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

— WESAAC 2013 —

Organizado por

Anarosa Alves Franco Brandão
Rafael Heitor Bordini
Jaime Simão Sichman

São Paulo, 26-29 Maio de 2013

Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCações — VII WESAAC / Brandão, A.A.F.; Bordini, R.H.; Sichman, J.S.; (Org). **ANAIS**.— — São Paulo, 2013.

207p. :il.
ISSN 2177-2096

1. Agentes Inteligentes. 2. Sistemas de Agentes de Software. 3. Ambientes para Agentes. 4. Aplicações de Agentes. I. Brandão, A.A.F. II. Bordini, R.H. III. Sichman, J.S.

Prefácio

Este documento contém os trabalhos apresentados na sétima edição do WESAAC (Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCações). O WESAAC 2013 foi realizado na cidade de São Paulo - SP, nas dependências da Universidade de São Paulo (USP), entre os dias 26 e 29 de maio de 2013, com o apoio da Escola Politécnica, do Instituto de Matemática e Estatística e do Centro de Computação Eletrônica da Universidade de São Paulo, da Sociedade Brasileira da Computação (SBC) e do Centro de Cultura Judaica de São Paulo.

Continuando a tradição da série WESAAC, os objetivos do evento continuam relacionados à integração de pesquisadores e estudantes de todos os níveis na área de Agentes e Sistemas de Agentes e divulgação das atividades de pesquisa dos diversos grupos de pesquisa do Brasil, com o intuito de facilitar o intercâmbio de conhecimentos. 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").

O histórico deste evento, que inicialmente foi denominado "Workshop - Escola de Sistemas de Agentes para Ambientes Colaborativos" e, a partir de sua quarta edição passou a ter a denominação atual, mostra o crescimento constante da comunidade de pesquisadores na área de agentes e sistemas baseados em agentes no Brasil. As três primeiras edições do evento tiveram uma abrangência regional, atingindo especialmente pesquisadores da região Sul do Brasil. A partir da quarta edição, realizada na cidade do Rio Grande - RS, aumentou-se o escopo do evento, ampliando sua abrangência de regional para nacional.

Nesta sétima edição do WESAAC, mantivemos a abrangência nacional, e ampliamos a participação internacional de pesquisadores destacados da área de sistemas de agentes, oriundos da Université Pierre et Marie Curie (UPMC)-França, da Université de Toulouse, França, da University of Otago, Nova Zelândia, e da Bar Ilan University, Israel. Além disso, também convidamos pesquisadores da indústria, notadamente a IBM.

Para esta edição, o evento recebeu uma variedade de contribuições. Foram submetidos 42 artigos, sendo 21 artigos completos e 21 artigos resumidos. Dentre os artigos completos, 13 foram aceitos para apresentação oral, divididas em três sessões técnicas, e 8 foram aceitos para apresentação na forma de poster. Dos artigos resumidos, 16 foram aceitos para apresentação como poster. Todos os artigos aceitos constam deste documento.

Gostaríamos de agradecer aos palestrantes convidados, Onn Shehory, Amal El Fallah Seghrouchni e Ana Bazzan que abrilhantaram o evento com suas palestras. Também agradecemos aos ministrantes de oficinas: Michael Winikoff, Fred Amblard, Amal El Fallah Seghrouchni, Jomi Fred Hubner e Sara Casare. Finalmente, agradecemos a todos os pesquisadores que submeteram os seus artigos, assim como aos membros do comitê de programa, aos revisores adicionais pelo criterioso trabalho desenvolvido e às nossas instituições (USP e PUC-RS). Um agradecimento especial às agências FAPESP, CAPES e CNPq, pelo fomento recebido e ao CCE-USP e ANSP,que tornaram possível o WESAAC 2013.

