: myTrust(100)
7                   timeToSpend(10000)
8                   timeToArrive(17514)}
9
10 agent m2: driver.asl {
11         beliefs: myTrust(450)
12                  timeToSpend(3000)
13                  timeToArrive(25307)}
```

Código 1. Arquivo JCM

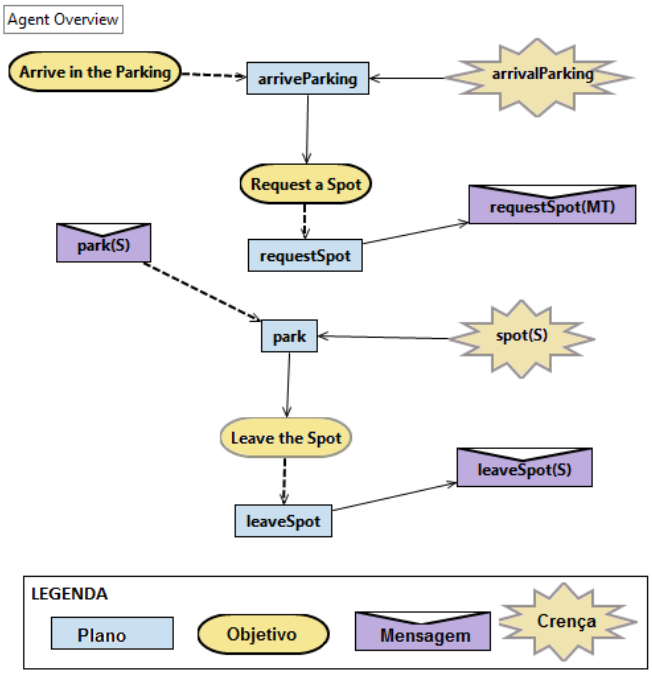


Figura 4. Perspectiva geral do agente *Driver*

As crenças iniciais do agente *driver* exibidas no código 1 e as crenças adquiridas também ilustradas na figura 4 são descritas nos itens abaixo.

- **myTrust(MT)**: Crença inicial do grau de confiança que o agente possui, sendo `MT` o valor correspondente da crença a respeito do grau de confiança;

- **timeToSpend(TS)**: Crença inicial do tempo que o agente permanecerá dentro do estacionamento, onde `TS` é o valor correspondente desse tempo em segundos;
- **timeToArrive(TA)**: Crença inicial a respeito do tempo que o agente levará para chegar ao estacionamento e requisitar uma vaga, sendo `TA` o valor representativo da crença em segundos;
- **spot(S)**: Crença adquirida durante a execução do SMA após a requisição aceita da vaga. O agente *manager* envia ao agente *driver* o identificador da vaga para que o agente utilize a vaga, sendo `S` o identificador dessa vaga;
- **arrivalParking**: Crença adquirida após a chegada do agente *driver* no estacionamento.

Além das crenças, os agentes implementados possuem objetivos e planos. Na figura 4 é apresentado um diagrama de perspectiva geral do agente *driver* desenvolvido por meio da metodologia Prometheus. Esse diagrama ilustra as crenças adquiridas, objetivos, planos e as mensagens que o agente troca durante a execução do SMA.

As crenças do agente *manager* não encontram-se no arquivo JCM, elas foram inseridas diretamente no arquivo MANAGER.ASL (vide código 2). A figura 5 apresenta o diagrama perspectiva geral do agente *manager*, ilustrando as suas crenças, objetivos, planos e trocas de mensagens com os agentes *drivers*.

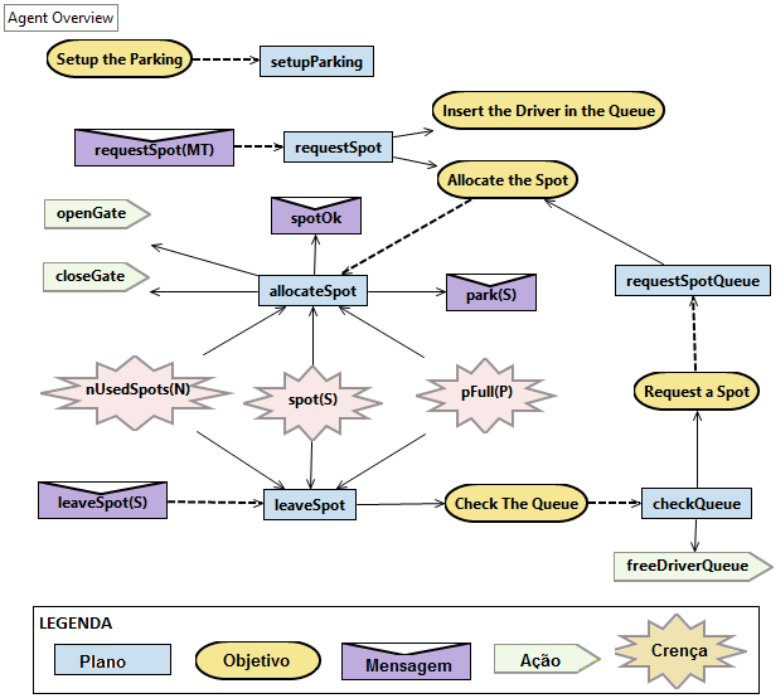


Figura 5. Perspectiva geral do agente *Manager*

No código 2 é apresentado uma parte do código MANAGER.ASL correspondente as crenças iniciais do agente *manager*. As crenças iniciais do agente *manager* exibidas na figura 5 e no código 2 são descritas a seguir.

```
1 nSpotsMAX(4).
2 nUsedSpots(0).
3 isFull(false).
4 pFull(0).
5 spot(0,0, "EMPTY").
6 spot(1,0, "EMPTY").
7 spot(2,0, "EMPTY").
8 spot(3,0, "EMPTY").
```

Código 2. Crenças e objetivos iniciais - Agente *Manager*

- **nSpotsUsed(N)**: Número de vagas utilizadas no momento, sendo `N` o valor correspondente inteiro correspondente a essas vagas;
- **nSpotsMAX(M)**: Número máximo de vagas que o estacionamento comporta, onde `M` é o valor inteiro máximo destas vagas;
- **isFull(C)**: Condição booleana para verificar se o estacionamento está cheio. Caso sim, os *drivers* que requisitarem uma vaga serão destinados para a fila, onde `C` expressa tal condição;
- **pUsage(P)**: Percentual de uso do estacionamento, sendo `P` um valor inteiro que representa essa porcentagem;
- **spot(S,COND,AG)**: Diferente do spot(S) do agente *driver* , essa crença determina as características da vaga em uma tupla de valores, onde o S é responsável pelo identificador da vaga; COND para verificar se há um agente ocupando a vaga e AG para identificar qual *driver* está ocupando a vaga;

Etapa 2: Implementação dos artefatos

Os artefatos em Cartago são os responsáveis por prover ações e funcionalidades para os agentes inseridos no ambiente. No contexto desse trabalho, o ambiente em si é o estacionamento. Para a versão do sistema do atual trabalho foram implementados dois artefatos: *Control* e *Gate*.

- **Artefato *Gate*:** Situando na entrada do estacionamento, responsável pela abertura e fechamento da cancela;
- **Artefato *Control*:** Responsável pelo gerenciamento dos *drivers* na fila, sendo inserindo-os ou selecionando o *driver* com as melhores condições de receber uma vaga.

A fila que o artefato *Control* gerencia é ordenada de acordo com o grau de confiança que cada *driver* possui, sendo assim, quanto maior o grau de confiança desse agente, mais rápido será a sua alocação para a vaga. Contudo, caso exista algum *driver* com um grau de confiança baixo, mas que já esteja na fila de espera por um tempo considerável. Esse *driver* não ficará aguardando na fila por mais tempo que um certo tempo limite. Na atual versão do SMA o limite é igual a 60 segundos, visto que o SMA aqui modelado é de um estacionamento privado e de médio porte. Portanto, caso existam motoristas na fila aguardando por uma vaga e uma vaga é liberada, o agente *manager* irá primeiro verificar se há algum *driver* com um tempo de espera na fila igual ou superior a 60 segundos. Se não houver, é analisado o grau de confiança e o agente com o maior valor recebe a vaga.

3. Resultados

Com o objetivo de demonstrar o impacto que o grau de confiança gera no processo de alocação de vagas, a seguir são descritos três cenários em que o sistema foi submetido com diferentes perfis de agentes *drivers*. Para cada perfil foi estipulado valores aleatórios para as crenças de cada agente. Na tabela 1 são apresentado os cenários C1, C2 e C3, bem como a quantidade de *drivers* presentes e a quantidade de vagas para cada cenário. A tabela 2 apresenta a configuração dos *drivers* com os seus respectivos valores para as crenças `timeToArrive`, `timeToSpend` e `myTrust`. Inicialmente o `MAPS` foi submetido a dez cenários para teste, porém neste trabalho serão descritos apenas os cenários mais críticos, os quais tem uma grande quantidade agentes *drivers* disputando por poucas vagas.

Tabela 1. Configuração dos cenários

Cenário	Número de *Drivers*	Número de vagas	Razão *Driver*/Vaga
1	50	1	50
2	50	2	25
3	10	1	10

Tabela 2. Configuração dos *drivers*

Agente	*timeToArrive(TA)*	*timeToSpend(TS)*	*myTrust(MT)*
m1	4	10	100
m2	2	3	450
m3	7	5	999
m4	3,5	7,5	4
m5	4,5	10	120
m6	6,7	9	770
m7	9	12	180
m8	10,29	8,5	10
m9	9,9	6,6	803
m10	4,6	6,6	5

Obs: Valores das crenças *timeToSpend* e *timeToArrive* são denotadas em segundos.

Os cenários propostos na tabela 1 apresentam de 10 a 50 agentes. Não foram utilizados mais agentes devido aos estacionamentos compreendidos pelo atual SMA serem de médio porte. A tabela 2 apresenta 10 perfis de agentes *drivers* . Para os cenários com 50 agentes foram criadas 5 instâncias de cada perfil para totalizar os 50 agentes propostos nos cenários C1 e C2.

Cenário 1

Neste cenário, tem-se 50 *drivers* disputando uma única vaga. A seguir na figura 6 é apresentado um gráfico do grau de confiança *versus* o tempo de alocação de vaga em segundos.

De acordo com o gráfico da figura 6, o grau de confiança impactou positivamente no tempo de alocação da vaga. Durante a execução do Cenário 1, existem vários *drivers* na fila de espera. Contudo, devido ao limite de tempo máximo de espera na fila, em certo

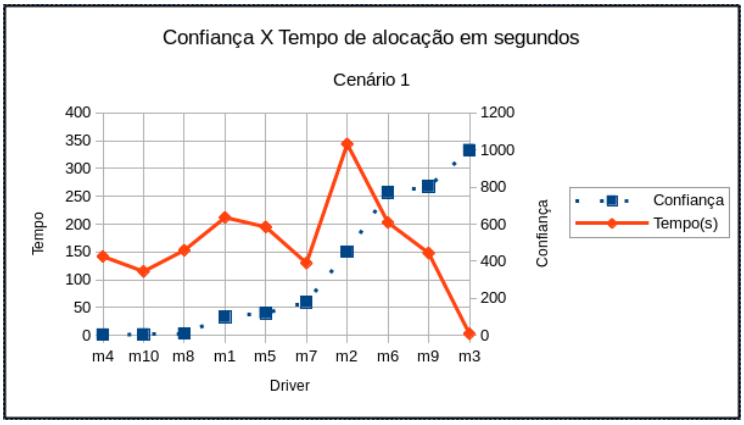


Figura 6. Cenário 1 - Relação grau de confiança *versus* tempo de alocação

momento será dada prioridade aqueles agentes que estão na fila por mais de 60 segundos. Ainda destaca-se que o *driver* `m3` (que possui um grau máximo de confiança - 999), tem o menor tempo de espera na fila de motoristas.

Cenário 2

Nesse cenário, há 50 agentes *drivers* para duas vagas. O gráfico da figura 7 apresenta a relação grau de confiança *versus* o tempo de alocação da vaga em segundos.

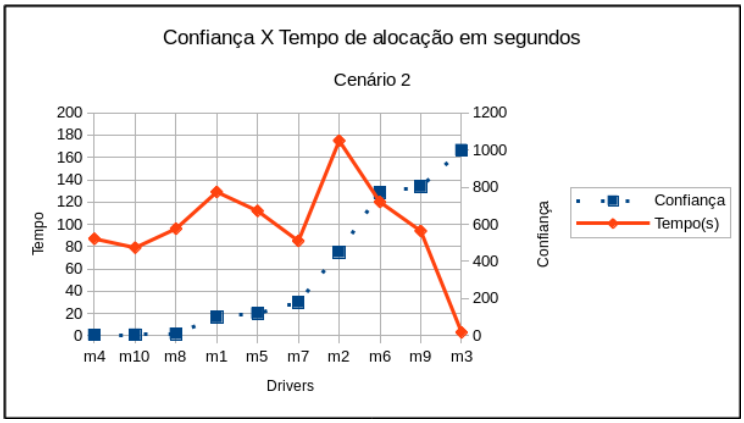


Figura 7. Cenário 2 - Relação grau confiança *versus* tempo de alocação

Neste cenário em contraste com cenário 1 possui uma vaga de estacionamento a mais disponível, diminuindo assim a média geral de tempo de alocação em segundos, sendo a média de tempo de alocação geral do Cenário 1 de 153 segundos e no Cenário 2 de 88 segundos. De modo simular ao cenário anterior, o grau de confiança impactou positivamente no tempo de alocação das vagas.

Cenário 3

No cenário 3 tem-se 10 agentes disputando uma única vaga. O gráfico apresentado na figura 8 demonstra o grau de confiança e seu impacto no processo de alocação de vagas.

Mesmo com um número reduzido de agentes utilizando o estacionamento, o grau de confiança ainda impactou no processo de alocação das vagas, mas um impacto menor

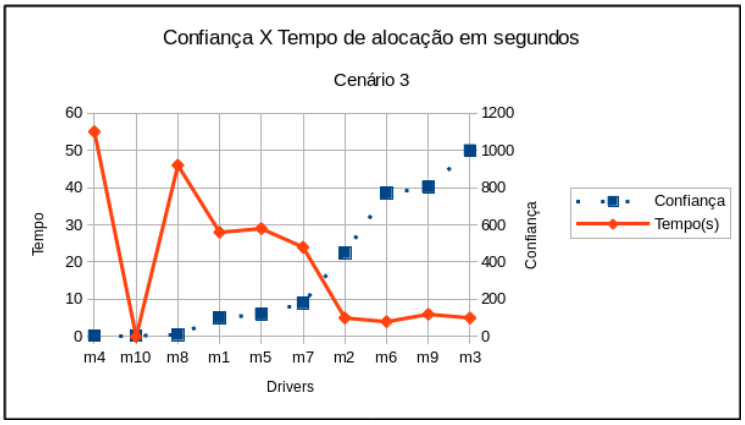


Figura 8. Cenário 3 - Relação grau de confiança *versus* tempo de alocação

em relação aos cenários anteriores. Porém, devido ao número reduzido de *drivers*, os tempos de espera não foram elevados ($>$ 60 segundos).

O principal objetivo da descrição dos três cenários é a análise do impacto que o grau de confiança gera na alocação de vagas dos agentes *Drivers*. Vale notar que durante a alocação de vagas também é analisado o tempo de espera do *Driver*. A fim de sumarizar os cenários apresentados, com base na tabela 2 foram selecionados agentes para cada nível de valor de *trust*: dois agentes (`m9` - 803) e (`m3` - 999) para o valor alto (700 $\rightarrow$ 999), um agente (`m2` - 450) para o valor médio (350 $\rightarrow$ 699) e dois agentes (`m4` - 4) e (`m7` - 180) para o valor baixo (0 $\rightarrow$ 349). A tabela 3 ilustra os agentes selecionados e seu tempo de alocação para receber uma vaga nos cenários C1,C2 e C3. Além disso é apresentado a média aritmética de alocação do agente nos cenários baseada no tempo de alocação para os cenários.

Tabela 3. Impacto do grau de confiança no processo de alocação de vagas

Agente	*myTrust (MT)*	*timeToArrive (TA)*	*timeToSpend (TS)*	C1	C2	C3	Média
m4	4	3,5	7,5	142	87	55	94.7
m7	180	9	12	130	85	24	79.7
m2	450	2	3s	344	175	5	175
m9	803	9,9	6,6	148	94	6	82
m3	999	7	5	3	3	5	3.7

Obs: Valores para as crenças: *timeToSpend*, *timeToArrive*, tempos de alocação de vaga para os cenários C1,C2,C3 e para a média são denotados em segundos.

De acordo com a tabela 3 é possível notar o impacto que o grau de confiança causa no processo de alocação de vagas. Entretanto há casos que esse valor entra em segundo plano devido ao tempo máximo de espera na fila de 60 segundos. O *driver* `m2` possui um grau médio de confiança e ainda assim possui a maior média, sendo assim nem sempre um *driver* com um valor médio terá um tempo de alocação médio. Por outro lado, na maioria dos casos dos *drivers* com um grau alto de confiança (*Driver* `m3` - *Trust*: 999) possuem em média um valor baixo para o tempo de alocação de uma vaga, em contrapartida os *drivers* com um grau baixo de confiança (*Driver* `m4` - *Trust*: 4) possuem em média um valor alto de tempo para alocação de uma vaga.

4. Conclusão

O principal objetivo do trabalho aqui apresentado é a modelagem e desenvolvimento de um SMA para alocação de vagas com base em graus de confiança. O desenvolvimento do SMA no *framework* JaCaMo proporcionou uma implementação robusta e flexível para futuras expansões do projeto. Além dos graus de confiança, há o fator do tempo limite de espera que influencia o processo de alocação de vagas, tornando-se prioridade para alguns *drivers* em relação à confiança em alguns cenários. O principal objetivo de inserir o tempo limite para alocação é priorizar também os *drivers* que estão a um longo tempo aguardando por uma vaga, todavia sem tirar o mérito dos agentes que possuem um grau de confiança alto. De fato, a inserção do tempo limite também busca a adequação do SMA a uma simulação de um caso mais próximo da realidade, visto que dificilmente um motorista iria esperar muito tempo na fila de um estacionamento privado.

Ainda é possível destacar possíveis extensões do projeto `MAPS`, como: (i) um artefato de persistência dos graus de confiança, de maneira que o grau de confiança possa ser incrementado ou decrementado conforme as utilizações dos *drivers*; (ii) artefato com uma interface gráfica do estado atual do sistema multiagente; (iii) organização social do SMA via Moise; (iv) futuras integrações com as plataformas Android e Arduino a fim de permitir que o `MAPS` seja aplicável em cenários reais.

Referências

Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., and Santi, A. (2013). Multi-agent oriented programming with jacamo. *Sci. Comput. Program.*, 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.

Di Napoli, C., Di Nocera, D., and Rossi, S. (2014). Negotiating parking spaces in smart cities. In *Proceeding of the 8th International Workshop on Agents in Traffic and Transportation, in conjunction with AAMAS*.

Di Nocera, D., Di Napoli, C., and Rossi, S. (2014). A Social-Aware Smart Parking Application. In *Proceedings of the 15th Workshop "From Objects to Agents* , volume 1269.

Gonçalves, W. R. C. and Alves, G. V. (2015). Smart parking: mecanismo de leilão de vagas de estacionamento usando reputação entre agentes. In *Anais do IX Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações – IX WESAAC*.

Parkingedge (2013). Best parking. Disponível em <http://www.bestparking.com>.

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.

SFPark (2015). Sfpark. Disponível em <http://www.sfpark.org>.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition.

Zhao, X., Zhao, K., and Hai, F. (2014). An algorithm of parking planning for smart parking system. pages 4965–4969. IEEE.

Retrospective, Relevance, and Trends of Software Agents, Environments and Applications School (WESAAC)

Enyo Gonçalves[1,4], Mariela Cortés[2], Marcos de Oliveira[1], Nécio Veras[3], Mário Falcão[2], Jaelson Castro[4]

[1] Universidade Federal do Ceará

[2] Universidade Estadual do Ceará

[3] Instituto Federal de Educação, Ciência e Tecnologia do Ceará

[4] Universidade Federal de Pernambuco

enyo@ufc.br, mariela@larces.uece.br, marcos.oliveira@ufc.br, necio.veras@ifce.br, mariofalcao.es@gmail.com, jbc@cin.ufpe.br

Abstract. *Multi-Agent Systems (MAS) software has been increasing dramatically in last years. In this context, the Software Agents, Environments and Applications School (WESAAC) is a Brazilian event of MAS which is in its tenth edition. When studying a research area, it is important to identify the most active groups, topics, the research trends and so forth. This study aims to investigate how the WESAAC is evolving, by analyzing the papers published in its 9 editions. We adopted a research strategy that combines scoping study and systematic review good practices. We identify the most active institutions and authors, the main topics discussed, the types of the contributions, the conferences and journals that have most referenced WESAAC papers, the publications with the greatest impact, and the trends in MAS. We found 116 papers over the 9 WESAAC editions, which were analyzed and discussed.*

1. INTRODUCTION

Nowadays, agent technology has been widely applied to solve a vast set of problems. Russell and Norvig [9] define an agent as an entity that can perceive its environment through sensors and act in the environment through actuators. According to Wooldridge [10], agents are complex entities with behavioural properties, such as (i) autonomy (i.e., they are able to execute without interacting with humans), (ii) social ability (i.e., they are able to interact by sending and receiving messages and not by explicit task invocation, (iii) Reactivity (i.e. The capacity of perceiving the environment and respond to its changes) and (iv) Proactiveness (i.e. a goal-directed behavior). Multi-Agent Systems (MAS) is the sub area of artificial intelligence that investigates the behavior of a set of autonomous agents, aiming to solve a problem that is beyond the capacity of a single agent [8]. The agent-oriented development paradigm requires adequate techniques to explore its benefits and features in order to support the construction and maintenance of this type of software [11].

In this context, the Software Agents, Environments and Applications School (WESAAC) provides a forum for researchers, practitioners and educators to present and

discuss the most recent innovations, trends, experiences and concerns in the field of software agents. The overall goal of the WESAAC is to provide an opportunity for researchers and students (both under-graduate and graduate) to discuss their goals, methods, and results of their research. The School aims to provide students with useful guidance on various aspects of the research from established researchers and the other student attendees. It also helps participants to establish a research and social network of their peers.

Motivated by the celebration of 10th edition of the WESAAC, this work aims to analyze the 116 full papers published in the WESAAC throughout its 9 previous editions. This review aims to identify the most active institutions and authors, the main topics discussed, the types of the contributions, the conferences and journals that have most referenced WESAAC papers, the most addressed topics, the publications with the greatest impact, and the trends in MAS.

The information provided in this paper may be useful in different contexts. For example, a newcomer (eg. new research student) will be able to identify main groups, main researchers and the work already developed. Moreover, topics that have not deserved much attention by the WESAAC may be identified and become the subject of new research projects. This kind of information will also be useful for setting up possible collaborative networks.

This paper is structured as follows. In Section 2, it is described the retrospective of the event. The research method is described in Section 3. In Section 4, it is presented the review results. In Section 5, it is discussed some threats to validity. Some related works are presented in Section 6. Finally, the conclusion and future works are presented in Section 7.

2. Retrospective of the Event

WESAAC provides a forum for researchers and a school for students. Initially called "Workshop - School of Agent Systems for Collaborative Environments" and then from the fourth edition, following suggestions from the participants of the first three editions, the event's scope was expanded without losing its trademark (acronym), and the interest that involved originally the researchers. In the fifth edition, the national scope of the event was maintained and strengthened, with the participation of prominent researchers from the MAS field, several institutions from Brazil, such as USP, IME / RJ, UFRGS, FURG, UFSC, UCPEL, UTFPR, and abroad, notably the University of Bologna (Italy), who presented lectures and led workshops on their research topics.

The first edition of WESSAC occurred in 2007 in the city of Pelotas- RS, at the Catholic University of Pelotas (UCPel), and it was coordinated by prof. Dr. Carlos Antonio da Rocha Costa. His second edition was in 2008 in the city of Santa Cruz do Sul-RS, at the University of Santa Cruz do Sul (UNISC), and it was coordinated by Prof. Dra. Rejane Frozza. The third edition happened in 2009 in the city of Caxias do Sul-RS, at the University of Caxias do Sul (UCS), under the coordination of prof. João Luis Tavares da Silva. The fourth edition took place in 2010 in the city of Rio Grande-RS, at the Federal University of Rio Grande (FURG), under the overall coordination of Prof. Dra. Diana Frances Adamatti. The fifth edition took place in 2011, in Curitiba-PR, at the Federal Technological University of Paraná (UTFPR), under the overall coordination of prof. Dr. Gustavo Alberto Giménez Lugo.

The sixth edition took place in 2012 in the city of Florianópolis-SC, at the Federal University of Santa Catarina (UFSC), under the coordination of prof. Dr. Jomi Fred Hübner. The seventh edition happened in the city of São Paulo-SP at the University of São Paulo (USP) in 2013, under coordination of Prof. Anarosa Alves Franco Brandão. The eighth edition occurred in Porto Alegre-RS at Pontifical Catholic University of Rio Grande do Sul (PUCRS) in 2014, under the organization of Prof. Rafael Bordini. The ninth and last edition took place in the city of Niterói-RJ in 2015, under the coordination of Prof. Viviane Torres da Silva [2].

All the proceedings and links to the home pages of WESAAC editions, and the results of this research are available in a web page created to this paper in: http://200.129.25.54/professor/necio/wesaac2016/site/#/.

3. RESEARCH METHOD

The experimental software engineering community has proposed reliable processes, guidelines and templates for conducting Systematic Literature Reviews (SLR) [6]. In this paper, we adopted the research strategy described in [4]; hence, we conducted a *scoping study* in order to "map out" the WESAAC previous editions.

While a systematic review is a means of identifying, evaluating and interpreting the available research findings related to a research question, topic area, or phenomenon [6], a scoping study is rather focused on examining the extent, range and nature of research activity, providing an overview in a specific area [4].

The set of steps applied in our paper were: (i) Protocol definition; ii) Research Questions definition; (iii) Conduction of Search; (iv) Data Extraction and Mapping; and (v) Data Analysis and Synthesis. These steps combine scoping study and systematic review good practices, such as protocol definition, to take advantage of both methodologies. The protocol has been designed and executed by four researchers with broad familiarity with MAS and one additional researcher that revised this protocol.

3.1. Research questions

The research questions that we intend to answer in this scoping study are:

RQ1 What are the main authors and institutions that published in WESAAC?

RQ2: What is the main language used in the papers?

RQ3. What are the main topics discussed in WESAAC?

RQ4. What are the types of the contributions?

RQ5. What is the impact of the publications of WESAAC in community?

RQ6. What are the trends in MAS?

3.2. Search Strategy, Data Sources and Study Selection

A manual search was conducted by four of the authors in all WESAAC proceedings in order to collect the studies, by examining the studies title and abstract. The following inclusion and exclusion criteria were used:

- Inclusion Criteria: All research papers published in the WESAAC;
- Exclusion Criteria: Short papers published in WESAAC.

The inclusion and exclusion criteria over the last 9 editions of WESAAC proceedings resulted in 116 studies to be further analyzed and classified.

3.3. Data extraction and synthesis

After the search and the selection processes, we performed a data extraction process by analyzing the 116 selected papers. In order to guide this data extraction, we used a predefined extraction form containing the following fields:

> *Identifier, Publication Year, Title, Language, Institution, Authors, Topic discussed, Type of contribution (model, tool, process, approach, method, etc.) and Number of citations by type (conference, journal, book, dissertation, thesis, other).*

This form enabled us to record full details of the papers under review and to be specific about how each of them addressed our research questions. The selection process and the data extraction were performed using a spreadsheet tool. In the next section, we present the results obtained in this scoping study.

4. Results

In this section, each research question is answered by analyzing different point of views, highlighting evidence gathered from the data extraction process.

RQ1. What are the main authors and institutions that published in WESAAC?

In these 9 editions, the WESAAC had the participation of 45 institutions and 230 authors.

From a total of 230 authors who published a paper on WESAAC editions, Antônio Carlos da Rocha Costa from Universidade Federal do Rio Grande do Sul (UFRGS)/Universidade Federal do Rio Grande (FURG) is leading the list with 18 published papers, followed by Graçaliz Pereira Dimuro (16) from Universidade Federal do Rio Grande (FURG) and Diana Francisca Adamatti (9) from Universidade Federal do Rio Grande (FURG). The list with the top four WESAAC authors is shown in Table 1. It is important to mention that the data shows the number of times in which an author is authoring or coauthoring an article.

Table 1. Top 14 Authors.

Author	Institution	#papers
Antônio Carlos da Rocha Costa	Universidade Federal do Rio Grande do Sul (UFRGS)/Universidade Federal do Rio Grande (FURG)	18
Graçaliz Pereira Dimuro	Universidade Federal do Rio Grande (FURG)	16
Diana Francisca Adamatti	Universidade Federal do Rio Grande (FURG)	9
Rejane Frozza	Universidade de Santa Cruz do Sul (UNISC)	8
Jomi Fred Hubner	Universidade Federal de Santa Catarina (UFSC)	5
Andrea Aparecida Konzen	Universidade de Santa Cruz do Sul (UNISC)	4
Cesar Augusto Tacla	Universidade Tecnológica Federal do Paraná (UTFPR)	4
Diego Rodrigues Pereira	Universidade Federal do Rio Grande do Sul (UFRGS)	4
Esteban de Manuel Jerez	Universidad de Sevilla	4
Jaime Simão Sichman	Universidade de São Paulo (USP)	4

Mariela Inés Cortés	Universidade Estadual do Ceará (UECE)	4
Saulo Popov Zambiasi	Universidade do Sul de Santa Catarina (UNISUL)	4
Viviane Torres da Silva	Universidade Federal Fluminense (UFF)/ IBM Research	4

As it has already been mentioned, 45 institutions have had at least one publication at WESAAC. See in Table 2 the 6 institutions with the most number of publications.

Table 2. Number of papers of the top 6 (six) institutions.

Institution	# papers
Universidade Católica de Pelotas (UCEPEL)	22
Universidade Federal do Rio Grande (FURG)	16
Universidade Federal de Santa Catarina (UFSC)	11
Universidade Federal do Rio Grande do Sul (UFRGS)	11
Universidade de Santa Cruz do Sul (UNISC)	8
Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS)	7

Despite all the universities listed in Table 2 are in the southern region, we can mention that the Universidade de São Paulo (7th place), University of Sevilla (9th place), State University of Ceará (12th place) and Federal Fluminense University (13th place). This shows that despite the hegemony of the Southern Brazilian universities, there is the participation of other regions of the country, and foreign universities as well. WESAAC started in south of Brazil and researchers of these region were involved in most of event editions, therefore it is natural that researchers and universities of the region are well classified.

After the identification of the most active authors and institutions, a verification of the most discussed topics was performed.

RQ2: What is the main language used in the papers?

We also investigated the language in which the papers were written and we identified the percentage of articles written in English and Portuguese. 94 papers (81%) were write in Portuguese and 22 papers (19%) were write in English.

RQ3. What are the main topics discussed in WESAAC?

All papers were analyzed in order to evaluate MAS topics that they have addressed. The list of discussed topics were based on the call for papers of previous editions of the WESAAC. Each paper was classified in one the topics.

The main topics defined according to the papers are shown in Table 3. “Applications of agents and multi-agent systems” is the main discussed topic (36.20%; 42 papers), followed by “agent-based software development” (16.37%; 19 papers). “Social simulations and agent-based simulation” follows closely with 14.65% (17 papers).

RQ4. What are the types of the contributions?

The purpose of this research question was to identify the contributions types of the WESAAC papers. We tried to list the types according to the classification presented in the

papers. Note that, similarly to other research questions, this question also allows a study to be included in only one category. The results of this question are presented in Table 4.

Table 3. The discussed topics in WESAAC.

Topic	#papers	%
applications of agents and multi-agent systems	42	36,2069
agent-based software development (programming languages, platforms, tools, methodologies)	19	16,37931
social simulations and agent-based simulation	17	14,65517
cooperation/coordination (negotiation, argumentation, reputation)	11	9,482759
agent organizations, societal issues, normative systems	9	7,758621
agent architectures and theories (BDI, belief revision, automated reasoning)	8	6,896552
agent communication	4	3,448276
(formal) specification and verification of multi-agent systems	4	3,448276

The predominant contribution that we identified was “Architecture/Framework” (18.96%; 22 studies), followed by “A multiagent System implementation” (18.10%; 21 studies), and “Tool” (15.51%; 18 studies).

Table 4. Types of contributions of WESAAC papers.

Type of Contribution	#papers	%
Architecture/Framework	22	18,96552
A multiagent System implementation	21	18,10345
Tool	18	15,51724
AI model/Formal model/Mathematical model/Ontology	16	13,7931
Simulation	8	6,896552
A multiagent Tecnique	5	4,310345
AI Technique implementation	5	4,310345
Algorithm	5	4,310345
Methodology	5	4,310345
Evaluation	5	4,310345
Multiagent System modeling	3	2,586207
Programming Language	2	1,724138
Discussion or Review	1	0,862069

The topic “AI model/Formal model/Mathematical model/Ontology” is proposed by more than 13,79% of the papers. The “Simulation” topic is proposed by more than 6,89% of the papers.

We also investigated if the contributions were empirically validated. Figure 1 shows that “simulation” is the main type of validation (19,82%, 23 papers), followed by “illustration” (14.65%, 17 papers). The validation “Experiment” was performed by

13.79% (16 papers), "comparison" was conducted by 8,62% (10 papers). 0.86% (1 paper) performed "survey". Some papers did not present any kind of validation (27,58%, 32 papers).

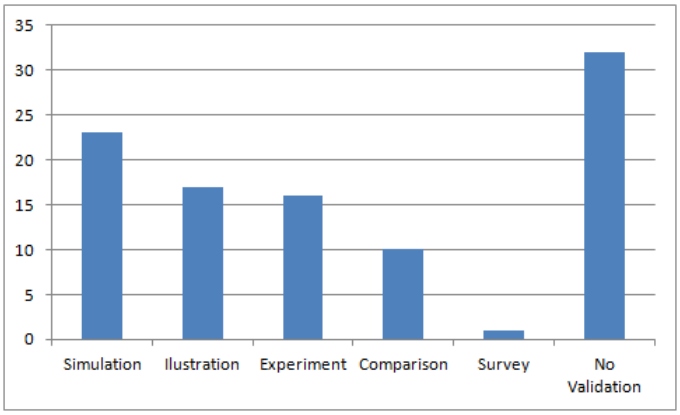


Figure 1. Empirical studies.

Q5. What are the publications of WESAAC community with the greatest impact?

The papers of the 9 WESAAC editions were cited, until 29th February 2016, 55 times by many publication types. The top 5 most cited papers are listed in Table 5. These papers represent the papers from the WESAAC that had the greatest research impact, considering citation count.

Table 5. TOP 5 WESAAC most cited papers.

Paper	Authors	Year	Citations
Agentes tutor e companheiro em um ambiente educacional baseado em estilos cognitivos	Rejane Frozza, Andréa Konzen, Alessandra Mainieri, Jacques Schreiber, Kurt Molz, Jorge Tautz, Ricardo Pedó, Jonas Dresch	2007	6
Modelando a Organização Social de um SMA para Simulacão dos Processos de Produção e Gestão Social de um Ecossistema Urbano: o caso da Horta San Jerónimo da cidade de Sevilla, Espanha	Flávia C. P. Santos, Glenda Dimuro, Thiago F. Rodrigues, Diana F. Adamati1,Graçaliz P. Dimuro, Antônio C. R. Costa, Esteban de Manuel Jerez	2012	5
Simulação Multiagente de uma Abordagem Evolutiva e Espacial para o Jogo do Ultimato	Luís Felipe K. Macedo, Murian dos Ribeiro, Stephanie L. Brião, Celso N. da Fonseca, Marilton S. de Aguiar, Graçaliz P. Dimuro	2012	4
Fred – um agente pedagógico mediador na construção do conhecimento	Julio Cezar Souza de Mello, Rejane Frozza	2007	4
Uma Arquitetura de Referência para Softwares Assistentes Pessoais Baseada em Agentes e SOA	Saulo Popov Zambiasi, Ricardo J. Rabelo	2011	4

The most referred paper is "*Agentes tutor e companheiro em um ambiente educacional baseado em estilos cognitivos",* published in 2007. It was cited 6 times and authored by Rejane Frozza, Andréa Konzen, Alessandra G. Mainieri, Jacques Schreiber, Kurt Molz, Jorge Tautz, Ricardo Pedó, Jonas Dresch.

The topics of the most cited papers are in accordance with the main topics discussed in WESAAC (RQ3). The second address the topic “agent organisations, societal issues, normative systems”. The first in third place papers, from 2012, is from “social simulations and agent-based simulation”. Finally, the other papers tied in third place cover “applications of agents and multi-agent systems” and “agent architectures and theories (BDI, belief revision, automated reasoning)”.

The first place paper addresses the “A multiagent System implementation”, the second is related to “Multiagent System modeling”, the first paper listed in thrid place papers covers “Simulation, Tool and Architecture/Framework”.

Regarding empirical studies, “illustration” was used to validate the paper in first place and the second place. The third place papers used “comparison”, not presented a validation and “simulation”, respectively.

RQ5. What is the impact of the WESAAC publications in community?

A deep investigation over the WESAAC papers shows the numbers of WESAAC referred papers over the years. This investigation was conducted by analyzing manually all citations of each 116 papers through Google Scholar.

Regarding the publication venues of the papers that cites WESAAC papers, we found the following distribution (see Figure 2): 51% in conferences (28 studies, which 9 studies in international conferences and 5 studies in WESAAC editions); 5% in journals (3 studies, which 2 studies in international conferences); 2% in Phd Thesis (1 study), 29% in Master Thesis (16 studies, which 1 international citation) and 13% (7 studies) in others types of documents (technical reports, presentations, etc.).

We also analyzed which conferences have most referenced WESAAC papers. Regarding conference papers, Table 6 shows the conferences that most cite WESAAC papers respectively.

Table 6. TOP 5 conferences.

Conference	#citations
Software Agents, Environments and Applications School (WESAAC)	5
Brazilian Workshop on Social Simulation	4
Latin American Informatics Conference (CLEI)	2

Based on these results, it is possible to see that the papers are serving as good sources to define and evolve the already proposed techniques, methods, processes and so on. However the references from WESAAC papers itself is considered low and could be encouraged in upcoming call for papers. Low amount of citations in journal papers and international conferences should be mentioned, probably it is related to few articles written in English (22 papers). Extended version of the best papers could be published in a journal, as other events been done, and this could encourage more authors to submit. Of the 55 citations, 15 are held by foreign researchers and 62 by researchers not published in WESAAC.

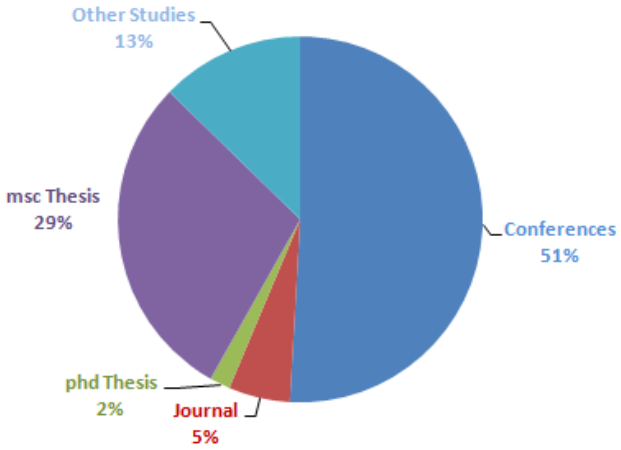


Figure 2. Publication venues of WESAAC papers.

RQ6. What are the trends in MAS?

The purpose of this question is to investigate the trends in MAS in relation to the topics addressed by the WESAAC papers and other papers of the MAS area. In this case we consider the main topics defined according to the papers showned in Table 3, i.e., the top 5 topics along the nine editions, and some papers related to our results.

Figure 3 shows the number of publications per topic/year. According to this figure, it is possible to observe an increase in the following topics: agent software development and agent organizations. On the other hand, we have observed a decrease in the number of papers of the following topics: agent applications and social simulation. The cooperation/coordination topic remained without great variations.

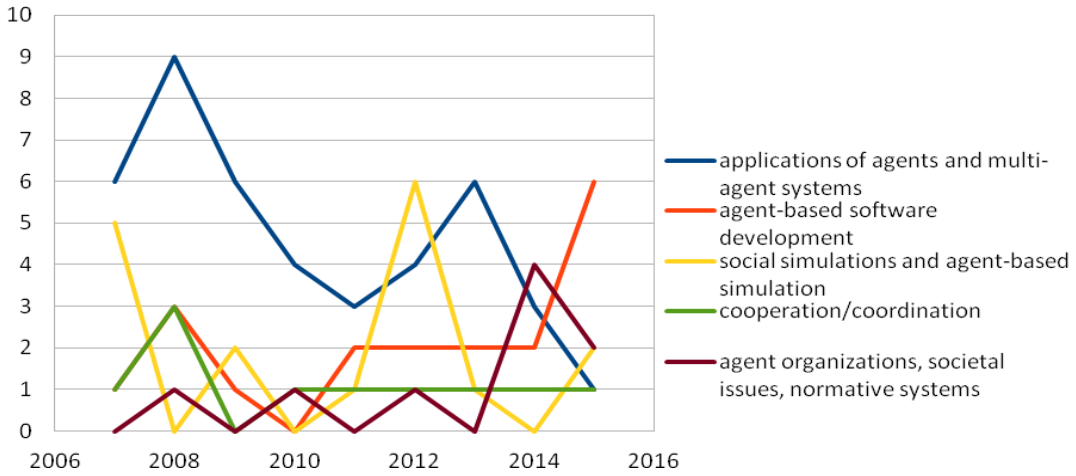


Figure 3. Number of WESAAC publications per topic/year

Figure 4 shows the number of publications per contribution/year, where we can note that all contributions have variations across the editions. According to this figure, it is possible to observe an increase in the last edition from the following contributions: “A Multiagent system implementation” and “Simulation”. On the other hand, it has been observed a decrease in the number of papers related to “Architecture/framework”. The “Tool” contribution has a variation and it was not proposed in last two editions of

WESAAC. The “AI model/Formal model/Mathematical model/Ontology” contribution remained without great variations.

The top of “Tool” contribution was in 2012 and “architecture/framework” was in 2010, with 5 papers each one. The top of “A Multiagent System implementation” was in 2008 with 6 papers. The top of “Simulation” was in 2012 and “AI model/Formal model/Mathematical model/Ontology” was in 2013 with 3 papers each one.

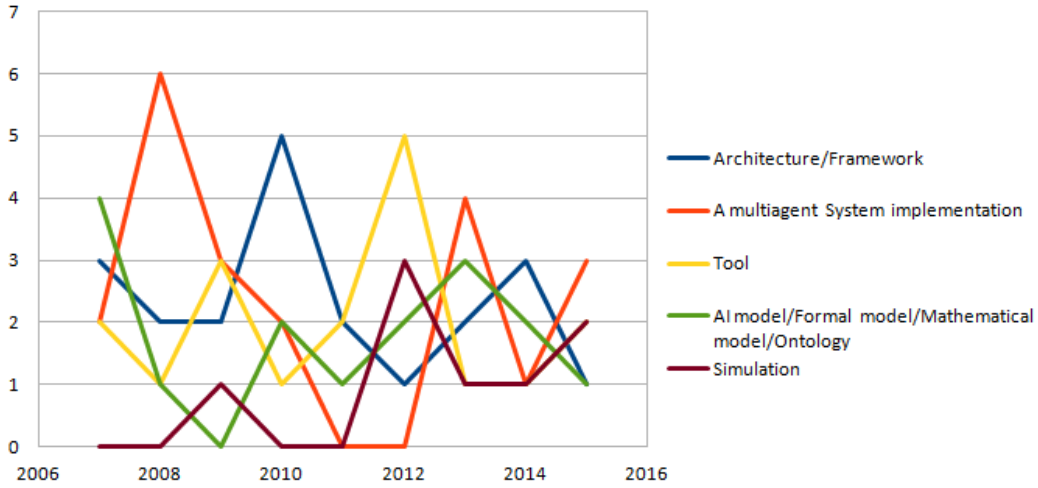


Figure 4. Number of WESAAC publications per contribution/year

Since 2005 specific technical challenges of agent-based computing are discussed: Trust and reputation, Virtual organization formation and management, Resource allocation and coordination, Negotiation, Emergence in large-scale agent systems, Learning and optimization theory, Methodologies, Provenance, Service architecture and composition, and Semantic integration [14].

In that roadmap the authors built fifteen recommendations about challenges appointed and some can found in the history of WESAAC such as: “Creation of tools, techniques and methodologies to support agent systems developers”, “Develop techniques for expressing and reasoning about trust and reputation, on both individual and social level, to enable interaction in dynamic and open environments”, and “Integrate agent components and features to enable the different theories, technologies and infrastructures to come together coherently”.

In other hand, recommendations such as “Develop a range of new techniques for learning and optimization in dynamic and unstable multi-agent environments, together with evaluation methods” and “Establish appropriate trade-offs between adaptability and predictability so that agents can exhibit behaviour, emergent or otherwise, that can be supported by tools and property verification.” are not exploited by authors in any paper.

According to [12], Agent Oriented Software Engineering can be divided in the following themes: Agent Programming Languages, Agent Oriented Frameworks, Agent Oriented Architectures, Agent Oriented Methodologies (Modeling Techniques) and Agent Communication (Content Languages, Interaction Mechanisms). The research community in the last almost 20 years, has explored these themes and the Applications built using technologies that come from them.

In [13], is made a report on the outcomes of the discussions of several researchers in the field, from universities and industry. They were from distinct areas and were engaged on work related with the following subjects: avionics and defence, smart cities, transportation, logistics, workflow management, decision support, healthcare, personal assistants, real-time control systems, power engineering, information integration, and virtual environments. They have concentrated the discussion in the following themes: Integration of MAS with other systems and technologies, and validation of MAS; Coordination and Organization; Tool, Languages and Technologies; and Component Oriented Agent Design.