São Paulo, Maio, 2013
Anarosa Alves Franco Brandão
Rafael Heitor Bordini
Jaime Simão Sichman

Support-Patrocínio

Organization - Organização

General Chair - Organização Geral

Anarosa Alves Franco Brandão	Universidade de São Paulo

PC Chair - Coordenação do Comitê de Programa

Rafael Heitor Bordini	Pontifícia Universidade Católica do Rio Grande do Sul

Local Chair - Organização Local

Jaime Simão Sichman	Universidade de São Paulo

Steering Committee - Comitê Consultivo

Rejane Frozza	Universidade de Santa Cruz do Sul
João Luis Tavares da Silva	Universidade de Caxias do Sul
Diana Francisca Adamatti	Universidade Federal do Rio Grande
Gustavo Gimenez-Lugo	Universidade Federal Tecnológica do Paraná
Jomi Fred Hübner	Universidade Federal de Santa Catarina

Program Committee - Comitê de Programa

Adamatti, Diana	FURG (Brasil)
Aguiar, Marilton Sanchotene de	UFPel (Brasil)
Alencar, Fernanda	UFPE (Brasil)
Balsa, João	Univ. Lisboa (Portugal)
Barbosa, Raquel	FURG (Brasil)
Bazzan, Ana L. C.	UFRGS (Brasil)
Boissier, Olivier	EMSE (FranÃ§a)
Bordini, Rafael Heitor	PUCRS (Brasil)
Brandão, Anarosa Alves Franco	USP (Brasil)
Campos, André	UFRN (Brasil)
Carine Webber	UCS, (Brasil)
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)
David, Nuno	ISCTE (Portugal)
Dimuro, Graçaliz	FURG (Brasil)
Ferreira Jr., Paulo R.	UFPel (Brasil)
Frozza, Rejane	UNISC (Brasil)
Giménez-Lugo, Gustavo	UTFPR (Brasil)
Gonçalves, Eder	FURG (Brasil)
Hubner, Jomi Fred	UFSC (Brasil)
Jaques, Patricia	UNISINOS (Brasil)
Koch, Fernando	IBM Research (Brasil)
Leite, Joao	Universidade Nova de Lisboa (Portugal)
Lemke, Ana Paula	IFRS (Brasil)

Lorenzi, Fabiana	UFRGS (Brasil)
Marchi, Jerusa	UFSC (Brasil)
Meneguzzi, Felipe	PUCRS (Brasil)
Moreira, Alvaro	UFRGS (Brasil)
Nunes, Ingrid	UFRGS (Brasil)
Okuyama, Fabio	IFRS (Brasil)
Rabelo, Ricardo J.	UFSC (Brasil)
Ribeiro, Marcelo Blois	GE Global Research (Brasil)
Ricci, Alessandro	University of Bologna (Italia)
Rodrigues, Maira	UFMG (Brasil)
Rosa, Paulo	IME/RJ (Brasil)
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)
Tedesco, Patricia	UFPE (Brasil)
Trigo, Paulo	ISEL (Portugal)
Vasconcelos, Wamberto	University of Aberdeen (UK)

Additional Reviewers - Revisores Adicionais

Brito, Maiquel De
Souza, Marlo
Lima, Allan
Schmitz, Tiago Luiz
Zateli, Maicon

Sumário

IV Short Papers - Resumos estendidos

Parte I

Invited Speakers - Palestras Convidadas

On agent collaboration, games and coalitions

Onn Shehory
Bar Illan University
IBM Research Labs
Israel
onn@il.ibm.com

***Abstract*—Agents in the context of others typically have to interact and collaborate to meet their goals. Agent collaboration calls for mechanisms from game theory and relaxation thereof. Multiple games have been considered to facilitate collaboration, and many mechanisms have been devised. Yet it appears that the most widely studied class of games and mechanisms surround coalition and team formation. Within coalitions, agents may jointly perform tasks that they would otherwise be unable to perform, or will perform poorly. To allow agent collaboration via coalitions, one should devise a coalition formation mechanism that exhibit desirable properties such as stability, fairness, optimality, and computational tractability. In this talk we will present agent attributes which affect interaction, games which facilitate interaction, and mechanism which implement feasible solutions to such games. Part of talk will focus on coalition formation mechanisms.**

Keywords—agents; collaboration; coalitions; games

Coordination in multi-agent systems: dimensions and mechanisms

Amal El Fallah Seghrouchni
LIP 6 – Université Pierre et Marie Curie
Paris, France
amal.elfallah@lip6.fr

***Abstract*—A multiagent system (MAS) is populated by multiple autonomous agents that interact to solve complex tasks, to enhance the system's overall utility while improving their individual performance. Hence, coordinating the behaviors of multiple agents acting in the same environment is an important issue in the multi-agent systems domain. This talk will address the MAS coordination as a process by which, a system of agents are lead to work together harmoniously. It will present the several dimensions of coordination as well as the main mechanisms developed in MAS field, in particular for cognitive agents.**

Keywords—multiagent systems; coordination; cognitive agents

Agents and Traffic Simulation

Ana Bazzan
Instituto de Informática - UFRGS
Porto Alegre - Brazil
bazzan@inf.ufrgs.br

***Abstract*—This talk addresses the following points: i) problems related to the increasing demand for mobility in modern society; ii) four facets of intelligent transport and traffic systems: modeling and simulation, advanced travelers' information systems (ATIS), management, traffic control and optimization, and new technologies (autonomous vehicles and automation of infra-structure); iii) how agents can contribute to make cities smarter; iv) current work and recent results of the multi-agent systems group at the computer science institute at UFRGS.**

Keywords—traffic simulation, advanced traveler´s information systems

Parte II

Short Courses - Oficinas

Coordination of Complex Systems based on Multi-Agent Planning

Amal El Fallah Seghrouchni
LIP 6 – Université Pierre et Marie Curie
Paris, France
amal.elfallah@lip6.fr

***Abstract*—Handling and the coordination of plans for the achievement of different goals is an important issue of planning, in particular when several agents (robots) are mobile within a shared and dynamic environment. This lecture will present an overview of planning technics and coordination mechanisms developed for multi-agent systems. Then, it will present some significant approaches we have developed for the coordination of temporal multi-agent plans. We present a first framework based on hybrid automata to represent and handle temporal plans of agents. The coordination of such plans and their synchronization within a multi-agent plan will be discussed and illustrated in the context of aircraft simulation. Then, we introduce a second framework where coordination mechanisms have been established to deal with temporal plans of different priorities. This framework will be illustrated on two scenarios : a Proactive-Reactive Coordination Problem (PRCP) where an agent has to modify its temporal plan in order to remove any conflicts with the plan of another agent having higher priority ; and a Coordinated Planning Problem (CPP), where an agent has to compute a plan for the achievement of its own goals, but without violating the constraints of another agent's higher priority plan and utilizing where possible the cooperative opportunities offered by the latter.**

Keywords—multiagent systems; coordination; cognitive agents; planning

Agent-Oriented Software Engineering: Current State and Future Directions

Michael Winikoff
University of Otago
New Zeland
michael.winikoff@otago.ac.nz

***Abstract*—The field of Agent-Oriented Software Engineering (AOSE) is concerned with the engineering aspects of developing agent-based systems, and how to support their development. Specifically, work in AOSE aims to provide practitioners with methodologies for the design of agent systems, and with supporting tools. A methodology can be seen as defining an overall process, where design artifacts (``models'') are used to capture key outcomes of the process. These design artifacts are expressed using one or more notations (which may be more or less formally defined). It is important that a methodology provide detailed usable guidelines for how to carry out key steps. For example, if a methodology says that the second step in the overall process is to identify the goals of the system, then this is not much use to the designer without some indication of the sorts of techniques that could be used to identify the goals. Work in the field varies in its focus: some papers take a higher-level view and describe whole methodologies, whereas others focus on a particular part or aspect of the software development process, for example, extending the modeling notation to better represent organizational aspects, or providing techniques for testing agent systems. This short-course will briefly review the history of AOSE, and then survey where the field of AOSE stands. Finally, the future of the field will be discussed: what are key directions and challenges for AOSE?**