In those surveys [13], the major concern was to drive efforts from the research community to find out why agent technologies were not embraced yet by the industry, and what is necessary to make that happen. They have pointed out some strategies to that end, such as: concentrate on making better tools and not only building new ones; create ways to validate and test MAS for safety, security, scalability, quality, maintainability, performance and interoperability, for example; componentize agent technology to better integrate it with other types of systems, concentrate on the scalability of the organizations and the runtime environments to support that. There is a concern as well to measure how the Agent Oriented techniques compare against each other, and with other technologies.

5. THREATS TO VALIDITY

There are some threats to the validity of our study, which we briefly describe along with the mitigation strategy for them.

In order to analyze the top authors, institutions, and countries, the number of papers for each one was considered. During this process, no distinction was made regarding the co-authoring of studies. The main author receives the same score as co-authors. We also have to consider that authors change of intuitions and countries. As the analysis of the citations of each 116 papers was performed manually, we considered it as an error prone activity. In order to mitigate this issue, some classifications were double checked.

Finally, we could have explored a broader set of data in order to investigate other aspects of the papers included in the review. Moreover, it is possible that some kind of inaccuracy or misclassification may have occurred in the data extraction performed in this scoping study.

6. RELATED WORKS

The analysis of how the Software engineering (SE) area as well as the Requirements Engineering is evolving has been the topic of some studies [3][4][5]. A scoping study was conducted by [3] in which the authors analyzed 512 papers of the 24 editions of Brazilian Symposium on Software Engineering (SBES), and understanding which is the impact of international research in this event. Their findings suggest that greater attention should be given to the SE area, with the aim to attract research from industry with real data, and also international collaboration.

The celebration of 25th anniversary of the SBES, as well as the realization of the Requirements Engineering Conference in Brazil for the first time, motivated [4] to conduct a mapping study aiming to have a closer look at the local RE community. Their

results showed that the Brazilian researchers have been extensively publishing at SBES and WER. Moreover, their findings reveal the better empirical validation maybe required.

A revision of 258 papers published at WER was performed by [5]. Their results are in a certain way similar as ours: Brazil is one of the countries with the most number of published papers in RE. Moreover, Universidade Federal de Pernambuco (UFPE) is one of the most active institution as far as publications at WER is concerned.

7. CONCLUSIONS AND FUTURE WORKS

In order to understand how the MAS community evolves, this study presented some findings resulted from a scoping study considering the 9 previous editions of WESAAC. In this sense, we identified the most active countries, institutions and authors, the main topics discussed, the types of the contributions, the conferences and journals that have most referenced WESAAC papers, the publications with the greatest impact, and trends in MAS.

Future works include an analysis to discuss about the schools (e.g. for their suitability, themes, repetition, impact on participants' dissertations/thesis or papers). An interview with experts in the area can be also a good way to understand the conference trends along the years [4].

ACKNOWLEDGMENTS

CNPq supported this work.

REFERENCES

[1] 10th Software Agents, Environments and Applications School (WESAAC) Available at: http://intelligentagents.github.io/wesaac-2016/en/

[2] 9th Software Agents, Environments and Applications School (WESAAC) Available at: http://www.inf.pucrs.br/felipe.meneguzzi/wesaac2014/history.html

[3] Dermeval, D.; Vilela, J.; Bittencourt, I.; Castro, J.; Isotani, S.; Brito, P.; Silva, A. Applications of ontologies in requirements engineering: a systematic review of the literature. In: Requirements Engineering Journal, 2015.

[4] Gomes, J.; Silveira, P.; Cruzes, D.; Snatana, E. 25 Years of Software Engineering in Brazil: An Analysis of SBES History. In: 25th Brazilian Symposium one Software Engineering (SBES), 2011, pp. 4-13.

[5] Laski, J.; Stancke, W.; Reinehr, S.; Malucelli, A. WER Overview: Retrospective, Trends and Relevance. In: CLEI Electronic Journal, vol. 17, n. 3, p. 4-4, 2014.

[6] Kitchenham, B; Charters, S. Guidelines for performing systematic literature reviews in software engineering. In: Technical report EBSE 2007–001, Keele University and Durham University Joint Report, 2007.

[7] Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. Systematic mapping studies in software engineering. In: EASE ’08: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, University of Bari, Italy, 2008.

[8] Jennings, N. R., 1996. Coordination Techniques for Distributed Artificial Intelligence. In: Foundations of Distributed Artificial Intelligence, pp. 187-210, Wiley.

[9] Russell, S. and Norvig, P., 2003. Artificial Intelligence: A Modern Approach, 2nd Ed., Upper Saddle River, NJ: Prentice Hall, ISBN 0-13-790395-2

[10] Wooldridge, M. An Introduction to Multiagent Systems. Second Edition. Willey, 2009.

[11] Zambonelli, F., Jennings, N., Wooldridge, M., 2001. Organizational Abstractions For The Analysis And Design Of Multi-Agent Systems. In: Agent-Oriented Software Engineering, LNCS 1957, Berlin: Springer, p. 127-141.

[12] Sturm, Arnon, and Onn Shehory. "Agent-Oriented Software Engineering: Revisiting the State of the Art." Agent-Oriented Software Engineering. Springer Berlin Heidelberg, 2014.

[13] Dix, Jürgen, et al. "Engineering multi-agent systems (Dagstuhl seminar 12342)." Dagstuhl Reports 2.8 (2012).

[14] Luck, Michael, Peter McBurney, Onn Shehory, and Steve Willmott. "Agent technology: computing as interaction (a roadmap for agent based computing)." University of Southampton on be half of AgentLink III, 2005.

Failure Prediction based on Monitoring Sequences of Actions and Action Duration

Giovani Farias[1], Ramon Fraga Pereira[1], Lucas Hilgert[1], Felipe Meneguzzi[1], Renata Vieira[1], and Rafael H. Bordini[1]

[1]Pontifical Catholic University of Rio Grande do Sul – PUCRS
School of Informatics – FACIN – Porto Alegre, Brazil

{giovani.farias,ramon.pereira}@acad.pucrs.br
lucaswhilgert@gmail.com
{felipe.meneguzzi,renata.vieira,rafael.bordini}@pucrs.br

Abstract. *An agent can attempt to achieve multiple goals and each goal can be achieved by applying various different plans. Anticipating failures in agent plan execution is important to enable an agent to develop strategies to avoid or circumvent such failures, allowing the agent to achieve its goal. Plan recognition can be used to infer which plans are being executed from observations of sequences of activities being performed by an agent. Symbolic Plan Recognition is an algorithm that represents knowledge about the agents under observation in the form of a plan library. In this paper, we use this symbolic algorithm to find out which plan the agent is performing and we develop a failure prediction system, based on information available in the plan library and in a simplified calendar which manages the goals the agent has to achieve. This failure predictor is able to monitor the sequence of agent actions and detects if an action is taking too long or does not match the plan that the agent was expected to be performing.*

1. Introduction

Recently, the number of real-world applications that deal with the need to recognise goals and plans from agent observations is on the rise. These applications can be found in fields such as human assisted living [Masato 2012], interface agent systems [Armentano and Amandi 2007], human-computer interaction [Hong 2001], traffic monitoring [Pynadath and Wellman 1995], and others. However, techniques that include the task of anticipating failures during agent plan execution have received relatively little attention. Multi-agent environments are dynamic since they are in a constant estate of change resulting from agents' actions. When these changes occur, a plan that was expected to work before, may fail. Thus, anticipating from agent observations when a plan is going to fail can be an important mechanism during the plan recognition process. Plan recognition approaches often do not make such inferences, which means that when an agent has no intention to complete or finish a plan these approaches continuously attempt to recognise what the agent is doing. In daily activities most people interrupt the plan that they are performing for some reason, such as, getting their attention drawn to something else, getting distracted by other events, or being interrupted by an emergency that needs immediate attention. In a plan recognition context, we consider that a plan is going to fail when the sequence of actions is taking too long or does not match the plan which the

observed agent should be performing at the moment. Our approach uses a calendar for managing some of the agent's goals over the near future, and when that information is available it facilitates our failure checking procedure, as well as plan recognition disambiguation. A plan failure can occur when an agent interrupts its current plan execution due to concurrent plans that need attention, or when an agent has to deal with conflicting plans. In this case, a mechanism to anticipate failures during agent plan execution can be useful in several situations, for example, helping an agent stay focused on a particular plan, or detecting when an agent is deviating from its regular activities.

Research on planning has focused on the modelling of actions with duration and stochastic outcomes, both theoretically as variants of Markov Decision Processes (MDP) [Mausam and Weld 2008], and domain description languages that express temporal planning (e.g., PDDL 2.1 [Fox and Long 2003], an extension of PDDL. In the literature, a similar approach to failure prediction, as we introduce in this paper, is plan abandonment detection. Geib and Goldman [Geib and Goldman 2003] proposes a formal model to recognise goal/plan abandonment in the plan recognition context, based on the PHATT (Probabilistic Hostile Agent Task Tracker) model [Goldman et al. 1999]. This formal model estimates the probability that a set of observed actions in sequence contributes to the goal being monitored, furthermore, Geib [Geib 2002] addresses some issues and requirements for dealing with plan abandonment, as well as intention recognition in the elderly-care domain.

In this paper we develop an approach to predict plan failure by monitoring agent's actions during its plan execution. Essentially, our approach to plan failure prediction features a mechanism that is composed of three modules. The first module is responsible for recognising the plan that the observed agent is executing. The second module checks if plans assigned to observed agent are being executed as scheduled in a predefined calendar. Lastly, the third module checks if actions are being executed as expected (i.e., not taking too long, and matching the current monitored plan). Thus, this approach can be used in complex software system, including health-care applications to improve functionality, such as activity recognition and task reallocation [Panisson et al. 2015] among agents representing human users, who collaborate to take care of a patient, by detecting if a person responsible for some activity of the patient is following his scheduled appointments; detecting problems, that may prevent the person in charge to attend to his obligations and send warning to the system.

2. Plan Recognition

Plan recognition can be defined as the task of recognising the intentions of an agent based on the available evidence, that is, agent actions, explicit statements about intentions, and agent preferences [Kautz and Allen 1986]. Plan recognition focuses on mechanisms for recognising the unobservable state of an agent, given observations of its interaction with its environment. In other words, a plan recognition system must have a mechanism that is capable of inferring agent intentions by observing the agent actions in the environment. This mechanism retrieves, from a given set of observations, one or more hypotheses of the agent's current plan of action. The practical knowledge used to infer plans is domain dependent and, therefore, is commonly specified beforehand for each specific domain. This domain dependent information is usually encoded as two parts of inputs for the recogniser: a set of observed actions and a set of plans and goals. More specifically, the inputs

to a plan recogniser are a set of goals that the recogniser expects the agent to carry out in the domain, a set of plans describing the way in which the agent can reach each goal, and a sequence of actions currently being performed by the observed agent (i.e, observations of agent actions). Thus, the plan recognition process itself consists in inferring the agent plan and determining how the observed actions contribute to performing it.

Symbolic plan recognition is a type of plan recognition mechanism that narrows the set of candidate intentions by eliminating the plans that are incompatible with current agent actions. Plans make up a plan library and can include preconditions, effects, and sub-goals. Generally, symbolic approaches assume that the observer has complete knowledge of the agent's possible plans and goals. Symbolic approaches handle the problem of plan recognition by determining which set of goals is consistent with the observed actions. Algorithms to recognise the intentions and plans executed by autonomous agents have long been studied in the Artificial Intelligence field under the general term of *plan recognition*. Kautz and Allen [Kautz and Allen 1986] focus on symbolic methods providing a formal theory of plan recognition. Usually, these approaches specify a plan library as an action hierarchy in which plans are represented as a plan graph with top-level actions as root nodes, and plan recognition is then reduced to a graph covering problem. The plan recognition process attempts to find a minimal set of top plans that explain the observations. For a good overview of plan recognition in general, see Carberry [Carberry 2001], and for the most recent research in the field of plan, intention, and activity recognition, see Sukthankar et al. [Sukthankar et al. 2014].

3. Symbolic Plan Recognition

The *Symbolic Plan Recognition* (SBR) [Avrahami-Zilberbrand and Kaminka 2005] is a method for complete, symbolic plan recognition that uses a plan library, which encodes agent knowledge in the form of plans. SBR extracts coherent hypotheses from a multi-featured observation sequence using a *Feature Decision Tree* (FDT) to efficiently match these observations to plan steps[1] in a plan library. An FDT is a decision tree, where each node represents an observable feature and each branch represents one possible value of this feature. Determining all matching plans from a set of observations features is efficiently achieved by traversing the FDT top-down until a leaf node is reached. Each leaf node is a pointer to a plan step in the plan library.

A plan library is represented by a single-root directed acyclic connected graph, which includes all possible plans that an observed agent may execute. The term plan is used here in a broader sense, representing behaviours, reaction plans, and recipes. Typically, a plan library has a single root node in which its children are top-level plans and all other nodes are simply plan steps. Furthermore, in a plan library, *sequential edges* specify the expected temporal order of a plan execution sequence and *decomposition edges* decompose plan steps into alternative sub-steps. The plan library has no hierarchical cycles. However, plans may have a (sequential) self-cycle, allowing a plan step to be executed during multiple subsequent time stamps. Each agent action generates a set of conditions on observable features that are associated with that action. When these conditions are included, the observations match particular plan steps. Figure 1 shows a plan library example based on Activities of Daily Living (ADL), which is a term used

[1] In this paper, we use "plan step" as a synonym for "action".

in health-care to refer to daily self-care activities of people. This plan library shows sequential links represented by dashed arrows and decomposition links represented by solid arrows. For instance, there is a decomposition link between *managing-medication* and *getting-up*, and a sequential link between *getting-up* and *using-bathroom*. The top-level plans are *managing-medication*, and *leisure*. Figure 1 does not show the set of conditions on observable features associated with plan steps, and circled numbers denote time stamps (e.g., *using-bathroom* has been considered a hypothesis at time stamp 2). The decomposition edges are shown only to the first child plan. Thus, the path *root* $\rightarrow$ *managing-medication* $\rightarrow$ *having-lunch* $\rightarrow$ *at-kitchen* $\rightarrow$ *taking-medication* can be a hypothesis for the current plan being executed by an observed agent.

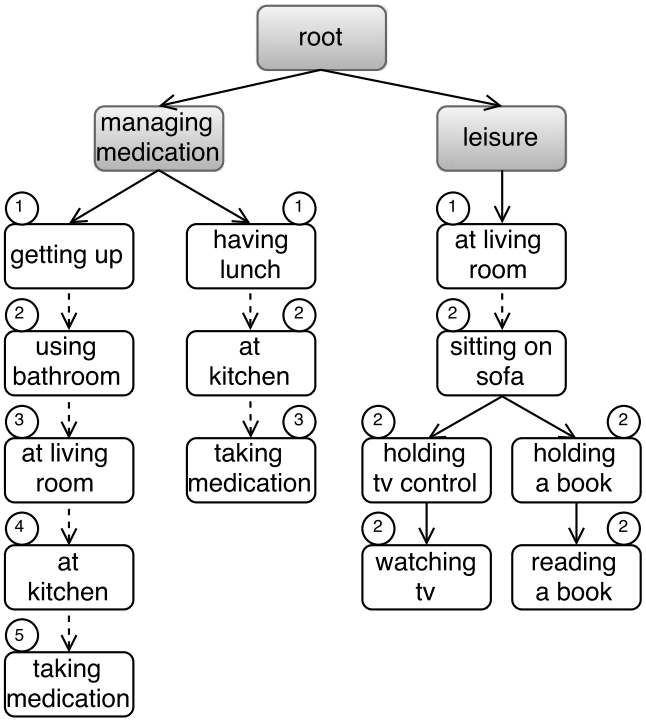


Figure 1. Example of plan library based on ADL.

4. Failure Predictor Components

The failure predictor is responsible for predicting plan failures during execution of an agent goal, more specifically, it tracks the execution of a goal and attempts to identify elements which can lead it to fail, for example, an action taking significantly more time than expected to conclude. The failure predictor is composed of three modules: the `SBR` module, which is responsible for recognising the plan that is being executed by the observed agent; an `Appointment Controller` module, which checks if the goals that are known to have been assigned to the agent are being executed as scheduled; and a `Plan-Step Controller` module, that checks if the plan steps (that compose the plan) are being executed as expected. These modules are presented, respectively, in Subsections 4.1, 4.2, and 4.3.

4.1. SBR Component

The `SBR` component implements the symbolic recogniser presented in Section 3. It is responsible for recognising the plan that an agent is currently executing, and it generates hypotheses about possible plans while the recognition is still not possible. This information is represented both as a list of candidate plans and as a hypotheses graph. As input, the `SBR` component receives *observations*, i.e., sets of contextual information about the observed agent and its actions. Examples of observations include the agent's global positioning coordinates, whether or not the agent is moving, or whether the agent is approaching a particular place, and any other contextual information that can be generated by an activity recognition process. As output, this component provides both the list of candidate plans and the hypotheses graph.

4.2. Appointment Controller

A plan library contains all known plans (agent goals) for a given domain, together with the sequence of actions that compose them, however, it does not define the time that an agent is expected to execute each plan, neither does it contain the time interval in which the plans have to be executed. These are essential information for the system to ensure that plans are being executed in an appropriate manner and to be able to detect potential failures in plan execution. The `Appointment Controller` component implements a simplified calendar which manages the agent goals and plans, it defines which plans of the plan library an agent is known to be responsible for, and at which time the agent is expected to execute some of them (to the extent that this is known in particular domains). This component also helps in disambiguation of candidate plans and in the early prediction of plan failures. It should be noted that only domain-related plans are kept in this individual calendar. Each entry (i.e., agent goal) in the `Appointment Controller` is composed of the following information:

- **Starting date** – Date in which the goal or plan is expected to be started;
- **Ending date** – Date in which the goal or plan is expected to end;
- **Title** – Title of goal or plan (e.g., *"managing-medication"*);
- **Description** (Optional) – Brief textual description of the goal (e.g., *"take medicine on time"*);
- **Related Plan ID** – Unique identifier of the relevant plan (i.e., the plan to achieve this goal), which corresponds to a top plan in the plan library (e.g., *"id:p1"*);
- **Tolerance** – Margin of error for the beginning and end times of each goal, e.g., some goals can start or end 5 minutes before (or after) the time for which it was originally scheduled without danger of the plan failing. This tolerance is necessary because, in real-world situations, goals usually do not start (or end) at the exact scheduled time.

Regarding schedule times, both the starting and ending dates of the goals are composed of day, month, year, hour, and minute (smaller units such as second, for example, are not necessary). The *tolerance* interval can be expressed in various time measures (e.g., hours or minutes).

4.3. Plan-Step Controller

The `Plan-Step Controller` monitors and analyses the plan execution (sequence of actions) to detect anomalies that can lead the plan to fail. The information necessary

for the `Plan-Step Controller` to operate is obtained through the `SBR` component, which provides information about the current plan and actions being performed; the plan library; and the file that contains information about expected time for each action execution. The plan library contains the known plans and the actions which need to be performed in a given plan for it finish successfully, besides the sequence in which these actions must be performed for a plan to be considered completed. However, it does not define when a plan should finish, neither the time in which actions must be executed. This type of information is important to detect anomalous behaviour during plan execution, such as:

- **Plan Interruption** – The plan execution is interrupted without all actions being completed;
- **Time Exceeded** – When an action takes significantly more time than expected, this typically leads to plan failure (e.g., being in a traffic jam);
- **Inconsistent Sequence** – The sequence of observed actions is inconsistent with the expected plan path in the plan library.

It is important to keep track of the actions being performed in order to be able to predict whether a plan is following the expected execution path. Thus, it is possible to identify a probable failure in plan execution and generate the required warnings according to failure type. The entry in file with data about expected execution time of each action is composed of the following information:

- **Plan-Step ID** – Unique identifier of related action, which corresponds to a plan step in the plan library (e.g., *"p1.11"*);
- **Label** – A label for identification the action (plan step) from the plan library (e.g., *"taking-medication"*);
- **Time** – Time that an action often take to be performed;
- **Tolerance** – A value whereby the ending time of the action is allowed to be delayed. For instance, an action can take 5 minutes in addition to its normal time to be performed. This tolerance is important for a real-world situation where actions can often take more time to be performed than an exact specified time.

5. Component Integration

The failure predictor components are integrated as presented in Figure 2. When the `SBR` (Section 3) is not able to determine the current plan (no plan or multiple plans were recognised) the `Appointment Controller` component is consulted (using the output of the plan recogniser). First, the component checks if there are plans scheduled for the moment in which it was consulted and, later, if a scheduled plan is in the list of candidate plans. During this verification the following situations might occur:

- There is no agent goal scheduled for the current time. In this situation, the `Appointment Controller` component has nothing to do, so the main cycle ends and the system awaits for a new observation;
- There is a plan scheduled for the current time, however, the candidate plan list is empty (no plans were recognised). In this case, the `Appointment Controller` component detects a failure in the scheduled goal execution (i.e., the goal that was expected to be executed at time the calendar was consulted) and a warning must be sent to the system, which should be able to handle this plan failure;

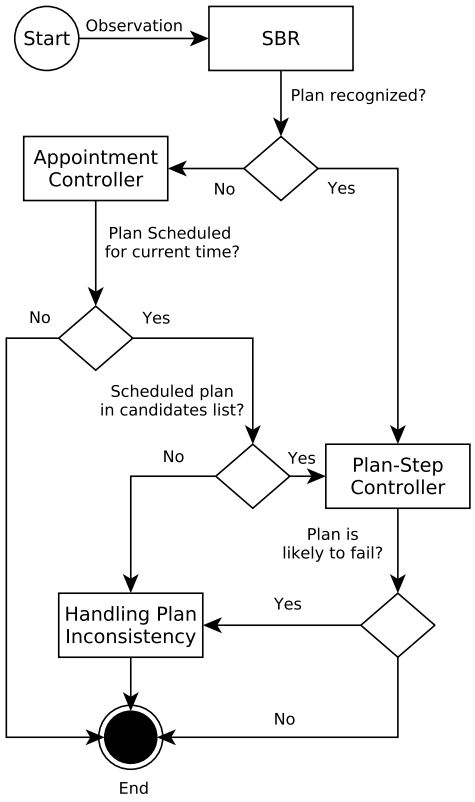


Figure 2. Components Integration.

- There is a plan scheduled for the current time and there are multiple plans in candidate plans list. In this case, the `Appointment Controller` verifies if the plan related to scheduled goal is present in plan candidate list. During this verification two situations might happen:
  - The plan related to the scheduled goal is in candidate plans list, thus, the referred plan is assumed to be the one that is currently being executed by agent;
  - The plan related to the scheduled goal is not in candidate plans list. In this situation, a failure is detected in the scheduled goal execution as it is not being executed by the agent as it should. Then, a warning related to the scheduled goal is sent to the `Handling Plan Inconsistency` step of the main cycle, in which the system will handle the plan related to the failing goal.

Regarding, goal scheduling in this initial implementation does not allow overlapping of goals (i.e., goals with coincident time intervals). That is, a new goal will not be added to calendar if it overlaps with an existing one. However, there is an exception for the overlapping rule regarding the starting and ending times. Two goals (A and B) are not considered as overlapping if the starting time of A is equal to the ending time of B. This exception is convenient, as sequential plans are usually scheduled with no time interval among them, e.g., A (1:00 p.m. to 2:00 p.m.) and B (2:00 p.m. to 4:00 p.m.) or A (4:00 p.m. to 5:00 p.m.) and B (2:00 p.m. to 4:00 p.m.).

The `Plan-Step Controller` is unable to disambiguate the list of candidate plans, thus, both `SBR` and `Appointment Controller` must inform only one goal and one plan step in each iteration with it. The `planController` (Algorithm 1) is part

of this component and responsible for handling such information, using the plan library and the data file with the plan step running times, in order to analyse the plan sequence (based on plan steps) and monitor the running time of each action. In this manner, it is possible to detect and report possible changes in plan execution in order to avoid possible failures.

Algorithm 1
`planController`(Current Goal g, Current PS p, List l)

```
1: Get plan step duration from l;
2: analyseCurrentGoal(g);
3: analyseCurrentPlanStep(p);
4: Inform possible failure;
```

Monitoring of the current plan is performed by the `analyseCurrentGoal` (Algorithm 2). Initially, the current goal is updated based on information received by `SBR` (Line 1) and a test is performed to check if it is a valid value (Line 2), after that, the algorithm checks if the current goal is equal to the previous goal (Line 5) that the agent was trying to achieve. If they are the same, it means agent is still carrying out the actions to achieve it, otherwise, the agent started to perform a new goal with a new plan. In this case, the algorithm has to verify if the previous goal was achieved successfully (Line 11) and the information in the plan library is used to check if the last plan step (of the previous goal) is a leaf node. If this is the case, it means that the plan was fully executed and probably has finished successfully, otherwise, the agent may have stopped performing the plan before its end or the agent is executing more than one plan at the same time, thus, the algorithm should send a warning about this possible failure (Line 15).

Algorithm 2
`analyseCurrentGoal`(Current Goal g)

```
1: current_goal ← g;
2: if current_goal = null then
3:    No goal can be checked;
4: else
5:    if current_goal = previous_goal then
6:       Goal g keeps running;
7:    else
8:       if previous_goal = null then
9:          Goal g started;
10:      else
11:         if previous_goal has finished in a leaf node then
12:            previous_goal finished and g started;
13:         else
14:            Goal g started;
15:            previous_goal stopped before ending;
```

The `analyseCurrentPlanStep` (Algorithm 3) is responsible for monitoring the execution of each action related to a goal, i.e., it analyses the sequence of execution

and the run time of each plan step. Initially, the current plan step is updated based on information received by `SBR`. The consistency of this information is checked and the algorithm then checks if the current plan step is equal to the previous plan step which the agent was performing (Line 5). If they are equal, it means that the agent is still performing the same plan step, thus, the algorithm has to check if the current action is within the time specified in the data file with the plan step running times. The `checkExecutionTime` (Line 7) receives as input the current plan step and checks if its run time is within the specified time, taking into account the specified tolerance for each action. If the agent is taking too long to perform some action, the algorithm detects this as unexpected behaviour and warns the system.

Algorithm 3
`analyseCurrentPlanStep`(Current Plan Step p)

```
1:  current_plan_step ← p;
2:  if current_plan_step = null then
3:     No plan step can be checked;
4:  else
5:     if current_plan_step = previous_plan_step then
6:        current_plan_step keeps running;
7:        checkExecutionTime(current_plan_step);
8:     else
9:        if isValidSequence(previous_plan_step, current_plan_step) then
10:          Current execution path is right;
11:       else
12:          Current execution path has changed;
13:       Update current_goal start time;
14:       previous_plan_step ← current_plan_step;
```

When the failure predictor detects a new action being performed by an agent, it is necessary to check if this action is part of the sequence of actions needed to accomplish the current goal (Line 9). The `isValidSequence` (Algorithm 4) uses the information in the plan library to check if the current plan step is part of a valid sequence of actions to achieve the current goal, receiving as input the previous plan step and the current plan step. If the current plan step contains a sequential parent node and this parent node is the previous plan step, it means the execution path is correct, otherwise, the current plan step does not match the execution path done so far to achieve the current goal.

Detecting whether the sequence of execution is valid is more complicated when the current plan step has a decomposition parent node, because the previous plan step does not have to be a parent node of the current plan step, but only be part of the current plan execution and follow the temporal order of the execution path. The `isPreviousNode` (Algorithm 5) algorithm receives as input the previous plan step and the current plan step. It checks the entire running sequence from current plan step node to previous plan step node in order to analyse the temporal order to determine if the current plan step is a valid sequence for current goal execution.

Algorithm 4
`isValidSequence`(Parent Node $parent$, Child Node $child$)

```
1: if child has a sequential parent then
2:    seq_parent ← child sequential parent;
3:    if seq_parent ≠ parent then
4:       return false;
5:    else
6:       return true;
7: else if child has a decomposition parent then
8:    dec_parent ← child decomposition parent;
9:    if dec_parent = parent then
10:      return true;
11:   else
12:      if isPreviousNode(parent, child) then
13:         return true;
14:      else
15:         return false;
16: else
17:   return false;
```

Algorithm 5
`isPreviousNode`(Parent Node $parent$, Child Node $child$)

```
1: if child has a sequential parent then
2:    seq_parent ← child sequential parent;
3:    if seq_parent = parent then
4:       return true;
5:    else
6:       return isPreviousNode(parent, seq_parent);
7: else if child has a decomposition parent then
8:    dec_parent ← child decomposition parent;
9:    if dec_parent = parent then
10:      return true;
11:   else
12:      return isPreviousNode(parent, dec_parent);
13: else
14:   return false;
```

6. Experiments

The objective of experiments is to show how our approach provides helpful reminders by monitoring and anticipating plan failure from agent observations, in this way, we model part of a scenario that represents agent behaviour based on Activities of Daily Living. These activities correspond to user single actions (e.g., *getting-up*, *watching-tv*, *reading-a-book*, *taking-medication*, *using-bathroom*), in this scenario, there is a person with disabilities who lives alone and needs constant monitoring to perform his daily activities, using plan library formalism, we model a set of plans for representing possible behaviour of this person, where some of these plans are shown in Figure 1.

To exemplify how our approach works, we schedule the top-level plan *managing-medication* to be performed between 7:00 a.m. and 7:30 a.m., in which, this schedule information is in the calendar (`Appointment Controller`). Considering the current part of the day as early morning, the sequence of activities to be performed in order to accomplish the plan *managing-medication* is: *getting-up* → *using-bathroom* → *at-living-room* → *at-kitchen* → *taking-medication*. According to this sequence, the user must move through the living room (i.e., plan step *at-living-room*) to complete the plan, however, this plan step is also part of the top-level plan *leisure*, in this case, the `SBR` component returns both top-level plans *managing-medication* and *leisure* when the current plan step is *at-living-room*. To deal with this ambiguity, our approach uses the `Appointment Controller` component to check if there is a top-level plan scheduled for the current time, if so, it discards those plans that are not scheduled for this time.

```
1 ...
2 [Info]:Current Top-Level Plan: managing-medications
3 [Info]:Current Plan Step (PS): at-living-room
4 [Info]:Checking time [at-living-room]
5 [Info]:PS [at-living-room] started at 7:15am
6 [Info]:PS [at-living-room] is running for 5 minutes
7 [Info]:PS [at-living-room] average time 3 minutes | tolerance 1 minute
8 [Warn]:PS [at-living-room] is taking too long
9 ...
```

Listing 1. Example of Failure Predictor Output.

The program output, presented in Listing 1, represents part of execution of the failure predictor approach, in a scenario where the user must take a medication in a strict time and during the plan execution the user's attention drawn to something else (e.g., the user stays at living room watching TV) and forgets to take his medication. In this case, the current top-level plan is *managing-medication* (Line 2) and the current plan step is *at-living-room* (Line 3). The failure predictor monitors the plan execution (Lines 4-6) and based on information in the `Plan-Step Controller`, i.e., average time of execution for each plan step and a time tolerance (Line 7), it is able to detect anomalies in plan execution and informs the system to try to address and correct these possible failures. In this example, the algorithm detects that a plan step is taking too long to be performed, then an alert message is generated (Line 8).

7. Conclusion

In this paper, we have developed a failure predictor based on plan recognition techniques and a calendar that includes *some* of the plans that the agent is known to be required to execute over time. Our main contribution is a system that anticipates plan failures by monitoring a sequence of agent actions during its plan execution. We have used this predictor as part of a system to support collaborative work in a scenario where family members and professional carers support an elderly person with a debilitating disease who lives alone. Although we currently only deal with plan failure *prediction*, as future work we can enable our system to elaborate alternative plans to avoid the detected failures (e.g., using planning techniques) rather than simply warning about possible plan failures.

Acknowledgements

Part of the results of this paper were obtained through the research project entitled "Semantic and Multi-Agent Technologies for Group Interaction", sponsored by Samsung Eletrônica da Amazônia Ltda. under the terms of Brazilian federal law No. 8.248/91.

References

Armentano, M. G. and Amandi, A. (2007). Plan recognition for interface agents. *Artificial Intelligence*, 28(2):131–162.

Avrahami-Zilberbrand, D. and Kaminka, G. A. (2005). Fast and complete symbolic plan recognition. In Kaelbling, L. P. and Saffiotti, A., editors, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 653–658. Professional Book Center.

Carberry, S. (2001). Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48.

Fox, M. and Long, D. (2003). PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20(1):61–124.

Geib, C. W. (2002). Problems with intent recognition for elder care. In *Proceedings of the AAAI-02 Workshop Automation as Caregiver*, pages 13–17.

Geib, C. W. and Goldman, R. P. (2003). Recognizing plan/goal abandonment. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 1515–1517, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Goldman, R. P., Geib, C. W., and Miller, C. A. (1999). A new model of plan recognition. *Artificial Intelligence*, 64:53–79.

Hong, J. (2001). Goal recognition through goal graph analysis. *Journal of Artificial Intelligence Research*, 15:1–30.

Kautz, H. A. and Allen, J. F. (1986). Generalized plan recognition. In Kehler, T., editor, *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI-86)*, pages 32–37. Morgan Kaufmann.

Masato, D. (2012). *Incremental Activity and Plan Recognition for Human Teams*. PhD thesis, University of Aberdeen.

Mausam and Weld, D. S. (2008). Planning with durative actions in stochastic domains. *Journal of Artificial Intelligence Research*, 31(1):33–82.

Panisson, A. R., Freitas, A., Schmidt, D., Hilgert, L., Meneguzzi, F., Vieira, R., and Bordini, R. H. (2015). Arguing About Task Reallocation Using Ontological Information in Multi-Agent Systems. In *12th International Workshop on Argumentation in Multiagent Systems*.

Pynadath, D. V. and Wellman, M. P. (1995). Accounting for context in plan recognition, with application to traffic monitoring. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, pages 472–481, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Sukthankar, G., Goldman, R. P., Geib, C., Pynadath, D. V., and Bui, H. H., editors (2014). *Plan, Activity, and Intent Recognition: Theory and Practice*. Elsevier.

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

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

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

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

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

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

1. Introdução

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

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

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

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

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

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

2. Trabalhos Relacionados

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

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

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

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

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

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

3. Materiais e Métodos

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

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

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

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

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

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

4. Análise e discussão

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

a) Construtividade

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

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

b) Suporte a análise comparativa

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

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

c) Integração

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

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

d) Maior nível de abstração

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

e) Mecanismos estruturantes com mais recursos

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

f) Escopo estendido

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

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

g) Separação de interesses

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

h) Técnicas leves

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

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

i) Especificação Multiparadigma

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

j) Análise polivalente

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

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

k) Especificação multiformato

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

l) Raciocínio apesar de erros

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

m) Feedback construtivo a partir de ferramentas

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

n) Suporte a evolução

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

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

o) Suporte ao reuso

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

p) Mensurabilidade de progresso

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

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

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

5. Conclusão

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

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

Referencias

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Um Jogo Dramático baseado na Teoria do Drama para Autorregulação de Processos de Trocas Sociais

Renata G. Wotter , Lucas T. B. Costa , Nelson de F. Traversi , Diana F. Adamatti , Graçaliz P. Dimuro

[1]C3 – Universidade Federal do Rio Grande (FURG)
Rio Grande – RS – Brazil

{renata.wotter, dianaada, gracaliz}@gmail.com,

{lucastubino, nelsontraversi}@furg.br

Abstract. *This paper presents a dramatic model for self-regulation of social exchange processes in multiagent systems, based on the concepts of Drama Theory. The model has five phases of dramatic resolution, which involve feelings, emotions, trust and reputation. Agents with different social exchange strategies interact each other in order to improve the quality of the interactions, through the equilibrium of their social exchange processes. The objective is to obtain a more natural model than the ones existing in the literature, which are based on (partially observable) Markov decision processes or in game theory, so that it can be applied in real-world applications. We aim at promoting more balanced and fair multiagent interactions, increasing the number of successful social exchanges and, thus, promoting the continuity of social exchanges.*

Resumo. *Este artigo apresenta um modelo dramático para autorregulação de processos de trocas sociais, baseado nos conceitos da Teoria do Drama. O modelo possui cinco fases de resolução dramática, as quais envolvem sentimentos, emoções, confiança e reputação. Os agentes com diferentes estratégias de trocas sociais interagem uns com os outros, buscando melhorar a qualidade das interações, com o equilíbrio dos processos de trocas sociais. O objetivo é obter um modelo mais natural que os existentes na literatura, baseados em processos de decisão de Markov (parcialmente observáveis) ou em teoria de jogos, de modo que possa ser aplicado no mundo real. Busca-se interações multiagentes mais balanceadas e justas, aumentando o número de trocas sociais bem sucedidas e, assim, promover a continuidade das trocas sociais.*

1. Introdução

A Teoria das Trocas Sociais de Piaget [Piaget 1995] tem sido utilizada como base para a análise de interações em Sistemas Multiagentes (SMA). Tais interações são denominadas trocas de serviços, as quais são avaliadas pelos agentes que interagem, gerando valores de trocas sociais, que são qualitativos e subjetivos [Dimuro et al. 2005]. Um problema fundamental neste contexto é o da regulação de trocas sociais [Grimaldo et al. 2007, Rodrigues 2007, Pereira et al. 2008, Dimuro et al. 2011, Macedo et al. 2014, Dimuro and da Rocha Costa 2015, Von Laer et al. 2015], no sentido de permitir, p.ex., a emergência de trocas balanceadas, levando ao equilíbrio social

[Piaget 1995] e/ou comportamento de justiça [Rabin 1993, Xianyu 2010]. Particularmente, este é um problema difícil quando agentes, adotando diferentes estratégias de trocas sociais, possuem informação incompleta sobre as estratégias de trocas dos demais agentes. Este é um problema crucial em sociedades de agentes abertas [Dimuro et al. 2011, Dimuro and da Rocha Costa 2015].

Em trabalhos anteriores [Dimuro et al. 2007, Pereira et al. 2008, Dimuro and da Rocha Costa 2015], Dimuro et al. e Pereira et al. introduziram diferentes modelos para o problema de regulação de troca social, com base em modelos híbridos de agentes BDI [1] e processos de decisão de Markov (parcialmente observáveis). Em [Macedo et al. 2014], Macedo et al. introduziram o Jogo de Autorregulação de Processos de Trocas Sociais (JAPTS), onde, os agentes, possuindo diferentes estratégias de trocas sociais, evoluem suas estratégias de troca ao longo do tempo por si só, promovendo interações mais equilibradas e justas, garantindo a continuidade das trocas. Em [Von Laer et al. 2015], Von Laer et al. analisaram o problema da autorregulação de processos de trocas sociais no contexto de um SMA baseado em agentes BDI, adaptando o jogo JAPTS para agentes Jason [Bordini et al. 2007] e introduzindo o aspecto cultural, onde a cultura da sociedade, agregando reputação dos agentes como crenças de grupo, influencia diretamente a evolução das estratégias de troca dos agentes, aumentando o número de interações de sucesso e melhorando o resultado dos agentes nas interações.

Em Teoria dos Jogos [Leyton-Brown and Shoham 2008], geralmente, um jogo definido pelas preferências e oportunidades dos jogadores é dado como fixo. Em 1991, Nigel Howard criou a Teoria do Drama [Howard 1994a, Howard 2006], uma extensão de teoria dos jogos, onde as preferências e escolhas dos personagens (jogadores) mudam sob a pressão das negociações pré-jogo. A teoria dos jogos tenta prever o resultado de um jogo com jogadores "racionais". No entanto, a teoria do drama mostra como aspirantes a jogadores, comunicando-se uns com os outros antes de um jogo, constroem não só o jogo que jogarão, mas também o resultado que esperam dele, sem a necessidade de prever um resultado. Além disso, a teoria do drama desafia o conceito teórico de jogo de "racionalidade". Ao analisar a comunicação pré-jogo, é descartada a hipótese de que os jogadores sabem o que querem, o que os outros querem, e o que eles e os outros podem fazer sobre isso, e que todas estas coisas são fixas [Howard 2006].

O objetivo deste artigo é propor um modelo dramático para autoregulação de processos de trocas sociais, aplicando os conceitos da teoria do drama ao jogo JAPTS, acrescentando sentimentos e expressões de emoções baseadas no modelo OCC [Ortony et al. 1988], para obter um modelo natural que se aproxima da realidade, de modo que possa ser utilizado em aplicações. O artigo está organizado como a seguir. A Seção 2 resume a base teórica do trabalho. A Seção 3 define o modelo dramático e uma prova de conceito da modelagem realizada. A Seção 4 é a Conclusão.

2. Referencial Teórico: a teoria das trocas sociais e a teoria do drama

De acordo com Piaget [Piaget 1995], uma troca social é qualquer sequência de ações entre dois sujeitos, tal que um dos sujeitos, pela realização de suas ações, preste um serviço para o outro, com a imediata avaliação qualitativa individual dos serviços prestados. Isto é, o

[1] BDI ("Beliefs, Desires, Intentions"), é um modelo de agente cognitivo introduzido em [Rao and Georgeff 1991].

agente atribui um valor ao seu investimento na realização de um serviço para outro agente e este último atribui um valor de satisfação por ter recebido tal serviço. Tais valores são chamados de valores de troca material. Em um processo de troca social, são gerados valores de débito e crédito, que permitirão a realização de trocas futuras. Débito e crédito são chamados de valores virtuais.

Uma troca social entre agentes envolve dois agentes i e j em duas etapas/estágios de trocas. Na **Etapa I** o agente i realiza um serviço para o agente j, sendo gerados os seguintes valores de troca: r_{ij} (valor de *Investmento* do agente i), s_{ji} (valor de *Satisfação* do agente i), t_{ji} (valor de *Débito* do agente j) e v_{ij} (valor de *Crédito* do agente i). Na **Etapa II** o agente i solicita para o agente j o pagamento do serviço realizado anteriormente para ele, e os valores gerados são análogos. Um processo de trocas sociais é uma seqüência destas etapas de trocas, em qualquer ordem. O equilíbrio é obtido quando o balanço dos valores de trocas para cada agente é em torno de um valor aceitável pela sociedade de agente, geralmente, em torno de zero.

Diferentemente da Teoria dos Jogos, que considera que o jogo é definido previamente pelas preferências e oportunidades dos jogadores, a Teoria do Drama [Howard 1990, Howard 1994a, Howard 1994b, Howard 2006] é uma teoria de como o próprio jogo pode mudar: como o dado jogo G pode ser transformado em um outro jogo G', o qual, por sua vez, pode ser transformado em G", etc. Estas transformações resultam de pressões que os jogadores colocam um sobre o outro durante as negociações pré-jogo, como eles trocam ameaças, promessas, persuasão emocional e argumentação racional.

A teoria do drama contribui para identificar mudanças causadas pela dinâmica interna das negociações pré-jogo, que descrevem processos racionais e irracionais de desenvolvimento humano e auto-realização, ao invés de apenas a escolha racional de de um determinado fim. Enquanto a teoria dos jogos expõe o comportamento racional, dirigido a objetivos, a teoria do drama mostra como, no curso de uma interação, as pessoas mudam e se desenvolvem. A racionalidade continua sendo importante, mas já não domina.

3. O Modelo Dramático de Autorregulação de Processos de Trocas Sociais

O modelo dramático de autorregulação de processos de trocas sociais proposto baseia-se nas cinco fases de resolução dramática da Teoria do Drama, que são representadas na Figura 1.

3.1. Fase 1: Definição de Cena

Nesta fase, o ambiente é definido com os personagens (agentes), estratégias de troca social dos agentes, reputação dos agentes, resultados e consequências.

As estratégias de troca social consideradas neste artigo são altruísmo, altruísmo fraco, egoísmo, egoísmo fraco e racionalidade. Por exemplo, um agente com a estratégia egoísmo é mais propenso a depreciar o serviço recebido e supervalorizar o serviço ofertado, que impacta nos valores de débito e crédito; o agente racional joga apenas pelo Equilíbrio de Nash [2]. As estratégias de trocas sociais são determinadas por vários fatores, como será explicado a seguir, mas, particularmente, pelo valor de investimento máximo

[2]Ver [Macedo et al. 2014] para uma discussão sobre o Equilíbrio de Nash do Jogo de Processos de Trocas Sociais.

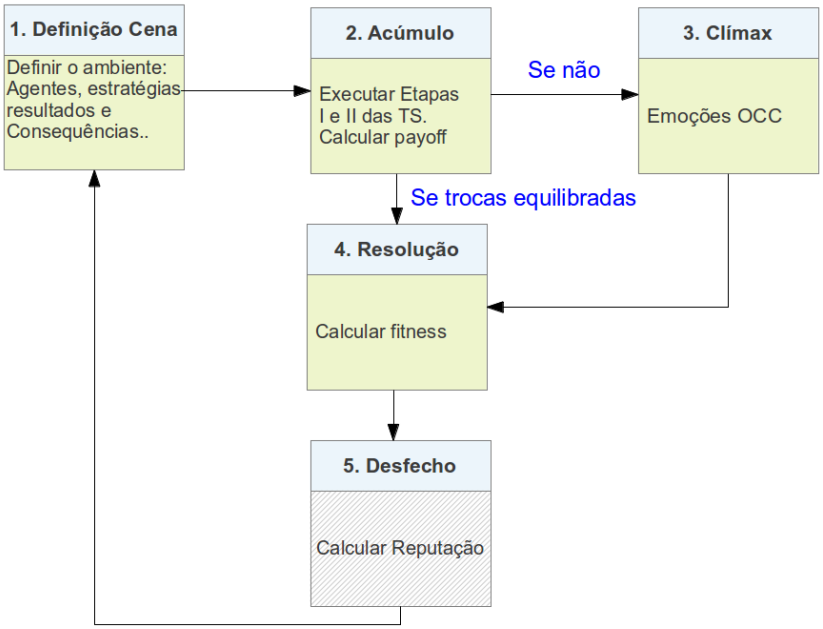


Figure 1. Fases da Resolução Dramática.

r^{max} que os agentes estão dispostos a fazer por um serviço realizado a outro agente, e o valor de satisfação mínima s^{min} que estão dispostos a aceitar, com $r, s \in [0, 1]$.