Keywords*—*agent-oriented software engineering, survey

Modelling social influence among agents

Fred Amblard
Université de Toulouse
Toulouse, France
frederic.amblard@ut-capitole.fr

***Abstract*—In this course, I will first present different social phenomena that could be describe as typical cases of social influence (opinion dynamics, culture formation, attitude dynamics…). Hereafter, I will present partly the theoretical ground of these phenomena mostly from social psychology. I will then present different simple models that enable to capture and render those dynamics. I will end this course on the particular role played but social networks in such a context, where they are at the same time the support of social influence among agents (I adopt the attitude of my friends) and also an effect of such social influence (I tend to become friend with people sharing the same attitude)**

Keywords—social simulation; modeling

Building AOSE Situational Methods Using Method Fragments

Sara Casare
Laboratório de Técnicas Inteligentes - USP
São Paulo - Brazil
sjcasare@uol.com.br

Abstract*—Multi-Agent Systems (MAS) provide a new paradigm for conceptualizing, designing, and implementing software systems, ranging from manufacturing to process control, air traffic control, and information management. They are particularly attractive for creating software that operates in distributed and open environments, such as the Internet, and which simulates scenarios that serve as basis to create public policies and strategies to deal with complex problems, such as rescue after natural disasters and evacuation of public facilities. Nevertheless, in order to be adopted by the software industry, a controlled and disciplined way to conduct software development projects related to the aforementioned domains is needed. Despite the research community efforts while proposing methods for structuring and guiding the development of MAS, AOSE methods are still at an early stage, mainly being applied in the context of academic projects. Moreover, the development of complex systems using MAS requires specific methods and then the use of Situational Method Engineering techniques for MAS seems to be a promising solution for it. Thus, considering the class of problems whose solutions are tailored for adopting the MAS paradigm and which depends on organization and coordination, it could be benefited by using an organization-centered approach, through the adoption of some organizational model for MAS combined with some agent (or multiagent)-based software development method. Currently, a project team that looks for a disciplined way to develop a MAS application involving such organizational characteristics will not find a method ready to be used. An example of such a real application could be an information system to support the adoption of strategies for evacuating huge facilities under bomb threats. In this minicourse we will address such problems, by introducing a process for building methods out of reusable parts of methods - the so-called method fragments - to support the development of organizational-centered MAS. At the end of the tutorial, its audience must be able to build MAS situational methods for such a class of problems. Also, they will be encouraged to use the process to build MAS situational methods tailored to other classes of problems as well

Keywords— multiagent systems; agent-oriented software engineering; situational method engineering

Programming Multiagent Systems

Jomi Fred Hubner
Departamento de Engenharia de Automação e Sistemas - UFSC
Florianópolis - Brazil
jomi@das.ufsc.br

***Abstract*—In this course we will give a survey about agent-oriented programming and show how this paradigm was combined with organization-oriented programming and environment-oriented programming to yield multiagent programming. It will focus in a particular platform called JaCaMO, an integration of the platforms Jason (for agent-oriented programming), Cartago (for environment-oriented programming) and MOISE (for organization-oriented programming). It includes a hands on session in a lab.**

Keywords— multiagent systems; multiagent programming

Parte III

Full Papers - Artigos Completos

A Language to Specify the Interaction Considering Agents, Environment, and Organization

Maicon R. Zatelli, Jomi F. Hübner
Department of Automation and Systems Engineering
Federal University of Santa Catarina (UFSC)
Florianópolis, SC, Brazil
{*maicon,jomi*}*@das.ufsc.br*

***Abstract*—Interaction is a subject widely investigated in multi-agent systems (MASs), but there are still some open issues. Beyond the interaction usual between agents, we can conceive other kinds, like the interaction between agents and environment, or between agents and organization. These other kinds allow us to consider several situations that are not limited to speech acts. For example, the interaction between agents and environment allow us to define actions and events in interaction protocols, which would not be possible to represent with just the concept of speech act. In this paper we propose a language to specify the interaction considering the environment, organization, and agents. We also present a sketch of a dynamic of execution and some examples of protocols.**