A consequência é representada por uma função $Q : X \rightarrow X$, onde X é o conjunto dos resultados individuais dos personagens ou agentes, ou seja, das suas estratégias. Os resultados individuais são um par composto pela aspiração (um futuro particular que gostaria de conseguir) e uma posição (um futuro que ele propõe aos outros).

No modelo dramático, o resultado de um agente é representado pelo investimento proposto (r^{prop}), ou seja, o futuro que ele propõe aos demais agentes; e pela satisfação esperada (s^{esp}), ou seja, a aspiração, um futuro particular que gostaria de alcançar.

3.2. Fase 2: Trocas

Nesta fase, que na Teoria do Drama é chamada de Acúmulo, um determinado frame $F = (Q, P)$ é selecionado. Onde, Q é o conjunto de resultados de cada agente e $P = (P_i | i \in C)$ é uma família de relações de preferência, uma para cada personagem ou agente i do elenco C, definida ao longo do conjunto X de resultados. Neste jogo, $(x, y) \in P_i$ significa que "o agente i prefere a estratégia x à estratégia y.

Após selecionado o frame, são executadas as etapas **I** e **II** das Trocas Sociais, como apresentado na Seção 2. Uma estratégia de troca social de um agente $\lambda = i, j$, é definida pela tupla:

$$(r_\lambda^{max}, r_\lambda^{prop}, r_\lambda^{efet}, s_\lambda^{min}, s_\lambda^{esp}, k_\lambda^{\rho t}, k_\lambda^{\rho v}), \quad (1)$$

onde: $r_\lambda^{max} \in [0, 1]$ e $s_\lambda^{min} \in [0, 1]$ representam os valores de investimento máximo que o agente λ fará por um serviço oferecido para outro agente, e o valor da satisafação

Table 1. Parâmetros das estratégias de trocas sociais

Estratégia	r^{max}	r^{prop}	r^{efet}	s^{min}	s^{esp}	a	b	c	d	$k^{\rho}t$	$k^{\rho}v$
Altruísmo	[0.8; 1]	[0.75; 0.99]	[0.8; 1]	[0.3; 0.49]	[0.3; 0.51]	0.8	0.8	0.2	0.2	0.2, $\rho = o$	0.2, $\rho = d$
Altruísmo Fraco	[0.61; 0.79]	[0.55; 0.75]	[0.55;0.79]	[0.5; 0.6]	[0.52; 0.65]	0.6	0.6	0.4	0.4	0.1, $\rho = o$	0.1, $\rho = d$
Egoísmo	[0.3; 0.49]	[0.25; 0.35]	[0.2;0.35]	[0.8; 1]	[0.85; 1]	0.2	0.2	0.8	0.8	0.2, $\rho = d$	0.2, $\rho = o$
Egoísmo Fraco	[0.5; 0.6]	[0.35; 0.55]	[0.34;0.6]	[0.61; 0.79]	[0.65; 0.85]	0.3	0.3	0.7	0.7	0.1, $\rho = d$	0.1, $\rho = o$
Racionalidade	[0; 0.29]	[0; 0.29]	[0; 0.29]	[0; 0.29]	[0; 0.29]	0	0	0	0	0	0

mínima que o agente λ espera pelos serviços recebidos, respectivamente; $r_{\lambda}^{prop} \in [0,1]$ e $s_{\lambda}^{esp} \in [0,1]$ são o investimento proposto que o agente λ fará por este serviço e a satisfação esperada pelo outro agente, respectivamente; $r_{\lambda}^{efet} \in [0,1]$ é o valor de investimento efetivo que o agente λ fará, pois dependendo da estratégia escolhida pelo agente, este investimento pode ser maior, menor ou igual ao proposto; $k_{\lambda}^{\rho t}, k_{\lambda}^{\rho v} \in [0,1]$ são, respectivamente, fatores de depreciação $(\rho = d)$ ou supervalorização $(\rho = o)$ de débito e crédito que caracterizam cada estratégia de troca. Portanto, existe um investimento máximo que o agente pode realizar, o investimento que o agente propõe e o investimento que o agente realmente efetua. Tais valores são apresentados na Tabela 1.

Nesta fase, são calculados o Payoff Suposto (*payoffSup*) e o Payoff Efetivo (*payoffEfet*) da troca social entre os agentes i e j, com as respectivas estratégias de troca:

$$(r_i^{max}, r_{ij}^{prop}, r_{ij}^{efet}, s_i^{min}, s_i^{esp}, k_i^{\rho t}, k_i^{\rho v}) \text{ e } (r_{ji}^{max}, r_{ji}^{prop}, r_{ji}^{efet}, s_j^{min}, s_j^{esp}, k_j^{\rho t}, k_j^{\rho v})$$

O *payoffSup* obtido nesta interação é avaliado pela função $p_{ij}^{sup} : [0,1]^4 \rightarrow [0,1]$, definido por:

$$p_{ij}^{sup} = \begin{cases} \dfrac{1 - r_{ij}^{prop} + s_{ij}^{esp}}{2}, & \text{se}(r_{ij}^{prop} \leq r_i^{max} \wedge s_{ji}^{esp} \geq s_j^{min}) \wedge (r_{ji}^{prop} \leq r_j^{max} \wedge s_{ij}^{esp} \geq s_i^{min}) \\ \dfrac{1 - r_{ij}^{prop}}{2}, & \text{se}(r_{ij}^{prop} \leq r_i^{max} \wedge s_{ji}^{esp} \geq s_j^{min}) \wedge (r_{ji}^{prop} > r_j^{max} \vee s_{ij}^{esp} < s_i^{min}) \\ 0, & \text{se}(r_{ij}^{prop} > r_i^{max} \vee s_{ji}^{esp} < s_j^{min}) \wedge (r_{ji}^{prop} > r_j^{max} \vee s_{ij}^{esp} < s_i^{min}) \end{cases} \quad (2)$$

O *payoffEfet* obtido nesta interação é avaliado pela função $p_{ij}^{efet} : [0,1]^4 \rightarrow [0,1]$, definido por:

$$p_{ij}^{efet} = \begin{cases} \dfrac{1 - r_{ij}^{efet} + s_{ij}^{esp}}{2}, & \text{se}(r_{ij}^{efet} \leq r_i^{max} \wedge s_{ji}^{esp} \geq s_j^{min}) \wedge (r_{ji}^{efet} \leq r_j^{max} \wedge s_{ij}^{esp} \geq s_i^{min}) \\ \dfrac{1 - r_{ij}^{efet}}{2}, & \text{se}(r_{ij}^{efet} \leq r_i^{max} \wedge s_{ji}^{esp} \geq s_j^{min}) \wedge (r_{ji}^{efet} > r_j^{max} \vee s_{ij}^{esp} < s_i^{min}) \\ 0, & \text{se}(r_{ij}^{efet} > r_i^{max} \vee s_{ji}^{esp} < s_j^{min}) \wedge (r_{ji}^{efet} > r_j^{max} \vee s_{ij}^{esp} < s_i^{min}) \end{cases} \quad (3)$$

O *payoffSup* e o *payoffEfet* do agente j são definidos de forma análoga.

Considerando um ambiente composto pelo elenco $C = 1, ..., m$ de m agentes, cada agente $i \in C$ interage com os outros $m-1$ agentes vizinhos $j \in C$, tal que $j \neq i$. A cada ciclo de interação, cada agente i avalia seus resultados materiais de troca social local com cada agente vizinho j, utilizando as funções de *payoffSup* e *payoffEfet* locais dadas nas Equações (2) e (3). Em seguida, o *payoffSup* e o *payoffEfet* totais recebidos por cada agente são calculados após cada agente ter realizado as duas etapas de troca com toda a sua vizinhança. Para p_{ij}^{sup} e p_{ij}^{efet} calculados pelas Equações (4) e (5), a alocação do *payoffProp* e do *payoffEfet* totais de uma vizinhança de m agentes é dada por:

$$X^{sup} = x_1^{sup}, ..., x_m^{sup}, \text{where } x_i^{sup} = \sum_{j \in C, j \neq i} p_{ij}^{sup} \quad (4)$$

$$X^{efet} = x_1^{efet}, ..., x_m^{efet}, \text{where } x_i^{efet} = \sum_{j \in C, j \neq i} p_{ij}^{efet} \quad (5)$$

Após calcular o *payoff efetivo* das trocas, é analisado o equilíbrio destas trocas. Idealmente, uma troca equilibrada é quando a divergência entre os payoffs de todas as trocas é nula. Porém, na prática, esta divergência se dá em torno de zero. Esta divergência entre os payoffs é calculada conforme a Equação 6:

$$D_i = \frac{1}{(m-1)} \sum_{i \neq j}^{j=[1..m]} |x_i - x_j| \leq \alpha \quad (6)$$

Onde, m é o número total de agentes e α é o fato de divergência.

Portanto, considera-se trocas equilibradas quando $D_i \leq \alpha$ para todas as trocas.

3.3. Fase 3: Climax

Se na fase 2 todas as trocas ocorrem de forma equilibrada, a fase 3 é ignorada e o jogo segue para a fase 4, onde as emoções terão peso nulo no cálculo do valor de *fitness*, definido por $\mathrm{F}_i(X^{efet})$ de um agente i.

Caso uma das trocas não seja equilibrada, todos os agentes migram para a fase 3. São consideradas quatro emoções do modelo OCC [Ortony et al. 1988]: gratificação, gratidão, remorso e raiva, representadas por $a_\lambda, b_\lambda, c_\lambda$ e d_λ, respectivamente. Observa-se que no modelo OCC existem três aspectos que alteram as reações do mundo: eventos, agentes e objetos. Os eventos são interessantes porque é possível analisar suas consequências, os agentes porque é possível analisar suas ações, e objetos porque os aspectos e propriedades destes objetos são analisados. As emoções escolhidas são parte de um grupo que foca na ação de um agente e nas consequências dos eventos [Adamatti and Bazzan 2003].

Esta fase é executada apenas uma vez, não obtendo necessariamente o equilíbrio das trocas, mas as emoções que influenciarão no cálculo do fitness da fase 4. O equilíbrio das trocas será alcançado com a autorrgeulação das trocas sociais.

Uma *estratégia de troca social espacial* de um agente $\lambda, \lambda = 1, ..., m$ é definida pela tupla:

$$(r_\lambda^{prop}, r_\lambda^{efet}, s_\lambda^{esp}, a_\lambda, b_\lambda, c_\lambda, d_\lambda, k_\lambda^{\rho t}, k_\lambda^{\rho v}), \quad (7)$$

onde $a_\lambda, b_\lambda, c_\lambda, d_\lambda$ refletem a influência das emoções no valor de fitness $\mathrm{F}_i(X^{efet})$ de um agente i, da seguinte forma:

- **Gratificação (a_i)**

$$\mathrm{F}_i(X^{efet}) = x_i^{efet} + a_i.max(x_i^{efet} - x_i^{sup}, o)$$

onde X^{efet} é a alocação do payoff efetivo total do agente i.

Gratificação é um sentimento positivo gerado no próprio agente que propôs a troca, quando o *payoff efetivo (payoffEfet)* do agente que praticou a troca (x_i^{efet}), obtido através do investimento efetivo, é maior que o *payoff suposto (payoffSup)*, obtido pelo investimento proposto por este agente (x_i^{sup}). Isto significa que, ao praticar um valor maior que o valor proposto, o agente se sente mais confiante e também gera um sentimento recíproco de gratidão no outro agente.

- **Gratidão (b_i)**

$$\mathrm{F}_i(X^{efet}) = x_i^{efet} + \frac{b_i}{(m-1)} \sum_{j \neq i} \max(x_j^{efet} - x_j^{sup}, 0)$$

onde X^{efet} é a alocação do payoff efetivo total do agente i.
Gratidão é um sentimento positivo gerado no agente que recebeu o serviço, quando o *payoff efetivo (payoffEfet)* do agente que praticou a troca (x_j^{efet}) é maior que o *payoff* que ele supunha receber no grupo (x_j^{sup}). Ao receber um valor maior que o proposto, o agente se sente grato ao agente que praticou o serviço, gerando uma boa reputação neste agente.

- **Remorso (c_i)**

$$\mathrm{F}_i(X^{efet}) = x_i^{efet} - c_i.max(x_i^{sup} - x_i^{efet}, o)$$

onde X^{efet} é a alocação do payoff efetivo total do agente i.
Remorso é um sentimento negativo gerado no próprio agente que propôs a troca, quando o *payoff efetivo (payoffEfet)* do agente que praticou a troca (x_i^{efet}) é menor que o *payoff suposto(payoffSup)*, obtido pelo investimento proposto por este agente (x_i^{sup}). Este sentimento gera um sentimento recíproco de raiva no outro agente, e consequentemente ficará com uma má reputação perante ao outro agente.

- **Raiva (d_i)**

$$\mathrm{F}_i(X^{efet}) = x_i^{efet} - \frac{d_i}{(m-1)} \sum_{j \neq i} max(x_j^{sup} - x_j^{efet}, o)$$

onde X^{efet} é a alocação do payoff efetivo total do agente i.

Raiva é um sentimento negativo gerado no agente que recebeu o serviço, quando o *payoff efetivo (payoffEfet)* do agente que praticou a troca (x_j^{prat}) é menor que o *payoff* que ele supunha receber no grupo (x_j^{sup}).
Diante disso, percebe-se que o equilíbrio é alcançado quando as emoções antagônicas se anulam.

3.4. Fase 4: Resolução

Após a execução das etapas **I** e **II** do processo de trocas sociais na fase de Acúmulo, havendo um equilíbrio, o jogo avança para a fase 4. Nesta fase, considerando o payoff efetivo obtido na fase 2, o agente i calcula seu grau de adaptação através de sua função *fitness* $\mathrm{F}_i : [0,1]^m \rightarrow [0,1]$, definida por:

$$\mathrm{F}_i(X^{efet}) = x_i^{efet}$$

onde X^{efet} é a alocação do payoff efetivo total do agente i.

Caso tenha sido executada a fase 3, as emoções geradas são acrescentadas à função *fitness*, representando a influência destas nos resultados do *payoffEfet* total dos agentes.

Seja X^{efet} a alocação do *payoffEfet* total de uma vizinhança de m agentes. A definição geral da função *fitness*, baseada em estratégia de trocas U_i de um agente i, é dada por:

$$F_i(X^{efet}) = \tag{8}$$
$$x_i + a_i max(x_i^{efet} - x_i^{sup}, 0) + \frac{b_i}{(m-1)} \sum_{j \neq i} max(x_j^{efet} - x_j^{sup}, 0) -$$
$$-c_i max(x_i^{sup} - x_i^{efet}, 0) - \frac{d_i}{(m-1)} \sum_{j \neq i} max(x_j^{sup} - x_j^{efet}, 0)$$

3.5. Fase 5: Desfecho

Após obter o valor da função *fitness*, a fase 5 é executada. Nesta fase, a reputação dos agentes é calculada. Para as ciências sociais, as reputações são definidas como um coletivo de crenças e opiniões que influenciam as ações dos indivíduos em relação aos seus pares. A reputação pode ainda ser vista como uma ferramenta social com o objetivo de reduzir a incerteza de se interagir com indivíduos de atributos desconhecidos. Para [Marsh 1994], reputação é geralmente definida como a quantidade de confiança inspirada por uma determinada pessoa em um ambiente ou domínio específico de interesse.

Dentro da ciência da computação, reputação e confiança tem ganhado crescente evidência nos últimos anos, especialmente na área da *Inteligencia Artificial Distribuída* (IAD), onde os Sistemas Multiagentes estão incluídos. Confiança e reputação são utilizadas como um meio de busca de parceiros. A reputação tem o poder de propagar a confiança e pode evitar que os agentes interajam desnecessariamente. Ver [Sabater and Sierra 2005, Sabater and Sierra 2002, Huynh et al. 2006, Yu et al. 2014].

[Rodrigues et al. 2015] desenvolveram um modelo de reputação baseado nos modelos REGRET [Sabater and Sierra 2001] e Hübner [Hübner et al. 2009]. A análise da reputação é divida em três dimensões: Dimensão Social, Dimensão Individual e Dimensão Ontológica, como proposto no modelo REGRET. Na Dimensao Social é analisada a efetividade do agente para com o seu grupo social; já na Dimensão Individual são analisadas as trocas diretas entre os agentes. Por fim, tem-se a Dimensão Ontológica, onde Dimensão Social e Individual se combinam para uma análise final.

Para este modelo dramático, utilizou-se o modelo de reputação proposto por [Rodrigues et al. 2015], considerando até o momento apenas a dimensão individual. Nesta fase 5, os *payoffs efetivos* obtidos através das trocas sociais serão armazenados em uma lista de tamanho v. O cálculo da reputação é dado por:

$$Rep = \frac{\sum_{j \in C, j \neq i} p_{ij}^{efet}}{size(v)} \tag{9}$$

Com as informações obtidas na fase de desfecho, o jogo retorna para a fase 1, onde poderá redefinir o ambiente a partir das novas estratégias, isto é, a partir da reputação

Table 2. Estratégias de Troca Social adotadas pelos agentes.

	r^{max}		r^{prop}		r^{efet}		min		s^{esp}		s^{efet}	
A_{ij}	i	j	ij	ji	ij	ji	i	j	ij	ji	ij	ji
A_{12}	1	0.8	0.8	0.7	0.85	0.75	0.51	0.6	0.55	0.65	0.75	0.85
A_{13}	1	0.6	0.75	0.5	0.75	0.45	0.51	0.8	0.55	0.9	0.45	0.8
A_{21}	0.8	1	0.7	0.8	0.75	0.85	0.6	0.51	0.65	0.55	0.85	0.75
A_{23}	0.8	0.6	0.7	0.45	0.72	0.5	0.6	0.8	0.62	0.85	0.5	0.75
A_{31}	0.6	1	0.55	0.75	0.55	0.75	0.8	0.51	0.85	0.51	0.8	0.6
A_{32}	0.6	0.8	0.5	0.7	0.5	0.7	0.8	0.6	0.85	0.62	0.85	0.55

Table 3. Valores Totais de *payoffSup*, *payoffEfet* e *fitness* das Trocas Sociais.

	x^{sup}	x^{efet}	F
A_1	0.77	0.57	0.382
A_2	0.93	0.55	0.218
A_3	1.32	0.62	-0.172
X^{sup}	3.02		
X^{efet}	1.74		

calculada, os agentes escolherão novos parceiros para realizar as trocas sociais, dando continuidade ao jogo. Antes do próximo ciclo, um novo *fitness* é calculado, como na Equação 10.

$$F^2 = F + / - (Rep.\beta) \tag{10}$$

onde, F é a função *fitness* calculada na fase 4, e β é o percentual de ajuste.

Se a reputação do agente é maior ou igual à média total dos payoffs, ou seja, é uma boa reputação, então o valor resultante de $(Rep.\beta)$ é somado ao valor do *fitness* anterior. Caso contrário, o agente está com uma má reputação, portanto, subtrai-se o valor do *fitness* anterior.

Antes de reiniciar o jogo, os agentes analisam os resultados do seu *fitness* anterior e *fitness* atual, ajusta sua estratégia de jogo de acordo com um vetor de ajuste com 27 probabilidades, aumentando, diminuindo ou mantendo constante os valores de investimento que pretende realizar, investimento máximo que pretende ofertar e a menor satisfação aceitável, r^{efet}, r^{max} e s^{min}, respectivamente. Então, os agentes redefinirão o ambiente a partir de novas estrtatégias, ou seja, os agentes escolherão novos parceiros para executar as trocas sociais, e iniciar a próxima rodada do jogo. Este processo é repetido em cada ciclo da simulação.

3.6. Prova de Conceito do Modelo Proposto

Considerando três agentes, A_1, A_2 e A_3. O Agente 1 tem uma estratégia **Altruísmo**, o Agente 2 tem uma estratégia **Altruísmo Fraco** eo Agente 3 tem uma estratégia **Egoísmo**. A Tabela 2 apresenta os valores de investimento e satisfação para cada troca social desses três agentes, de acordo com suas estratégias.

Após realizar as trocas socials entre os agentes, *payoffSup* and *payoffEfet* são calculados usando as Equações (2) and (3), obtendo os valores mostrados na Tabela 4. Obtidos estes valores, calcula-se os valores totais do *payoffSup* e *payoffEfet* para cada troca social, conforme a Tabela 3.

Visto que as trocas sociais entre os agentes 2 e 3 não foram equilibradas, a fase 3 (Clímax) do jogo é realizada e as emoções são consideradas na fase 4 para calcular a função *fitness* através da Equação (8). Os resultados da fase 4 são apresentados na Tabela 4.

Table 4. Valores calculados nas fases 2 e 4 do jogo.

			Etapas de Troca para p^{efet}			
A_{ij}	p^{sup}	p^{efet}	**Etapa I**	**Etapa II**	**0 etapas**	$fitness$
A_{12}	0.37	0.45		x		0.58
A_{13}	0.4	0.12	x			0.04
A_{21}	0.47	0.55		x		0.65
A_{23}	0.46	0			x	-0.45
A_{31}	0.65	0.62		x		0.37
A_{32}	0.67	0			x	-0.9

Table 5. Análise do Processo de Trocas Sociais.

	Gratificação	**Gratidão**	**Remorso**	**Raiva**	**Why?**
A_{12}	x	x			Nas Etapas I e II $r^{efet} > r^{prop}$ $s^{efet} > s^{esp}$
A_{13}			x	x	Na Etapa II $r^{efet} < r^{prop}$ $s^{efet} < s^{min}$
A_{21}	x	x			Nas Etapas I and II $r^{efet} > r^{esp}$ $s^{efet} > s^{esp}$
A_{23}			x	x	Nas Etapas I and II $s^{efet} < s^{esp}$
A_{31}			x	x	Na Etapa II $r^{efet} = r^{esp}$ $s^{efet} < s^{esp}$
A_{32}			x	x	Nas Etapas I and II $s^{efet} < s^{esp}$

A Tabela 5 apresenta a conclusão analítica de cada processo de troca social entre os Agentes 1, 2 e 3, considerando as emoções envolvidas.

Ao calcular o *fitness* total (F) dos agentes em relação ao grupo, observa-se que nenhum dos três agentes sentiu ou causou emoções positivas (gratidão e gratificação). Comparando os resultados obtidos entre pares, percebe-se que no grupo as emoções negativas prevalecem, anulando as emoções positivas.

4. Conclusão

Foi introduzido o modelo de um jogo dramático de autorregulação de processos de trocas sociais. No mundo real, as trocas sociais não se dão de forma exclusivamente racional, frequentemente envolvendo sentimentos e emoções. Sendo assim, surgiu a possibilidade de aplicar a teoria do drama ao jogo de autorregulação de processos de trocas sociais.

Aplicando os conceitos da teoria do drama e inserindo o modelo de confiança e reputação ao modelo dramático desenvolvido, espera-se obter um jogo de simulação de trocas sociais em um ambiente que se aproxime do mundo real.

O jogo dramático está sendo implementado em NetLogo, e até o momento tem-se as fases 1, 2, 3 e 4 do modelo dramático. A fase 5 (confiança e reputação) e a evolução das estratégias ao longo das interações está sendo implementada. A Figura 2 apresenta o ambiente do jogo, onde é possível configurar o cenário do jogo através das estratégias, verificar a quantidade de trocas realizadas, bem como o resultado da função *fitness*, considerando a fase 3 (Clímax) ou não.

Após o término da implementação, serão realizadas simulações com diferentes composições de sociedade de agentes e cenários para analisar a evolução das estratégias e dos processos de trocas sociais ao longo do tempo.

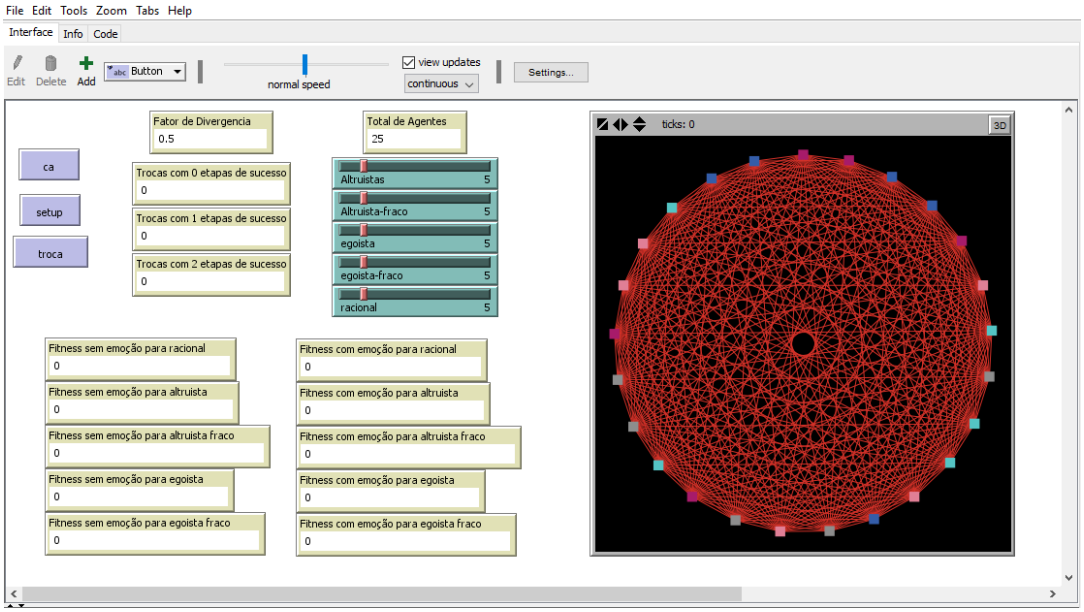


Figure 2. Ambiente do Jogo Dramático.

References

Adamatti, D. F. and Bazzan, A. (2003). Afrodite - ambiente de simulação baseado em agentes com emoções. In *Proceedings of ABS 2003 - Agent Based Simulation*, Montpellier.

Bordini, R. H., Hübner, J. F., and Wooldrige, M. (2007). *Programming Multi-agent Systems in AgentSpeak Using Jason*. Wiley Series in Agent Technology. John Wiley & Sons, Chichester.

Dimuro, G. P., Costa, A. C. R., Gonçalves, L. V., and Hübner, A. (2007). Centralized regulation of social exchanges between personality-based agents. *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, v.4386 of *LNCS*, p. 338–355. Springer, Berlin.

Dimuro, G. P., Costa, A. C. R., and Palazzo, L. (2005). Systems of exchange values as tools for multi-agent organizations. *Journal of the Brazilian Computer Society*, 11:27–40.

Dimuro, G. P., Costa, A. R. C., Gonçalves, L. V., and Pereira, D. (2011). Recognizing and learning models of social exchange strategies for the regulation of social interactions in open agent societies. *Journal of the Brazilian Computer Society*, 17:143–161.

Dimuro, G. P. and da Rocha Costa, A. C. (2015). Regulating social exchanges in open MAS: The problem of reciprocal conversions between POMDPs and HMMs. *Information Sciences*, 323:16 – 33.

Grimaldo, F., Lozano, M., and Barber, F. (2007). Coordination and sociability for intelligent virtual agents. *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, v. 4870 of *LNCS*, p. 58–70. Springer, Berlin.

Hübner, J. F., Vercouter, L., and Boissier, O. (2009). Instrumenting multi-agent organisations with artifacts to support reputation processes. In *Coordination, Organizations, Institutions, and Norms in Agent Systems IV*, v. 8386 of *LNCS*, p. 96–110. Springer, Berlin.

Howard, N. (1990). Soft game theory. *Information and Decision Technologies*, 16:215–227.

Howard, N. (1994a). Drama theory and its relation to game theory. part 1: Dramatic resolution vs. rational solution. *Group Decision and Negotiation*, 3(2):187–206.

Howard, N. (1994b). Drama theory and its relation to game theory. part 2: Formal model of the resolution process. *Group Decision and Negotiation*, 3(2):207–235.

Howard, N. (2006). *What is Drama Theory?* (acessed on January, 2016).

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

Leyton-Brown, K. and Shoham, Y. (2008). *Essentials of game theory: A concise, multidisciplinary introduction*. Morgan & Claypool, California.

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. *21st ECAI 2014, Proceedings*, number 263 in Frontier in Artificial Intelligence and Applications, p. 573–578, Netherlands. IOS Press.

Marsh, S. (1994). *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling.

Ortony, A., Clore, G. L., and Collins, A. (1988). *The Cognitive Structure of Emotions*. Cambridge University Press, Cambridge.

Pereira, D., Gonçalves, L., Dimuro, G. P., and Costa, A. R. C. (2008). Towards the self-regulation of personality-based social exchange processes in multiagent systems. *SBIA 2008*, v. 5249 of *LNCS*, p. 113–123. Springer, Berlin.

Piaget, J. (1995). *Sociological Studies*. Routlege, London.

Rabin, M. (1993). Incorporating fairness into game theory and economics. *The American Economic Review*, 86(5):1281–1302.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. *Proceedings og the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, p. 473–484, San Mateo. Morgan Kaufmann.

Rodrigues, H. D. N., Adamatti, D. F., and Dimuro, G. P. (2015). Modelagem de agentes BDI-Fuzzy submetidos ao processo de reputação. In *X Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações*, p. 143, Niterói. UFF.

Rodrigues, M. R. (2007). *Social techniques for effective interactions in open cooperative systems*. PhD thesis, University of Southampton, Southhampton.

Sabater, J. and Sierra, C. (2001). Regret: A reputation model for gregarious societies. In *Proceedings of the Fourth Workshop on deception Fraud and Trust in Agent Societies*, p. 61–70.

Sabater, J. and Sierra, C. (2002). Reputation and social network analysis in multi-agent systems. In *Proceedings of the First International Conference on Autonomous Agents and Multiagents Systems, AAMAS 2002*, p. 475–482. ACM.

Sabater, J. and Sierra, C. (2005). Review on computational trust and reputation models. *Artif. Intell. Rev.*, 24(1):33–60.

Von Laer, A., Dimuro, G. P., and Adamatti, D. F. (2015). Analysing the influence of the cultural aspect in the self-regulation of social exchanges in MAS societies: An evolutionary game-based approach. *Progress in Artificial Intelligence*, v. 9273 of *LNCS*, p. 673–686. Springer.

Xianyu, B. (2010). Social preference, incomplete information, and the evolution of ultimatum game in the small world networks: An agent-based approach. *Journal of Artificial Societies and Social Simulation*, 13:2.

Yu, H., Miao, C., An, B., Shen, Z., and Leung, C. (2014). Reputation-aware task allocation for human trustees. In *Proceedings of the 13th AAMAS – 2014*, p. 357–364, New York. IFAAMAS/ACM.

Um sistema multiagente com leilões para a seleção de pacientes numa clínica odontológica.

Rodrigo Rabenhorst[1], Cleo Zanella Billa[2]

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

rodrigoraben@gmail.com[1], cleobilla@furg.com[2]

Abstract. *This article discusses the use of a multi-agent system (MAS) with auctions in order to improve the patient trial process in an academic clinical dentistry. The clinic is composed of 13 units, which one with its own speciality. We propose a MAS model where each unit is represented by an agent and they use auctions to negotiate and decide who is going to attend the patient. Each unit bids the auction based on its own availability and speciality.*

Resumo. *Esse artigo propõe a utilização de um Sistema Multiagente (SMA) com leilões para melhorar o processo de triagem de uma clínica acadêmica odontológica. A clínica é composta por 13 unidades, cada uma com sua especialidade. Nesse trabalho, é proposto um modelo onde cada unidade é representada por um agente e elas negociam o atendimento a um paciente através de leilões. Cada unidade calcula o seu lance baseado na sua disponibilidade e especialidade.*

1. Introdução

O atendimento hoje na clínica odontológica, Jequittí III da Universidade Federal de Pelotas, é feito considerando-se uma seleção de pessoas segundo suas necessidades, as quais são avaliados por um profissional e alunos. Após, estes são encaminhados ao preenchimento das vagas de atendimento cujo critério principal é a urgência e o segundo preencher a lacuna das agendas das clínicas em um curto período, como este método é de um período muito curto, muitas unidades acabam perdendo pacientes que potencializariam as aulas práticas do curso de odontologia da Universidade Federal de Pelotas e deixam de prestar um serviço mais apropriado a comunidade.

Este trabalho tem como objetivo propor um modelo de triagem, para melhorar o atendimento dos pacientes que chegam a clínica, buscando diminuir o tempo de espera de um paciente para tratamento e aumentar o atendimento especializado. O modelo propõe o uso leilões como método de negociação entre as unidades, representadas por agentes, de atendimento.

[Fiani 2006], apresenta, mediante abstrações, como funciona o processo na tomada de decisões dos agentes que interagem entre si, baseado na compreensão da lógica de cada situação onde estão envolvidos. Além disso, o estudo da teoria dos jogos, propicia

desenvolver a capacidade de raciocinar estrategicamente, explorando as possibilidades de interação de tais agentes.

Quando feita a decomposição do problema, ficou claro a necessidade de um sistema colaborativo e que tivessem uma interação flexível. Com isto, foi tomada a decisão de modelar um SMA, aonde possível fosse explorar as áreas de conhecimento e a disponibilidade em cada uma das unidades clínicas disponíveis para atendimento. Visando qualificar o atendimento e facilitando o agendamento destes pacientes.

O artigo está organizado da seguinte maneira: a seção 2 apresenta brevemente os conceitos da teoria dos jogos, leilões e triagem. A seção 3 apresenta os trabalhos relacionados que buscaram soluções a problemas semelhantes ao problema enfocado neste artigo. Já a seção 4 expõe uma visão sobre o cenário atual da Clínica Jequittí III e propõe o modelo para transformar o processo de triagem num sistema SMA. Finalmente, a seção 5 aponta as considerações finais.

2. Conceitos básicos

[Wooldridge e Jennings 1995] e [Garcia e Sichman 2003] definem agente como uma entidade real ou virtual, imersa num ambiente sobre o qual ela está apta a agir, e disposta de uma capacidade de percepção e representação parcial deste ambiente, que pode se comunicar com outros agentes, e denota um comportamento autônomo – consequência de suas observações – de seu conhecimento e das suas interações com os demais agentes.

Segundo [Girardi 2004], os sistemas multiagente são utilizados para a construção de sistemas complexos, e a sua arquitetura permite esquematizar as propriedades e a estrutura de interação entre agentes, garantindo a funcionalidade deste sistema.

Nesta seção, são apresentados os conceitos teóricos fundamentais para este trabalho, tais como: sistema multiagente, teoria dos jogos e leilões.

2.1 Sistemas multiagente

Conforme [Cunha e Wotter e Rabenhorst 2015], a área de sistemas multiagente (SMA) estuda a inserção de agentes autônomos em um universo multiagente. O objetivo final é garantir que os agentes interagem para solucionar um problema. Em um SMA, os agentes e suas interações devem ser projetados com o propósito de se tornarem independentes do contexto da aplicação-alvo. Assim, desenvolve-se um projeto reutilizável onde os componentes podem ser empregados em aplicações similares.

De acordo com [Girardi 2004], um SMA com política de cooperação precisa que agentes expressem as suas necessidades aos demais a fim de realizar uma determinada tarefa. Tal processo de cooperação ocorre por meio de tarefas compartilhadas - auxílio mútuo nas tarefas entre eles mesmos e transmitindo os resultados à comunidade.

2.2 Teoria dos Jogos

[Fiani 2006], com o uso da teoria dos jogos, apresenta mediante abstrações, como funciona o processo de tomada de decisão dos agentes que interagem entre si, considerando a lógica de cada situação na qual estão envolvidos. O estudo da teoria dos

jogos vai ainda mais longe, porque estuda a capacidade de raciocinar estrategicamente explorando as possibilidades de interações dos agentes, possibilidades estas que nem sempre correspondem à intuição.

Na teoria da escolha racional, busca-se entender como os jogadores tomam suas decisões em situações de interação estratégica. É preciso conhecer as preferências dos envolvidos objetivando nortear as suas escolhas. Assim, eles compartilham suas intenções, procurando um vínculo de associação entre os elementos [Fiani 2006].

2.3 Leilões

[Fiani 2006] define leilão como um modo de negociação definido por uma série de regras visando especificar a forma de determinação do vencedor e da quantia a ser paga por ele. Uma característica marcante dos leilões é a presença de assimetria de informações, tornando necessária sua caracterização. Justifica-se tal necessidade pois diferentes tipos deste mecanismo podem ocasionar resultados divergentes.

Leilões podem também ser definidos quanto à sua natureza (oferta, demanda ou duplicidade), quanto à forma dos lances (aberto ou fechado) e quanto ao método de determinação do valor de fechamento (primeiro ou segundo preço). Além disso, pode ou não ser determinado um preço de reserva, equivalente ao menor lance válido para a participação no leilão, ou ainda ser usado sequencialmente (*multi-round*), por meio de procedimento interativo de atualização de lances a cada interação [Fiani 2006].

2.4 Triagem de Pacientes

A triagem é o processo do qual os pacientes são separados por ordem de acordo com a sua condição. Para a clínica em questão esta seleção é feita para preencher as vagas de atendimento das disciplinas práticas do curso de odontologia da Universidade Federal de Pelotas. O processo é feito da seguinte maneira: os pacientes vão diariamente a procura de atendimento odontológico, passam por uma avaliação com os profissionais e são encaminhados para os atendimentos daquela semana, caso se encaixem nas especialidades e ainda existam vagas. Caso contrário, os pacientes devem retornar na semana seguinte e tentar novamente um atendimento.

Esse método faz com que muitos pacientes sejam perdidos, e a universidade além de precisar destes pacientes, deixa de atender a comunidade carente. Sendo este um problema colaborativo, justifica-se a escolha de um SMA para aprimorar o sistema de seleção de pacientes, visando diminuir a perda de pacientes pelo tempo de espera e buscando atender da melhor forma os pacientes e alunos dentro das clínicas disponíveis.

3. Trabalhos Relacionados

Nesta seção apresenta alguns trabalhos sobre SMA, leilões e triagem explicando o processo de desenvolvimento, bem como os conceitos aplicados, que mostram o uso de

duas modalidades diferente de leilões aplicada a casos semelhantes, os quais serviram para a proposta do modelo de triagem da Clínica Jequittí III.

3.1. Uma Proposta para Leilões Digitais com Multiagente e Negociação Dinâmica

[Carvalho 2006] discorre sobre a implementação de um ambiente de comércio eletrônico, baseado em multiagente, que utiliza cenários simulados da realidade, com a finalidade dos agentes executarem uma variedade de funções típicas do comércio eletrônico.

Segundo [Carvalho 2006], um sistema de comércio eletrônico contempla várias fases que abrangem desde a identificação das necessidades até a determinação dos processos de compra e venda de produtos. Porém esse estudo envolve, somente aquelas da procura por fornecedores. Em relação às compras, os autores aplicam as técnicas de SMA e a prática de leilões.

A fase de procura por fornecedor é feita mediante a busca em vários leilões, cada um representando uma loja em potencial. Desobrigando o cliente a se preocupar em pesquisar informações a respeito das lojas.

A negociação é realizada considerando um processo denominado leilão inglês, ou seja, um mecanismo no qual cada participante é livre para aumentar sua proposta.

Quando encerradas as propostas, o leilão termina e o maior arrematante é o vencedor em razão do preço oferecido. Estratégias de agentes baseiam-se em uma série de lances em função das ofertas pessoais, das suas expectativas sobre os outros arrematantes, e os lances ocorridos no passado.

3.2 Smart Parking: Mecanismo de Leilão de Vagas de Estacionamento Usando Reputação entre Agentes

[Gonçalves e Alves 2015] criaram um ambiente multiagente, com dois tipos de agentes - o agente motorista e o agente estacionamento. O primeiro representa o motorista e seu veículo no sistema e tem o objetivo de interagir no compartilhamento de informações com outros agentes, incluindo o segundo tipo. O agente estacionamento representa o próprio estacionamento, ou seja, existe um agente para cada vaga, e ele é responsável pelo seu próprio gerenciamento por meio de leilões.

O agente motorista determina os objetivos - o local e o global. O primeiro é de interesse próprio desse agente que é conseguir a reserva de uma vaga. Já o segundo diz respeito à cooperação e à organização, da melhor forma possível, do estacionamento, otimizando como um todo a sua operacionalização. Os agentes estacionamento têm por intuito configurar as trocas de mensagens entre si e tomam decisões importantes referentes ao uso do estacionamento levando em conta a reputação dos motoristas quanto ao bom comportamento nesse sistema.

Essas informações subsidiam o aproveitamento do recurso - e delimitam quais os critérios utilizados na escolha do motorista mais adequado a fim de ocupar uma determinada vaga. As que não estão em uso são negociadas pelo agente estacionamento. Elas se encontram organizadas em uma fila prioritária com vistas a serem leiloadas - e a alocação de cada uma para um determinado motorista é feita por um leilão.

Nesses casos, o modelo de leilão utilizado é denominado leilão holandês [Krishna 2009], ou seja, é um tipo de arremate no qual o item a ser leiloado, em razão do preço muito alto, dificilmente é adquirido pelos compradores. Quando não há nenhum interessado o valor diminui e o item passa a ser então novamente leiloado, até encontrar um comprador que aceite a proposta.

3.3 Simulação Baseada em Agentes para Alocação Pessoal em Procedimento de Classificação de Risco na Emergência de um Hospital

Devido ao grande número de pacientes nos setores de emergência dos hospitais públicos, há uma necessidade crítica de lidar com esse incremento, por meio de um sistema de admissão de pacientes que os classifique com agilidade, priorizando o atendimento daqueles que apresentam maior risco. Para tanto, é preciso que existam profissionais qualificados para executar o procedimento de classificação de risco de pacientes, e de uma forma de avaliação do desempenho, que permita a proposição de mudanças, tais como a forma de alocação desses profissionais, para contribuir com a melhoria da qualidade de atenção aos pacientes recém-admitidos [Andrade 2010].

O objetivo principal do trabalho é proporcionar ao administrador hospitalar uma solução para o problema de alocação dos agentes envolvidos, em um procedimento de classificação de risco. Uma simulação social baseada em agentes foi criada, para emulação da dinâmica das equipes nos modelos de classificação de risco, incorporando a interação, a tomada de decisão, e a qualificação dos agentes, permitindo a mensuração do desempenho da equipe no procedimento.

[Andrade 2010], afirma que os sistemas de apoio à decisão baseados em simulação social multiagente são razoavelmente adequados para simular o comportamento de pessoas trabalhando em equipe, como um procedimento de classificação de risco hospitalar. Pode-se dizer que, estes sistemas são importantes no auxílio ao processo decisório da administração hospitalar em ambientes de incerteza.

Para realizar as triagens, o autor modela o procedimento de classificação de risco, através de algoritmos, considerando cada uma de suas tarefas como um processo a ser conduzido por agentes com qualidades, habilidades e experiências, sintetizadas por um índice numérico.

4. O Cenário

A Clínica Jequittí III, situa-se dentro da Universidade Federal de Pelotas, no prédio da Faculdade de Odontologia. Por se tratar de um ambiente acadêmico, no curso há divisões das áreas de conhecimento em treze disciplinas, e em aulas práticas. Estas são ministradas na clínica com a participação de voluntários da comunidade ou de pessoas indicadas pela rede básica de saúde após atendimento em postos de saúde.

4.1. A Clínica

O atendimento hoje na clínica é feito considerando-se uma seleção de pacientes segundo suas necessidades, as quais são avaliados por um profissional e alunos. Os pacientes são encaminhados ao preenchimento das vagas de atendimento cujo critério principal é a urgência. Assim, as unidades, como são chamadas cada uma das especialidades, têm pacientes à espera de tratamento, e os responsáveis pela triagem, tentam evitar vagas ociosas.

Os exames da triagem costumam seguir uma rotina. Inicialmente o paciente entra na sala de triagem, um profissional lhe faz algumas perguntas com o intuito de conhecer o histórico e os hábitos odontológicos. Depois, ele passa por exames, tais como anamnésia, clínico, avaliação de estresse oclusal e periodontal. - Com base nessas informações, o paciente é encaminhado às filas de atendimento de uma das unidades, ainda disponíveis para a semana.

As clínicas possuem data semanal certa para funcionar, assim, o profissional responsável pela triagem tenta preencher primeiro as lacunas da próxima clínica, desconsiderando, portanto, a necessidade de o paciente ser encaminhado a outra clínica mais apropriada.

4.2. O Modelo Gerado para a Clínica

Para criar o primeiro modelo, e reduzir um pouco a complexidade, ao invés de trabalhar com as 13 unidades, foi gerado um modelo que trabalha com apenas três destas unidades, que tratam de restauração dentária e que ocorrem em maior incidência. Estas unidades são as Unidades Odontológicas (UCO) I, II e III. Elas atendem os mesmos casos de dentistícas, e alguns de periodontia e endodontia, mas a posição do dente afetado e a face deste dente é um fator determinante para gerar um lance de maior ou menor interesse pelos agentes.

O sistema de chegada dos pacientes até a parte dos primeiros exames clínicos, continua igual. Mas o responsável pela triagem, insere no sistema o prontuário deste paciente e as informações são lidas pelo agente responsável pelo leilão aplicado ao modelo inglês. Ele informa para as demais agentes unidades todos os problemas encontrados no primeiro exame e estes baseados em suas crenças, desejos e intenções, geram lances e enviam para o agente leiloeiro, que informa a todos os agentes o vencedor com o melhor lance ou informa que não há mais vagas para atendimentos no período de agenda. Ao final o

agente leiloeiro devolve ao usuário do sistema a clínica que irá atender, a data e o horário.

A seguir foi criado baseado na forma como a clínica Jequittí III realiza sua seleção de pacientes, buscando melhorar o atendimento, diminuindo a evasão de pacientes pela demora ou forma como o agendamento ocorre hoje. No modelo proposto a avaliação do paciente continua ocorrendo da mesma forma, porém agora a clínica possui um agendamento que abre vagas para os próximos trinta dias, o que permite ao profissional da triagem oferecer o atendimento mais apropriado para o paciente, deixando que a seleção das vagas disponíveis através dos Agentes que junto as suas crenças, desejos e intenções podem fazer uma avaliação sobre cada prontuário e gerar um lance demonstrando o quanto aquele paciente se enquadra para as suas especialidades.

As vagas e o agendamento de cada unidade é controlada pelos Agentes a quem representam. A Figura 1 apresenta o Diagrama de Caso de Uso do Sistema. Ao chegar um paciente na clínica, antes de qualquer exame é feita uma pesquisa sobre se ainda há vagas disponíveis para atendimentos, independente da especialidade. E só depois com o exame realizado pelo profissional da triagem, o agente leiloeiro recebe o prontuário deste paciente e o envia para os demais agentes para iniciar o leilão. Cada agente unidade calcula o valor do seu lance com base na sua especialidade e na sua agenda.

A forma de leilão escolhida para adaptar a este caso, é o modelo de leilão inglês, muito semelhante ao trabalho de [Carvalho 2006], no qual o lance é determinado através de um processo dinâmico que estabelece o vencedor. Este processo dinâmico ocorre de forma ascendente. Neste leilão, o lance vencedor pode ser anunciado pelo agente leiloeiro, que recebeu um lance baseado nos desejos de cada unidade representada por um agente. A negociação encerra, com a mensagem enviada pelo leiloeiro a todos os agentes, informando o agendamento ou que não há vagas disponíveis para o período. Como os lances são gerados baseados nos desejos e no número de vagas ociosas de cada unidade, o paciente só não consegue uma vaga com o especialista mais indicado.

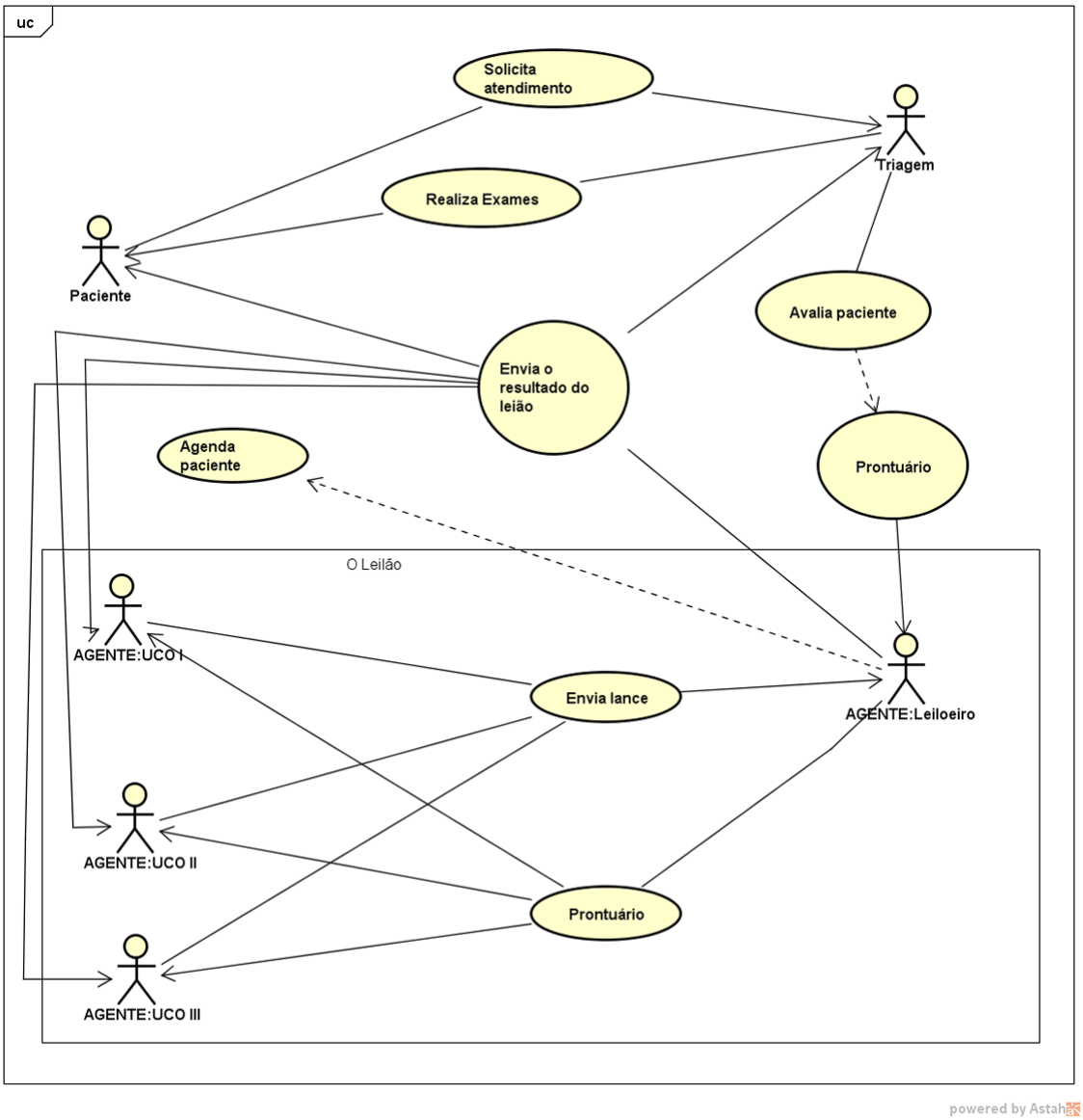


Figura 1. Estudo de Caso do Modelo.

4.2.2 O Diagrama de Sequência

O diagrama de sequência da Figura 2 mostra a comunicação entre os diversos agentes participantes do processo. Na figura são representados os seguintes agentes: Paciente, Triagem (Profissional responsável pela primeira avaliação), Leiloeiro e as unidades clínicas odontológicas, UCOI, UCO II e UCO III.

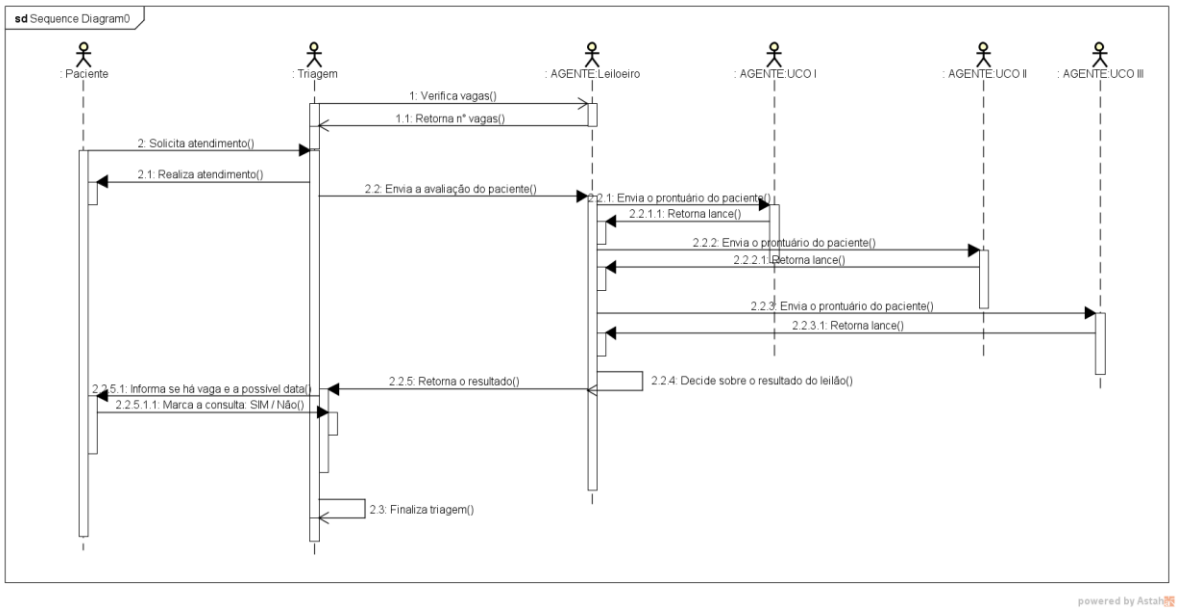


Figura 2. Diagrama de Sequência do Modelo.

Conforme mostra na Figura 2, o profissional responsável pela triagem recebe a informação sobre as vagas disponíveis nas agendas dos agentes UCO I, UCO II, UCO III que representam as unidades especializadas através do Agente leiloeiro, e fica aguardando a chegada de possíveis pacientes. Com a chegada de um paciente, o agente triagem realiza os exames básicos e registra no sistema para que o agente leiloeiro possa receber o prontuário e realizar o leilão. O agente leiloeiro abre o leilão enviando a cada agente unidade o prontuário do paciente. Cada agente unidade calcula o lance baseado no seu interesse em atender o paciente e informar ao agente leiloeiro. Quando todos os agentes unidades enviarem seus lances, o agente leiloeiro encerra o leilão e define o vencedor. O agente leiloeiro comunica a todos os agentes a data, a hora e a unidade que irá atende-lo.

Fluxo Básico:

a) O agente triagem consulta as vagas disponíveis nas clínicas;
b) O agente paciente solicita atendimento;
c) O agente triagem realiza os exames básicos;
d) O agente triagem envia para o agente leiloeiro o prontuário do paciente;
e) O agente leiloeiro abre um leilão e envia para os agentes unidades, o prontuário para avaliação do caso e aguarda como retorno um lance;
f) Os agentes unidades UCO I, UCO II e UCO III enviam seus lances ao agente leiloeiro;
g) O agente leiloeiro envia para todos os envolvidos o resultado final do leilão e o agendamento do paciente com data e hora.
h) O agente da unidade vencedora atualiza sua agenda.

4.2.3 Os Agentes

A seguir são mostrados os agentes do sistema e suas crenças, desejos e intenções iniciais.

Nome do Agente: Leiloeiro
Desejos: Conduzir um leilão.
Crenças: Conhece os agentes que participam do leilão.
Conhece o agente triagem.
Intenções: * Abrir o leilão;
* Receber os lances;
* Encerrar o leilão;

Nome do Agente: UCO I
Desejo: Tratar pacientes com problemas de cavidade oclusal;
Tratar pacientes com problema de face livre;
Ocupar vagas ociosas;
Crenças: Agenda da unidade;
Intenções: * Receber prontuários;
* Gerar um lance baseado nos seus desejos;
* Enviar um lance.

Nome do Agente: UCO II
Desejos: Tratar pacientes com problemas de cavidade oclusal;
Tratar pacientes com problemas de cavidade oclusal e ou mais um ou dois proximais;
Tratar pacientes com problema de face livre.
Ocupar vagas ociosas;
Crenças: Agenda da unidade;
Intenções: * Receber prontuários;
* Gerar um lance baseado nos seus desejos;
* Enviar um lance.

Nome do Agente: UCO III
Desejos: Tratar pacientes com problemas de cavidade oclusal e ou mais dois proximais;
Tratar pacientes com problema de face livre;
Ocupar vagas ociosas;
Crenças: Agenda da unidade;
Intenções: * Receber prontuários;
* Gerar um lance baseado nos seus desejos;
* Enviar um lance.

5. Considerações Finais

Este artigo apresenta um modelo de sistema multiagente que utiliza leilões para realizar a triagem de pacientes para uma clínica odontológica e assim distribuir da melhor forma estes pacientes dentro de suas unidades especialistas.

No modelo, os agentes que representam as unidades possuem em seus desejos e crenças os procedimentos que eles querem e podem atender. As decisões sobre os lances de cada unidade, são formados através das informações dos prontuários e das agendas de cada clínica. Suas intenções podem ser idênticas, mas seus desejos e a sua agenda serão fatores determinantes para gerar o valor do lance a ser ofertado ao agente leiloeiro.

Referências

Andrade, Sylvio Flávio. "Simulação Baseada em Agentes Para Alocação de Pessoal em Procedimento de Classificação de Risco na Emergência de um Hospital". 2010. Tese de Doutorado. Universidade Federal do Rio de Janeiro.

Carvalho, Christiano Guimarães de; Rathie, Aman. "Uma proposta para leilões digitais com multiagente e negociação dinâmica". CEP, v. 70910, p. 900, 2006.

Cunha, Rafhael Rodrigues; Wotter, Renata Gomes; Rabenhorst, Rodrigo Machado. Hardwares e sistemas multiagente: um estudo sobre arquiteturas híbridas. Revista Brasileira de Computação Aplicada, v. 7, n. 2, p. 2-15, 2015.

Fiani, Ronaldo. "Teoria dos Jogos–Com Aplicação em Economia, Administração e Ciências Sociais". 2ª Edição. São Paulo-SP: Campus, 2006.

Garcia, A. C. B. e Sichman, J. S. (2003). Agentes e Sistemas Multiagentes, capítulo 11, páginas 269–306. Barueri, SP: Manole. In Sistemas Inteligentes: Fundamentos e Aplicações, Rezende, S. O., Editor.

Girardi, Rosario. "Engenharia de Software baseada em Agentes. "Procedimentos do IV Congresso Brasileiro de Ciência da Computação (CBComp 2004). 2004.

Gonçalves, Wesley RC; Alves, Gleifer Vaz. "Smart Parking: mecanismo de leilão de vagas de estacionamento usando reputação entre agentes". Universidade Federal do Paraná (UTFPR) – Ponta Grossa, PR. Brasil, 2015, proceedings do WESAAC 15.

Krishna, Vijay. "Auction theory". Academic press, 2009.

Woodridge, Michael; Jennings, Nicholas R. "Agent theories, architectures, and languages: a survey". In: Intelligent agents. Springer Berlin Heidelberg, 1994. p. 1-39.

Tomada de Decisão em Sistemas Multiagente utilizando personalidade e emoções

Gerson A. Urban Filho, Diana F. Adamatti

Centro de Ciências Computacionais - Universidade Federal do Rio Grande
Rio Grande – RS – Brazil

{gersonurb, dianaada}@gmail.com

Abstract. *This paper proposes a model of multi-agent system based on exchanges which can be simulate on each agent, emotions and personality. The aim of this paper is to propose a simulation model that aims to simulate the social behavior from an exchange, where each agent will have its particular personality. This personality will be responsible for influence the agent's decision. Once the agent decides to take action, expectation emotions are generated and subsequently with the action already completed, emotions are generated regarding the completion of this action. These emotions cause changes in personality, allowing the agent's adaptation to the environment they live in and their mode of interaction with it.*

Resumo. *Neste trabalho é proposto um modelo de sistema multiagente baseado em trocas onde é possível simular em cada agente, emoções e personalidade. O principal objetivo deste artigo é propor um modelo que visa simular o comportamento social através de trocas, onde cada agente terá sua própria personalidade. Esta personalidade será responsável por influenciar a tomada de decisão do agente. Uma vez que o agente decide realizar uma ação, são geradas emoções pela expectativa da realização da ação e posteriormente, com a ação já finalizada, são geradas emoções referentes à conclusão desta. Estas emoções provocam variações na personalidade, possibilitando a adaptação do agente ao meio em que vive e seu modo de interação com o mesmo.*

1. Introdução

As emoções e o comportamento do indivíduo são alvo de estudos há muito tempo[Watson e Rayner 1920]. Conforme o entendimento sobre o comportamento humano cresceu, foram criadas técnicas para simulação de emoções ou personalidade em agentes.

Muitas destas técnicas são utilizadas em sistemas multiagente. Entretanto, na maior parte destes sistemas, os agentes recebem uma personalidade pré-definida. Este modelo propõe uma integração entre a personalidade e a emoção, tentando simular como ocorre esse processo no comportamento humano [Reisenzein e Weber, 2008], onde a personalidade de um agente pode interferir nas reações e decisões que ele irá tomar, consequentemente, interferindo no modo em que as emoções são geradas.

Uma vez inserido em uma sociedade amigável, é provável que o agente esteja mais propenso a receber boas ações e portanto sentir “boas” emoções. A quantidade das emoções sentidas e a qualidade delas, podem fazer com que a personalidade de uma agente varie [Nakao et al., 2000], podendo se tornar mais parecida com a personalidade de agentes a sua volta.

Atualmente o que se sabe sobre a personalidade humana é que ela é formada basicamente por dois fatores, o biológico [Bouchard et al., 1990; Bouchar, 1994] e o sociológico [Hopwood e Donnellan, 2011]. O fator biológico indica que cada indivíduo já nasce com uma pré disposição genética, para ter um tipo de comportamento e personalidade. O fator sociológico indica que a personalidade e o comportamento de uma pessoa pode ser determinado e influenciado de acordo com a sociedade em que se está inserido.

Este artigo propõe um modelo capaz de simular a variação, e portanto a formação, que ocorre com a personalidade de um agente no decorrer do tempo e em diferentes condições sociais e ambientais. O modelo trata, de maneira simplificada, as questões biológicas. É utilizado um modelo emocional que possibilita diferentes respostas para os mesmos estímulos (emoções) sentidos, de acordo com os diferentes tipos de personalidade. Os modelos psicológicos utilizados como base são o modelo OCEAN [Digman, 1990] de personalidades e o modelo OCC [Ortony et al., 1988] de emoções.

O artigo está dividido em 5 seções. Na seção 2 são apresentados os conceitos teóricos envolvidos neste trabalho. A seção 3 apresenta os modelos propostos e utilizados. Os resultados obtidos, o comportamento do modelo e sua análise são encontradas na seção 4 e, por fim, a seção 5 apresenta as conclusões e os trabalhos que serão realizados futuramente.

2. Referencial Teórico

Nesta seção serão introduzidos os três principais temas estudados para o desenvolvimento do artigo: sistemas multiagente, modelagem de personalidade e modelagem de emoções.

2.1. Sistemas Multiagente

Como o modelo proposto visa determinar as emoções a serem sentidas e a formação da personalidade de um indivíduo, é interessante simulá-lo em um Sistema Multiagente (SMA). Os SMA são uma subárea da Inteligência Artificial Distribuída (IAD), que diferentemente da Inteligência Artificial clássica que simula o comportamento em um único indivíduo, a IAD é capaz de simular um comportamento entre vários indivíduos, seu comportamento social, entendendo suas interações [Ferber, 1999].

Em um SMA, cada agente pode ter suas crenças e desejos, possibilitando assim que cada indivíduo ‘pense’ e aja de forma diferente.

Segundo Wooldridge (2002) um agente é uma entidade encapsulada capaz de resolver problemas que possuem autonomia, reatividade, pró-atividade e habilidade social. Um agente é definido como uma entidade cognitiva consciente, capaz de

expressar sentimentos, percepções e emoções, assim como os seres humanos. Estes agentes possuem características especificas como benevolência, mobilidade, conhecimento, crença, intenções e racionalidade.

Uma vez que cada agente tenha sua própria personalidade, é possível que existam variações comportamentais entre eles, assim como em uma sociedade, onde cada indivíduo, ao sofrer uma mesma ação, possa ter reações completamente diferentes, de acordo com seus pensamentos e ideologia.

2.2. Modelagem de personalidade

Buscando entender o comportamento humano e suas reações adversas para uma mesma situação, pesquisadores e filósofos tentam definir o funcionamento da personalidade humana a muito tempo. Durante pesquisas, grupos independentes de pesquisadores [Tupes e Christal, 1961; Digman, 1990; Costa, e McCrae, 1992; Goldberg, 1993], definiram, de forma empírica, a personalidade humana como o conjunto de cinco grandes fatores, o que ficou conhecido como *Big Five* ou OCEAN.

Estes cinco fatores são Abertura à experiência (*Openness*), Escrupulosidade (*Conscientiousness*), Extroversão (*Extraversion*), Altruísmo (*Agreeableness*) e Neuroticismo (*Neuroticism*). De forma a estar de acordo com o acrônimo OCEAN, serão utilizados os nomes em inglês de cada fator (pois cada letra representa um dos fatores). Cada pessoa tem um peso diferente para cada um destes fatores, sempre tendo todos os fatores representados por algum valor.

Cada fator tem uma significância e pode ser influenciado de formas diferentes. Um indivíduo com valor alto em *Openness* está mais disposto a ter novas experiências, no geral tem tendências a ser original, curioso, criativo e que busca novas vivencias e experiências.

Um indivíduo com alto valor em *Conscientiousness* é um indivíduo que tem mais foco em seus objetivos, geralmente é mais sistemático e tem disciplina para alcançar o que deseja.

O fator *Extraversion*, como já pode ser deduzido, é referente à extroversão que o indivíduo tem, a facilidade de ser bastante sociável e comunicativo.

Agreeableness é o fator que mede o quão altruísta o indivíduo é. Quando se tem um valor alto de altruísmo, o indivíduo é mais amigável e está mais a disposição de ajudar quando necessário.

Por fim, um indivíduo com valor alto em *Neuroticism* é um indivíduo muito neurótico, que tem medo de que as coisas dêem errado. Um indivíduo assim pode se tornar um ser mais ansioso, irritado, temperamental ou mal-humorado.

2.3. Modelagem de Emoções

Emoções são consideradas um dos principais fatores que influenciam em nossas vidas, em nosso modo de agir e de pensar. Por assim serem, existem muitas pesquisas a seu respeito e portanto muitas definições diferentes [Marsella et al., 2010]. Dentre estas definições, algumas são mais apropriadas para a modelagem computacional do que outras. Um dos modelos que mais se destacam para tal aplicação é o OCC (acrônimo de

seus criadores Ortony, Clore e Collins) [Ortony et al., 1988]. O modelo OCC divide as emoções em três categorias principais, chamadas de: *event-based emotions, agent-based emotions* e *object-based emotions.*

Event-based emotions podem ser definidas como emoções que resultam da consequência de eventos que ocorrem com outros indivíduos ou com o próprio indivíduo. Dentro destas emoções para o próprio indivíduo estão: *Joy, Distress, Satisfaction, Fears-Confirmed, Relief* e *Desapointment;* e como emoções para os outros estão: *Happy-For, Pity, Gloating* e *Ressentment.*

Agent-Based emotions são emoções sentidas em relação à alguma ação, tanto de outros indivíduos como de si mesmo. Estas emoções são: *Pride* e *Shame* para si, e *Admiration* e *Reproach* para outros indivíduos.

Object-Based Emotion são emoções direcionadas à objetos. São definidas duas emoções: *Love* e *Hate.*

Em nosso trabalho, por motivos de simplificações, não utilizaremos *Love* e *Hate.* Assim, serão simuladas apenas 20 emoções, ao invés das 22 contempladas pelo modelo OCC.

3. O modelo proposto

O modelo pode ser dividido em três partes: a parte da escolha da ação, influenciada pelos recursos e a personalidade do agente; a parte da análise da consequência da ação, que é realizada através da percepção e é responsável por gerar a emoção; e por fim, a parte da atualização da personalidade através da emoção sentida. Estas etapas podem ser visualizadas no fluxograma demonstrado na Figura1.

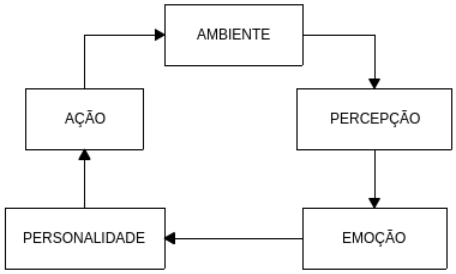


Figura1: Fluxograma comportamental do modelo.

Primeiramente, será explicado o modelo de mundo e suas regras, pois compreendendo o estudo de caso será mais fácil o entendimento de como o fluxograma da Figura 1 funciona e como deverá ser o comportamento de cada agente.

3.1. O modelo de mundo e suas regras

Como o objetivo do trabalho é fazer com que agentes tenham personalidade e esta interfira em suas ações, foi necessário criar um ambiente multiagente com um conjunto de regras.

Este modelo foi desenvolvido para simplificar ao máximo as ações e demonstrar de uma maneira clara como a personalidade pode influenciar nas ações de um agente.

Portanto, foi desenvolvido um modelo baseado em trocas de recursos entre os agentes.

Para um entendimento melhor do modelo imagine o seguinte contexto:

Em uma cidade existem apenas pequenos agricultores e cada um destes agricultores produz apenas um determinado alimento. Um agricultor pode consumir o que produz, entretanto para ter uma melhor saúde, ele deve variar sua dieta pelo menos uma vez por semana. Como cada agricultor pode produzir apenas um tipo de recurso, ele deve realizar trocas com outros agricultores de sua cidade.

Assim, é definido o ambiente proposto, onde cada agricultor é representado por um agente e cada 'cultivo' é representado por um recurso.

No modelo desenvolvido:

- Se um agente não consumir um recurso diferente dentro de um número N de dias ele morre;
- Após consumir um recurso diferente, o agricultor tem mais N dias para consumir outro recurso diferente novamente;
- A produção de recurso (PR) de cada agente deve ser de pelo menos N recursos para cada N dias, a fim de a cada dia poder ser consumido um recurso, próprio ou trocado.
- Um agente só pode fazer uma solicitação de troca a cada dia, entretanto pode receber mais de uma solicitação.
- Deve existir um valor M, que defina a quantidade máxima de estoque para cada recurso.

3.2. A escolha de uma ação

Cada agente tem em suas crenças a quantidade de cada recurso que ele obtém e a quantidade de dias que ele tem para consumir um novo recurso. As escolhas das ações são baseadas nestas duas crenças e em seus pesos de personalidade.

Uma solicitação de troca é realizada quando o agente tem um *desejo de troca*. O desejo de troca é definido pela necessidade que o agente tem de obter um novo recurso, ou seja, os dias que ele ainda tem para consumir e a quantidade de recursos extras estocados. Além disto é utilizado também os pesos de *Conscientiounness* e *Extrovert*, desta forma, o desejo de roca depende também do quanto o agente é focado para alcançar um objetivo e o quanto ele tem facilidade em se comunicar com outros agentes.

Se houver um desejo de troca será necessário escolher o agente para troca e definir a oferta que será proposta.

A escolha do agente é definida por três fatores. O peso *Openness* que representa o quão o agente é aberto à tentativa de uma troca com outro agente, o peso *Neuroticism* que verifica o quão ele terá medo da troca dar ou não certo, e de sua última troca efetuada. Se a última troca foi boa, ele terá um peso maior do que se a troca tiver sido ruim ou negada.

Por fim, a oferta será calculada com base na necessidade e no *Agreeableness* do

agente, onde quanto maior for a necessidade do agente, mais recursos ele ofertará para o outro agente, e, da mesma maneira, quanto maior o altruísmo, mais ele se importará em fazer uma boa oferta para o outro agente.

Para facilitar, o agente que envia a proposta denomina-se *Ag1*, e o que recebe de *Ag2*. Após *Ag1* definir todos estes parâmetros, ele envia uma mensagem para o *Ag2*, indicando que quer efetuar uma troca, e também sua oferta, ou seja, quantos recursos de *Ag1* por quantos recursos de *Ag2*. Esta ação será responsável pela primeira parte para gerar as emoções.

Quando *Ag2* receber a proposta, ele irá avaliar se ela é boa ou não. A avaliação da proposta é feita através de dois parâmetros: o *Agreeableness* de *Ag2*, pois se ele tiver um alto nível de altruísmo, ele tentará ajudar o outro agente, aceitando a troca e também através da verificação de quão boa é a oferta de *Ag1*, verificando se está ganhando ou perdendo recursos com a troca. Se o *Ag2* aceitar, eles irão trocar os recursos. Esta etapa será responsável por gerar as outras emoções, as quais serão introduzidas na seção 3.3.

3.3. A geração de Emoções

Para uma emoção ser gerada, é preciso que o agente execute alguma ação e tenha a percepção de como esta ação ocorre no ambiente em que ele está inserido. No modelo proposto, a ação que o agente fará será a de solicitação de troca e as emoções sentidas, através de sua percepção, podem ser divididas em quatro partes.

A primeira parte ocorre quando o *Ag1* fizer a solicitação de troca e, portanto, é referente às emoções de expectativa. Dentre as emoções de expectativa estão *Hope* e *Fear*. Elas serão sentidas de acordo com o nível de *Neuroticism* de *Ag1*. Se *Ag1* for muito neurótico, ele sentirá medo de *Ag2* e não aceitará sua oferta; caso contrario, ele sentirá esperança de *Ag2* e aceitará sua proposta.

A segunda parte é composta por quatro emoções: Satisfaction, Fears-Confirmed, Relief e Disapointment, que podem ser chamadas de emoções de realização, sendo que Satisfaction e Disapointment são consequências de Hope, onde ocorrendo Satisfaction quando a troca solicitada é aceita e Disapointment quando é negada por Ag2. Da mesma maneira ocorre com Relief e Fears-Confirmed que são consequências de Fear.

As emoções da terceira e da quarta parte podem ser chamadas de emoções de avaliação. Estas são geradas após o agente analisar se ele efetuou uma boa troca ou não. Na terceira parte, podem ser geradas as emoções de avaliação para o próprio agente. Estas emoções são: *Joy, Distress, Gloating, Ressentment, Admiration, Reproach, Gratification, Remorse, Gratitude* e *Anger*, as quais são influenciadas pelos pesos OCEAN e pela aceitação ou rejeição da oferta por *Ag2*. Na quarta parte, podem ser geradas as emoções de avaliação para o outro agente (*Ag2*). Estas emoções são: *Happy-For, Pity, Pride e Shame, a*s quais são influenciadas pela proposta feita para *Ag2*, se foi ruim para ele ou não.

Por fim, ao verificar estas emoções, será calculada qual emoção será sentida. Para tal, foi utilizado o modelo proposto por [Egges.; Kshirsagar; Magnenat-Thalman; 2004]. Este modelo escolhe a emoção a ser sentida através de matrizes e vetores de pesos influenciados pelo OCEAN.

3.4. A atualização da personalidade

Por fim, a última parte de nosso modelo, a atualização dos pesos OCEAN através das definições das emoções sentidas. Para realizar esta atualização da personalidade, verifica-se cada emoção sentida e em quais pesos ela tem influência. Posteriormente é feito um cálculo de atualização que utiliza como base o peso da personalidade no instante anterior.

Para entender o funcionamento da atualização é importante, primeiramente, que seja entendido como funciona a variação da personalidade. Cada peso OCEAN pode ser representado pelo intervalo [0,1], onde 0 significa a abstinência de um determinado peso e 1 a total presença. Exemplificando, se um agente tem valor 0 em *Agreeableness* (A=0) significa que é um agente muito egoísta, se tiver valor 1 (A=1), significa que é muito altruísta.

Quando um agente tem um peso OCEAN com valor próximo aos limites (0 ou 1), a variação de sua personalidade é menor que quando seu peso é intermediário, ou seja, um agente com A=0,1 tem menos variação positiva em seu peso OCEAN do que um agente com A = 0,6 quando expostos a um mesmo evento de altruísmo.

Para que seja possível este comportamento de variação, os valores OCEAN foram representados por funções sigmoidais. O que possibilita maiores variações quando seu peso é médio, e menores variações em seus extremos.

A função que representa os pesos OCEAN é dada pela Equação 1.

$$f(x) = \frac{1}{1 + e^{-kx}} \quad (1)$$

Onde $x \in R$ e varia de [-inf,+inf]; $i \in N$ e i = [0,4] podendo representar cada índice dos 5 fatores OCEAN; $F(x) \in R$ e (x)=[0,1] representando o valor real de cada personalidade; e k é uma constante positiva que multiplica x.

Para atualizar os pesos de personalidade, trabalha-se com os valores de x. Como cada agente pode ser inicializado com valores diferentes de personalidade, é necessário primeiramente encontrar cada valor inicial de x. Aplicando regras matemáticas básicas e propriedades logarítmicas na Eq.1 obtêm-se a Equação 2.

$$x = \frac{-1}{k} * \ln\left(\frac{1 - f(x)}{f(x)}\right) \quad (2)$$

Para realizar a atualização do peso OCEAN foram definidas duas equações: uma irá definir o comportamento de subida e a outra o comportamento de descida de cada peso OCEAN. Nestas equações são utilizados dois tipos de fatores de personalidade, a personalidade social, que até o momento tratamos como $f(x)$ e a personalidade biológica (*pBio*) a qual foi dada uma breve introdução na seção 1.

Ao contrário da personalidade social, a personalidade biológica nunca muda, ela é constante por toda a vida do agente. A *pBio* é responsável por fazer com que um indivíduo seja propenso a ter certos comportamentos, em nosso modelo será responsável por influenciar na personalidade final do agente. Ou seja, um agente que tenha sua *pBio* definida como muito egoísta, por mais altruístas que sejam as iterações

que este agente sofra ao longo da vida, ele nunca será completamente altruísta.

Portanto a equação responsável por elevar o valor de personalidade, pode ser representada pela seguinte Equação 3.

$$x(t+1) = x + \frac{f(x) + C * (pBio - f(x))}{fD + fT} \quad (3)$$

E a equação referente a descida da curva é apresentada pela Equação 4.

$$x(t+1) = x + \frac{(1 - f(x)) + C * (pBio - f(x))}{fD + fT} \quad (4)$$

Onde: *fD* é uma constante de divisão, para que a mudança em *x* ocorra de forma suavizada.

fT é o fator temporal da equação. A cada iteração este fator aumenta, diminuindo cada vez mais a variação sofrida por *x*. Ele é responsável por dar um peso às emoções ocorridas anteriormente, para que uma emoção que foi sentida em algum instante continue tendo seu peso, mesmo que pequeno, na personalidade atual do agente. Ou seja, uma emoção anteriormente sentida pelo agente, deixará sua marca pelo resto das iterações que ele sofrer. Assim, é possível que as emoções sentidas tenham um peso na formação final da personalidade do agente.

C é uma constante de multiplicação responsável por definir o quanto a personalidade biológica influenciará na personalidade final do agente.

4. O comportamento do modelo

O ambiente multiagente foi simulado com a utilização de 4 agentes distintos. Cada simulação continha 10000 iterações e foram feitas 100 simulações do cenário, para se ter um comportamento médio padrão assim como o seu desvio.

O modelo proposto se comporta da seguinte forma: quando o ambiente é hostil, os agentes nele inseridos tendem a ficar com seus pesos de personalidade baixos, com exceção de *Neuroticism* o qual funciona de modo contrário dos demais pesos. E, de modo contrário, quando os agentes estão inseridos em um ambiente amigável, seus pesos OCEAN tendem a subir, com exceção de *Neuroticism* que tende baixar.

Entretanto, existem várias formas de tornar o ambiente hostil ou amigável. As principais variáveis por alterar o ambiente são os pesos OCEAN de cada agente, a quantidade de dias que um agente tem para consumir um recurso diferente e a quantidade de recursos extras que o agente produz por dia.

A personalidade de cada agente, através dos pesos OCEAN, é capaz de modificar o ambiente (modificando a sociedade), pois como já foi explicado, um agente com alto nível de Altruism, por exemplo, aceita muitas trocas e solicita muitas trocas boas, deixando o ambiente melhor e mais amigável. Já a quantidade de dias que um agente tem para consumir um novo recurso é importante, pois diz o quanto o agente pode ficar "parado" apenas armazenando recurso próprio. Se esta quantidade de dias for muito baixa, o ambiente será mais hostil, pois os agentes estarão em constante tentativas de trocas, e com alto consumo de recursos, talvez nem sempre terão recursos suficientes para troca.

Além disto, a quantidade de recursos extras que cada agente produz por dia, é também essencial por tornar o ambiente hostil ou não, pois é através deste parâmetro que os agentes terão ou não recursos para solicitar trocas. Se o valor de produção excedente for baixo, os agentes viverão em um local hostil, com poucos recursos e consequentemente farão poucas trocas. Já em um sistema onde a produção de recursos excedentes é alta, os agentes terão mais recursos disponíveis, tanto para solicitar trocas como para aceitar solicitações de troca.

Para a análise dos resultados foi setado um ambiente "ideal" onde há uma quantidade razoável de recursos produzidos por dia e pouca necessidade de consumo de recursos diferentes. Neste ambiente foram realizadas 4 simulações distintas, as quais serão comparadas em pares.

Nas duas primeiras simulações os pesos de personalidade biológica são definidos como (0,5; 0,5; 0,5; 0,5; 0,5) e a comparação se dá através da sociedade inserida, onde na primeira simulação os outros agentes (sociedade) tem seus pesos pBio setados como (0,9; 0,9; 0,9; 0,9; 0,9), tentando representar uma sociedade amigável, enquanto na segunda simulação os outros agentes tem seus pesos pBio setados como (0,1; 0,1; 0,1; 0,1; 0,9), tentando, desta forma, simular uma sociedade mais hostil. A tabela 1 demonstra a comparação dos resultados entre a primeira e a segunda simulação enquanto a figura 1 demonstra de forma gráfica os valores finais dos pesos OCEAN obtidos em cada simulação e a figura 2 demonstra a media dos dias vividos pelo agente em cada simulação.

	O	C	E	A	N	Média de dias vividos
S1	0,613597695	0,61359575	0,599993546	0,599989	0,163143	10000
S2	0,45692914	0,456934228	0,547484084	0,547482	0,492084	6423
Desvio S1	0,003481252	0,003481212	2,19E-05	2,19E-05	0,011507	-
Desvio S2	0,07575408	0,075746742	0,037940173	0,037938	0,045771	-

Tabela1 : Comparação entre resultados das simulações 1 e 2

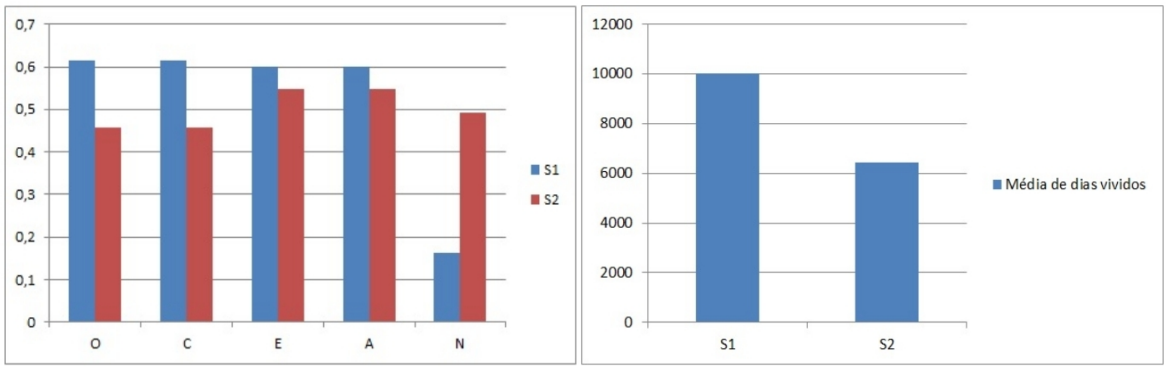


Figura1 : Gráfico comparativo dos pesos OCEAN finais e dos dias vividos, para um mesmo agente inserido em sociedades diferentes

Como pode ser visto, a variação da personalidade entre as duas simulações não foi tão grande. Entretanto, isto ocorre pelo fato da pBio do Agente estudado ser (0,5;

0,5; 0,5; 0,5; 0,5). Isto acaba influenciando a personalidade do agente para que continue no ’meio termo’.

Mas a principal característica a ser notada é a variação da personalidade final do agente analisado. Onde, o agente inserido em uma sociedade amigável teve seus pesos de personalidade amentados, com exceção a *Neurotcism*, e o agente inserido em uma sociedade mais hostil teve uma tendência a diminuir seus pesos OCEAN. Um fato que pôde ser analisado também é que a variação dos pesos OCEAN na segunda simulação foi menor do que na primeira simulação, isto se deve a quantidade de dias vividos pelo agente, onde em uma sociedade amigável o agente tem uma sobrevida maior, podendo assim sofrer mais tempo de influência em sua personalidade do que o agente da segunda simulação. Isto pode ser analisado através dos dias vividos, demonstrado na tabela 1 e também através do desvio padrão dos pesos de personalidade, também demonstrado na tabela 1, onde o desvio padrão é maior pelo fato de que em algumas simulações o agente morria no inicio da simulação enquanto em outras conseguia chegar até o final.

Na terceira e quarta simulação foi analisado o agente inserido em um ambiente e em uma sociedade amigável, porém, desta vez, na terceira simulação o agente analisado tinha seus pesos *pBio* como um agente amigável (0,9; 0,9; 0,9; 0,9; 0,1), enquanto na quarta simulação o agente tinha seus pesos *pBio* como um agente que tende a ser mais hostil (0,1; 0,1; 0,1; 0,1; 0,9).

Esta análise foi realizada para demonstrar como um agente distinto se comporta quando inserido em um mesmo ambiente e uma mesma sociedade.

Os valores dos resultados desta simulação podem ser verificados na tabela 2, assim como na figura 2, onde é representada de forma gráfica a comparação dos pesos OCEAN finais.

	O	C	E	A	N	Média de dias vividos
S3	0,951125466	0,951121481	0,676922476	0,676915481	0,387177	9900
S4	0,389023517	0,389033205	0,393435859	0,393440657	0,807095	9900
Desvio S3	0,045543346	0,045542948	1,79E-02	0,017890854	0,011674	-
Desvio S4	0,016996282	0,016994879	0,010826273	0,010825787	0,031004	-

Tabela2 : Comparação entre os resultados das simulações 3 e 4

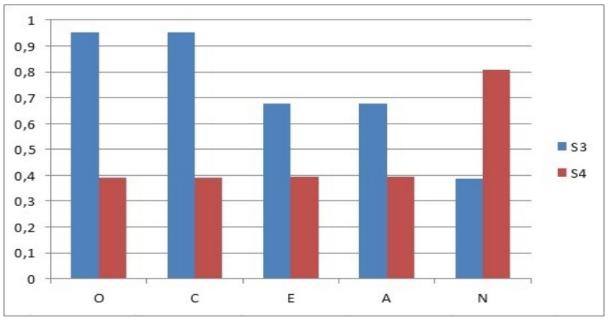


Figura3 : Gráfico dos valores OCEAN finais nas simulações 3 e 4.

O que pode ser inicialmente analisado é o fato de por estar em uma sociedade amigável, o agente teve em média uma vida muito maior.

Assim como esperado, na terceira simulação, o agente teve seus pesos OCEAN aumentados, devido a ele estar em uma sociedade amigável e também a seus pesos de personalidade biológica. Entretanto, na quarta simulação os pesos O, C, E, e A de personalidade do agente se mantiveram próximos à 0,4. Isto ocorre por dois fatores, como o agente está inserido em um ambiente muito amigável, ele tende a ficar muito amigável, entretanto, por seus pesos de pBio serem muito baixos, a personalidade do agente chega a um limite onde não consegue mais aumentar. O que causa este fenômeno é o valor que se dá para a constante C, na equação de acréscimo e decréscimo dos pesos OCEAN, pois, como a explicado, esta constante define o quanto a personalidade biológica influenciará na personalidade final do agente.

Estas simulações demonstram os fatores envolvidos na adaptabilidade do agente de acordo com o ambiente e a sociedade em que ele é inserido. Demonstram também que os métodos desenvolvidos durante a proposta do trabalho funcionam de maneira genérica e eficaz.

5. Conclusão e Trabalhos Futuros

Neste trabalho foi demonstrado como é possível realizar uma simulação multiagente onde cada agente tem sua própria personalidade, a qual influencia diretamente em seus desejos e em seus objetivos finais.

Foi proposto um modelo de mundo simples e com regras simples, para ser utilizado em implementações que utilizam SMA. Modelo este que pode ser usado para simular problemas que envolvem teoria dos jogos, trocas sociais ou até mesmo modelos multiagente que simulam competição ou cooperação entre agentes.

Além disto, foi demonstrada uma metodologia para utilizar a personalidade em agentes, com a finalidade de trabalhar com a tomada de decisão do mesmo. Propondo como pode ser definida a personalidade para que elas interfiram nas ações do agente. A partir desta metodologia, é possível integrar a personalidade de um agente para diversas aplicações.

A principal contribuição do trabalho está no modo em que as emoções sentidas são capazes de modificar a personalidade do agente. Uma vez que este modelo pode ser usado em aplicações onde é necessário que um agente se adapte à um meio. No estudo de caso, foram utilizados vários agentes que formavam uma sociedade, mas o modelo pode ser usado em outras aplicações, como jogos ou problemas que envolvem IHC (Interface Humano Computador), em que haja a necessidade de um agente se adaptar e realizar ações de acordo com seu fator de personalidade e com o ambiente à sua volta.

Como trabalhos futuros, espera-se desenvolver um sistema de reputação entre os agentes, onde tem-se uma “memória” sobre as trocas anteriormente realizadas (boas ou ruins) para que a escolha de um agente para solicitar trocas, possa ser influenciada por outras métricas. Deseja-se modificar a parte das escolhas das emoções sentidas, de forma que uma rede neural faça as escolhas das emoções, bem como um sistema utilizando lógica Fuzzy, para comparar os resultados e verificar quais modelos tem um

melhor comportamento. Também é necessário que se entenda mais a fundo o funcionamento do comportamento humano, a fim de melhorar as métricas utilizadas e aproximá-las o máximo possível da realidade do comportamento humano.

Referências

Bouchard Jr. T.J., David T, Lykken DT, McGue M, Segal NL, Tellegen A (Oct 12, 1990) “Sources of human psychological differences: the Minnesota study of twins reared apart”. Science, v250 n4978 p223.

Bouchard Jr. T.J., (Jun. 17, 1994) “Genes, environment, and personality” Science, New Series, Volume 264, Issue 5166, 1700-1701.

Costa, P.T., Jr., McCrae, R.R.; (1992) Revised NEO Personality Inventory (NEO-PI-R) and NEO Five-Factor Inventory (NEO-FFI) manual.Odessa, FL: Psychological Assessment Resources.

Christopher J. Hopwood, M. Brent Donnellan (2011 Mar) “Genetic and environmental influences on personality trait stability and growth during the transition to adulthood: A three wave longitudinal study” J Pers Soc Psychol. 100(3): 545–556.

Digman, J.M., (1990) "Personality structure: Emergence of the five-factormodel," Annual Review of Psychology, 41, 417-440.

Egges, A., Kshirsagar, S., Magnenat-Thalman, N., (2004) “Generic personality and emotion simulation for conversational agents”. Computer Animation and Virtual Worlds. Volume 15 Issue 1, p. 1 – 13.

Ferber J., (1999) “Multi-agent systems: An introduction to distributed artificial inteligence”, Addison-Wesley.

Goldberg, L.R., (1993) "The structure of phenotypic personality traits,"American Psychologist, 48, 26-34.

Kazuhisa Nakao, Jyo Takaishi, Kenji Tatsuta, (2000), “The influences of family environment on personality traits” Psychiatry and Clinical Neurosciences.