***Keywords*-interaction; language; AEIO; environment; organization**

I. Introduction

This work is based on the AEIO approach (*Agent*, *Environment*, *Interaction*, *Organization*) [1], which conceives a multi-agent system (MAS) as composed of four basic components: agents, environment, interaction, and organization. Therefore, an MAS is not only based on the existence of agents, but there are other elements equally important. The developer should be able to see each of these parts clearly and separately.

Nowadays, it already exists many works about agents, organization, and environment. There are tools to specify, develop, and execute each one. For example, an MAS developer is able to build the environment by means of CArtAgO [2], the organization by means of AGR [3], ISLANDER [4], Moise [5], and so forth, and finally, the agents by means of GOAL [6], JADE [7], Jason [8], and so on. There are also tools to link these components to work together, such as JaCaMo [9]. However, none of the current tools provide features to specify and execute the interaction considering the existence of the three other components, that is, to define the interaction between agents, between agents and environment, and between agents and organization. As a consequence, the interaction is specified inside of the other MAS components, which results in difficulties to maintain, to reuse code, to debug, to work with open systems, etc.

In [10] we presented some advantages to separate the interaction of the other MAS components. With a separated interaction component it is possible to provide tools to improve debugging, because we can monitor the MAS execution from the interaction viewpoint. Moreover, as our proposal considers the interaction with the environment by means of actions or events, it is possible to represent how the agents have to proceed to interact with the several elements in the environment. We can also improve the use of open systems, where the agents can be heterogeneous. Open systems can be improved because the agents do not need to know in advance how to interact with the several elements in the MAS, but they just need to know how to handle the interaction component. Afterwards, the agents can follow the interaction specification to interact with the other MAS elements. Therefore, the migration of the agents to other MAS is also facilitated. In addition, the proposed model helps the agents to accomplish their organizational goals. The agents usually receive the goals related to their roles, but they do not receive what they must do to achieve them. In this case, the interaction helps the agents with a well-defined sequence of steps, including actions, messages, and events, which institutionalizes how the agents should interact to achieve the goals. Finally, a separated interaction component improve the reuse of code and the maintenance in an MAS because the whole interaction code is written separately from the rest of the system, which facilitates to visualize, to locate, and to change/update the interaction code.

Since the current languages are not suited to specify the interaction with the other MAS components, in this paper, our aim is to propose a programming language to specify interaction protocols considering the organization, the environment, and the agents (section III). This language follows the interaction model introduced in [10], where an *unified* and *coherent* interaction model is proposed (section II). In the proposed language, we have added some features to represent interaction protocols for several different scenarios. Furthermore, we also present a sketch of a dynamic of exe-

The authors are grateful for the support given by CNPq, grants 140261/2013-3 and 306301/2012-1