Marsella, S., Gratch, J., Petta, P., (2010) “Computational Models of Emotion”, Oxford: Oxford University Press.

Ortony, A., Clore, G.L., Collins A., (1988) “The Cognitive Structure of Emotions”, Cambridge University Press.

Reisenzein, R and Weber, H. (2008) “Personality and emotion”. In Corr, P and Matthews, G, editors, Cambridge Handbook of Personality. Oxford University Press.

Tupes, E.C., Christal, R.E.; (1961) "Recurrent Personality Factors Based on Trait Ratings," Technical Report ASD-TR-61-97, Lackland Air Force Base, TX: Personnel Laboratory, Air Force Systems Command.

Watson, John B.; Rayner, Rosalie (1920) “Conditioned Emotional Reactions”. Journal of Experimental Psychology, 3(1), 1-14.

Wooldridge, M. (2002) “An Introduction to Multi-agent Systems”, New York: Wiley.

Modelo de Reputação Fuzzy Aplicado a um Sistema Multiagente

Henrique D. N. Rodrigues[1], Diana F. Adamatti[1], Graçaliz P. Dimuro[1], Glenda Dimuro[2], Esteban de Manuel Jerez[2]

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

[2]Depto de Expresión Gráfica Arquitectónica, Universidad de Sevilla, Seville, Spain

{henriquedonancio,dianaada,gracaliz}@gmail.com

Abstract. *This paper presents a reputation model applied to a multi-agent system for simulation of regulatory policy and reputation of the social organization of the Sán Jerónimo Vegetable Garden, located in Seville, Spain. Therefore, we are used BDI agents with fuzzy beliefs to investment analysis and satisfaction of services as well as a reputation model as a performance measure of their activities within the project, implemented by JaCaMo framework.*

Resumo. *Este artigo apresenta um modelo de reputação aplicado a um sistema multiagente para simulação de políticas normativas e reputação da organização social da Horta Urbana Sán Jerónimo, localizada em Sevilha, Espanha. Para tanto, foi utilizado agentes BDI com crenças fuzzy para análise de investimento e satisfação sobre serviços, assim como um modelo de reputação como medida de desempenho de suas atividades dentro do projeto, implementados através do arcabouço JaCaMo.*

1. Introdução

Este trabalho tem como objetivo o desenvolvimento de ferramentas SMA para simulação de trocas sociais e cooperação entre agentes, tendo como cenário, o projeto social da Horta Urbana "Sán Jerónimo", localizada em Sevilha, Espanha, coordenado pela ONG "Ecologistas en Acción".

Em sistemas multiagentes, o conhecimento sobre outros agentes é construído através de experiências passadas seja de forma direta ou indireta. Um modelo de reputação fuzzy busca tratar essa questão de forma subjetiva sobre como o comportamento de um agente afeta as expectativas de outro agente [Rubiera et al. 2001].

Os agentes participantes desse projeto são os hortelãos e aspirantes a hortelãos, que são os agentes sociais. Técnicos e secretaria assumem o papel de agentes governamentais, responsáveis pela verificação das normas aplicadas aos agentes sociais.

O regulamento da horta é um conjunto que conta um total de quarenta normas que visa estabelecer melhor convívio entre os usuários da horta e a administração, além de resguardar seus direitos e orientar suas ações.

O modelo de reputação adotado é baseado em modelos tais como REGRET [Sabater and Sierra 2001] e [Hübner et al. 2009]. A análise da reputação é divida em três

dimensões: Dimensão Social, Dimensão Individual, Dimensão Ontológica, como proposto no modelo REGRET. Na Dimensão Social é analisada a efetividade do agente para com o seu grupo social, já na Dimensão Individual é analisada as trocas diretas entre os agentes. Por fim, têm se a Dimensão Ontológica, onde Dimensão Social e Individual se combinam para uma análise final.

Este artigo apresenta um modelo de reputação aplicado a um SMA para simular trocas que não envolvem bens materiais, como também a política interna da organização, tendo como métrica de desempenho social a reputação do agente dentro do contexto do projeto Horta Sán Jerónimo. Entende-se como trocas que não envolvem bens materiais, aquelas que tanto na contratação e/ou realização do serviço não há bens envolvidos, gerando apenas débitos e créditos virtuais.

O modelo de agentes com representação de crenças *fuzzy* (BDI-Fuzzy) foi escolhido devido a sua maior complexidade e abrangência em análises subjetivas e individuais de serviços, através da *Atitude de Avaliação de Serviços*.

Para implementação prévia do modelo foi utilizado a plataforma Jason[Bordini et al. 2007], um interpretador da linguagem AgentSpeak(L), baseada na arquitetura BDI [Rao 1996, Rao and Georgeff 1992], juntamente com o arcabouço CArtAgO [Ricci et al. 2014] e o arcabouço para prover a simulação de políticas públicas MSPP (Modeling and Simulation of Public Policies) [Santos and Costa 2012, Santos et al. 2012].

O artigo está organizado como descrito a seguir. A Seção 2 apresenta a plataforma Jason, o arcabouço CArtAgO, e o modelo organizacional . A Seção 3 expõe as características do modelo de reputação adotado juntamente com caracteríticas fuzzy de avaliação de trocas entre os agentes sociais. A Seção 4 apresenta a prévia simulação do SMA baseado na Horta San Jéronimo, e a Seção 5 faz uma análise dos resultados obtidos até o momento.

2. A Plataforma JaCaMo

A plataforma JaCaMo [Bordini and Hübner 2014] é um arcabouço para programação de Sistemas Multiagentes constituído de três ferramentas.

O Jason [Bordini et al. 2007] é um interpretador da linguagem AgentSpeak(L), baseada na arquitetura BDI. Um aspecto importante dessa plataforma é sua implementação em Java, e portanto multi-plataforma. A comunicação entre agentes no Jason é baseada na teoria de atos de fala. Os agentes ao se comunicarem, geram crenças e estas por sua vez podem desencadear planos.

O arcabouço CArtAgO (Common ARTifact infrastructure for AGents Open environments) [Ricci et al. 2014] é baseado no modelo Agentes e Artefatos (A & A) para modelar e projetar Sistemas Multiagente. Com essa ferramenta é possível criar artefatos estruturados em espaços abertos onde agentes podem se unir de forma a trabalhar em conjunto.

Os Artefatos são recursos e ferramentas construídas de forma dinâmica, usados e manipulados por agentes para apoiar/realizar suas atividades individuais e coletivas. O ambiente como também os recursos disponíveis no mesmo podem ser modelados na forma de um artefato CArtAgO.

O modelo organizacional MOISE+ [Hübner 2003] é uma ferramenta com intuito de modelar a organização de SMA. Consiste na especificação de três dimensões: a estrutural, onde definem-se papéis e ligações de heranças e grupos; a funcional, onde é estabelecido um conjunto de planos globais e missões para que as metas sejam atingidas; e a deôntica, que é a dimensão responsável pela definição de qual papel tem obrigação ou permissão para realizar cada missão.

3. Modelo de Reputação

O modelo de reputação segue a estrutura adotada em REGRET [Sabater and Sierra 2001], onde a reputação é uma composição de três dimensões: Dimensão Social, Dimensão Individual e Dimensão Ontológica.

Modelos de reputação fuzzy já existem na literatura como [Rubiera et al. 2001] e outros direcionados para e-commerce e redes P2P. Esta abordagem no entanto estende modelos existentes como Regret [Sabater and Sierra 2001], capaz de fornecer métricas para as políticas públicas em cenários de ambiente cooperativo.

3.1. Dimensão Social

A dimensão social avalia aspectos coletivos em relação ao agente. Em [Hübner et al. 2009] essa avaliação é feita analisando a efetividade do agente em relação a normas (obrigações), a participação do agente e seus resultados.

No projeto HSJ, os agentes devem desempenhar algumas obrigações, tais como pagamento de mensalidades e renovação de suas inscrições, além do cumprimento das normas. A participação também é um fator a ser levado em conta uma vez que o projeto determina que o agente participe de reuniões que servem para discutir e orientar suas práticas. Os resultados, são todas ações que o agente venha executar coletivamente.

Cada agente tem como crença um respectivo valor referente ao grau de importância[1] que ele atribui a estes aspectos que são compartilhados entre o grupo. A avaliação individual dada por cada individuo é dada pela Equação (1) [Hübner et al. 2009]:

$$avaliacao(\alpha) = \frac{\gamma p(\alpha) + \delta o(\alpha) + \epsilon r(\alpha)}{\gamma + \delta + \epsilon} \quad (1)$$

Onde os fatores γ, δ e ϵ definem a importância da participação, obediência e resultados, respectivamente.

Esses fatores são independentes entre os agentes sociais, podendo assumir valores por convenção entre 1 a 10, definindo assim o grau de relevância que o agente determina para o atributo multiplicado pelo fator. Dessa forma, o valor mínimo dado para reputação de determinado agente é 0 e o valor máximo é 1.

[1]O valores são independentes para cada agente afim de descentralizar o modelo proposto em [Hübner et al. 2009], tendo o agente uma avaliação mais individual dos aspectos coletivos.

3.2. Dimensão Individual

A dimensão individual é resultado das interações diretas entre os agentes. Em [Sabater and Sierra 2001], essa dimensão é tratada como a mais confiável, pois expressa resultados de interações diretas com o agente alvo, ou seja, uma avaliação dada pelo resultado da interação entre os agentes envolvidos.

Na abordagem desse trabalho, os agentes trocam serviços não econômicos que não envolvem trocas monetárias ou de bens. Os agentes hortelãos no projeto HSJ, tem como base de avaliação o investimento em realizar um serviço e a satisfação ao receber um serviço. Ao avaliar um investimento, o agente leva em consideração três fatores [Farias et al. 2013]:

- A *dificuldade*, que representa o quão difícil é para o agente realizar o serviço solicitado. Assume valores no intervalo de 0 e 10, onde valores próximos de 10 representam um grau de dificuldade maior.

- O *custo*, que representa os gastos para realizar o serviço. Assume valores entre 1 e 100, onde valores próximos de 100 indicam gastos maiores na realização do serviço.

- O *tempo*, que representa o tempo gasto na execução do serviço. Assume valores entre 1 e 90, onde valores próximos de 90 indicam um tempo maior empregado na realização do serviço.

Para a análise da satisfação como resultado de um serviço recebido é levado em consideração:

- A *qualidade*, que representa o grau de excelência do serviço. Assume valores entre 0 e 10 onde quanto maior o valor, maior o grau de excelência do serviço.

- O *preço*[2], que representa o valor dos gastos com a contratação do serviço. Assume valores entre 0 e 100, onde valores próximos de 0 indicam gastos menores.

- O *tempo*, que representa o tempo gasto esperando a execução do serviço. Assume valores entre 1 e 90, onde quanto menor o valor, menor o tempo esperado na realização do serviço.

Um agente ao requisitar que outro agente preste um serviço, calcula a satisfação esperada através de uma função de pertinência fuzzy e uma base de regras e com base na sua "Atitude de Avaliação de Serviço" [Farias et al. 2013].

Tabela 1. Base de Regras Fuzzy para Análise do Investimento Baseado na Dificuldade

Base de Regras Fuzzy do Investimento (Dificuldade)
$R^{(1)}$: IF dificuldade IS baixa THEN investimento IS baixo
$R^{(2)}$: IF dificuldade IS média THEN investimento IS médio
$R^{(3)}$: IF dificuldade IS alta THEN investimento IS alto

[2]Para o estudo de caso esse fator não é levado em consideração.

A Atitude de Avaliação de Serviço é a composição de um ou mais atributos que pertencem a esse serviço. Isto significa que o agente pode analisar o investimento na realização do serviço "plantar", baseando-se exclusivamente na dificuldade de se executar essa ação, e em contrapartida analisar a satisfação em receber este serviço analisando por exemplo, o tempo e a dificuldade. Cada agente possui sua Atitude de Avaliação de Serviço independentemente.

Ao receber o serviço o agente calcula a satisfação real, assim como o agente que fez o investimento calcula o investimento real. A relação entre satisfação esperada e satisfação real, gera crédito para quem prestou o serviço e débito para quem recebeu o serviço.

Tabela 2. Base de Regras Fuzzy para Análise do Débito

Base de Regras Fuzzy (Satisfação Esperada X Satisfação Real)
$R^{(1)}$: IF satisfação esp. IS baixa AND satisfação real IS baixa THEN débito IS médio
$R^{(2)}$: IF satisfação esp. IS baixa AND satisfação real IS média THEN débito IS alto
$R^{(3)}$: IF satisfação esp. IS baixa AND satisfação real IS alta THEN débito IS alto
$R^{(4)}$: IF satisfação esp. IS média AND satisfação real IS baixa THEN débito IS baixo
$R^{(5)}$: IF satisfação esp. IS média AND satisfação real IS média THEN débito IS médio
$R^{(6)}$: IF satisfação esp. IS média AND satisfação real IS alta THEN débito IS alto
$R^{(7)}$: IF satisfação esp. IS alta AND satisfação real IS baixa THEN débito IS baixo
$R^{(8)}$: IF satisfação esp. IS alta AND satisfação real IS média THEN débito IS baixo
$R^{(9)}$: IF satisfação esp. IS alta AND satisfação real IS alta THEN débito IS médio

3.3. Balanço Material e Virtual

O balanço material de um agente (α), no modelo BDI-Fuzzy, é obtido através de uma avaliação fuzzy (qualitativa) obtido pelos valores materiais de investimento R_a e satisfação S_a, gerados na troca de serviços com um outro agente, segundo uma avaliação pessoal do agente [Farias et al. 2013].

A escala com os termos linguísticos para representação do balanço material e virtual são:

T_{bm} = $<$ *muito desfavorável, desfavorável, equilibrado, favorável, muito favorável* $>$

Tendo o seu valor normalizado representado pela função de pertinência fuzzy [Figura 1] :

A cada troca realizada onde o agente presta um serviço (faz um investimento), e recebe um serviço (gera uma satisfação) com um agente alvo, é gerado um balanço material. A cada troca o agente registra o balanço material em um vetor *v* . Esse acumulado de trocas gera a reputação com o agente alvo, de acordo com a equação (2):

$$\sum_{i=1}^{\text{size}(v)} \frac{v_i * a_n}{\text{size}(v)}, \tag{2}$$

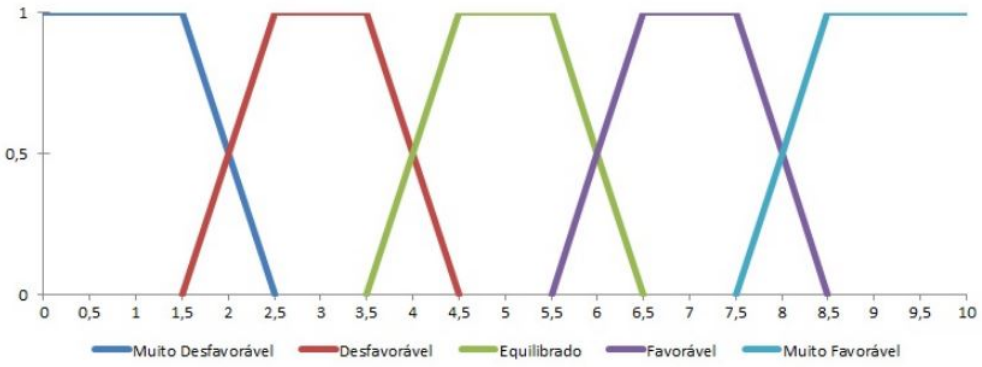


Figura 1. Escala para variáveis linguísticas [Farias et al. 2013]

onde a_n = a_1 + (n - 1) * β, sendo β dado por $\beta = \frac{a_1+1}{\text{size}_{(v)+1}}$ e a_1 = 0.1

Uma vez que o agente presta um serviço, ele gera crédito em relação a quem recebe o serviço, e o agente que recebe o serviço contrai um débito para quem prestou o serviço. Uma vez realizada a troca, ou seja, o agente realiza um investimento ao prestar um serviço e gera uma satisfação ao receber um serviço, os valores de crédito e débito gerados são analisados através do "Balanço Virtual".

O Balanço Virtual é atualizado a cada novo crédito e débito gerados na prestação ou recebimento de um serviço através da base de regras [Farias et al. 2013]:

Tabela 3. Avaliação Fuzzy do Balanço Virtual Quando o Agente Realiza um Serviço (Balanço Virtual x Crédito)

	baixo	**médio**	**alto**
muito desfavorável	muito desfavorável	desfavorável	equilibrado
desfavorável	desfavorável	equilibrado	favorável
equilibrado	equilibrado	favorável	muito favorável
favorável	favorável	muito favorável	muito favorável
muito favorável	muito favorável	muito favorável	muito favorável

Tabela 4. Avaliação Fuzzy do Balanço Virtual Quando o Agente Recebe um Serviço (Balanço Virtual x Débito)

	baixo	**médio**	**alto**
muito desfavorável	muito desfavorável	muito desfavorável	muito desfavorável
desfavorável	desfavorável	muito desfavorável	muito desfavorável
equilibrado	equilibrado	desfavorável	muito desfavorável
favorável	favorável	equilibrado	desfavorável
muito favorável	muito favorável	favorável	equilibrado

O agente ao ser solicitado a executar uma das três tarefas que compõe o cenário deste estudo: plantar, irrigar e colher, sempre requisita auxílio a outro agente. O critério de escolha acerca de quem poderá ser solicitado é baseado no Balanço Virtual e segue os seguintes passos:

1. Balanço Virtual *muito desfavorável*: Esse tipo de balanço é preferível uma vez que o agente possui muito mais créditos do que débitos em relação ao agente

alvo.

2. Balanço Virtual *desfavorável*: Uma relação com o agente alvo com balanço virtual desfavorável é a segunda opção quando não há uma relação com balanço virtual muito desfavorável, pois o agente possui uma quantidade de créditos relativamente maior que débitos.

3. Balanço Virtual *equilibrado*: O agente possui uma relação equilibrada de trocas com o agente alvo. Esse balanço virtual ainda é preferível a pedir um desconhecido uma vez que o agente terá que pedir auxílio e consequentemente gerar um débito, porém trocas anteriores atingiram um equilíbro.

4. Balanço Virtal *favorável/muito favorável*: Quando o agente não possui sequer um balanço equilibrado com outro agente, ele então requisita a quem ele contraiu mais débitos para que indique outro agente. O agente não requisita o agente com que possui mais débitos pois seu balanço já é negativo e ele busca atingir um equilibrio.

- *Indicação*: Os agentes ao serem solicitados para que indiquem alguém para fazer um serviço, buscam indicar aqueles com que: tiveram uma satisfação alta em receber o serviço, como segunda opção aqueles com que teve uma satisfação media, e como última escolha, não havendo nenhuma das demais opções, aqueles com que obteve satisfação baixa.

3.4. Dimensão Ontológica

A Dimensão Ontológica é a combinação das dimensões Social e Individual. Como já dito, a Dimensão Individual é mais confiável em relação a Dimensão Social, uma vez que a primeira expressa as relações diretas entre os agentes. A equação que produz essa combinação é expressa em (3):

$$D_o(\alpha) = \frac{\vartheta D_s(\alpha) + \varphi D_i(\alpha)}{\vartheta + \varphi} \tag{3}$$

Onde os fatores ϑ e φ definem a importância da Dimensão Social e Dimensão Individual respectivamente, sendo $\vartheta < \varphi$ e aplicados a todos os agentes. O valor desses fatores deve ser escolhido de acordo a necessidade de se valorizar a Dimensão Individual em relação a Dimensão Social. Ressalta-se que D_i e D_s devem estar normalizados.

4. Análise da Dimensão Individual

Cada agente, possui valores de investimento esperado que em um primeiro momento é informado ao agente que receberá o serviço, gerando uma satisfação esperada. Em uma segunda etapa, ao concluir o serviço o agente gera uma satisfação real, que corresponde aos valores reais investidos na realização do serviço avaliado tanto no primeiro momento quanto no segundo de acordo com a Atitude de Avaliação de Serviço de cada parte envolvida.

Tabela 5. Investimento Esperado/Real do Agente Cícero

	Dificuldade	Custo	Tempo
Serviço Plantar	6.3	50	47
Serviço Colher	5	40	60

Tabela 6. Investimento Esperado do Agente Genaro

	Dificuldade	Custo	Tempo
Serviço Plantar	8	70	62
Serviço Colher	8.5	65	70

Tabela 7. Investimento Real do Agente Genaro

	Dificuldade	Custo	Tempo
Serviço Plantar	9	88	75
Serviço Colher	9.2	80	85

Tabela 8. Investimento Esperado do Agente Pedro

	Dificuldade	Custo	Tempo
Serviço Plantar	3.2	35	50
Serviço Colher	2.3	20	28

Tabela 9. Investimento Real do Agente Genaro

	Dificuldade	Custo	Tempo
Serviço Plantar	2.8	30	36
Serviço Colher	1.7	10	11

No cenário criado existem três agentes, Cícero, Genaro e Pedro, que trocam serviços entre si. As trocas nesse caso específico, acontecem até que o Balanço Virtual atinja estados através da Heurística de Busca de Parceiros a condição de *favorável* ou *muito favorável*, quando então o agente pede indicação para o agente alvo sobre quem possa executar o serviço. Assim, agentes que investem menos, acabam experimentando variação de valores inferiores aqueles que por exemplo, superam as expectativas. Isso condicionado a Atitude de Avaliação de Serviço, uma vez que por exemplo, o agente Pedro investe menos nos quesitos *dificuldade*, *custo* e *tempo*, mas um investimento menor em *tempo* tem significado positivo, portanto, se avaliado somente pelo tempo gasto na execução das tarefas, o agente Pedro pode obter a maior desempenho entre os agentes envolvidos.

5. Conclusão

As simulações iniciais indicam que o modelo híbrido de reputação adotado contempla tanto o desempenho coletivo, tanto o individual dos agentes envolvidos de forma satisfatória, favorecendo aqueles que melhor desempenham seus papéis tanto em relações diretas de trocas como também no coletivo em relação a política normativa (ver [Rodrigues et al. 2013].

Em trabalhos futuros espera-se que mais agentes possam interagir, trocando informações e indicando agentes para serviços, como proposto na heurística de busca

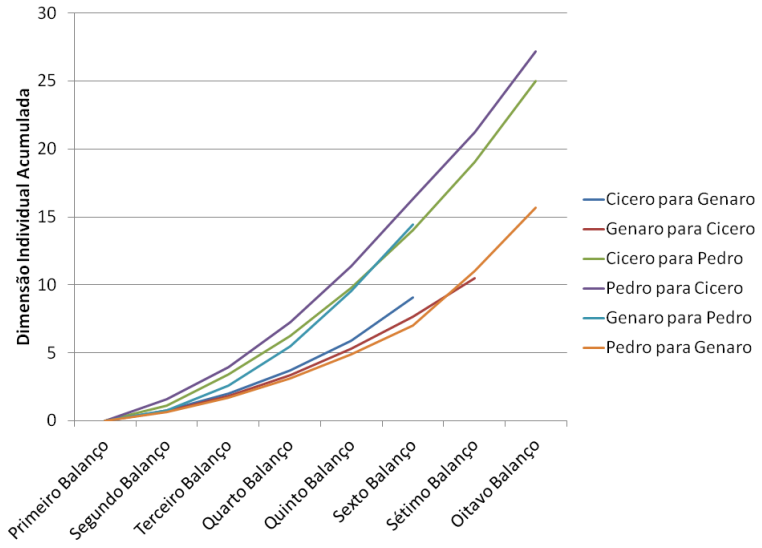


Figura 2. Individual Dimension Evaluation based on Cost and Time for Investment and Time for Satisfaction

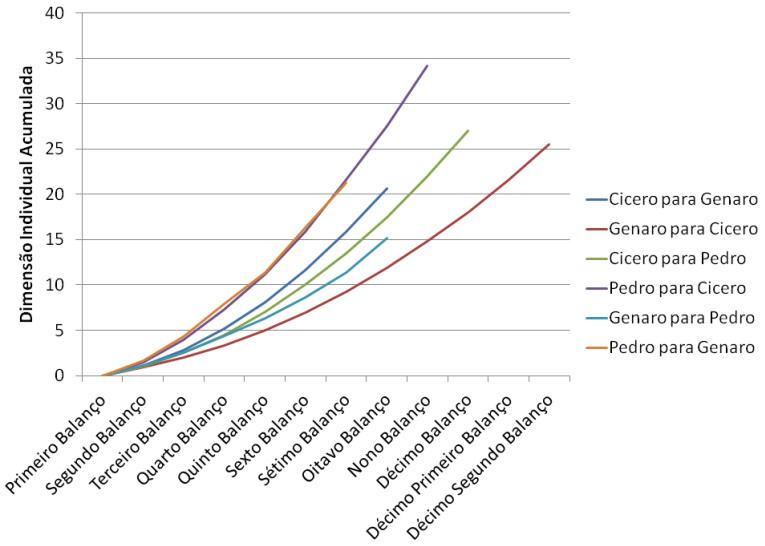


Figura 3. Individual Dimension Evaluation based on Cost and Time for Investment and Quality and Time for Satisfaction

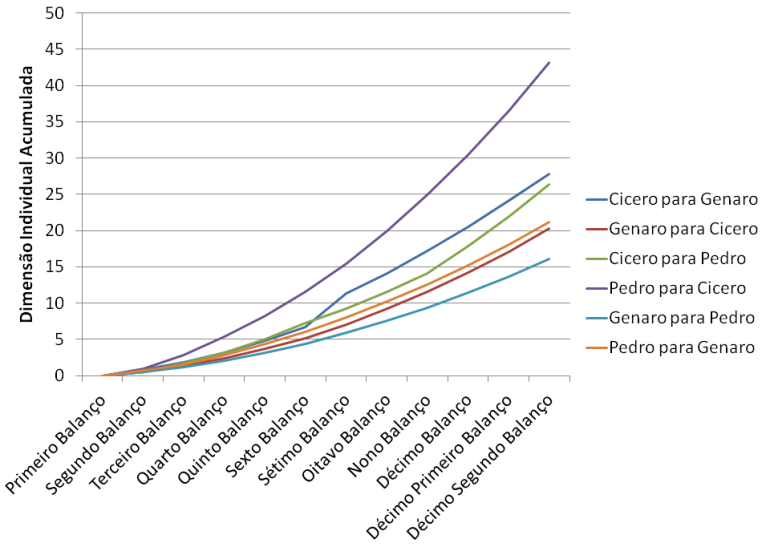


Figura 4. Individual Dimension Evaluation based on Difficulty for Investment and Quality and Time for Satisfaction

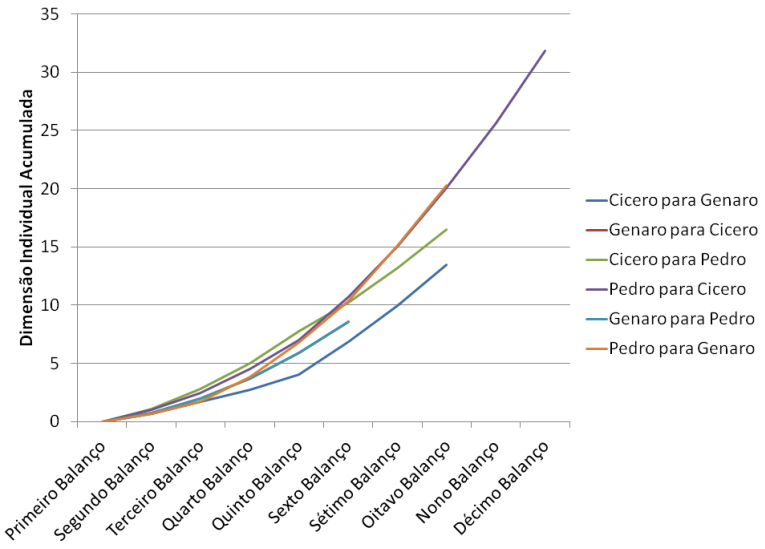


Figura 5. Individual Dimension Evaluation based on Difficulty for Investment and Time for Satisfaction

por parceiros, assim como integrar aspectos sociais coletivos afim de obter a Dimensão Ontológica aqui apresentada.

Agradecimentos

Este trabalho é apoiado pelo CNPq (Proc. 560118/10-4, 305131/2010-9, 476234/2011-5) , FAPERGS (Proc. 11/0872-3) e Projeto RS-SOC (FAPERGS Proc. 10/0049-7).

Referências

Bordini, R. H. and Hübner, J. F. (2014). Jacamo project. http://jacamo.sourceforge.net/.

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

Farias, G. P., Dimuro, G., Dimuro, G., and Jerez, E. D. M. (2013). Exchanges of services based on Piaget's theory of social exchanges using a BDI-fuzzy agent model.

Hübner, J. F. (2003). *Um Modelo de Reorganização de Sistemas Multiagentes*. PhD thesis, Universidade de São Paulo, São Paulo.

Hübner, J. F., Vercouter, L., and Boissier, O. (2009). *Instrumenting Multi-Agent Organisations with Artifacts to Support Reputation Processes*. Springer.

Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, LNCS. Springer, Berlin.

Rao, A. S. and Georgeff, M. P. (1992). An abstract architecture for rational agents. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92), Cambridge, MA, October 25–29, 1992*, pages 439–449. Morgan Kaufmann.

Ricci, A., Santi, A., and Piunti, M. (2014). CArtAgO (common atifact infrastructure for agents open environments).

Rodrigues, H. D. N., Santos, F. C. P., Dimuro, G., Adamatti, D. F., Jerez, E. M., and Dimuro, G. P. (2013). A mas for the simulation of normative policies of the urban vegetable garden of san jerónimo, seville, spain.

Rubiera, J. C., Lopez, J. M. M., and Muro, J. D. (2001). A fuzzy model of reputation in multi-agent systems. In *Proceedings of the fifth international conference on Autonomous agents*, pages 25 – 26.

Sabater, J. and Sierra, C. (2001). Regret: A reputation model for gregarious societies. In *Proceedings of the Fourth Workshop on deception Fraud and Trust in Agent Societies*, pages 61–70.

Santos, I. and Costa, A. C. R. (2012). Toward a framework for simulating agent-based models of public policy processes on the jason-cartago platform. In *Proceedings of the Second International Workshop on Agent-based Modeling for Policy Engineering in 20th European Conference on Artificial Intelligence (ECAI)- AMPLE 2012*, Montpellier. Montpellier University.

Santos, I., Mota, F. P., Costa, A. C. R., and Dimuro, G. P. (2012). Um framework para simulação de políticas públicas aplicado ao caso da piracema, sob o olhar da teoria dos jogos. Porto Alegre. SBC.

A Conceptual Middleware for Adaptive Sanctioning in Normative Multi-Agent Systems

Igor Conrado Alves de Lima[1], **Luis Gustavo Nardin**[2], **Jaime Simão Sichman**[3]

[1]IME – University of São Paulo, São Paulo, SP – Brazil

[2]CMCI – University of Idaho, Moscow, ID – USA

[3]Polytechnic School – University of São Paulo, São Paulo, SP – Brazil

igorcal@ime.usp.br, gnardin@uidaho.edu, jaime.sichman@poli.usp.br

Abstract. *The use of the normative approach to govern Multi-Agent Systems has been motivated by the increasing interest in balancing between agents' autonomy and global system control. In Normative Multi-Agent Systems (NMAS), despite the existence of norms specifying the rules of how agents ought or ought not to behave, agents have the autonomy to decide whether or not to act in compliance with them. A suitable way to govern agents is using sanction-based enforcement mechanisms. These mechanisms provide agents with a certain level of autonomy while controlling them through the application of sanctions. Here we present a conceptual middleware that makes use of an adaptive sanctioning enforcement model to improve the level of norm compliance in NMAS by enabling agents to choose among several categories of sanctions.*

1. Introduction

Multi-Agent Systems (MAS) are characterised by a set of heterogeneous agents that interact among themselves (e.g., cooperate, compete, and negotiate) in order to perform their tasks and achieve their goals [Wooldridge 2009]. An important characteristic of agents in MAS is the capability to operate autonomously. If not properly governed, however, such autonomy may result in degraded emerging properties of the system [Johnson et al. 2012].

For the last two decades, the normative approach has attracted the attention of the scientific community as a means to tackle the issue of MAS governance. Such interest is due to the fact that norms have been recognised as playing a key role in regulating humans' behaviours and maintaining the social order in the human society [Castelfranchi 1995, Castelfranchi 2000, Conte et al. 1998, Verhagen 2000, Boella et al. 2006, Boella et al. 2008, Hollander and Wu 2011, Andrighetto et al. 2013]. In agreement with Boella *et al.* [Boella et al. 2006], we refer to norms as "a principle of right action binding upon the members of a group and serving to guide, control, or regulate proper and acceptable behaviour", and to normative as a qualifier of something "conforming to or based on norms". Normative Multi-Agent Systems (NMAS) are then the integration of normative concepts into MAS. This approach has been motivated by the increasing interest in balancing between agents' autonomy and control in MAS [Verhagen 2000].

Norms by themselves, however, do not guarantee that agents will comply with them. Agents have autonomy to decide whether or not to comply with or violate such norms. A possible form to govern NMAS is by using enforcement mechanisms that

may motivate agents to comply with or prevent agents from violating norms. Here we propose the use of sanctioning as an enforcement mechanism. Sanction is a reaction or response to a norm violation or compliance used as a means to achieve social order [Castelfranchi 2000]. Sanctions may be either direct (material) or indirect (social). Direct sanctions have an immediate effect on the resources of the target agent (e.g., by imposing fines), whereas indirect sanctions may affect future interactions of the sanctioned agent (e.g., by affecting its reputation) [Cardoso and Oliveira 2009].

There are two complementary sanctioning approaches. One is the *trust and reputation* approach in which given two agents, A and B, A may sanction B by performing any action that, positively or negatively, affects B's reputation. The other is the *norm enforcement* approach in which a non-complaint behaviour is negatively sanctioned and a compliant behaviour positively sanctioned by the institutions in the NMAS [Nardin 2015].

A considerable amount of sanction enforcement mechanisms may be found in the literature. Some examples of these mechanisms are presented in [Cardoso and Oliveira 2009, Centeno et al. 2011, Centeno et al. 2013, Criado et al. 2013, Daskalopulu et al. 2002, De Pinninck et al. 2010, Luck et al. 2003, Mahmoud et al. 2012a, Mahmoud et al. 2012b, Modgil et al. 2009, Villatoro et al. 2011]. However, they lack the support of at least one of the following: multiple categories of sanctions; potential association of multiple sanctions with multiple norms; or the decision-making to determine the most adequate sanction to apply depending on different contextual factors. As shown in [Nardin 2015], these are all desirable requirements for sociotechnical systems, all of which we intend to address with the middleware proposed in this work.

Here we propose a middleware to enforce norms in NMAS using sanctions in order to achieve increased levels of norm compliance. The remaining sections are organised as follows: Section 2 briefly describes the adaptive sanctioning enforcement model used in the middleware; Section 3 introduces the proposed middleware; and Section 4 reports future works.

2. Sanctioning Model

Our conceptual middleware is based on the sanctioning enforcement model proposed by Nardin and colleagues [Nardin et al. 2016, Nardin 2015]. This enforcement model is based on the social approach in which the agents themselves are responsible for performing an adaptive and auto-organised peer control. For such purpose, agents are endowed with mechanisms to monitor their peers, assess their behaviours, and apply sanctions whenever necessary.

The model is comprised of a *sanctioning enforcement process* and a *sanctioning evaluation model*. The sanctioning enforcement process enables agents to reason about and adapt their behaviour regarding possible sanctions (Figure 1). It has four stages: (i) violation detection; (ii) sanction determination; (iii) sanctioning process; and (iv) assimilation. Five capabilities (Detector, Evaluator, Executor, Controller, and Legislator) enact these stages by using two data repositories (De Jure and De Facto).

The repositories are centralised and used to store information about norms and sanctions. The De Jure repository stores norms and sanctions, as well as links between

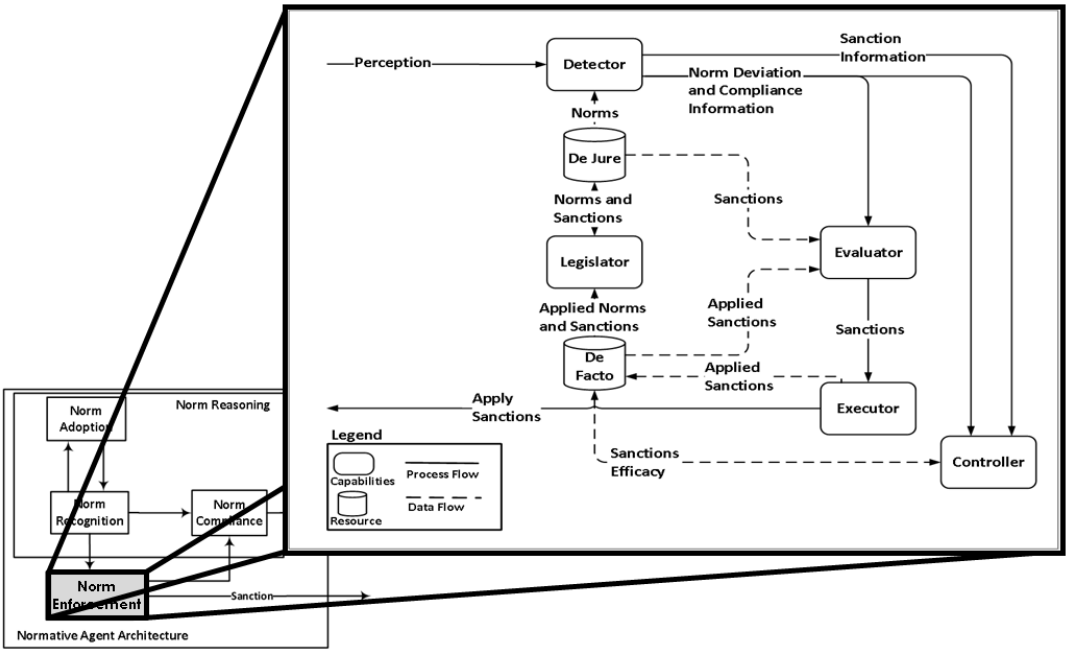


Figure 1. Sanctioning enforcement process model [Nardin 2015].

them, which are also known by the agents. One norm may be related to s different sanctions, whereas one sanction may be triggered by n different norms, as shown in Figure 2. The De Facto repository stores information about the applied sanctions and other relevant information used to assess their efficacy.

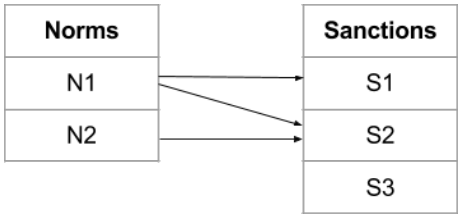


Figure 2. Relationship of norms and sanctions in the De Jure repository.

The five capabilities of the sanctioning enforcement process are:

- **Detector** – perceives the environment and detects any norm violation or compliance, and sanctions applied by other agents;
- **Evaluator** – obtains information from De Jure and De Facto to determine whether and which sanctions to apply;
- **Executor** – applies a sanction;
- **Controller** – monitors the outcomes of applied sanctions to evaluate their efficacy and records information about sanctions applied by other agents;
- **Legislator** – updates De Jure based on an assessment of De Jure and De Facto.

3. Proposed Middleware

Here we propose a middleware to implement the adaptive sanctioning enforcement model for NMAS briefly described in Section 2. This middleware will be developed as artefacts in the JaCaMo platform [Boissier et al. 2013].

The JaCaMo platform introduces a new promising programming paradigm called "multi-agent oriented programming" by integrating three orthogonal programming paradigms. It puts together agent-oriented programming, organisation-oriented programming, and environment-oriented programming in a synergistic way while preserving separation of concerns. A system built with JaCaMo is given by an agent organisation programmed in $\mathcal{M}$OISE$^+$ [Hübner et al. 2007], autonomous agents programmed in *Jason* [Bordini et al. 2007], and shared distributed artefact-based environments programmed in CArtAgO [Ricci et al. 2009]. Artefacts are building-blocks which provide services in addition to functions that make individual agents work together in a MAS and shape the agent environment according to the system needs. We chose JaCaMo due the high-level first-class support it provides for developing agents, environments, and organisations in synergy.

The middleware architecture is shown in Figure 3. The figure shows a couple of sample agents (*Agent A* and *Agent B*) in an environment. These agents may possess the five capabilities aforementioned (Detector, Evaluator, Executor, Controller, and Legislator) and are able to use the De Jure and De Facto repositories available on the environment. *Judge X* and *Legislator Y* represent two special artefacts provided by the middleware. *Judge X* is responsible for applying formal and direct sanctions. For example, suppose *Agent A* reports to *Judge X* a possible norm violation caused by *Agent B*. *Judge X*, based on De Jure and De Facto repositories, decides whether a sanction is appropriate and, if so, chooses which sanction to apply and its level of severity. *Legislator Y* is responsible for updating norms and sanctions in De Jure based on the assessment of the De Jure and De Facto repositories.

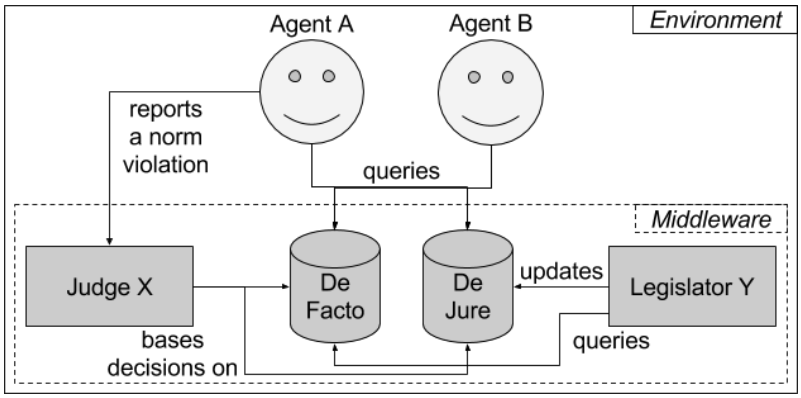


Figure 3. Middleware architecture.

The main advantage of our proposed middleware is the focus on flexibility and adaptability. By using our middleware, agents are free to choose the best way to sanction a violator or complier agent. In addition to being able to decide between formal or informal sanctions, they may also determine the level of severity of the informal ones. Due to the fact that these decisions are all dependent on the current context and historic facts, our middleware can, therefore, assure high level of flexibility and adaptability for norm enforcement in NMAS.

As direct consequence of the flexibility provided, the main disadvantage of our middleware is the limited control and predictability of the final results. As the sanctioning

mechanism depends on system's history and evolution, this influences how agents will learn and apply sanctions.

4. Future Works

This work is at an early stage of development and still in its conceptual phase. A first possible step would be to create a $\mathcal{M}$OISE$^+$ model, whose roles correspond to the five capabilities of the normative enforcement process. A next step would be to perform tests with the current architecture. Next, we intend to decentralise the De Facto repository meaning that every agent will have its own De Facto while De Jure remains centralised like the "penal code" in an environment-level of the NMAS.

Acknowledgements

This work has been supported by CNPq - Brazil.

References

Andrighetto, G., Governatori, G., Noriega, P., and van der Torre, L. W. (2013). *Normative multi-agent systems*, volume 4. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Boella, G., Torre, L., and Verhagen, H. (2008). Introduction to the special issue on normative multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 17(1):1–10.

Boella, G., Van Der Torre, L., and Verhagen, H. (2006). Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, 12(2-3):71–79.

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.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. John Wiley & Sons.

Cardoso, H. L. and Oliveira, E. (2009). Adaptive deterrence sanctions in a normative framework. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*, pages 36–43. IEEE Computer Society.

Castelfranchi, C. (1995). *Cognitive and social action*. Psychology Press.

Castelfranchi, C. (2000). Engineering social order. In *Engineering societies in the agents world*, pages 1–18. Springer.

Centeno, R., Billhardt, H., and Hermoso, R. (2011). An adaptive sanctioning mechanism for open multi-agent systems regulated by norms. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 523–530. IEEE.

Centeno, R., Billhardt, H., and Hermoso, R. (2013). Persuading agents to act in the right way: An incentive-based approach. *Engineering Applications of Artificial Intelligence*, 26(1):198–210.

Conte, R., Castelfranchi, C., and Dignum, F. (1998). *Autonomous norm acceptance*. Springer.

Criado, N., Argente, E., Noriega, P., and Botti, V. (2013). MaNEA: A distributed architecture for enforcing norms in open MAS. *Engineering Applications of Artificial Intelligence*, 26(1):76–95.

Daskalopulu, A., Dimitrakos, T., and Maibaum, T. (2002). Evidence-based electronic contract performance monitoring. *Group decision and negotiation*, 11(6):469–485.

De Pinninck, A. P., Sierra, C., and Schorlemmer, M. (2010). A multiagent network for peer norm enforcement. *Autonomous Agents and Multi-Agent Systems*, 21(3):397–424.

Hollander, C. D. and Wu, A. S. (2011). The current state of normative agent-based systems. *Journal of Artificial Societies and Social Simulation*, 14(2):6.

Hübner, J. F., Sichman, J. S., and Boissier, O. (2007). Developing organised multiagent systems using the $\mathcal{M}$OISE$^+$ model: programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering*, 1(3-4):370–395.

Johnson, M., Bradshaw, J. M., Feltovich, P. J., Jonker, C., Van Riemsdijk, B., and Sierhuis, M. (2012). Autonomy and interdependence in human-agent-robot teams. *Intelligent Systems, IEEE*, 27(2):43–51.

Luck, M. et al. (2003). Modelling norms for autonomous agents. In *Computer Science, 2003. ENC 2003. Proceedings of the Fourth Mexican International Conference on*, pages 238–245. IEEE.

Mahmoud, S., Griffiths, N., Keppens, J., and Luck, M. (2012a). Efficient norm emergence through experiential dynamic punishment. In *ECAI*, volume 12, pages 576–581.

Mahmoud, S., Villatoro, D., Keppens, J., and Luck, M. (2012b). Optimised reputation-based adaptive punishment for limited observability. In *Self-Adaptive and Self-Organizing Systems (SASO), 2012 IEEE Sixth International Conference on*, pages 129–138. IEEE.

Modgil, S., Faci, N., Meneguzzi, F., Oren, N., Miles, S., and Luck, M. (2009). A framework for monitoring agent-based normative systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 153–160. International Foundation for Autonomous Agents and Multiagent Systems.

Nardin, L. G. (2015). *An Adaptive Santioning Enforcement Model for Normative Multiagent Systems*. PhD thesis, Universidade de São Paulo.

Nardin, L. G., Balke, T., Ajmeri, N., Kalia, A. A., Sichman, J. S., and Singh, M. P. (2016). Classifying sanctions and designing a conceptual sanctioning process model for socio-technical systems. *The Knowledge Engineering Review*, 31(2):142–166.

Ricci, A., Piunti, M., Viroli, M., and Omicini, A. (2009). Environment programming in CArtAgO. In *Multi-Agent Programming:*, pages 259–288. Springer.

Verhagen, H. (2000). *Norm autonomous agents*. PhD thesis, Citeseer.

Villatoro, D., Andrighetto, G., Sabater-Mir, J., and Conte, R. (2011). Dynamic sanctioning for robust and cost-efficient norm compliance. In *IJCAI*, volume 11, pages 414–419.

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

Integrando Requisitos Organizacionais à Modelagem de Sistemas Multiagente Normativos

Emmanuel Sávio Silva Freire[1] e Mariela Inés Cortés[2]

[1]Instituto Federal do Ceará (IFCE)
Rua Deoclécio Lima Verde, s/n – 63.507-110 – Iguatu – CE – Brasil
savio.essf@gmail.com

[2]Universidade Estadual do Ceará (UECE)
mariela@larces.uece.br

Abstract. *Normative multiagent systems are used to development complex systems. Because this, many languages and techniques are proposed to help the modelling and programming phases of these systems. However, the gathering and analysis of requirements phases is essential for a best understanding of users' needs. In this context, this paper addresses a integration between gathering and analysis of requirements phase and modelling phase. Thus, a mapping between an istar framework and NorMAS-ML language is proposed. Because it is a preliminary work, it doesn't have results to compare with related work.*

Resumo. *Os sistemas multiagente normativos são utilizados para o desenvolvimento de sistemas complexos. Para isso, muitas linguagens e técnicas foram propostas para auxiliar as fases de modelagem e implementação destes sistemas. Entretanto, a fase de levantamento e análise de requisitos é essencial para um melhor entendimento das necessidades dos usuários. Neste contexto, este artigo aborda a integração entre as fases de levantamento e especificação de requisitos e a fase de modelagem. Para tanto, um mapeamento entre o framework istar e a linguagem NorMAS-ML é proposta. Vale ressaltar que, por se tratar de um trabalho preliminar, ainda não se dispõem de resultados para comparação com trabalhos relacionados.*

1. Introdução

Os sistemas multiagente normativos (SMAN's) são formados por um conjunto de agentes inteligentes que interagem entre si para alcançarem os seus objetivos. Entretanto, necessitam observar as normas definidas nesses sistemas para verificar quais as ações que são permitidas, obrigadas ou proibidas de serem executadas [Silva, Braga e Figueiredo, 2010].

Neste contexto, linguagens de modelagem [Freire et. al, 2012] [Silva, Braga e Figueiredo, 2010] e de implementação [Lopes et.al, 2011] [Rocha Jr, Freire e Cortés, 2013] foram propostas para auxiliar o desenvolvimento de tais sistemas. Dentre elas, destaca-se a linguagem NorMAS-ML. Por meio dela, é possível modelar todas as entidades e as características desses sistemas. Entretanto, essa linguagem, mesmo tendo uma ferramenta de modelagem, não dá suporte a fase de levantamento e análise de requisitos.

Segundo Sommerville (2007), a fase de levantamento e análise de requisitos propicia um melhor entendimento das reais necessidades dos usuários do sistema. Assim, é possível detalhar e priorizar os requisitos permitindo que aqueles essenciais ao sistema possam ser implementados inicialmente. Com isso, o usuário tem uma maior probabilidade de receber um sistema que resolva as suas necessidades.

Neste sentido, o *framework istar* (i*) [Yu, 1995] foi proposto para auxiliar a fase de requisitos por meio de um modelo conceitual composto pelo: (i) modelo de dependência estratégica e (ii) modelo estratégico de razão. Por meio deles, os engenheiros de requisitos conseguem modelar os *stakeholders* como atores e suas intenções como objetivos. Assim, consegue-se obter uma compreensão sobre os relacionamentos da organização e sobre as razões envolvidas nos processos de decisão [Yu, 1995].

No contexto de SMAN's, a metodologia Tropos utiliza o *framework istar* para a fase de requisitos integrada com a fase de modelagem. Entretanto, Tropos não permite a modelagem das características das normas presentes nos SMAN's. Assim, o presente artigo propõe a integração do *framework istar* (fase de requisitos) com a linguagem NorMAS-ML, que permite a modelagem de SMAN's, possuindo uma ferramenta de modelagem integrada juntamente com uma linguagem de implementação (JAMDER 2.0 [Rocha Jr, Freire e Cortés, 2013]).

O presente artigo está estruturado como segue: a Seção 2 apresenta o referencial teórico acerca do *framework istar* e da linguagem NorMAS-ML. A Seção 3 discute como a fase de requisitos é incluída no processo de desenvolvimento de SMAN's juntamente com a proposta de integração e mapeamento de *istar* e NorMAS-ML. Finalmente, a Seção 4 apresenta as conclusões e os trabalhos futuros.

2. Referencial Teórico

2.1. O *Framework istar*

O *framework istar* [Yu, 1995] possui uma estrutura conceitual para reconhecer motivações e intenções sobre as características de um processo. Com isso, auxilia no processo de levantamento e análise de requisitos. Ele é composto por dois modelos: (i) de Dependência Estratégica (SD), que fornece uma descrição intencional do processo em termos de uma rede de relacionamentos de dependência entre atores do ambiente, e (ii) Estratégico da Razão (SR) que apresenta uma descrição estratégica do processo, fornecendo uma análise dos meios para permitir que os objetivos possam ser cumpridos por meio das contribuições dos atores. Os dois modelos são utilizados nas fases iniciais do Tropos para capturar as intenções dos *stakeholders* e as responsabilidades dos sistemas.

2.2. A Linguagem NorMAS-ML

A linguagem NorMAS-ML (*Normative Multi-Agent Systems Language*) foi definida por Freire et. al (2012) por meio da extensão da linguagem MAS-ML [Silva, Choren e Lucena, 2008] por meio da inclusão dos conceitos estáticos das normas [Silva, Braga e Figueiredo, 2010]. Vale ressaltar, que todas essas linguagens são extensões de UML. Por meio dela, é possível modelar as características dos SMAN's juntamente com os

aspectos estáticos das normas. Adicionalmente, NorMAS-ML possui uma ferramenta de modelagem capaz de gerar código a partir dos seus modelos.

3. Discussão

Por meio de NorMAS-ML, pode-se gerar modelos para representar os SMAN's, tanto na fase de modelagem quanto na fase de implementação. Entretanto, essa linguagem não dá suporte a fase de levantamento e análise de requisitos. Segundo Sommerville (2007), a fase de requisitos propicia um maior entendimento do sistema a ser desenvolvido, pois o analista de requisitos pode interagir com os usuários para conhecer as suas necessidades.

Neste sentido, Yu (1995) propos o *framework istar* que permite suporte à fase de requisitos por meio da identificação das necessidades de cada um dos *stakeholders* juntamente com o motivo para cada uma delas. Assim, Tropos inclui os modelos apresentados no *framework istar* na fase inicial de desenvolvimento. Adicionalmente, Melo et. al (2015) também apresentou um mapeamento do *framework istar* para modelos da UML, mais especificamente, para o diagrama de classes.

Como NorMAS-ML é uma extensão de UML, infere-se que se pode fazer um mapeamento entre *istar* e essa linguagem. Entretanto, na literatura, apenas trabalhos relacionando *istar* com SMA's são apresentados. Assim, precisa-se incluir o conceito normativo presente em NorMAS-ML. Segundo Silva, Braga e Figueiredo (2010), as normas são restrições sobre as ações dos agentes, indicando quais as ações que podem, devem ou não devem ser executadas. Por outro lado, Sommerville (2007) afirma que a especificação de sistemas necessita dos requisitos funcionais e não funcionais. O primeiro está relacionado com a funcionalidade em si, enquanto o segundo está relacionado com restrições funcionais do sistema. Assim, poderia pensar em mapear as normas como requisitos não funcionais.

Entretanto, ao analisar o conceito de regras de negócio, Sommerville (2007) indica que as regras são restrições impostas pelo processo de negócio da organização que irá utilizar o sistema. Logo, comparando este conceito com o de normas, pode-se verificar a semelhança entre eles. Portanto, as normas podem ser mapeadas como regras de negócio.

Adicionalmente, é necessário verificar como o mapeamento será realizado para as entidades presentes na linguagem. Vale ressaltar que Melo et. Al (2015) apresentou a formalização do mapeamento entre *istar* e o diagrama de classes da UML. Logo, como NorMAS-ML possui um diagrama de classes, pode-se utilizar a mesma estratégia utilizada por Melo et. al (2015). Por outro lado, as entidades ambiente, organização e agente presentes na linguagem precisam ser representadas em *istar*.

Neste contexto, pode-se definir pontos de extensão em *istar* considerando os conceitos de NorMAS-ML. Para tanto, precisa-se analisar a estrutura do *framework* e da linguagem e compará-las com o intuito de identificar as semelhanças existentes. Em seguida, deve-se verificar quais alterações devem ser realizadas no *framework* e/ou na linguagem e, finalmente, verificar a compatibilidade do mapeamento e da extensão. Para a validação da extensão, deve-se utilizar o estudo de caso modelado por [Freire et. al (2012) utilizando NorMAS-ML juntamente com os requisitos representados em *istar* analisados por Santos (2008). Por se tratar de um mesmo estudo de caso, pode-se modelá-lo com a extensão proposta para verificar a compatibilidade.

4. Conclusão e Trabalhos Futuros

Este artigo apresentou uma proposta de integração entre as fases de requisitos e de modelagem para permitir um melhor entendimento e modelagem de SMAN's. Para tanto, é proposta a utilização do *framework istar* juntamente com a linguagem NorMAS-ML para essa integração. Decidiu-se utilizar *istar* por conta da sua relevância para a análise e representação de requisitos. Por outro lado, resolveu-se escolher NorMAS-ML por conta da sua estrutura baseada em UML que permite a modelagem as entidades típicas dos SMAN's e possui ferramenta de modelagem integrada com um *framework* para desenvolvimento. Adicionalmente, espera-se, como trabalhos futuros: (i) analisar e integrar as entidades de NorMAS-ML e do *istar*, (ii) formalizar a integração por meio de ontologias, e (iii) verificar a consistência da extensão por meio de um estudo de caso.

Referências

Freire, E. S. S., Cortés, M. I., Gonçalves, E. J. T., 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). Proceedings of the 14th International Conference on Enterprise Information Systems.

Lopes, Y. S., Gonçalves, E. J. T., Cortés, M. I., Freire, E. S. S. (2011). Extending JADE Framework to Support Different Internal Architectures of Agents. In: The Ninth European Workshop on Multi-agent Systems (EUMAS), 2011, Maastricht (The Netherlands). Proceedings of the Ninth European Workshop on Multi-agent Systems.

Melo, J., Sousa, A., Agra, C., Júnior, J., Castro, J., Alencar, F. (2015). Formalization of Mapping Rule form Istar to Class Diagram in UML. In: 29th Brazilian Symposium on Software Engineering, 2015.

Rocha Jr., R. M., Freire, E. S. S., Cortés, M. I. (2013). Estendendo o Framework Jamder para Suporte à Implementação de Sistemas Multi-Agente Normativos. In: IX Simpósio Brasileiro de Sistemas de Informação (SBSI), 2013, João Pessoa. Anais do IX Simpósio Brasileiro de Sistemas de Informação (SBSI).

Santos, B. S. (2008). IStar Tool - Uma proposta de ferramenta para modelagem de i*. Dissertação de Mestrado. Recife: Universidade Federal de Pernambuco, Centro de Informática.

Silva, V., Braga, C., 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.

Silva, V. T., Choren R., Lucena, C. (2008). MAS-ML: a multi-agent system modelling language, In IJAOSE, Modeling Lang. for Agent Systems,(2)4pp.382-421.

Sommerville, I. (2007). Engenharia de Software. 8 ed. São Paulo: Pearson Addison-Wesley.

Yu, E. (2002). Agent-Oriented Modelling: Software Versus World, In: Proceedings of the Agent-Oriented Software Engineering (AOSE'01), Edited by Wooldridge, M., Weiss, G. and Ciancarini, P., LNAI, Vol. 2222, Springer-Verlag, p. 206 – 225.

Rede Bayesiana de Emoções e sua Implementação em um Jogo Computacional baseado em Agentes

Gustavo Carneiro Fleck[1], Adriano Werhli[1], Diana F. Adamatti[1]

1 – Laboratório de Simulação Social e Ambiental – Centro de Ciências Computacionais

Universidade Federal de Rio Grande (LAMSA/C3/FURG) – Rio Grande – RS

gustavofleck@furg.br, werhli@gmail.com, dianaada@gmail.com

Resumo. *Este artigo apresenta um estudo de caso onde é criado um jogo simples, onde os agentes têm personalidades baseadas no modelo de redes Bayesiana de emoções. Os resultados mostram que, dependendo as personalidades definidas, diferentes situações ocorrem, e com isso a jogabilidade é alterada.*

Palvras-chave: *Sistemas Multiagentes, Rede Bayesiana de Emoções, Modelo OCC*

1. Introdução

A emoção humana é alvo de inúmeros estudos em diversas áreas do conhecimento, sendo o meio computacional uma delas. A Inteligência Artificial (IA) é uma área multidisciplinar que visa a simulação da capacidade humana de pensar, tomar decisões, resolver problemas e nesse caso, sentir. Os Sistemas Multiagentes oferecem a possibilidade de simular várias dessas situações através da interação, de agentes e o meio, resultando em uma representação, aproximada, do comportamento humano.

A Rede Bayesiana é uma ótima ferramenta a ser usada nessa simulação, pois nos oferece um raciocínio probabilístico onde podemos adicionar a imprevisibilidade ao agente, e assim obter uma representação mais aproximada, da mente humana.

Este artigo tem como objetivo, através de uma Rede Bayesiana baseada no modelo OCC de emoções anteriormente definida, desenvolver um jogo que conta com um agente com diferentes personalidades, a fim de simular um comportamento próximo da mente humana, mostrar a diferença de desempenho, e as emoções com maior relevância em cada personalidade.

O artigo está estruturado da seguinte maneira: a seção 2 apresenta o referencial teórico desse estudo. A seção 3 explica o estudo de caso implementado e os resultados obtidos. Na seção 4 são as conclusões e os trabalhos futuros.

2. Referencial Teórico

2.1. Emoções e o Modelo OCC

As emoções são algo ainda incompreendido, de certa forma, na ciência, o que torna sua simulação algo muito mais complicado. A mente humana é imprevisível e é impossível criar um padrão para as emoções, já que todo o ser humano age de forma diferente ao receber os mesmos estímulos.

O modelo OCC de emoções foi proposto no livro “The Cognitive Structure of Emotions” (Ortony et al., 1988) por Ortony, Clore e Collins. Esse modelo é capaz de identificar a partir de estímulos gerados em um ambiente, as emoções que serão sentidas pelo agente. Ele usa três tipos de geradores de estímulos: eventos, agentes e objetos. Toda emoção gerada é resultado de um ou mais estímulos. O modelo é composto por 22 emoções, onze positivas e onze negativas, como mostra a Figura 1, e se baseia na diferenciação das reações de valências positivas e negativas, ou seja, a partir de um evento, variáveis são atribuídas a fim de gerar uma emoção positiva ou negativa.

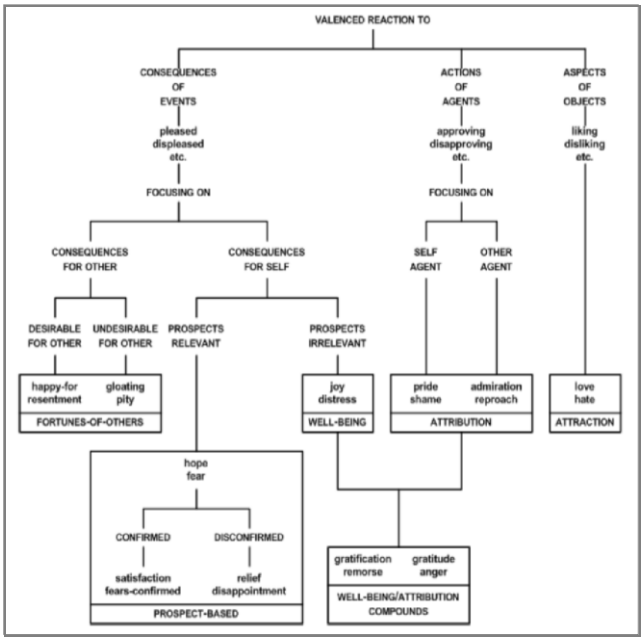


Figura 1. Estrutura do Modelo OCC (Ortony et al., 1988)

2.2. Redes Bayesianas

Tendo em vista as dificuldades encontradas para a modelagem de problemas reais, tais como falta de dados, impossibilidade de coleta e até mesmo imprecisão dos mesmos, optou-se pelo uso de um método que utiliza o raciocínio probabilístico, acreditando-se ser uma boa alternativa. As redes Bayesianas são modelos gráficos para raciocínio baseado na incerteza. Podemos defini-las como sendo uma combinação da Teoria Probabilística e a Teoria dos Grafos (Pearl, 1988). O tipo de probabilidade utilizado nas redes Bayesianas é o condicional. Esse tipo de probabilidade é o que depende de acontecimentos anteriores. Representa-se uma probabilidade condicional por P (A|B), que significa a probabilidade de que o evento A ocorra dado a ocorrência de um evento B. As redes Bayesianas são compostas por nós, arestas, tem direção estabelecida e não possuem ciclos. Cada nó, ou nodo, representa a variável e as arestas assumem as probabilidades de ocorrer o evento.

2.3. Sistemas Multiagentes

De mesma forma como nas organizações humanas as atividades, muitas vezes, são realizadas por um grupo de pessoas que trabalham de modo cooperativo, onde existem decisões individuais que afetam o grupo, em sistemas Multiagentes (SMA) as pessoas

são representadas por agentes artificiais, os quais se relacionam em um ambiente de forma a buscar soluções para problemas de forma cooperativa, compartilhando informações, evitando conflitos e coordenando a execução de atividades (Adamatti, 2003). Um agente é uma entidade computacional com um comportamento autônomo que lhe permite decidir suas próprias ações (Alvares e Sichman, 1997). Tratando-se de Sistemas Multiagentes, o fato dos agentes serem autônomos designa o fato dos agentes terem existência própria, independentemente da existência de outros agentes. As aplicações desenvolvidas sobre um SMA têm o objetivo de simular situações reais. Em uma simulação baseada em agentes, o fenômeno real é decomposto em um conjunto de elemento e suas interações. Para cada elemento é modelado como um agente, resultando em um modelo geral onde os agentes interagem entre si e com o ambiente (Frozza, 1997).

2.4. Modelo de Rede Bayesiana de Emoções

A Rede Bayesiana de Emoções foi desenvolvida por Neves (2014) com base no modelo OCC, levando em consideração o fato de possuir uma estrutura de simples tradução computacional e por ser um modelo bastante abrangente, e com isso adicionando a imprevisibilidade necessária nas emoções. Seu funcionamento é o mesmo do modelo OCC.

O modelo proposto por Neves (2014) é apresentado na Figura 3.

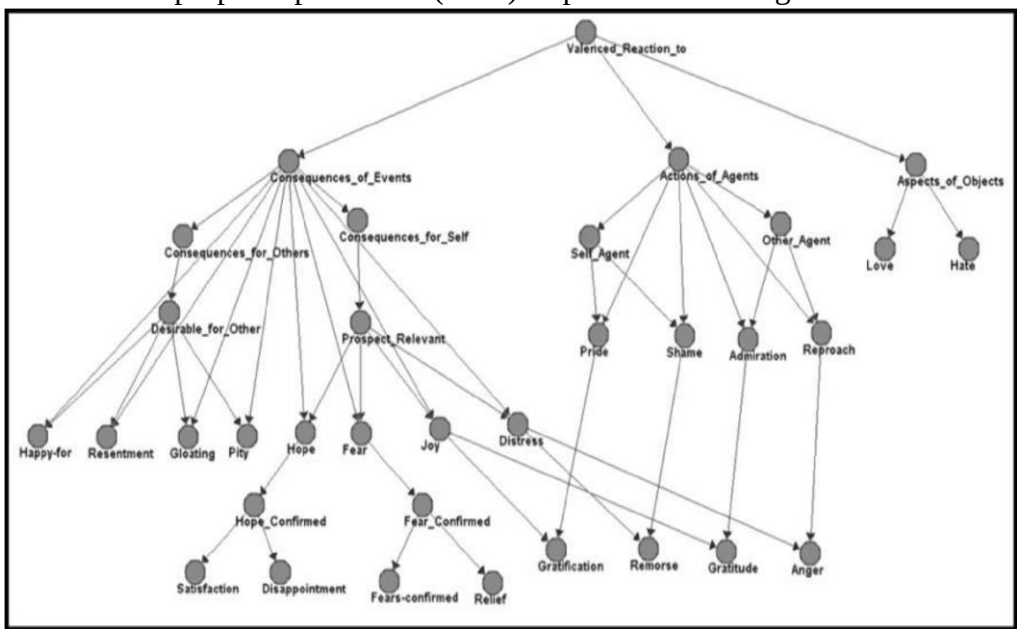


Figura 3. Rede Bayesiana de emoções (Neves, 2014).

3. Estudo de Caso

Após criação da rede Bayesiana e da definição das probabilidades, é necessário adicionar a rede Bayesiana de emoções a um ambiente multiagentes de forma a validá-la e propiciar a avaliação de sua eficácia. Então foi desenvolvido um cenário onde pode-se adicionar vários eventos e analisar o comportamento do agente.

Para este trabalho foi escolhido criar um jogo onde os NPCs (Non-Player Character) são tratados como agentes, e a partir disso, implementar a Rede Bayesiana de Emoções, de forma a gerar um sistema Multiagente mais complexo e com mais variáveis a serem analisadas. A Rede Bayesiana de emoções foi implementada junto ao código principal do jogo, que está na linguagem JAVA, para gerar emoções no NPC e mostrar as diferenças que essa rede pode trazer, fazendo com que o jogo, que à primeira vista seria igual em todas as situações, tenha a imprevisibilidade do estado emocional do NPC. Mudando assim a jogabilidade e muitas vezes o caminho para chegar ao objetivo final.

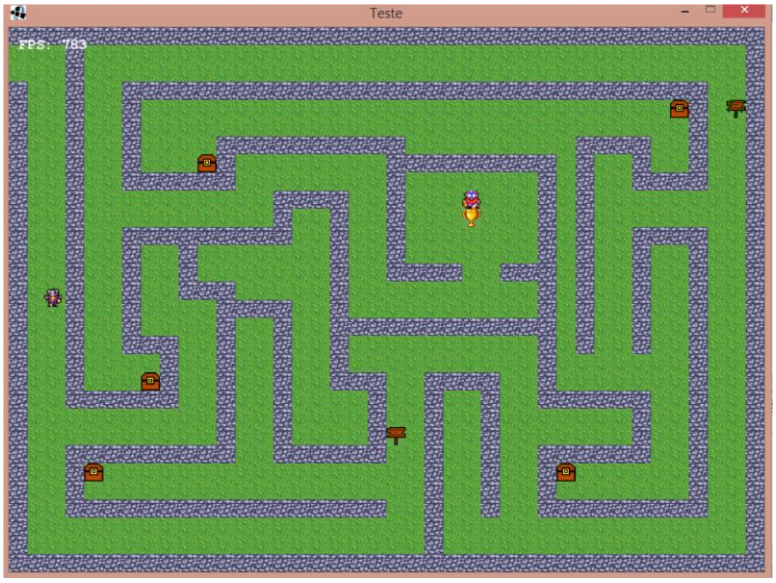


Figura 4. Interface do labirinto.

No jogo proposto, o jogador tem que chegar ao fim de um labirinto, onde ele pode encontrar eventos que alteram seu humor. Ao encontrar um jogador (humano), o agente (NPC) pode ser estimulado positivamente ou negativamente, dependendo do objetivo definido para ele. Também pode se deparar com duas placas que revelam sua posição em relação ao fim do labirinto, onde o jogador alcança seu objetivo, que também alteram seu comportamento. A Figura 4 apresenta a interface do jogo proposto (labirinto). Quando o NPC não está sobre o efeito de nenhuma emoção, ele tem uma velocidade moderada ao caminhar. Sobre o efeito de emoções positivas essa velocidade aumenta, e na presença de emoções negativas ela diminui e pode haver uma certa desorientação no agente. Para a análise de desempenho do agente sobre o efeito de cada personalidade, levou-se em conta o tempo necessário para alcançar o objetivo, onde quanto mais negativas suas emoções, maior o tempo gasto para chegar ao fim do labirinto.

3.1 Resultados

Para definir as diferentes personalidades dos agentes foram feitas mudanças nas probabilidades das emoções base utilizadas na rede Bayesiana proposta em Neves (2014). As emoções consideradas positivas tiveram suas probabilidades de ocorrer aumentadas, dentro do código fonte da rede Bayesiana, de 95% para 99%, para a personalidade **otimista**, e as emoções consideradas ruins foram aumentadas de 95%

para 99%, para a personalidade **pessimista**. A personalidade **neutra** não sofreu nenhuma alteração, pois a probabilidade de emoções positivas deve ser a mesma de emoções negativas.

Com essas pequenas mudanças, pode-se observar uma real diferença entre as personalidades, tendo a personalidade neutra como base, quando o agente é definido como pessimista, ele se mostra menos eficaz no desempenho de sua tarefa. O contrário acontece ao se atribuir a personalidade otimista. Foi possível analisar a relação das emoções sentidas em cada personalidade com a conclusão da tarefa proposta para o agente. Todos os resultados obtidos são uma média de vinte execuções do jogo. Na figura 5, é possível demonstrar a diferença do tempo de execução em cada personalidade, onde as personalidades **otimista, pessimista e neutra** demoraram 18, 35 e 23 segundos, respectivamente.

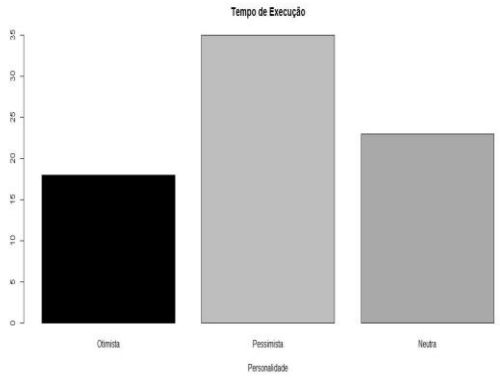


Figura 5. Tempo por personalidades.

Na figura 6 pode-se ver a diferença do somatório total das emoções por personalidade. O agente **otimista** teve um somatório total de 0.96099363778; o **pessimista** de 0.61085707054; e o agente sobre o efeito da personalidade **neutra** teve um somatório total de emoções de 0.710249673273. Esse somatório é obtido através da soma de todas emoções positivas e negativas geradas durante a execução e quanto maior o número, maior a intensidade das emoções positivas.

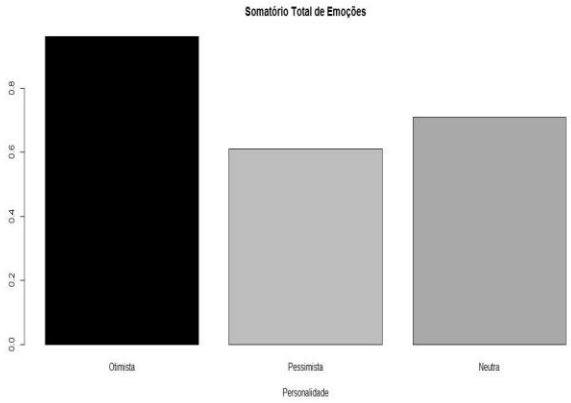


Figura 6. Somatório das emoções por personalidade.

5. Conclusões

Através da modelagem de emoções em sistemas multiagentes foi desenvolvido um jogo simples, onde as emoções do agente (NPC) alteram a forma como o jogo é executado. As emoções definidas para os agentes buscam, cada vez mais, uma proximidade com as emoções expressadas pelas pessoas. Através da utilização de um modelo de emoções é possível criar algoritmos para tomada de decisão de agentes artificiais.

A Rede Bayesiana foi aplicada com sucesso no cenário do labirinto, e se mostrou uma ótima alternativa para a modelagem de emoções em softwares que utilizem o Java como principal linguagem de programação.

O Sistema Multiagente desenvolvido (jogo) é formado por agentes cognitivos, e serve para simular as ações de um agente conforme as mudanças que ocorrem no ambiente. Utilizando os agentes cognitivos, é possível ter um histórico de ações (memória) e através deste histórico, pode-se avaliar emoções positivas e negativas geradas no agente por um estímulo proveniente do ambiente.

Como trabalhos futuros, pretende-se realizar mais testes como personalidades em cenários mais complexos, como jogos com mais NPCs e regras, e Interface Homem Máquina (adaptação da interface as emoções do usuário).

6. Referências

ADAMATTI, D. F. **AFRODITE - Ambiente de Simulação Baseado em Agentes com Emoções**. Dissertação de Mestrado. Universidade Federal do Rio Grande do Sul. Porto Alegre. 2003.

ALVAREZ, L. O.; SICHMAN, J. Introdução aos Sistemas Multiagentes. **Jornada de Atualização em Informática**, Brasília, 1997. 1-38.

COZMAN, F. G. Bayesian Networks in Java: User manual and download. **JavaBayes**, 2001. Disponivel em: <http://www.cs.cmu.edu/~javabayes/Home/index.html>. Acesso em: 9 Abril 2013.

NEVES, F. S. **Modelagem de Emoções Usando Redes Bayesianas**. Dissertação de Mestrado. Universidade Federal do Rio Grande. Rio Grande, 2014.

FROZZA, R. **SIMULA: Ambiente para Desenvolvimento de Sistemas Multiagentes Reativos**. Dissertação de Mestrado. Universidade Federal do Rio Grande do Sul. Porto Alegre. 1997.

ORTONY, A.; CLORE, G.; COLLINS, A. **The Cognitive Structure of Emotions**. Cambridge: Cambridge University Press, 1988.

PEARL, J. **Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference**. 1a. ed: Morgan Kaufmann, 1988.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence A Modern Approach**. 2a. ed. Upper Saddle River, New Jersey: Prentice Hall, 2003.

WERHLI, A. V. **Reconstruction of Gene Regulatory Networks from Postgenomic Data**. Tese

Modelando a Curva de Crescimento do *Mycobacterium tuberculosis* com a utilização de simulação multiagente: um estudo de caso para a variável *Quorum sensing*

Marcilene Fonseca de Moraes[1], Diana F. Adamatti[1], Albano Oliveira de Borba[2], Adriano Velasque Werhli[2]

[1]Programa de Pós-Graduação em Modelagem Computacional – Universidade Federal do Rio Grande (FURG)
Caixa Postal 474 – 96.203-900 – Rio Grande – RS – Brazil

[2]Centro de Ciências Computacionais– Universidade Federal do Rio Grande (FURG)
Caixa Postal 474 – 96.201-900 – Rio Grande – RS – Brazil

{marcilenemoraes,dianaadamatti}@furg.br,{albano.b06,werhli}@gmail.com
@gmail.com

Abstract. *This paper describes a proposal for a study of the growth curve of the tuberculosis and aims to simulate the curve with minimum error possible comparing to experimental (real data) and still show that the use of the concept of Quorum sensing is essential in this process. To develop the model, were created an environment based on agents, where the variables of them use probability distributions.*

Resumo. *Este trabalho descreve uma proposta de estudo da curva de crescimento do Mycobacterium tuberculosis e tem como objetivo simular a curva com um padrão de crescimento semelhante em relação a experimental (dados reais) e ainda mostrar que o uso do conceito de Quorum sensing é essencial para que isso aconteça. Para implementar o modelo foi criado um ambiente baseado em agentes, onde as variáveis desses agentes utilizam distribuições de probabilidades.*

1. Introdução

A tuberculose (TB) é um grande problema de saúde pública, que afeta predominantemente países de baixa e média renda, se desenvolvendo também entre imigrantes, partes mais pobres e vulneráveis de países ricos (Lönnroth et al., 2015). Segundo Burgos e Pym (2002), o *Mycobacterium tuberculosis*, causador da tuberculose, é um dos patógenos bacterianos mais bem sucedidos na história da humanidade.

Diante destas circunstâncias, o estudo da curva de crescimento do *Mycobacterium tuberculosis* torna-se extremamente importante, já que através deste estudo podem-se testar hipóteses, verificar reações do bacilo a fármacos e ainda pode ajudar no desenvolvimento de novos (Von Groll, 2010).

No entanto, o bacilo da tuberculose possui uma taxa de crescimento populacional muito lenta, esse comportamento faz com que experimentos *in vitro* sejam algo demorado e de alto custo.

Neste contexto, surgem os sistemas multiagentes, área da inteligência artificial que permite, por meios de suas ferramentas, simular regras de comportamento de determinado sistema computacionalmente.

De acordo com Wooldridge (2009), os sistemas multiagentes são uma poderosa e flexível ferramenta para modelagem deste tipo de sistema, pois desta maneira pode-se analisar o comportamento de cada indivíduo, ao invés de uma média dos comportamentos do mesmo.

Por muitos anos, pesquisadores acreditavam que as bactérias existiam como células individuais, agiam como populações de células independentes e se reproduziam quando em condições favoráveis. No entanto, nas últimas décadas diversos estudos mostraram que esses microrganismos podem se comunicar através do *Quorum sensing*.

O mecanismo *Quorum sensing* baseia-se na produção e difusão de pequenas moléculas sinalizadoras que podem ser detectadas pelas bactérias. Esse processo acontece quando há percepção de alta densidade celular, possibilitando a toda população iniciar uma ação, uma vez que concentração crítica tem sido alcançada (Whitehead et al., 2001).

Este trabalho modela a curva de crescimento do bacilo da tuberculose, utilizando sistemas multiagentes, onde as variáveis da simulação utilizam distribuições de probabilidades, tornando assim, o modelo desenvolvido mais similar ao modelo de crescimento real. E tem como objetivo, elucidar que a utilização do conceito de *Quorum sensing* na modelagem da curva do *Mycobacterium tuberculoses* possibilita a obtenção de um comportamento mais próximo, quando comparado à curva real.

2. Curva de Crescimento do *Mycobacterium tuberculosis*

Quando o *Mycobacterium tuberculosis* é inoculado em um meio que contenha todos os nutrientes necessários para sua duplicação, ele tende a multiplicar-se.

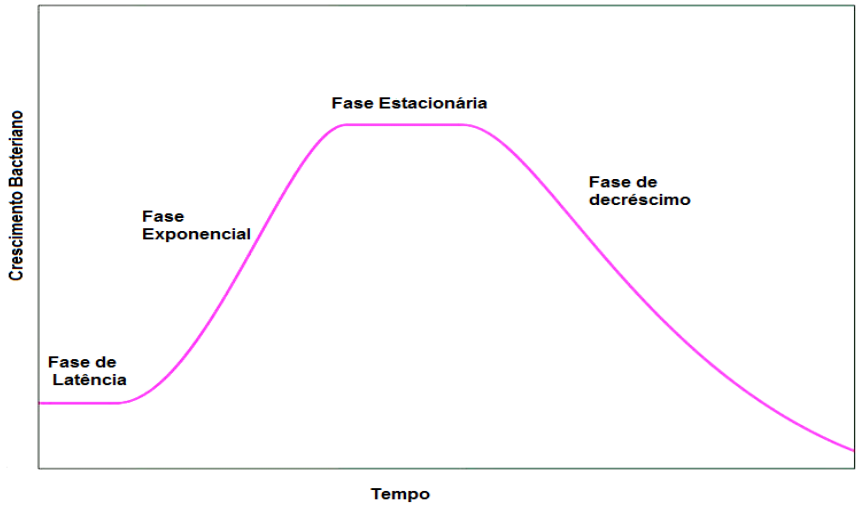


Figura 1. Curva de crescimento e suas fases (Todar, 2013).

As bactérias inicialmente se ajustam ao novo meio (fase de latência) até que elas possam começar o processo de divisão regularmente (fase exponencial). Quando seu crescimento tornasse limitado, as células param de se dividir (fase estacionária), até que finalmente eles morrem pela inviabilidade do ambiente (Todar, 2013).

3. Modelo de Crescimento Proposto

Quando as curvas de crescimento são obtidas através de experimentos *in vitro*, as informações que essas curvas apresentam são o produto de diversos fatores da dinâmica populacional. Logo, não é possível extrair informações isoladas, como quanto as bactérias consomem, ou quanto deixam de consumir depois de atingida a saturação do ambiente, ou ainda qual a proporção de moléculas sinalizadoras *(Quorum sensing*) são necessárias para decretarem a saturação.

Diante dessas circunstâncias, foi necessário inferir quanto as variáveis que interferem no crescimento populacional, tendo como única referência à observação dos resultados obtidos.

Para simular a dinâmica populacional foi utilizado o ambiente de programação NetLog. O modelo baseado em agentes implementado simula um ambiente onde os agentes representam *Mycobacterium tuberculosis.*

Os agentes do modelo possuem regras específicas de comportamento, que são modeladas como variáveis dos agentes. Essas regras são essenciais para que eles representem seu papel no ambiente e interaja como ele.

A simulação tem como divisão temporal o *tick*. A cada *tick*, os agentes realizam uma ou mais ações, estas ações são modeladas por funções estabelecidas no modelo, sendo elas: *alimenta, prossegue, sinaliza* e *reproduz.*

Assim como muitos fenômenos mensuráveis presentes em nosso dia a dia tendem a se distribuir conforme alguns modelos probabilísticos teóricos, tem-se por hipótese que as principais variáveis que modelam a curva de crescimento do bacilo *Mycobacterium tuberculosis* também podem se distribuir conforme algumas delas.

Como muitas variáveis aleatórias biológicas se ajustam a distribuição normal (Callegari-jacques, 2003), ou seja, os valores centrais são mais frequentes e os extremos, mais raros, sendo os valores muito baixos tão pouco frequentes quanto os muito altos. Assumiu-se que as variáveis do modelo também se distribuem normalmente.

A Figura 2 demonstra o ciclo de vida do agente, esclarecendo as ações e decisões que eles devem tomar a cada *tick*.

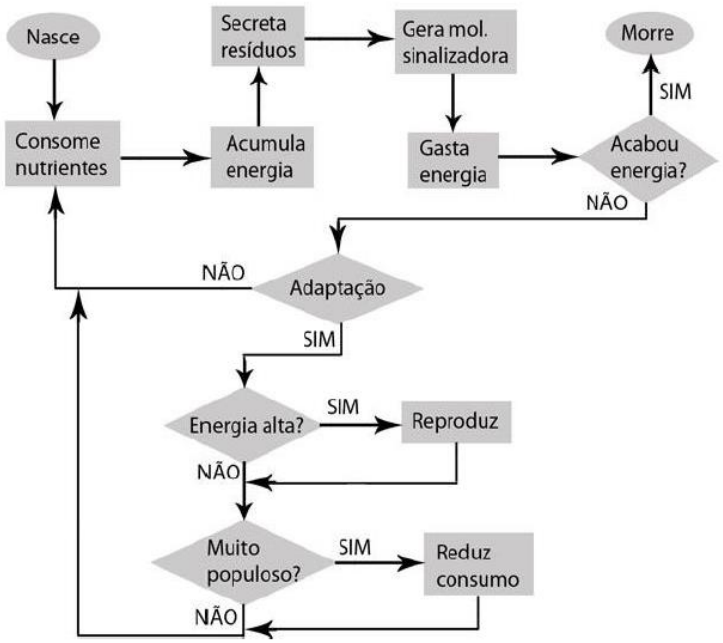


Figura 2. Fluxograma do Ciclo de vida dos agentes (Werlang, 2013).

O agente inicia o ciclo consumindo nutrientes do *patch* onde se encontra, a taxa de consumo é dada pela variável *consumo*. Os nutrientes absorvidos são transformados em energia que será depositada na reserva do agente. E após a metabolização dos nutrientes, ele secreta os resíduos provenientes no ambiente.

Depois de desempenhadas essas tarefas, uma quantidade de energia é subtraída da reserva do agente. Essa quantidade é dada pela variável *func_vitais*. Se a energia do agente, quantificada pela variável *energia*, for menor que 1, o agente morre; caso contrário, ele prossegue para as outras etapas da modelagem.

Passado o período de adaptação, estipulado pela variável *tick_reprodução*, o agente finalmente esta apto a se reproduzir. No entanto, um fator limitante para a reprodução é a energia disponível na sua reserva, se a energia for maior ou igual a estabelecida pela variável *energia_reproduz*, o agente reproduz; se não, ele continua suas funções sem reproduzir-se, até obter a energia necessária.

A última etapa do ciclo consiste na verificação da densidade populacional do ambiente. Caso o agente decrete situação de saturação, dada pela variável *sensor_bacteria*, ele libera um molécula sinalizadora (*Quorum sensing*) indicando aos outros que o ambiente está cheio e logo reduz seu consumo.

4. Resultados

Os resultados obtidos através das simulações foram comparados com os resultados observados em experimento *in vitro*. Na Figura 3 tem-se o resultado da simulação usando a modelagem baseada em agentes, utilizando o conceito de *Quorum sensing* na modelagem.

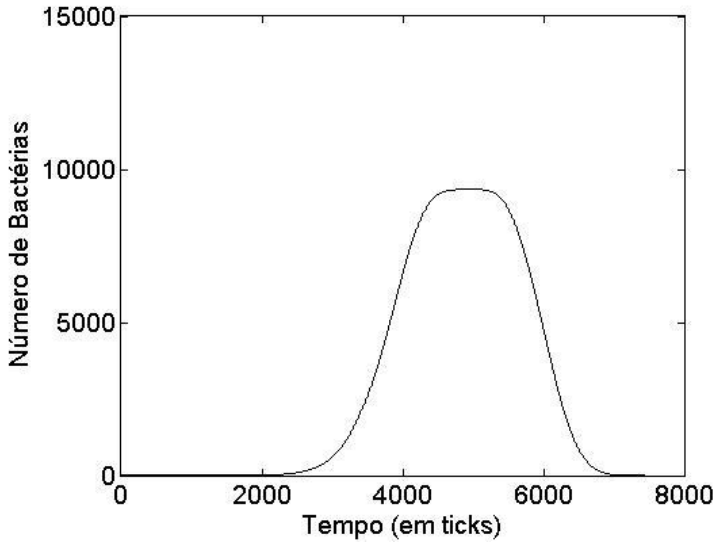


Figura 3. Curva de crescimento simulada

Na Figura 1 tem-se o resultado do crescimento bacteriano obtido por experimentos laboratoriais. O comportamento do bacilo da TB em ambos casos, Figuras 1 e 3, seguiu um padrão com relação ao número de bactérias em função do tempo.

Na simulação realizada com um ambiente diferenciado, Figura 4, sem a presença de moléculas sinalizadoras, onde o agente não entra em estado de consumo reduzido, pode-se ver também certa similaridade com curva de crescimento, no entanto a fase estacionária, visível nas Figuras 1 e 3, é praticamente despercebível na Figura 4.

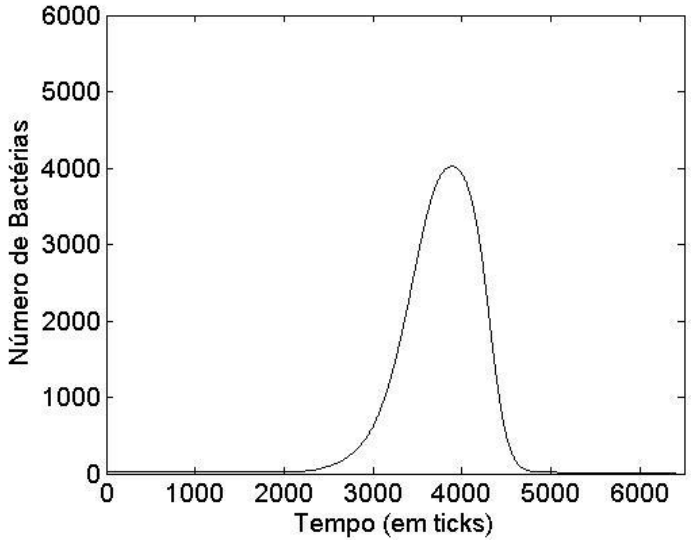


Figura 4. Curva de crescimento simulada sem a utilização do conceito de *Quorum sensing*.

E quanto comparada com a curva de crescimento da Figura 3, em número de bactérias e tempo de simulação, a mesma apresenta uma população máxima menor e o tempo de sobrevivência inferior.

Este comportamento se justifica pelo fato de que a detecção do ambiente estar superpopuloso, não ocorreu, as bactérias consumiram os nutrientes do ambiente desenfreadamente, acumulando muita energia em sua reserva, e por isso se duplicando ligeiramente. Isso fez com que os nutrientes acabassem rapidamente, resultando em uma morte prematura de todos os agentes do ambiente.

5. Conclusões

Este trabalho modelou a curva de crescimento do *Mycobcterium tuberculosis* utilizando sistemas multiagentes. Os agentes representavam a bactérias em seu meio, a forma como interagiam umas com as outras e com o ambiente.

Na modelagem da curva foi utilizado o conceito de *Quorum sensing*, onde os agentes mudavam seu comportamento quando atingido a saturação para obter um melhor aproveitamento do ambiente. E para tornar o modelo mais similar a um modelo de crescimento real, os valores das variáveis dos agentes eram gerados utilizando a distribuição de probabilidade normal.

O modelo baseado em agentes desenvolvido apresentou resultados satisfatórios, uma vez que a curva de crescimento modelada revelou-se similar a curva real. Essa similaridade torna o modelo bastante útil para verificação de hipóteses, já que as simulações levam minutos, em oposição aos experimentos *in vitro* que levariam dias. E ainda ressalta-se a importância da continuidade dos estudos sobre o *Quorum sensing*, uma vez que a inserção da variável no modelo foi imprescindível para a obtenção da similaridade.

Como trabalho futuro tem-se a intenção de promover uma ação conjunta com um especialista em crescimento bacteriano com a finalidade de poder comparar numericamente as saídas obtidas do modelo desenvolvido com as de um experimento *in vitro* obtidas em laboratório.

Referências

Burgos, M. e Pym, A. (2002), Molecular epidemiology of tuberculosis, European Respiratory Journal, v.20, n.36, p.54–65.

Callegari-jacques, S.M. (2003), Bioestatística: princípios e aplicações, Porto Alegre: Artmed.

Lönnroth, K and Migliori, G. B. and Abubakar. I. et al. (2015), Towards tuberculosis elimination: an action framework for low-incidence countries, European Respiratory Journal, v.45, n.4, p.928–952.

Todar, k. (2013), The growth of bacterial populations, Todar's Online Textbook of Bacteriology.

Von groll, A. (2010), Fitness of Mycobacterium tuberculosis associated to genotypes and drug resistance: new approaches for understanding the transmission dynamics of tuberculosis. Ghent University. Ghent, p. 137.

Whitehead, N. A., Barnard, A. M. L., Slater, H.,Simpson, N. J. L. e Salmond, G. P. C. (2001), Quorum-sensing in Gram-negative bacteria, FEMS Microbiology Reviews, Amsterdam, v. 25, n. 4, p.365-404.

Wooldridge, M. (2002), An Introduction to Multiagent Systems, England: John Wiley & Sons.

An Analysis of Javino Middleware for Robotic Platforms Using Jason and JADE Frameworks

Dayana Junger[1], João Victor Guinelli da Silva[1], Carlos Eduardo Pantoja[1]

[1]CEFET/RJ – UnED Nova Friburgo – Av. Gov. Roberto da Silveira, 1900 – Nova Friburgo – RJ – Brasil

dayanacomputer@hotmail.com, {joao.silva, pantoja}@cefet-rj.br

Abstract. *Robotics is the field of knowledge in which it is possible to use agent-oriented programming languages to develop robots capable of interacting with their environment by using sensors and actuators. Some situations require that agents try to achieve tasks in the physical world by both sending commands to actuators and perceiving data from sensors. Based on this, Javino middleware can be employed to interconnect Java-based MAS and hardware devices. Thus, our main objective in this paper is to prove the usefulness of using Javino to perform the communication between MAS applications and Arduino boards. For this, we conduct experiments to analyze the Javino's performance in Jason and JADE frameworks and discover if there is any condition in which this middleware should not be used.*

1. Introduction

Intelligent agents are computational systems that can both perceive and act, by themselves, into their environment. They can work together with other agents, forming a Multi-Agent System (MAS) [Wooldridge, 2009]. Similarly, robots are physical agents that use their sensors and actuators to get information about the environment and act on it based on their own decisions [Matarić, 2007]. Because of the similarity between these definitions, the idea of embedding MAS in robots to implement the decision-making mechanism is widely used; however, it is not a simple task, and some works try to achieve this using different approaches.

In [Soriano et al., 2013], for example, for each robot there is a computer running a simulated agent implemented using the Java Agent Development Framework (JADE) [Bellifemini et al., 2004]. In this work, the robots and its related computers communicate through Bluetooth. On the other hand, in [Lazarin and Pantoja, 2015], the authors embed software agents programmed using Jason framework [Bordini et al., 2007] into a Raspberry Pi[1]. The MAS embedded can control Arduino boards through the serial port using Javino middleware.

Considering this, our main objective of this paper is to prove the usefulness of using Javino to perform the communication between MAS applications and Arduino[2] boards. For this, we perform experiments to analyze the Javino's performance in the frameworks mentioned above to discover if there is any condition in which this middleware should not be used. Since Javino employs serial ports, if different agents compete for the same port at the same time, conflicts may occur.

[1] www.raspberrypi.org/
[2] www.arduino.cc/

The rest of this paper is structured as follows: in Section 2, we introduce concepts from Jason, JADE and Javino. We analyze the middleware in Section 3. In Section 4, some related works are discussed and in Section 5, the conclusion is seen.

2. Development of MAS

Taking either Jason or JADE along with Javino is a procedure which can be used to allow the development of robots using Arduino boards being controlled by the MAS. However, there are some divergences between Jason and JADE that influence the way each multi-agent system is embedded into robotic devices. This section presents some basic concepts of Jason, JADE and Javino.

2.1. Jason Framework

Jason is composed of agents which have plans, desires, intentions and beliefs. These agents can be situated in a simulated environment, in which it is possible to program which perceptions they can access. In the environment, it is programmed the external actions that agents can execute to respond to a stimulus. Furthermore, Jason includes some extensions, such as facilities for communication.

Jason agents implement a reasoning where they perceive the environment and updates their belief base with the perceptions received. Any update in the belief base generates an event which is added to an event list. After that, the agent checks for acceptable messages in its mailbox. Next, an event is chosen, and all the agent's plans are selected to be analyzed. Then a function gets one of these plans to be executed. Finally, if there is more than one ready-to-use intention, a function takes one of them, and the first non-executed action from this intention is performed.

2.2. JADE Framework

JADE [Bellifemini et al., 2004] is a framework for MAS development which is based on the programming language Java, and that fully implements the Foundation for Intelligent Physical Agents (FIPA) specification. Its main objective is to provide usable and accessible services for different-knowledge-level developers, which do not need to know FIPA.

In this framework, the agent platform is composed of one or more containers, in which the agents live, that are linked to the main container. The main container has some special responsibilities in the platform, such as managing information about containers and agents. Furthermore, it also hosts two special sorts of agents, one that provides the white page services, and another one that provides the yellow page services.

2.3. Javino Middleware

Javino [Lazarin and Pantoja, 2015] is a middleware used for performing the communication between an Arduino and a Java program that implements a protocol specified to provide an error-detection mechanism for data exchange through the serial port. That middleware is composed of a double-sided library, in which one acts on the Java side while another one acts on the Arduino side.

The main objective of the protocol is to provide a communication bridge between the hardware and the MAS by ensuring that there is no error in the information received from any side during data communication. The authors suggest that this error detection may be particularly useful for embedded MAS because a wrong message can stop the effectors or/and even the agent, or, even worse, let the agent behave wrongly by taking damaged data.

3. Middleware Analysis

In order to evaluate Javino usage along with robotic agents, we executed experiments using Jason and JADE to verify if the middleware can transmit data with an acceptable loss rate; if it is possible to use Javino when there is competition for the same serial port; if the delaying time or the perception sizes have any influence on data transmission; and if the framework employed influences the middleware usage.

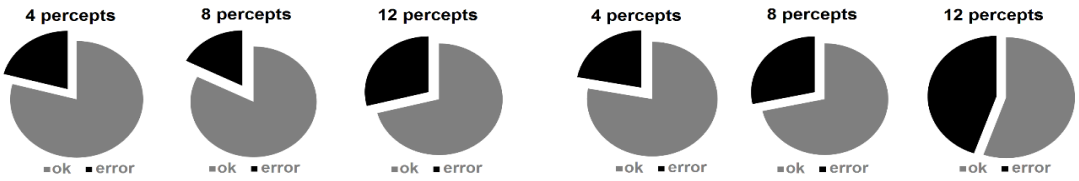
To perform the tests, we employed a robot with 12 ultrasonic sensors, which could send perceptions for the agent-oriented software by using our approach. We divided the experiments into two types: *Listen* mode and *Request* mode. In *Listen* mode, the controller sends data from sensors continuously, and, in the *Request* mode, the agent requests data by sending a serial message to the controller. For each one of these experiments, the robot sent 4, 8 and 12 different perceptions to the MAS. Furthermore, in the *Listen* mode, the delaying time-tested between each sent message was 100, 300, 500 and 700 milliseconds (ms). We also varied the number of agents *(n)* which were competing for the same serial port by starting with $n=1$. For each experiment performed, we monitored the messages sent to MAS for 5 minutes.

After the experiments execution with the *Listen* mode, it was possible to verify the percentage of correct messages sent for 4, 8 and 12 perceptions (Figure 2a) in Jason. The error rates were, respectively, 21%, 18% and 29% of 2271, 2122 and 1900 messages received from Javino. Firstly, we perceived that while the size of the message increased, the number of received messages decreased, since it needs more computational time for processing and verifying the correctness of the message.

In JADE, the error rates for 4, 8 and 12 perceptions were 22%, 28% and 45% of 2409, 2118 and 1983 messages sent (Figure 2b). Javino presented the same behaviour of Jason when the size of messages increased. JADE presented error rates higher than Jason, especially in messages with 12 perceptions. It could have occurred because the reactive architecture of JADE is slightly faster than the BDI one, so the agent tries to get more messages from sensors. However, it was necessary to identify which was causing the bottleneck of errors in communication. In both cases, we considered all messages sent without considering the delaying time.

For this, we analyzed the messages sent with a delaying time of 100, 300, 500 and 700 milliseconds. In Jason, the correct message rates with delaying time of 100 ms for 4, 8 and 12 perceptions were, respectively, 2%, 33% and 45% (Figure 3a). When the delaying time increased to 300 ms, the rates of correct messages were 100%, 99% and 99%. We noticed that while the delaying time decreased, the error rate increased. It

makes sense since there were several messages arriving at almost the same time. Then, the agent was trying to get messages with errors.



(a) **Percentage of correct messages sent with *n* perceptions using Jason.**

(b) **Percentage of correct messages sent with *n* perceptions using Jason.**

Figure 2: Percentage of correct messages sent from Javino to Jason (a) and Javino to JADE (b).

In JADE, the rate of correct messages with a delaying time of 100 ms for 4, 8 and 12 perceptions were 99%, 99% and 0.1%. When the delaying time increased to 300 ms, the rates were 99%, 99% and 88% (Figure 3b). In this case, the JADE lost some messages, but it was not significantly different from Jason. Until this point, all the experiments used only one agent. It was important to verify the behaviour of Javino and the MAS when n agents were competing for the same serial port.

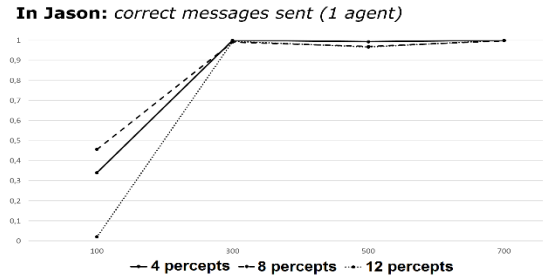
Then, in this experiment, which uses Jason, it is possible to observe the rate of correct messages in which the delaying time was related to more than one agent (Figure 3c). When one or more agents competed for the same serial port, the error rates increased, significantly, since only one agent should manipulate a serial port. The MAS stopped running after 2 or 3 minutes. In JADE, the rates of correct messages sent when more than one agent was competing for the serial port were 14% and 0% for 2 and 3 agents, respectively (Figure 3d). Therefore, we noticed that it is not recommended to use Javino with 2 or more agents accessing the same serial port since the agent competes for the port. However, if the agents access different serial ports, there is no problem in using Javino.

After the experiments execution with the *Request* mode, by using the same parameters of the former experiments, we noticed that the rate of correct messages sent was 100% for 4, 8 and 12 perceptions. When it was considered the delaying time, the rate of correct messages was 100% for all parameters. It happened because of the characteristic of the Request mode since the agent is responsible for requesting the information from the controller, which response by transmitting the perceptions from sensors. Consequently, the delaying time (used in the controller programming), in this situation, does not influence the middleware performance, since the requirement depends on the agent and do not depend on the controller.

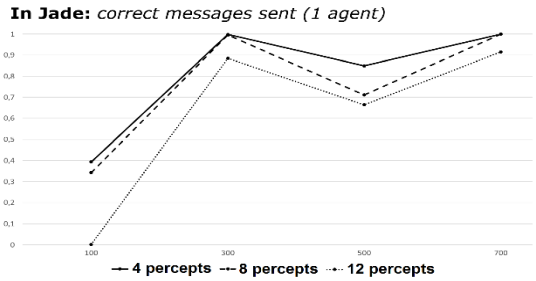
4. Related Work

In this section, we discuss some related works, which employ robotic agents. In [Jensen, 2010], the author presents an extended version of Jason that combines this framework along with Lego robots. Furthermore, it proposes, as an extension, several

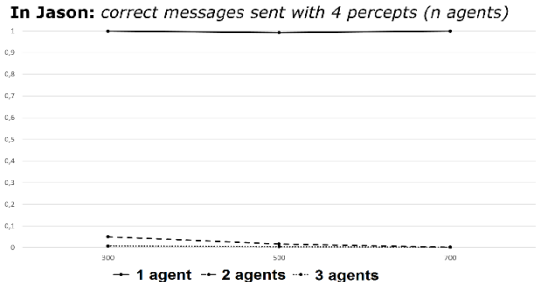
internal actions to establish the communication between physical agents and the Lego Mindstorm NXT toolkit. It states that a Lego agent can be represented as a Jason agent. It uses Bluetooth as a communication channel and LeJOS as the middleware between Jason and Lego agents. Additionally, it presents a communication process between Lego agents. Despite being able to communicate, the author says that the robots performance is slow, and the robotic platforms are tied to Lego Mindstorm NXT. Differently, we propose an architecture which provides the possibility of developing the same functionalities by using any robotic platform and, for instance, Jason and JADE frameworks.



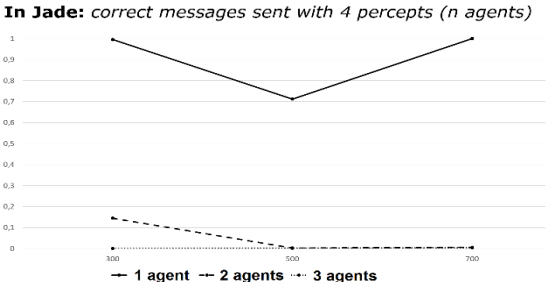
(a) Serial port usage by 1 agent in Jason.



(b) Serial port usage by 1 agent in JADE.



(c) Serial port usage by *n* agent in Jason.



(d) Serial port usage by *n* agent in Jason.

Figure 3: Messages received *per* delaying time by Jason (a) and by JADE (b). Agents competing for the same serial port in Jason (c) and JADE (d).

Describing the implementation of MAS into embedded systems with restricted resources is a procedure discussed in [Soriano et al, 2013] and it suggests that it is possible to develop different robotic applications by using JADE. As the communication time is taken into account, in the sense that it is an important aspect to be considered to avoid the system uncertainty, a kinematic control is locally employed to avert this sort of problem, and it uses two hardware platforms (Lego Mindstorms NXT and E-puck) to effect practical tests.

In one of these tests, the robots are supposed to follow points by starting from the closest one to a well-known position. In the tests, each robot is connected via Bluetooth to a computer in which the correspondent software agent is running, and these computers are part of a network that forms the MAS by using JADE. Then, these robots

can perform cooperative actions into the environment. However, the methodology approach does not address the issue of embedding directly an agent-oriented programming in the hardware. In this approach is necessary a computer to implement the agent's functionality of the robot.

5. Conclusions

This paper presented an analysis of Javino, a middleware that implements a communication protocol with error detection that can be employed in robotics and MAS. To perform the tests, it was used a 12-sensor robot to analyze Javino's performance along with a MAS, which was programmed through Jason and JADE. In the experiments performed by using the Listen mode, we showed that it is not possible to use Javino with a delaying time smaller than 100 ms. It is recommended to use it by setting up a waiting time greater than 300 ms, independently of message sizes (which should be up to 256 bytes). In the Request mode, there were no errors related to communication in both agent-oriented frameworks. We stated that if more than one agent competes for the same serial port in which the controller is connected, the Javino does not work properly.

For further works, we aim to provide a mechanism to allow the communication between robotic agents through Javino. For this, we aim to extend the Javino's protocol to address this issue and test it in different MAS frameworks. In this protocol, it must be provided a way to enable robotic agents (which are embedded along with individual MAS) communicate with each other. So, the agent's' architecture must be modified to support this new communication protocol.

References

Bellifemini, F., Caire, G., Greenwood, D., (2004). Developing Multi-Agent Systems with JADE. John Wiley and Sons, London.

Bordini, R.H., Hubner, J.F., Wooldridge, W. (2007). Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley and Sons, London.

Jensen, A.S., (2010). Implementing Lego Agents Using Jason, http://arxiv.org/pdf/1010.0150.pdf, November.

Lazarin, N.M., Pantoja, C.E. (2015). A Robotic-agent Platform for Embedding Software Agents using Raspberry Pi and Arduino Boards. In Proceedings of 9th Software Agents, Environments and Applications School: WESAAC'15, Niterói, Brazil.

Matarić, M. J. (2007). The Robotics Primer. MIT Press.

Soriano, A., Marín, L., Valera, Á., Vallés M. (2013). Multi-Agent Systems Integration in Embedded Systems with Limited Resources to Perform Tasks of Coordination and Cooperation. In: Proceedings of 10th International Conference on Informatics in Control, Automation and Robotics: p. 140 - 147, ICINCO' 13, Reykjavik, Iceland.

Wooldridge, M. (2009). An Introduction to Multi-Agent Systems. John Wiley and Sons, London.

Protocolo para diálogos argumentativos no auxílio da decisão consensual em sistemas multiagentes

Ayslan T. Possebom[1], **Mariela Morveli**[1], **Cesar A. Tacla**[1]

[1] Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI)
Universidade Tecnológica Federal do Paraná (UTFPR)

{possebom,morveli.espinoza,cesar.tacla}@gmail.com.br

Abstract. *Many argumentation systems applied to decision-making in multi-agent systems have been proposed. However, less attention has been given to cases where the decision should be obtained by consensus. This paper introduces a protocol for argumentation-based dialogues that support the consensual decision making where the dialogue is dynamic with no pre-established speech sequence and it allows identifying to what extent the set of acceptance or rejection used in the formula of the arguments is known by the agents. This paper connects argumentation and consensual decision-making in multi-agent systems.*

Resumo. *Diversos sistemas de argumentação aplicados na tomada de decisão em sistemas multiagentes foram propostos. No entanto, pouca atenção foi dada aos casos onde a decisão deve ser obtida de forma consensual. Apresentamos um protocolo para diálogos baseado em argumentação para apoio na tomada de decisão consensual onde o diálogo é dinâmico, não havendo uma sequência de fala pré-estabelecida, permitindo identificar em que medida o conjunto de apoios ou rejeições é conhecida pelos agentes. Este trabalho conecta argumentação e tomada de decisão consensual em sistemas multiagentes.*

1. Introdução

O consenso pode ser visto como um processo de construção do senso comum. Neste processo, a comunicação entre os membros envolve a troca de argumentos de modo que cada membro possa obter ou fornecer informações adicionais sobre o problema de decisão. A argumentação consiste em um processo de tomada de decisão que visa atingir algum acordo sobre o que é acreditado pelos agentes que possuem informações incompletas ou contraditórias. Diferentes frameworks para argumentação abstrata ([Dung 1995] [Vreeswijk 1997] [Bench-Capon 2002]) e lógica ([García and Simari 2004] [Besnard and Hunter 2009] [Dung et al. 2009]) foram propostos. Estes frameworks tratam da identificação da aceitabilidade de argumentos e da estrutura interna destes argumentos.

As abordagens de tomada de decisão utilizando argumentação aplicadas a sistemas multiagentes geralmente atribuem forças ou pesos nos argumentos (ex: [Amgoud and Prade 2009]) para identificar argumentos relevantes, ou então, compartilhamento de tabelas de valores representando a influência do argumento para determinado agente (ex: [Fan et al. 2014]). Estas abordagens não permitem identificar a uniformidade de opiniões dos agentes. Para que o consenso seja atingido, a maioria dos agentes devem

estar de acordo com as informações apresentadas no diálogo e também serem capazes de aceitar as informações que são conhecidas pela maioria dos agentes.

Este trabalho tem como objetivo propor um protocolo para diálogos a ser aplicado em um sistema multiagente para auxiliar na tomada de decisão consensual onde a ordem de fala dos agentes é dinâmica. Esta proposta se baseia no que ocorre em reuniões presenciais nas quais muitas pessoas participam da decisão, debatem sobre determinado assunto e precisam se inscrever para fornecer seus argumentos aos demais membros do grupo. Neste trabalho trataremos apenas a questão de modelagem do problema de decisão e os recursos necessários para que o diálogo aconteça em busca do consenso. Regras de decisão e aceitabilidade de argumentos, bem como revisão de crenças e modelagem do oponente não serão tratados nesta abordagem inicial. O diálogo dinâmico permite que apenas os agentes que possuem argumentos sejam selecionados para expor suas ideias.

O trabalho está organizado como segue: Seção 2 traz a definição do problema de decisão e dos agentes. Seção 3 introduz o protocolo definido para o diálogo entre os agentes. Em seguida, apresentamos as conclusões e perspectivas futuras na seção 4.

2. Argumentos, Agentes e Tomada de Decisões

Tendo-se um conjunto *Ag* de agentes $\{ag_1,...,ag_n\}$ com *n>1* participando de um diálogo, cada agente ag_i usa uma *linguagem base* para criar arqumentos utilizando suas crenças, desejos e intenções. A definição de *argumento* utilizada neste trabalho pode ser encontrada em [Amgoud et al. 2002]. Cada agente expressa seus argumentos com base em seus conhecimentos. Neste sentido, pode haver inconsistência na base de conhecimento do agente ou conflitos entre os argumentos gerados com os argumentos já apresentados anteriormente por outros os agentes. Estes conflitos podem gerar duas formas principais de ataques entre argumentos: *undercut* e *rebuttal*, conforme definidos em [Parsons and McBurney 2003].

Para que os agentes possam gerar e comparar argumentos, bem como dialogar e tomar decisões, cada agente deve ser constituído por base de conhecimento e um conjunto de tabelas de diálogos.

Definição 1 *(Agente) Um agente ag_i é um par <Σ, DT>, onde Σ=$K_i \cup G_i \cup KO^i_j$ formando os estados mentais K_i (conhecimentos do agente ag_i sobre o ambiente), G_i (objetivos do agente, representando preferências sobre alternativas de decisão ou características desejáveis em uma decisão) e KO^i_j com j $\neq$ i (informações obtidas de outros agentes ag_j ou informações consensuais), e DT é um conjunto de tabelas de diálogos, sendo uma tabela para cada alternativa de decisão.*

Definição 2 *(Tabela de diálogo) Uma tabela de diálogo $Dialog_{(x)}$ é uma tupla <κ, Ag, Arg, Apoio, Contra, Ataca_a> onde:*

- *x é uma alternativa de decisão*
- *κ é o número sequencial que representa um argumento;*
- *Ag é o agente ag_i que emite o argumento;*
- *Arg é um argumento ou contra-argumento;*
- *Apoio$\subseteq$Ag representa quais agentes apóiam qual fórmula (premissa ou conclusão) do argumento;*

- *Contra⊆Ag representa quais agentes rejeitam alguma fórmula do argumento (premissa);*
- *Ataca_$a_x \in \kappa_y$ com x < y é o número da sequência que o argumento faz um ataque (Rebuttal ou Undercut).*

Os argumentos são construídos pelos agentes na tentativa de contra-argumentar algum outro argumento já apresentado na sequência de diálogos ao qual o agente se contrapõe. O processo de argumentação consiste em construir argumentos a favor ou contra outros argumentos, avaliar os argumentos e aplicar algum princípio de comparação destes argumentos para avaliar as alternativas de decisões [Amgoud et al. 2005].

Podemos definir um framework de tomada de decisão concensual baseada em argumentação como:

Definição 3 *(Framework de tomada de decisão consensual) Um framework de tomada de decisão consensual AF baseada em argumentação é uma tupla <(Ag, TS), D, Att, T, DT, $\lhd_{Pref}$> onde:*

- *(Ag, TS) é um par onde Ag={ag_1,...,ag_n}, n>1, é um conjunto (finito) de agentes e TS $\in \mathbb{Z}$, representa o grau de confiança do agente ag_i;*
- *Um conjunto (finito) de alternativas de decisão D={d_1,...,d_m}, m>1;*
- *Um conjunto (finito) de atributos Att={att_1,...,att_l}, l>0;*
- *O tempo máximo de espera para que um agente apóie ou rejeite um argumento;*
- *Um conjunto (finito) de tabelas de diálogo DT={$dialog_{(d_1)}$,...,$dialog_{(d_m)}$};*
- *Uma pré-ordem (parcial ou completa) sobre D, definida com base nos argumentos apresentados em Dialog, representada por $\lhd_{Pref}$.*

O framework de tomada de decisão consensual baseada em argumentação apresenta $\lhd_{Pref}$ como resultado. A relação $d_1 \lhd_{Pref} d_2$ indica que a alternativa de decisão d_1 tem preferência sobre d_2 e, portanto, representa a alternativa que apresenta argumentos com maior número de apoios.

Exemplo 1 *Seja AF formado por um conjunto de agentes que decidiram sobre o destino de uma viagem de férias AF = <{(ag_1,5), (ag_2,5), (ag_3,5)}, {Florianópolis, Curitiba, Maceió,}, {barato, verão, praia, perto}, 10, {$dialog_{(Florianópolis)}$, $dialog_{(Curitiba)}$, $dialog_{(Maceió)}$}, {Florianópolis, Maceió, Curitiba}>. Três agentes dialogaram fornecendo argumentos pró ou contra cada uma das opções de decisão levando em consideração os atributos (barato, verão, praia, perto). Por meio de algum mecanismo de decisão e uma sequência de diálogos, obteve-se $\lhd_{Pref}$={Florianópolis, Curitiba, Maceió}. Neste framework de diálogo observa-se que a maioria dos agentes apoiaram Florianópolis.*

O grau de confiança pode ser utilizado quando existe a necessidade de estabelecer maior importância nos argumentos fornecidos por determinado agente. Por exemplo, se determinado agente possui uma especialidade sobre o assunto em discussão, seus argumentos oferecem um maior impacto na decisão. Durante a etapa de construção de argumentos, um agente ag_i deve buscar por fatos e/ou regras em Σ. Preferencialmente, o agente deve criar argumentos apoiados pelas bases K_i e G_i. Quando existir incompatibilidade entre K_i e KO_i, a base KO_i deverá ser usada, pois contém dados que são amplamente aceitos (consenso sobre determinada informação).

3. Estrutura do Diálogo

Um diálogo representa a troca de mensagens entre os agentes participantes da argumentação, senguido regras que conduzem a sequência destas mensagens. Para construir este protocolo de diálogos, precisamos especificar os seguintes parâmetros: definição do framework de argumentação AF contendo todos os seus elementos, criação da estrutura que representará a ordem de fala dos agentes e definição de mecanismo para o cálculo da força dos argumentos (consenso).

O protocolo de diálogo proposto utiliza um recurso chamado Artefato. Artefatos consistem em um conjunto de recursos e ferramentas que os agentes compartilham e usam cooperativamente para executar suas atividades em um ambiente [Ricci and et al. 2009]. Cada artefato é composto de um conjunto de operações e um conjunto de propriedades observáveis. Utilizamos um artefato de coordenação de falas formado por uma listagem indicando a ordem em que os agentes podem fazer seus respectivos movimentos. Sempre que um agente possuir algum argumento e desejar apresentá-lo aos demais agentes, ele deve se registrar no artefato e aguardar por sua vez. Um agente pode: (1) se registrar na lista da sequência de falas; (2) consultar a listagem; e (3) cancelar sua inscrição. O agente que ocupa a primeira posição da lista está habilitado a fornecer seus argumentos. Quando a fala do agente se encerra, sua posição da lista é removida e o próximo agente inscrito estará habilitado a encaminhar argumentos. Se no decorrer da argumentação um agente revisar suas crenças e desistir de argumentar, ele poderá solicitar sua exclusão do registro. Vale ressaltar que um agente pode ocupar apenas uma posição nesta listagem.

Cada movimento de diálogo representa um ato de fala (atos ilocucionários). No diálogo argumentativo para a tomada de decisões consensual, propomos os seguintes tipos de fala:

1. *Propose(S,h)*: onde S é um conjunto de fórmulas de L representando o suporte do argumento e h a conclusão. Usado quando o agente deseja enviar um argumento ou contra-argumento no jogo de diálogos;
2. *Challenge(Ag, h)*: onde Ag é o agente que apoiou a conclusão h em uma fala. Usado quando um agente é contra determinado argumento apresentado e decide conhecer os motivos do apoio ao argumento;
3. *Ask-if(Ag, Φ)*: onde Ag é um agente do sistema e Φ é uma fórmula em L. Usado para perguntar ao agente Ag se ele possui Φ (fato) em sua base de crenças;
4. *Query-if(Ag, Φ)*: onde Ag é um agente do sistema e Φ é uma fórmula em L. Usado para perguntar ao agente Ag se ele possui alguma regra que conclua Φ em sua base de crenças;
5. *Inform(Ag, Ψ)*: onde Ag é um agente do sistema e Ψ é uma resposta. Esta resposta poderá ser um argumento (resposta de *Challenge*), um valor booleano true ou false (resposta de *Ask-if*) ou uma regra (resposta de *Query-if*);
6. *Accept(κ, Φ)*: onde κ indica o número da sequência de diálogos em que um argumento foi apresentado e Φ representa a parte do argumento em que o agente está apoiando (premissa ou conclusão);
7. *Refuse(κ, Φ)*: onde κ indica o número da sequência de diálogos em que um argumento foi apresentado e Φ representa a parte do argumento em que o agente está rejeitando (apenas premissa).

Os tipos de fala *Challenge*, *Ask-if* e *Query-if* não precisam ser broadcast. Estas falas podem ser direcionadas a determinados agentes com o objetivo de obter informações que não estão presentes na base de conhecimentos do agente. Com estas informações, a base KO_j^i será populada dinamicamente no decorrer dos diálogos contendo fatos e regras dos argumentos apresentados e informações solicitadas aos outros agentes. Desta forma, novos argumentos e contra-argumentos com informações conhecidas por outros agentes poderão ser formados.

A intenção do *Accept* e *Refuse* é identificar o quanto um argumento ou parte do argumento é aceito pelos demais agentes do sistema. A idéia do consenso é que os agentes entrem em acordo sobre uma opção de decisão. Neste sentido, quando um argumento utiliza informações que a maioria dos agentes acredita, ele tende a ser aceito pelos outros agentes e passa a ter uma maior importância no diálogo e na decisão. Este cálculo da importância/força do argumento não será abordado neste trabalho.

O protocolo de diálogos e estratégias dos agentes segue conforme as etapas apresentadas para cada alternativa de decisão:

1. O primeiro movimento da discussão κ=0 contém uma alternativa de decisão como conclusão do argumento;
2. Os agentes verificam em suas bases de crenças se estão de acordo ou não com o argumento apresentado em κ e fornecem suas avaliações (*Accept* ou *Refuse*);
3. Os agentes podem atualizar suas bases de crenças com as informações apresentadas nos argumentos anteriores (caso não tenham sido contestadas) e com informações adicionais solicitadas aos demais agentes (*Challenge*, *Ask-if*, *Query-if*);
4. Os agentes verificam em suas bases de crenças se possuem algum contra-argumento. Em caso positivo, se inscrevem na ordem de fala para que possam apresentá-los. Caso um possível contra-argumento de um agente utilize fórmulas que foram rejeitadas, este agente pode se retirar da fila;
5. O agente que estiver na primeira posição da fila pode apresentar todos os seus contra-argumentos e então é removido da fila (*Propose*);
6. O processo retorna em 2 até que não existam mais agentes na ordem de fala e nenhum argumento seja fornecido no tempo definido no framework de tomada de decisões.

Cada argumento fornecido no diálogo receberá uma força que representará a sua importância perante todos os agentes do sistema. Quanto maior esta força, maior o consenso de que esta informação deva ser acreditada pelos agentes. Para o cálculo da força dos argumentos deverá ser considerado a confiança do agente emissor do argumento juntamente com os apoios e rejeições recebidos. Os argumentos emitidos por agentes com maior credibilidad possuem um maior impacto na decisão final, tendo-se a sua especialidade sobre o assunto em discussão. Em contrapartida, caso a maioria dos agentes não concordem com uma determinada informação, esta informação pode ser tida como falsa, pois neste caso será difícil atingir consenso.

A partir da sequência de diálogos, podemos criar grafos da argumentação e calcular extensões de argumentação [Dung 1995], tais como grounded ou preferred, argumentos céticos ou crédulos, entre outros. Estas extensões podem indicar a relação de preferências entre argumentos. Este mecanismo não faz parte do escopo do protocolo de diálogos tratado neste trabalho.

4. Conclusões

Este trabalho apresentou uma proposta inicial de protocolo de diálogos entre agentes a ser utilizado quando a tomada de decisões deve ser consensual e baseada em argumentação. Neste protocolo não é pré-estabelecido uma ordem de fala, priorizando apenas nos agentes que desejam emitir seus argumentos. Em particular, esta proposta permite identificar o quanto cada fórmula usada nos argumentos são conhecidas pelos agentes do sistema indicando o consenso sobre determinada informação, conduzindo-nos a associar preferências sobre determinados argumentos apresentados no diálogo.

Nas próximas etapas, precisamos definir como será realizado o cálculo da força dos argumentos, bem como determinar quando um agente irá atualizar suas crenças. Adicionalmente, preciamos definir como será realizado o mecanismo de tomada de decisões, visto que teremos diferentes estratégias de argumentação para estabelecer a ordem de preferência do grupo em relação às alternativas de decisão.

Referências

Amgoud, L., Bonnefon, J.-F., and Prade, H. (2005). An argumentation-based approach to multiple criteria decision. In *Symbolic and quantitative approaches to reasoning with uncertainty*, pages 269–280. Springer.

Amgoud, L., Maudet, N., and Parsons, S. (2002). An argumentation-based semantics for agent communication languages. In *ECAI*, volume 2, pages 38–42.

Amgoud, L. and Prade, H. (2009). Using arguments for making and explaining decisions. *Artificial Intelligence*, 173(3):413–436.

Bench-Capon, T. (2002). Value based argumentation frameworks. *arXiv preprint cs/0207059*.

Besnard, P. and Hunter, A. (2009). Argumentation based on classical logic. In *Argumentation in Artificial Intelligence*, pages 133–152. Springer.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357.

Dung, P. M., Kowalski, R. A., and Toni, F. (2009). Assumption-based argumentation. In *Argumentation in artificial intelligence*, pages 199–218. Springer.

Fan, X., Toni, F., Mocanu, A., and Williams, M. (2014). Dialogical two-agent decision making with assumption-based argumentation. In *Proceedings of the AAMAS 2014*, pages 533–540. IFAAMAS.

García, A. J. and Simari, G. R. (2004). Defeasible logic programming: An argumentative approach. *Theory and practice of logic programming*, 4(1+ 2):95–138.

Parsons, S. and McBurney, P. (2003). Argumentation-based dialogues for agent coordination. *Group Decision and Negotiation*, 12(5):415–439.

Ricci, A. and et al. (2009). Environment programming in cartago. In *Multi-Agent Programming: Languages, Tools and Applications*, pages 259–288. Springer.

Vreeswijk, G. A. (1997). Abstract argumentation systems. *Artificial intelligence*, 90(1):225–279.

Implementação de Recursos em um Smart Parking baseado em Sistemas Multiagente

Felipe Felix Ducheiko[1], **Lucas Fernando Souza de Castro**[1], **Gleifer Vaz Alves**[1]

[1]Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná (UTFPR)
Caixa Postal 84.016 – 210 – Ponta Grossa – PR – Brasil

felipeducheiko@alunos.utfpr.edu.br, l.castropg@gmail.com, gleifer@utfpr.edu.br

Abstract. *One of the major problems faced every day by the population of the cities is to find a free parking spot. Usually, in times great traffic flow to find a parking spot, a driver will waste time or will drive a long way until find a spot. Therefore, wasting fuel and generating traffic jams. The main purpose of MAPS project (Multiagent Parking System) is develop a multiagent system to allocate parking spots. Specifically, this article presents the implementation of additional resources to the MAPS project, so as to facilitate the expansion and implementation of the project.*

Resumo. *Um dos grandes problemas enfrentado todos os dias pelos habitantes das cidades é o de encontrar uma vaga de estacionamento livre. Normalmente em horários de grande fluxo é gasto uma considerável parcela de tempo, ou são percorridas longas distâncias até que seja encontrada uma vaga, o que desperdiça combustível e gera engarrafamentos. Pensando nisto o projeto MAPS (MultiAgent Parking System) é idealizado com objetivo de desenvolver um sistema Multiagente para alocação de vagas de estacionamento. Especificamente, este artigo apresenta a implementação de recursos adicionais ao projeto MAPS, para assim viabilizar a expansão e aplicação do mesmo.*

1. Introdução

O desenvolvimento da sociedade irrompe em muitos benefícios, mas consequentemente em alguns malefícios. Novas tecnologias são utilizadas para aprimorar e aumentar os benefícios já trazidos por tecnologias anteriores ou solucionar novos problemas. O conceito de Cidades Inteligentes refere-se ao uso de tecnologias da informação na tentativa de solucionar problemas das cidades a fim de melhorar a qualidade de vida da população.

Segundo [Batty et al. 2012] o conceito de Cidade Inteligente surgiu durante a última década com a fusão de várias ideias, tendo o intuito de melhorar a eficiência e a competitividade das cidades, criando novas maneiras para solucionar problemas. A essência do conceito é integrar as tecnologias que até agora têm sido desenvolvidas separadamente umas das outras. Mas que tem ligações claras em seu funcionamento e podem ser desenvolvidas de forma integrada.

Cidades possuem inúmeros desafios a serem resolvidos, dentre eles destacam-se os de mobilidade urbana. Segundo [Koster et al. 2014] cerca de 40% do tráfego em Nova York é gerado por carros à procura de vagas de estacionamento, o que ocasiona um agravamento dos congestionamentos e por conseguinte aumenta a emissão de poluentes.

Quando se percebe que a grande demanda de vagas de estacionamentos não está sendo satisfeita, normalmente provém a noção de que a solução é um aumento quantitativo do número de vagas. Porém, nem sempre essa é a solução mais sensata, pois a utilização das mesmas vagas de modo mais inteligente pode solucionar ou amenizar o problema. Um *Smart Parking* (estacionamento inteligente) pode ser composto por dispositivos de *hardware*, capazes de detectar o nível de ocupação dos estacionamentos, e *softwares* integrados para gerir a atribuição desses espaços. Normalmente tais sistemas são concebidos para auxiliar os motoristas na localização de vagas disponíveis [Nocera et al. 2014].

Dentre os vários modelos computacionais que podem ser usados para a implementação de um *Smart Parking* destacam-se os Sistemas Multiagentes (SMAs). Em [Wooldridge 2009] SMAs são definidos como sendo sistemas compostos de vários elementos computacionais que realizam interações, sendo tais elementos conhecidos como agentes. Esses sistemas possuem duas características importantes: primeiramente são, ao menos em certa medida, capazes de ações autônomas e em segundo lugar têm a capacidade de interagir uns com os outros pela interação análoga às interações sociais humanas. Além disso, os agentes então envolvidos em um ambiente onde eles podem ter uma organização, comunicação entre outros aspectos.

Tendo justamente o objetivo de aplicar métodos e técnicas de SMA, na criação de uma solução para alocação de vagas e gerenciamento de um *Smart Parking*, foi concebido o projeto MAPS *(MultiAgent Parking System)*. Para o desenvolvimento deste projeto está sendo utilizado o framework JaCaMo [JACAMO 2011].

Especificamente, o trabalho aqui apresentado tem como objetivo principal desenvolver dois recursos adicionais para que sejam incorporados ao projeto MAPS: a implantação de uma interface gráfica e a persistência de dados. Tais recursos visam facilitar a utilização e possibilitar a expansão do projeto.

O restante do artigo está organizado da seguinte maneira: na Seção 2 descreve-se o projeto MAPS. Na Seção 3 são apresentados detalhes sobre a implementação de recursos adicionais. Na Seção 4 encontram-se as considerações finais.

2. Projeto MAPS

O projeto MAPS *(MultiAgent Parking System)* é desenvolvido no GPAS (Grupo de Pesquisa em Agentes de *Software* - UTFPR - PG) com a meta principal de elaborar soluções para estacionamentos inteligentes. Os primeiros resultados do MAPS são apresentados no trabalho de [Castro 2015], onde foi implementado um SMA por meio do framework JaCaMo para alocação de vagas em estacionamentos[1].

Para implementar o sistema foram criados dois tipos de agentes (implementados no Jason) : os agentes *drivers* que interagem e utilizam o sistema multiagente e o agente *manager* que é responsável por informar, alocar e gerenciar as vagas do estacionamento.

O sistema também é composto por dois artefatos (implementados no Cartago) : o artefato *control* que é responsável pelo gerenciamento dos motoristas na fila e também o artefato *gate* que é responsável pela abertura e fechamento da cancela.

A alocação de vagas é realizada conforme o grau de confiança (*degree of trust*,

[1]Repositório GitHub: github.com/MAPS-UTFPR/MAPS

ou apenas *trust*) de cada motorista. Tal conceito é corroborado em [Huynh et al. 2006] quando afirmam que a confiança e a reputação são temas centrais para a interação efetiva em um SMA aberto em que os agentes entram e saem do sistema. Em [Gonçalves and Alves 2015] é discutida uma proposta para o uso e cálculo do *trust* em um *Smart Parking*. Contudo, a atual versão do MAPS ainda não implementa o conceito de grau de confiança, pois não há armazenamento de valores de confiança. Após, a Seção 3.2 irá discutir tal questão.

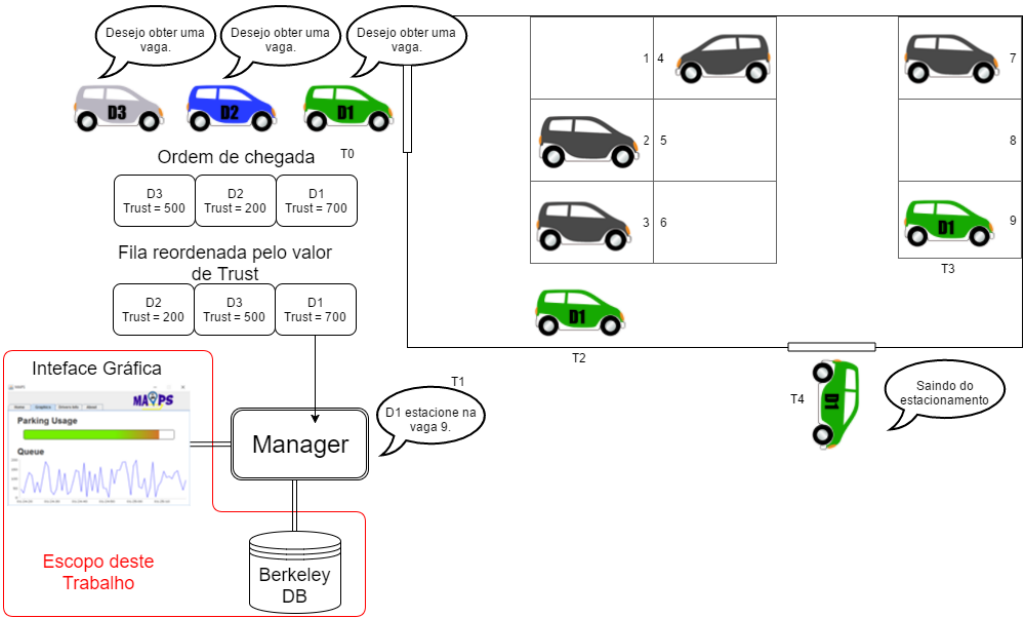


Figura 1. Diagrama Projeto MAPS - Fonte: Autoria Própria

Na Figura 1 é apresentado um diagrama que ilustra o funcionamento geral do Projeto MAPS. Neste diagrama é possível identificar os agentes *drivers* que chegam no estacionamento e requisitam vagas. O agente *Manager* gerencia a fila de *drivers*. Os instantes de tempos (t0 até t4) ilustram as diferentes ações de um agente no SMA.

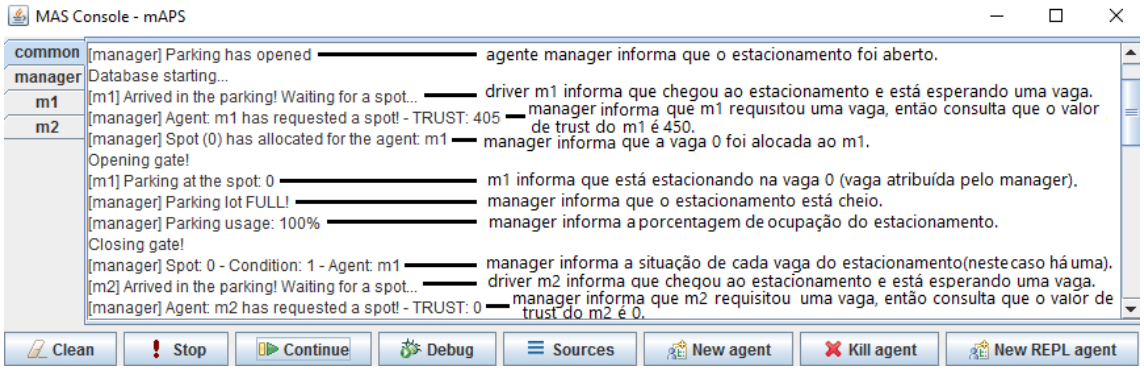


Figura 2. Console JaCaMo

Com a finalidade de acompanhar a execução do SMA o JaCaMo possui um console simplificado. Na Figura 2 é apresentado o início da execução do MAPS através do console e uma breve explicação das interações que ocorrem entre os agentes.

3. Implementação de Recursos

Esta seção aborda detalhes da implementação dos recursos adicionais. Na Subseção 3.1 é apresentada a Interface, ao passo que na Subseção 3.2 a implementação do Banco de Dados é discutida.

3.1. Interface gráfica

A implementação desta primeira interface gráfica do MAPS é voltada para os responsáveis pelo controle e gestão do *smart parking*, a fim de que possam ter acesso rápido a informações cruciais para otimizar o seu desempenho. A implementação está em fase de desenvolvimento, conforme é possível visualizar na Figura 3.

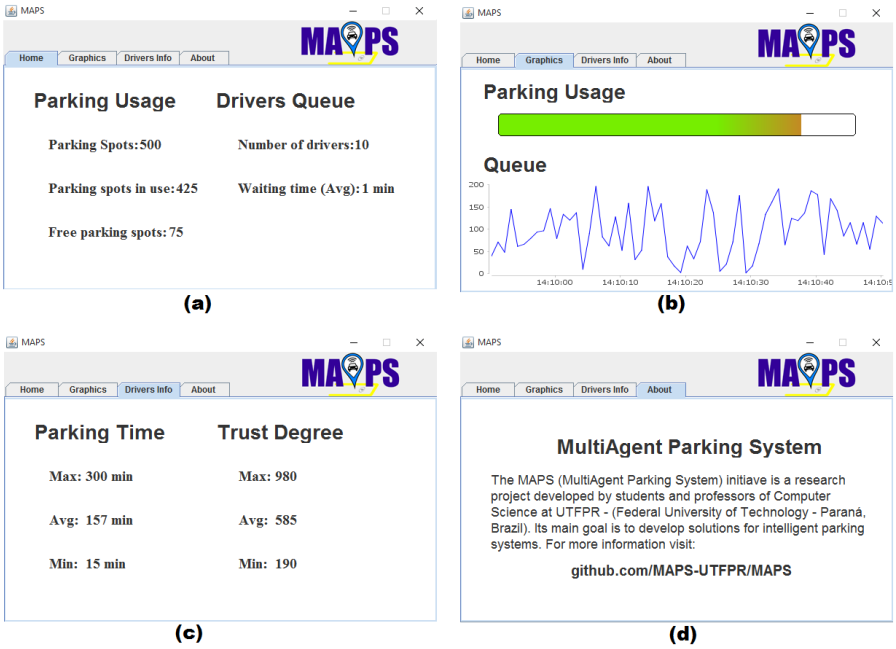


Figura 3. Protótipos das telas

Para desenvolver o protótipo foram definidos os seguintes requisitos: (i) design intuitivo e descomplicado; (ii) simplicidade, assim evitando uma sobrecarga cognitiva do usuário; (iii) desenvolvimento em Java, para facilitar a integração com o JaCaMo; e (iv) uma interface facilmente extensível, devido as futuras alterações nas versões do sistema.

Na tela inicial da interface gráfica, que pode ser visualizada na Figura 3 (a), o usuário já possui informações sobre o nível de ocupação do *smart parking* (número total de vagas, o número de vagas livres e em uso) e sobre a situação da fila de espera (número de motoristas na fila e o tempo médio de espera).

Na aba *graphics*, que pode ser visualizada na Figura 3 (b), o usuário possui informações sobre o nível de ocupação do *smart parking*, através de um gráfico de barra e sobre a situação da fila de espera, através de um gráfico de linha que relaciona o horário e quantidade de motoristas na fila. Estas informações são apresentadas através de gráficos simples para que sejam facilmente verificadas.

Na Figura 3 (c) tem-se a aba *drivers info* que traz os valores máximo, mínimo e médio do tempo de permanência *(Parking time)* e do *trust* dos motoristas que frequenta-

ram o *smart parking*. Essas informações são úteis para melhorias futuras, por exemplo se o valor médio do *trust* estiver alto pode ser um indício de que novos motoristas teriam dificuldade de conseguir vagas. Logo, essas informações podem ajudar o agente *Manager* a calibrar os valores para assegurar o bom funcionamento do SMA. Observando a Figura 3 (d) é possível visualizar a aba *About*, que trás informações básicas sobre o projeto.

Como já demonstrado, na Figura 2, o console do JaCaMo exibe o passo a passo da execução do MAPS, o que não é suficiente para o objetivo aqui abordado, visto que exibe apenas informações básicas do funcionamento do sistema. Contrastando, a interface exibe informações complementares como gráficos e valores máximos, médios e mínimos.

3.2. Persistência de Dados

A implementação inicial do MAPS possuía os valores de confiança fixos, ou seja, os agentes *drivers* sempre tinham o mesmo valor de confiança perante o *manager*. Com a implementação da persistência dos dados é possível que os valores sejam salvos para futuras utilizações de um mesmo agente *driver*. A partir da implementação da persistência dos valores de *trust*, é possível gerenciar estes valores baseado em um histórico de utilização. Na versão atual do projeto, o grau de confiança é obtido de forma simples e direta. Basta incrementar 20 unidades ao valor de confiança, cada vez que um dado *driver* utilizar o *smart parking*. O valor de 20 unidades é usado considerando que um *driver* poderia utilizar cinco vezes em uma semana o estacionamento e obter a pontuação igual a 100 unidades, o qual representa 10% da pontuação máxima (1.000 unidades).

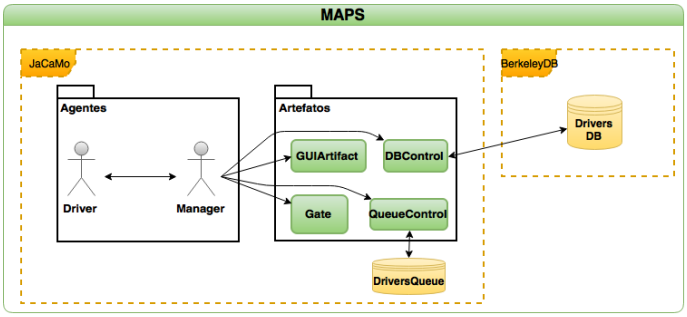


Figura 4. Projeto MAPS: perspectiva DB

O gerenciamento da persistência dos dados dá-se por meio de um artefato em Cartago chamado *DBControl*, o qual é responsável pelo controle dos valores de confiança e informá-los para o *manager*. O banco de dados utilizado para o armazenamento dos valores de confiança e dos dados dos *drivers* segue o modelo NO-SQL (*Not Only SQL*) do tipo chave-valor, devido a sua alta performance, escalabilidade e simplicidade [Moniruzzaman and Hossain 2013]. Com o objetivo de manter as características do modelo chave-valor na implementação do banco de dados no MAPS, foi utilizado o banco BerkeleyDB da Oracle. Porém a versão utilizada foi a *Java Edition* pra facilitar a integração com o Cartago, visto que o Cartago utiliza a linguagem Java.

A Figura 4 ilustra os componentes Jason, Cartago e do Banco de Dados implementados no projeto MAPS. Em especial, os novos recursos descritos neste trabalho, a saber: artefatos do Cartago (*DBControl* e *GUIArtifact*), bem como a adição do BerkeleyDB como banco de dados para armazenar os dados dos agentes *drivers*.

4. Considerações Finais

Este trabalho apresenta a implementação de uma interface gráfica voltada ao controle e gestão do MAPS e a implementação da persistência de dados. A etapa de implementação desta primeira interface gráfica é uma evolução para o projeto MAPS, pois pode auxiliar o agente *manager* a tomar decisões. Por exemplo, avaliar se os *drivers* estão permanecendo pouco tempo no estacionamento, ou seja, se há grande rotatividade de vagas. Se isso ocorre, o *manager* pode reorganizar a dinâmica das vagas para facilitar o estacionamento. Além da interface gráfica, há a extensão da persistência dos dados, a qual proporciona que o grau de confiança dos agentes *drivers* seja utilizado e incrementado de acordo a sua utilização.

O emprego do *framework* JaCaMo facilitou a implantação dos recursos aqui propostos, visto que no desenvolvimento do artefatos em Cartago através da linguagem Java proporciona um alto nível de abstração para o desenvolvimento dos recursos.

O projeto MAPS continua em execução, e pretende-se desenvolver os seguintes trabalhos: (i) desenvolver um aplicativo móvel onde os agentes *drivers* poderão requisitar as vagas; (ii) embarcar o sistema utilizando o Arduino e Raspberry Pi [Lazarin and Pantoja 2015]; (iii) criar outras formas para calcular o grau de confiança; e (iv) utilização do simulador de trânsito Sumo.

Referências

Batty, M., Axhausen, K., Fosca, G., Pozdnoukhov, A., Bazzani, A., Wachowicz, M., Ouzounis, G., and Portugali, Y. (2012). Smart Cities of the Future.

Gonçalves, W. R. C. and Alves, G. V. (2015). Smart Parking: mecanismo de leilão de vagas de estacionamento usando reputação entre agentes. 9º Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações. Niterói - RJ. jun. 2015.

Huynh, T. D., Jennings, N. R., and Shadbolt, N. R. (2006). An integrated trust and reputation model for open multi-agent systems.

JACAMO (2011). The JaCaMo approach. Disponível em <http://jacamo.sourceforge.net/?page_id=40>.

Koster, A., Koch, F., and Bazzan, A. L. (2014). Incentivising Crowdsourced Parking Solutions.

Lazarin, N. M. and Pantoja, C. E. (2015). A Robotic-agent Platform For Embedding Software Agents using Raspberry Pi and Arduino Boards. 9º Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações. Niterói - RJ. jun. 2015.

Moniruzzaman, A. and Hossain, S. A. (2013). Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. *arXiv preprint arXiv:1307.0191*.

Nocera, D. D., Napoli, C. D., and Rossi, S. (2014). A Social-Aware Smart Parking Application.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. J. Wiley, New York, 2nd edition.

Checagem de Consistência de Modelos de Sistemas Multiagentes Normativos Utilizando MAS-ML *Tool*

Igor B. Nogueira[1], Mariela I. Cortés[1], Enyo J. T. Gonçalves[2]

[1]Universidade Estadual do Ceará, Fortaleza– Brasil

[2]Universidade Federal do Ceará

igor.bnog@gmail.com, mariela@larces.uece.br, enyo@ufc.br

Abstract. *Modeling tools are important allies in the development of complex software systems. The accuracy and detail may vary, but the models need to be correct and precise. This paper presents the evolution of the modeling tool for multi-agent systems MAS- ML tool in order to provide support for checking consistency of the internal properties and dependencies between diagrams.*

Resumo. *Ferramentas de modelagem são aliadas importantes no desenvolvimento de sistemas de software complexos. O detalhe e o rigor podem variar, mas em qualquer caso, os modelos precisam ser corretos e precisos. Este artigo apresenta a evolução da ferramenta de modelagem para sistemas multiagentes MAS-ML tool de forma que forneça apoio à checagem de consistência das propriedades internas e dependências entre diagramas.*

1. Introdução

A modelagem de sistemas é o processo de desenvolvimento de modelos abstratos em que cada modelo representa uma visão diferenciada do sistema [Sommerville 2011]. Os modelos precisam ser corretos no uso da notação e a consistência entre as diferentes visões precisa ser garantida. No entanto, analisar e estabelecer a boa formação de diagramas é uma tarefa difícil, principalmente no caso de sistemas complexos.

Uma linguagem de modelagem normalmente fornece um conjunto de diagramas através dos quais diferentes visões do sistema podem ser capturadas. Inconsistências inter-diagramas são definidas como resultantes da violação às propriedades de interdependência entre modelos. Por outro lado, uma inconsistência intra-diagrama está relacionada a violações a propriedades internas de um diagrama analisado isoladamente.

O método Observed-MAS [Brandão 2005] propõe a análise sistemática de modelos de SMAs descritos na versão original de MAS-ML através do uso de ontologias. Consequentemente, o método não suporta a versão mais atual da linguagem. O presente artigo aborda a implementação de um mecanismo que possibilite a checagem de propriedades estruturais e detecção de inconsistências intra e inter-diagramas utilizando a ferramenta MAS-ML Tool contemplando os diagramas definidos em MAS-ML 3.0.

2. Consistência de Modelos

O gerenciamento da consistência entre modelos envolve um conjunto de métodos e ferramentas que possibilitem estabelecer e manter a consistência entre diversos artefatos criados e utilizados por múltiplos *stakeholders* [Spanoudakis e Zisman 2001], [Kuster 2004].

3. Evolução de MAS-ML *tool*

3.1. Consolidação da Ferramenta

MAS-ML *tool* [Farias et al. 2009], [Gonçalves et al 2015] [Feijó 2012] [Freire et al. 2012] [Lima et al. 2013] é uma ferramenta que funciona como um *plug-in* da plataforma Eclipse[1] e serve como ambiente de modelagem para SMAs. O desenvolvimento da ferramenta ocorreu de forma gradativa onde, para cada diagrama foi desenvolvido um componente ou plugin independente, o que dificulta a utilização da ferramenta para a modelagem e análise global do sistema.

A ferramenta foi gerada a partir dos *plug-ins* GMF[2] (*Graphical Modeling Framework*) e EuGENia[3], responsável por automatizar e facilitar os procedimentos necessários para o desenvolvimento de diagramas utilizando o GMF. A estratégia adotada para a implementação do plug-in da ferramenta segue a abordagem orientada por modelos a partir do metamodelo consolidado em MAS-ML 3.0.

3.2. Checagem de Consistência de Modelo

A checagem de consistência em MAS-ML *tool* integrada dá-se por meio de verificações a nível interno de cada modelo através da checagem intra-diagrama, e por meio de verificação que leva em consideração a interdependência entre os modelos analisados, através da checagem inter-diagramas.

A checagem intra-diagrama é baseada nas regras de validação considerando as propriedades internas do modelo especificadas em MAS-ML 3.0 (Tabela 1).

Tabela 1 – Regras de validação intra-diagrama em MAS-ML 3.0.

Regra	Descrição	Implementação em OCL
Regra 1	Se um agente possui percepção, uma norma não pode restringi-la.	(self.resourceOfNorm->notEmpty() = true and self.normRestrict->notEmpty() = true and self.normRestrict.agentClass->notEmpty() = true and self.resourceOfNorm.feature->notEmpty() = true and self.resourceOfNorm.feature.oclIsTypeOf(Perception) = true and self.normRestrict.agentClass.ownedPerception->exists(perc | perc.name = self.resourceOfNorm.feature.name) = true) implies false
Regra 2	Se um agente possui planejamento, uma norma não pode restringi-la.	(self.resourceOfNorm->notEmpty() = true and self.normRestrict->notEmpty() = true and self.normRestrict.agentClass->notEmpty() = true and self.resourceOfNorm.feature->notEmpty() = true and self.resourceOfNorm.feature.oclIsTypeOf(Planning) = true and self.normRestrict.agentClass.ownedPlanning->exists(plann | plann.name = self.resourceOfNorm.feature.name) = true) implies false
Regra 3	Se um agente possui função-próximo, uma norma não pode restringi-la.	(self.resourceOfNorm->notEmpty() = true and self.normRestrict->notEmpty() = true and self.normRestrict.agentClass->notEmpty() = true and self.normRestrict.agentClass.owendAction->notEmpty() = true and self.normRestrict.agentClass.owendAction->exists(a | a.actionSemantics <> ActionSemantics::DefaultSemantics) = true) implies false
Regra 4	Se um agente possui função de formulação de objetivo, uma norma não pode restringi-la.	
Regra 5	Se um agente possui função de formulação de problema, uma norma não pode restringi-la.	
Regra 6	Se um agente possui função utilidade, uma norma não pode restringi-la.	

[1] http:// www.eclipse.org
[2] http://eclipse.org/gmf-tooling/
[3] http://www.eclipse.org/epsilon/doc/articles/eugenia-patching/

Por outro lado, a checagem de consistência inter-diagramas em MAS-ML *tool* proposta se baseia na definição de propriedades inter-diagramas, isto é, propriedades de interdependência entre diagramas. A Tabela 2 apresenta as propriedades de interdependência entre diagramas estáticos de MAS-ML, incluindo o diagrama de normas e de acordo com o método Observed-MAS [Brandão 2005].

Tabela 2 – Propriedades de Interdependência entre Diagramas Estáticos

Diagrama 1	Diagrama 2	Regra
Classes	Organização	Toda classe de agente ou de organização modelada num diagrama de classes precisa, necessariamente, estar modelada em algum diagrama de organização.
Papel	Organização	Toda classe de papel de agente ou de papel de objeto definida num diagrama de papel precisa estar definida em algum diagrama de organização.
	Classes	Toda classe de objeto modelada num diagrama de papel precisa estar modelada em algum diagrama de classes.
Classes	Normas	Toda classe de objeto, agente ou de organização modelada num diagrama de classes precisa, necessariamente, estar modelada em algum diagrama de normas.
Organização		Toda classe de agente, classe de organização, classe de papel de agente ou de papel de objeto definida num diagrama de organização precisa estar definida em algum diagrama de normas.
Papel		Toda classe de papel de agente ou de papel de objeto definida num diagrama de papel precisa estar definida em algum diagrama de normas.

3.3. Verificador Inter-Diagramas

A fim de fazer a checagem de consistência entre modelos, um algoritmo foi proposto a fim de categorizar os diagramas e verificar suas propriedades internas. Tomando as diferenças entre os diagramas estáticos de MAS-ML 3.0, percebe-se que o diagrama de normas é o mais abrangente, em relação a entidades e relacionamentos contemplados, seguido do diagrama de organização, papel e classes, respectivamente.

Seguindo a ordem de precedência dos diagramas e suas propriedades, o mecanismo de categorização considera a seguinte lógica: (i) modelo que possui, no mínimo, uma entidade norma definida em sua estrutura é categorizado como diagrama de normas; (ii) modelo que possui, no mínimo, uma entidade organização é categorizado como diagrama de organização; (iii) modelo que possui, no mínimo, um papel de agente ou papel de objeto sem organização e ambiente definidos em sua estrutura é categorizado como diagramas de papel e (iv) modelo que não atende às condições anteriores é categorizado como diagrama de classes. Com os modelos categorizados e, a partir das informações contidas nos arquivos dos diagramas projetados em MAS-ML *tool* (.masml), as propriedades de interdependência entre modelos podem ser verificadas (Algoritmo 1).

O mecanismo de categorização e a definição das propriedades inter-diagramas foram implementados em Java e a manipulação dos arquivos .masml foi facilitada através do JDOM [JDOM, 2015].

Algoritmo 1 – Interdependência entre Diagramas de Classes e Normas

```
Boolean containElement = true;
String inconsistencyWarning = "Rule 1: Inconsistency Class-Norm Diagrams - ";
String elementValue = "";
String valueElement = "";
switch(type){
   case ClassDiagram:
      if(type2 == Diagram.Type.NormDiagram){
         for(Element element2:agentClassChildren2){
             valueElement = element2.getAttributeValue("name");
             element2 = element; element2.setAttribute("name", valueElement);
```

```
            }
            if(!agentClassChildren2.contains(element)){ containElement = false;}
        }
        if(!containElement){ inconsistencyWarning +=  "Agent Not Defined"; }
        containElement = true;
        for(Element element: classChildren){
            for(Element element2: classChildren2){
              valueElement = element2.getAttributeValue("name");
              element2 = element; element2.setAttribute("name", valueElement);
             }
             if(!classChildren2.contains(element)){ containElement = false; }
         }
         if(!containElement){ inconsistencyWarning += "Class Not Defined"; }
         containsElement = true;
         for(Element element: organizationClassChildren){
            for(Element element2: organizationClassChildren2){
                valueElement = element2.getAttributeValue("name");
                element2 = element;
                element2.setAttribute("name", valueElement);
            }
            if(!organizationClassChildren2.contains(element)){ containElement = false;}
          }
          if(!containElement){ inconsistencyWarning += "Organization Not Defined"; }
          containElement = true;
}
```

4. Estudo de Caso

Para ilustrar a checagem de consistência de modelos utilizando a ferramenta MAS-ML *tool* integrada, foi usado TAC-SCM [Collins et al., 2006].

5.1 Identificação das Entidades do Sistema

O ambiente *TACEnvironment* possui uma organização principal *TacOrganizacao* e os agentes Comprador (*BuyerAgent)*, que pode desempenhar o papel de Comprador, Vendedor (*SellerAgent*), que pode desempenhar o papel de Vendedor (Seller), Entregador (*DeliveryAgent*), que pode desempenhar o papel de Entregador (Delivery), o agente fornecedor (*SupplierAgent*), que pode desempenhar o papel de fornecedor, e o agente gerente, que pode desempenhar o papel de gerente.

A Figura 1 (a) ilustra o diagrama de classes, especificando as relações dos agentes entre si e entre os agentes e o ambiente. Os relacionamentos que os papéis possuem entre si podem vistos na Figura 2 (b).

As seguintes normas foram definidas no contexto do TAC-SCM:

- N1: O agente vendedor é proibido de ter acesso aos pagamentos (proibição);
- N2: Todos os vendedores da organização TAC-Organization têm permissão para atualizar o estoque (permissão);
- N3: Os compradores da organização TAC-Organization são obrigados a pagar pelos itens que eles compraram (obrigação);
- N4 (Punição para a violação da N3): Os compradores que violaram N3 são proibidos de comprar itens (proibição).
- N5: O agente comprador é proibido de verificar as atualizações de lançamento de novos lances (proibição).

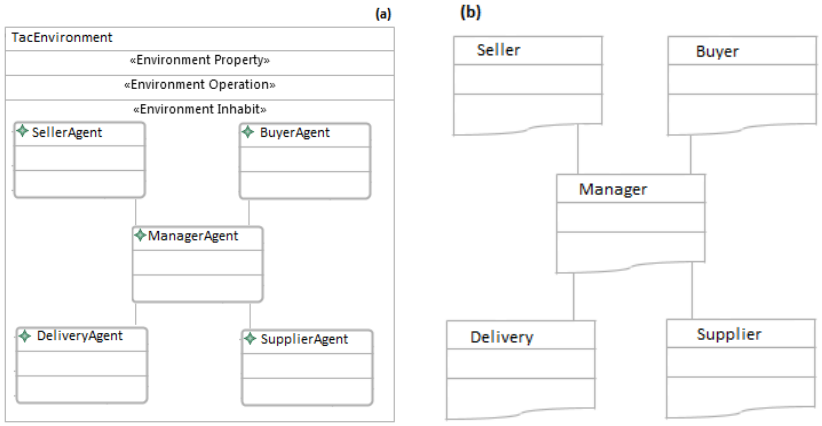


Figura 1. Diagrama de Classes (a) e Diagrama de Papel (b) para o TAC-SCM

A Figura 2 representa o diagrama de normas relacionando as normas ao ambiente juntamente com as entidades e a organização que está no contexto de N1 e N2.

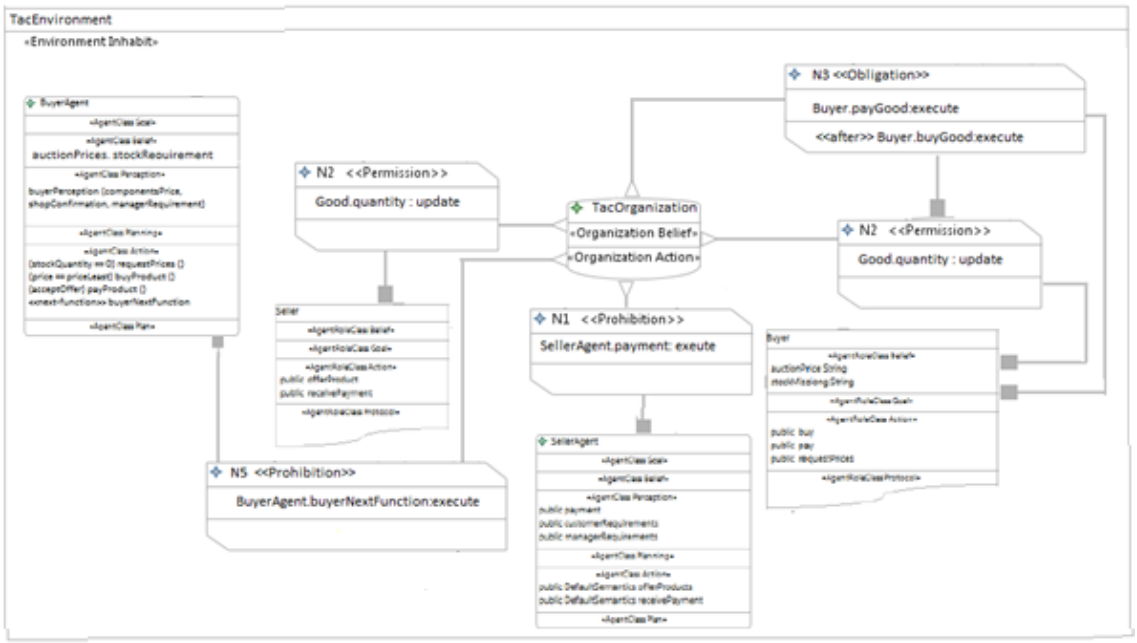


Figura 2. Modelagem do Diagrama de Normas

O processo de verificação de consistência entre os diagramas inicia executando o mecanismo de categorização a partir dos arquivos .masml. No estudo de caso são avaliados o diagrama de classes e o diagrama de normas. A partir da checagem das condições de categorização o tipo de diagrama é estabelecido em cada caso. Com o diagrama-origem (do tipo Diagrama de Classes) e o destino (do tipo Diagrama de Normas) categorizados, a checagem de consistência inter-diagramas pode ser feita.

Considerando que toda classe de objeto, de agente ou de organização modelada no diagrama de classes precisa, necessariamente, estar contemplada no diagrama de normas, uma inconsistência é detectada uma vez que os agentes *SupplierAgent, DeliveryAgent* não foram modelados. Consequentemente a mensagem *"Rule 1: Inconsistency Class-Norm Diagrams - Class Not Defined"* é exibida ao projetista.

6. Conclusão e Trabalhos Futuros

Este artigo apresenta a integração e evolução da ferramenta MAS-ML *tool* propiciando a checagem de consistência entre os modelos de sistemas multiagentes normativos modelados na ferramenta. A integração dos vários plug-ins da ferramenta MAS-ML *tool* seguiu a abordagem orientada por modelos, e na versão consolidada contempla os diagramas definidos em MAS-ML 3.0, a saber: classes, organização, papéis e normas.

A checagem de consistência de modelos em MAS-ML *tool* integrada deu-se por meio de verificações a nível interno de cada modelo através da checagem intra-diagramas baseada em regras OCL, e inter-diagramas com base na definição de propriedades de interdependência. Como trabalhos futuros podemos citar a extensão da ferramenta MAS-ML *tool* para abranger aspectos dinâmicos e estáticos de forma integrada, e incluir a checagem de consistência desses diagramas.

Referências

Brandão, A. A. F. (2005) “Um método para estruturação e análise de modelos de sistemas multiagentes baseado em ontologias”, Tese de doutorado. Rio de Janeiro: PUC, Departamento de Informática.

Collins, J.; Arunachalam, R.; Sadeh, N.; Eriksson, J.; Finne, N.; Janson, S. The Supply Chain Management Game for the 2007 Trading Agent Competition. 2006.

Farias, K.; Nunes, I.; Silva, V. T.; Lucena, C. J. P. (2009) “MAS-ML *Tool*: Um ambiente de modelagem de sistemas multi-agente”, *Fifth Workshop on Software Engineering for Agent-oriented Systems* (SEAS@SBES 09), Brazil.

Feijó, A. R. (2012). “Evolução da Ferramenta MAS-ML tool para a Modelagem dos Diagramas de Papéis e Sequência”. Monografia. Departamento de Computação, UECE.

Freire, E. S. S.; Cortés, M. I.; Gonçalves, E. J. T.; Lopes, Y. S. (2012) “A Modeling Language for Normative Multi-Agent Systems”. In: 13th AOSE, Valencia (Spain)

Gonçalves, E., Cortés, M., Campos, G., Lopes, Y., Freire, E., Silva, V., Farias, K., Oliveira, M. (2015) "Mas-ml 2.0: Supporting the modelling of multi-agent systems with different agent architectures." Journal of Systems and Software, V. 108, 77-109.

JDOM, disponível em:<http://www.jdom.org/>, acessado em 20/02/2016.

Kuster, J. (2004) “Consistency management of object-oriented behavioral models,” Ph.D. dissertation, Universit¨at Paderborn.

Lima, F. R. O.; Feijó, A. R; Rocha Jr., R. M.; Nogueira, I. B.; Goncalves, E. J. T. ; Freire, E. S. S. ; Cortes, M. I . (2013) “Dynamic Modeling of Multi-Agent Systems Using MAS-ML Tool”. In: VII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações (WESAAC), São Paulo.

Sommerville, I. (2011) “Software Engineering”. 9ª ed. Boston: Pearson.

Spanoudakis, G. and Zisman, A. (2001) “Inconsistency management in software engineering: Survey and open research issues,” in in Handbook of Software Engineering and Knowledge Engineering. World Scientific, pp. 329–380.

Modelagem da Teoria da Identidade Social em Sistemas Multiagente

Jader Saldanha[1], **Narúsci Bastos**[1],
Graçaliz Dimuro[1], **Cleo Billa**[1] **e Diana Adamatti**[1]

[1]Universidade Federal do Rio Grande
Centro de Ciências Computacionais - C3
Caixa Postal 474 - 96201-900 - Rio Grande - RS
Tel: +5553 3233 6623

{jadersaldanha,dianaadamatti}@furg.br

{naruscibastos,cleo.billa,gracaliz}@gmail.com

Resumo. *Simulações Baseadas em Agentes (SBA) contribuem em variados tópicos de modelos do mundo real. Henri Tajfel e seus colaboradores, por meio de experimentos, comprovaram que a existência de uma categorização social dentro e fora de grupos, provoca algum tipo de favorecimento a seus semelhantes. Estes estudos originários motivaram a Teoria da Identidade Social (TIS) que enfatiza a dimensão social do comportamento individual e grupal, ao postular que o indivíduo é moldado pela sociedade e pela cultura. Trocas sociais (TS) em Sistemas Multiagentes (SMA) são objeto de estudo em diversos contextos, nos quais as relações sociais são interpretadas como TS. Um problema fundamental discutido é a regulação das TS. A modelagem da TIS em sistemas de TS pode ser um contraponto na análise da autorregulação destes processos. Portanto, um modelo preliminar de SBA com os estudos originários da TIS foi desenvolvido.*

1. Introdução

[Woolridge and Wooldridge 2001] conceitua SMA como sistemas compostos por múltiplos elementos computacionais interagindo, conhecidos como agentes, dos quais possuem duas capacidades: uma (ou mais) ação autônoma de decidir por eles mesmos o que precisam para satisfazer seus objetivos, e capacidade de interação com outros agentes, não somente trocando dados mas incorporando atividades sociais como cooperação, coordenação, negociação etc. Dentre as possibilidades que os SMA trazem ao estudo de diversos campos, o autor ainda menciona sua colaboração a pesquisa de sistemas sociais artificiais, por exemplo.

Segundo [Rojas 2015], a análise de modelos e teorias de interatividade da sociedade humana permite observar que existem diversas formas de interação, que podem ser em diferentes níveis. Por exemplo, pode haver uma troca de informações, uma negociação ou uma discussão, um desenvolvimento de visões compartilhadas de um ambiente, ou até a formação ou dissolução de estruturas organizacionais. Essa forma de interação pode colaborar para a resolução de problemas em SMA.

Não obstante, em um modelo de interação social real entre seres humanos, discriminações sociais intergrupais acontecem. [Tajfel et al. 1971] [Tajfel 1970] por meio

de experimentos comprovaram que por uma mera categorização social dentro e fora de grupos, é provocado algum tipo de favorecimento a seus semelhantes. Os estudos originários de [Tajfel et al. 1971] e seus colaboradores trouxeram a TIS que segundo a autora [Ferreira 2010], procurava enfatizar a dimensão social do comportamento individual e grupal, ao postular que o indivíduo é moldado pela sociedade e pela cultura. Nesse sentido, defende que as relações intergrupais estão intimamente relacionadas a processos de identificação grupal e de comparação social. A autora ainda menciona que a TIS apoia-se em três postulados básicos: (1) o autoconceito é derivado da identificação e pertença grupal; (2) as pessoas são motivadas a manter uma autoestima positiva; (3) as pessoas estabelecem uma identidade social positiva mediante a comparação favorável de seu próprio grupo (*in-group*) com outros grupos sociais (*out-groups*). Nesse sentido, quando tal comparação não se mostra favorável ao próprio grupo, elas irão adotar diferentes estratégias para recuperar o favoritismo de seu próprio grupo, como forma de assegurar uma autoestima positiva.

Há pesquisas que elucidam trajetória na tentativa de modelagem da TIS em SMA, dentre elas [Lustick 2002] [Grier et al. 2008] [Prada et al. 2012], que ressaltam a importância de características da psicologia social em agentes. Na literatura, TS em SMA são objeto de estudo em diversos contextos, nos quais as relações sociais são interpretadas como TS ([Pereira 2008] [Gonçalves 2009] [Farias 2012] [Macedo 2013] [von Laer 2014] [Rojas 2015]). Como investigado por [Macedo 2013], um problema fundamental discutido na literatura é a regulação das TS, por exemplo, a emergência de trocas equilibradas ao longo do tempo levando ao equilíbrio social e/ou comportamento de equilíbrio/justiça. A modelagem da TIS em sistemas de TS pode ser um contraponto na análise da autorregulação destes processos. Sendo assim, os resultados obtidos em uma autorregulação com características da TIS poderiam autorregular o sistema de uma maneira eficaz e eficiente? Existiriam agentes que não efetuariam trocas com grupos de não pertencimento? Estas questões motivaram, dentro de uma disciplina de pós-graduação, a investigação da TIS em SMA e assim, um modelo preliminar de uma SBA foi desenvolvido.

O trabalho está organizado da seguinte maneira: na seção 2 é descrito o paradigma dos grupos mínimos, na seção 3 o modelo proposto em SMA, e na seção 4 as conclusões parciais.

2. O Paradigma dos Grupos Mínimos

[Tajfel 1970] deu início aos estudos de experimentação em discriminação intergrupal questionando se a discriminação poderia estar ligada a conflitos sociais ou a algum histórico de hostilidade. Aparentemente, [Tajfel 1970] concluiu que o simples fato de uma divisão em grupos é suficiente para disparar um comportamento discriminatório.

[Amâncio 1993] relata, em seus estudos de categorização de sujeitos onde era provocada uma diferenciação entre as categorias sociais que se traduziam em uma avaliação positiva da categoria de pertencimento em detrimento a outra, e portanto tornou-se necessário analisar se essa categorização também se traduziria em uma discriminação intergrupal, ou seja, em um favorecimento intergrupal (*ingroup*) ao invés do grupo de fora (*outgroup*). Foi então que esse objetivo levou a construção do paradigma dos grupos mínimos que investigava em um projeto de [Tajfel 1970] e [Tajfel et al. 1971] (e seus

colaboradores) as condições emergentes de uma discriminação intergrupal que pretendia investigar as condições minimas do efeito da categorização na discriminação intergrupal.

[Tajfel et al. 1971] descrevem a avaliação do efeito da categorização social em comportamento intergrupal em uma situação onde nem um cálculo de interesse individual ou atitudes prévias de hostilidades poderiam determinar comportamento discriminatório contra outro grupo. Este trabalho é uma extensão de [Tajfel 1970] com os experimentos descritos mais detalhadamente. Tendo como base o trabalho original de [Tajfel et al. 1971], dois experimentos iniciais foram conduzidos. O experimento 1, foi semelhante ao 2, apenas diferenciando-se na categorização social e na avaliação de estratégias grupais. Na próxima seção descreve-se resumidamente o experimento 2, o qual (dadas restrições computacionais) foi implementado na ferramenta de simulação multiagente Netlogo. Optou-se pelo desenvolvimento de um modelo do experimento 2, dada sua viabilidade próxima a uma implementação em uma ferramenta multiagente.

2.1. Experimento 2

Os objetivos neste experimento eram: validar os resultados do experimento 1 utilizando uma diferente categorização intergrupal e explorar sistematicamente a atração exercida nos sujeitos nas decisões intergrupais por algumas variáveis das quais são consideradas relevantes. No primeiro experimento os autores encontraram dificuldades em avaliar algumas questões dado o tipo de matrizes utilizadas para escolhas grupais.

O experimento foi divido em duas partes, onde na primeira era induzida a categorização intergrupal dos sujeitos, neste caso utilizando a preferência estética na escolha de pinturas de Klee e Kandinsky. A segunda parte consistiu na explicação dos artefatos (matrizes e folhetos) utilizados para a atribuição de recompensas e penalidades aos indivíduos. As matrizes se constituíam por duas linhas de colunas variáveis, onde a primeira linha correspondia a uma recompensa a algum indivíduo e a segunda a penalidade (estas linhas poderiam estar dispostas de forma aleatória). Cada indivíduo selecionaria uma coluna destas matrizes, onde pela organização numérica, envolveria estratégias de benefício (ou não) grupal. Para o experimento 2 foram definidas as seguintes estratégias: MJP (*maximum joint payoff*) um tipo de escolha onde ambos os indivíduos são beneficiados igualmente. MIP (*maximum ingroup payoff*) um tipo de escolha onde corresponde ao numero mais alto de pontos que pode ser atribuído a um membro do seu próprio grupo (*ingroup*). MD (*maximum difference in favour of the ingroup)* a diferença máxima de pontos entre dois indivíduos do qual a escolha pertence, essa diferença se refere a um membro do seu próprio grupo (*ingroup*). Na próxima seção descreve-se o modelo criado no Netlogo.

3. Modelo Proposto em Netlogo

Um dos objetivos desta pesquisa é incorporar características da TIS na análise da autorregulação dos processos de TS, um Jogo de Autorregulação dos Processos de Trocas Sociais (JAPTS) foi desenvolvido por [Macedo 2013], e portanto, as estruturas de dados necessárias para a incorporação do JAPTS e da TIS são necessárias. O JAPTS foi desenvolvido em Netlogo, então o modelo inicial do experimento 2 também.

Foram criados três grupos (A, B, C) compostos por no mínimo 2 agentes podendo totalizar até 100 em cada grupo. Cada grupo possui 6 matrizes iguais, que foram retiradas

do experimento original de [Tajfel et al. 1971]. O trecho de código apresentado a seguir mostra como foi criado em NetLogo o grupo A: da linha 1 a 5 contém as especificações dos agentes; a linha 4 permite que o usuário escolha o número de agentes pertencentes ao grupo na tela inicial do sistema. A partir da linha 6 até a 13 são criadas as matrizes.

Código usado para criar o grupo A no Netlogo

```
1.to setup
2.clear-all
3.set-default-shape groupsA "person"
4.create-groupsA initial-number-A
5.set color red
6.ask groupsA [forward 4]
7.ask groupsA[
8.set mA1 matrix:from-row-list [[19 18 17 16 15 14 13 12 11
10 9 8 7][1 3 5 7 9 11 13 15 17 19 21 23 25]]
9.set mA2 matrix:from-row-list [[23 22 21 20 19 18 17 16 15
14 13 12 11][5 7 9 11 13 15 17 19 21 23 25 27 29]]
10.set mA3 matrix:from-row-list [[7 8 9 10 11 12 13 14 15
16 17 18 19][1 3 5 7 9 11 13 15 17 19 21 23 25]]
11.set mA4 matrix:from-row-list [[11 12 13 14 15 16 17 18
19 21 21 22 23][5 7 9 11 13 15 17 19 21 23 25 27 29]]
12.set mA5 matrix:from-row-list [[1 2 3 4 5 6 7 8 9 10 11
12 13 14][14 13 12 11 10 9 8 7 6 5 4 3 2 1]]
13.set mA6 matrix:from-row-list [[18 17 16 15 14 13 12 11
10 9 8 7 6 5][5 6 7 8 9 10 11 12 13 14 15 16 17 18]]
]
```

Inicialmente, cada agente escolhe randomicamente uma coluna de cada uma das matrizes do seu grupo, para que posteriormente possa ser verificado a qual ”estratégia” pertence a sua escolha. As estratégias são MD, MIP e MJP, para que seja possível a verificação das escolhas de cada agente. Cada ”estratégia” é composta de uma matriz cujo os valores são os estipulados por [Tajfel et al. 1971]. Os valores foram escolhidos de forma aleatória a partir das seis matrizes, onde: as 2 colunas da direita e esquerda mais próximas do centro pertencem a estratégia MD; as duas colunas mais para a esquerda pertencem a estratégia MIP, e as mais da direita a estrategia MJP. As matrizes pertencentes as estratégias são definidas no trecho de código a seguir:

Código usado para definir as estratégias

```
1. set  md matrix:from-row-list [[19 18 17 9 8 7 23 22 21 13
12 11 7 8 9 17 18 19  11 12 13 21 22 23 1 2 3 12 13 14 18
17 16 7 6 5 ] [1 3 5 21 23 25 5 7 9 25 27 29 1 3 5 21 23 25
5 7 9 25 27 29 14 13 12 3 2 1 5 6 7 16 17 18]]

2. set  mip matrix:from-row-list [[16 15 14 12 11 10 20 19
18 16 15 14 10 11 12  14 15 16 14 15 16 18 19 20 4 5 6 9 10
11 15 14 13 10 9 8][7 9 11 15 17 19 11 13 15 19 21 23 7 9
11 15 17 19 11 13 15 19 21 23 11 10 9 6 5 4 8 9 10 13 14
```

```
15]]

3. set mjp matrix:from-row-list[[13 17 7 8 12 11][13 17 8
7 11 12]]
```

Logo após a construção das matrizes são realizados os seguintes passos:

1. Cada agente escolhe uma coluna aleatoriamente de cada uma das matrizes do seu grupo;
2. Para cada uma das escolhas dos agentes é verificada a que estratégia pertence e incrementa-se um a uma variável de estratégia;
3. É mostrado na tela o número de escolhas de cada grupo para cada estratégia.

Esta modelagem inicial será utilizada na análise de um cenário do Jogo de Autorregulação dos Processos de Trocas Sociais (JAPTS), desenvolvido por [Macedo 2013]. O objetivo é incorporar as diferentes estratégias de benefício grupal e responder algumas das questões iniciais de pesquisa apresentadas na introdução.

4. Conclusões Parciais

Inicialmente [Tajfel 1970] [Tajfel et al. 1971] e seus colaboradores propuseram estudos de experimentação intergrupal questionando se a discriminação poderia estar ligada a conflitos sociais ou mesmo a algum histórico de hostilidade. O autor concluiu que o simples fato de uma divisão em grupos poderia disparar um comportamento discriminatório. Estas pesquisas iniciais motivaram o desenvolvimento da Teoria da Identidade Social (TIS). A TIS vem sendo pesquisa dentro da esfera de SMA, contudo sem abordar sistemas de TS. Um dos pontos de contribuição é a autorregulação desses processos, onde é abordado pelo JAPTS em [Macedo 2013]. Motivados pela incorporação da TIS e do JAPTS foi desenvolvido um modelo inicial do experimento original conduzido por [Tajfel et al. 1971]. Como trabalho futuro pretende-se de fato realizar a incorporação dos dois modelos e criar cenários de teste para responder algumas questões como: a autorregulação pelo uso da TIS é eficiente e eficaz? Como as identidades são formadas a partir de grupos? Quais modelos de diferentes autores que utilizaram a TIS e a SBA podem contribuir em sistemas de TS?

Agradecimentos

Os autores desta pesquisa gostariam de agradecer a agencia financiadora CAPES e o Centro de Ciências Computacionais - C3 - e ao Laboratório de Sistemas Multiagentes e Simulação Social e Ambiental (LAMSA) da Universidade Federal do Rio Grande por proverem os recursos necessários para desenvolver esta pesquisa.

References

Amâncio, L. (1993). Identidade social e relações intergrupais. *VALA, Jorge*.

Farias, G. (2012). Um modelo de agentes bdi- fuzzy para trocas de serviços não - econômicos com base na teoria das trocas sociais. Master's thesis, Programa de Pós-graduação em Modelagem Computacional da Universidade Federal do Rio Grande.

Ferreira, M. C. (2010). *A Psicologia Social contemporanea: principais tendencias e perspectivas nacionais e internacionais*, volume 26. scielo.

Gonçalves, L. V. (2009). Uma arquitetura de agentes bdi para auto-regulação de trocas sociais em sistemas multiagentes abertos. Master's thesis, Programa de Pós-graduação em Computação da Universidade Catolica de Pelotas.

Grier, R. A., Skarin, B., Wolpert, L., and Lubyansky, A. (2008). Scipr: A computational model to simulate cultural identities for predicting reactions to events.

Lustick, I. (2002). Ps-i: A user-friendly agent-based modeling platform for testing theories of political identity and political stability. *Journal of Artificial Societies and Social Simulation*, 5(3).

Macedo, L. (2013). Uma abordagem evolucionária e espacial para o jogo da autorregulação de processos de trocas sociais em sistemas multiagentes. Master's thesis, Programa de Pós-graduação em Modelagem Computacional da Universidade Federal do Rio Grande.

Pereira, D. R. (2008). Construção de planos bdi a partir de políticas ótimas de pomdps, com aplicação na auto-regulação de trocas sociais em sistemas multiagentes. Master's thesis, Programa de Pós-graduação em Computação da Universidade Catolica de Pelotas.

Prada, R., Raimundo, G., Dimas, J., Martinho, C., Peña, J. F., Baptista, M., Santos, P. A., and Ribeiro, L. L. (2012). The role of social identity, rationality and anticipation in believable agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1175–1176. International Foundation for Autonomous Agents and Multiagent Systems.

Rojas, Y. (2015). Trocas sociais em sistemas multiagentes: Tranferência de confiança com base na reputação e na relação de dependência. Master's thesis, Programa de Pós-graduação em Computação da Universidade Federal do Rio Grande.

Tajfel, H. (1970). Experiments in intergroup discrimination. *Scientific American*, 223(5):96–102.

Tajfel, H., Billig, M., Bundy, R., and Flament, C. (1971). Social categorization and intergroup behaviour. *European Journal of Social Psychology*, 1(2):149–178.

von Laer, A. G. (2014). Autorregulação de processos de trocas sociais em sma: Um modelo de sociedade de agentes bdi evolucionários e culturais no contexto do jacamo. Master's thesis, Programa de Pós-graduação em Computação da Universidade Federal do Rio Grande.

Woolridge, M. and Wooldridge, M. J. (2001). *Introduction to Multiagent Systems*. John Wiley & Sons, Inc., New York, NY, USA.