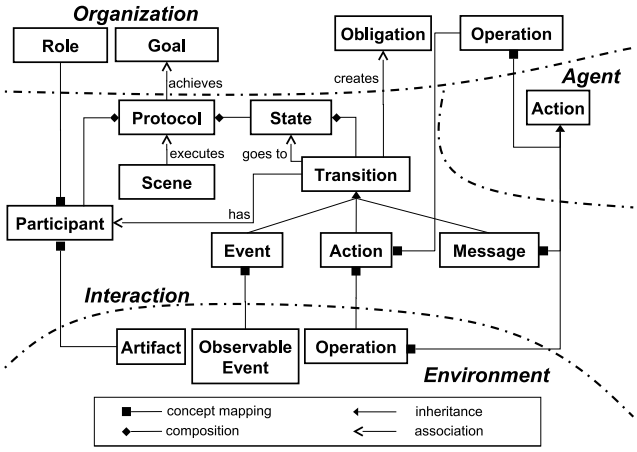

Figure 1: Conceptual model.

```
protocol        ::= "protocol" <ID> "{" description
                                        goals
                                        participants
                                        states
                                        transitions "}"
description     ::= ("description" ":" <STRING> ";")?
goals           ::= "goals" ":" (goal)+
goal            ::= <STRING> ";"
participants    ::= "participants" ":" (participant)+
participant     ::= participantId partDescription ";"
partDescription ::= ("agent" role | "artifact" type) partCardinality
partCardinality ::= ("all" | ("min" <INTEGER>)? ("max" (<INTEGER> | "+"))?)
participantId   ::= <ID>
type            ::= <STRING>
role            ::= <STRING>
states          ::= "states" ":" (state)+
state           ::= stateId ("initial" | "final")? ";"
stateId         ::= <ID>
transitions     ::= "transitions" ":" (transition)+
transition      ::= stateId "-" stateId "#" (occurrence | timeout | import)
timeout         ::= "timeout" <INTEGER> ";"
import          ::= "import" <STRING> mapping ";"
mapping         ::= "mapping" "{" (mapFromTo)+ "}"
mapFromTo       ::= participantId participantId ";"
occurrence      ::= pCardOccur "--" duty "->" pCardOccur ((trigger)+ | ";")
pCardOccur      ::= participantId ("[" <INTEGER> "]")?
duty            ::= dutyType <STRING>
dutyType        ::= ("event" | "action" | "message" "[" <ID> "]")
trigger         ::= ("trigger" pattern (":" content)? | ":" content) ";"
pattern         ::= <STRING>
content         ::= <STRING>
```

Figure 2: Language grammar.

cution (section IV) and some examples of protocols. Finally, before conclusion, we show the results and discussion with some related work.

II. Conceptual Model

This section briefly presents how the several MAS components are conceptually integrated with the interaction. Only the core ideas of the model are described here and examples will be presented in the next sections. More details can be found in [10].

In this model, the concepts of the other components were mapped onto the interaction concepts. Figure 1 shows the four MAS components and the links between the interaction and the others. In order to keep the figure clear and clean, we only show the concepts that were used in the model. The most important concept in our model is the interaction protocol, which is composed of a set of participants, transitions, states, and goals. Each transition links two states and it can be fired by an event, a message, or an action. The following paragraphs briefly explain how the other components are connected.

Protocols are linked to organizational concepts[1] in four points (top of Figure 1). When a protocol finishes successfully, an organizational goal is considered achieved. Other organizational concepts used in the interaction component are the roles, which constrain the participation of agents in the protocol; the obligations, which agents have to follow in order to accomplish the protocol; and the operations, which are the actions that some agent can perform in the organization such as adopt or leave some role.

The environment[2] also has some important concepts to be considered in the interactions. We mapped the concept of artifact onto a participant in the interaction component, which constrains the participation of artifacts in the protocol; the operations, which represent the actions that the agents can perform in the environment; and finally, the observable events, which agents can perceive in the environment such as an alarm, the color of something, etc.

In the end, the agent component provides the concepts of action, which can be some action performed in the environment or in the organization, and the message exchange, which represents the use of communicative acts to interact with the other agents.

III. A Language to Specify Interaction Protocols

In this section, we map the concepts presented in Figure 1 onto a programming language to specify interaction protocols[3]. Figure 2 presents the language grammar with its non-terminal symbols. A protocol is composed of a name, a description (represented by the non-terminal `description`), goals that will be achieved (represented by the non-terminal `goals`), participants (represented by the non-terminal `participants`), states (represented by the non-terminal `states`), and transitions (represented by the non-terminal `transitions`). We explain the main language features in detail by means of some examples.

Protocol 1 presents a first example, where the aim is to perform an election between the agents. The participation of the agents is defined in line 5, which state that they must play the role `elector` in the organization. The protocol includes the participation of a ballot box artifact to help the agents to vote in an anonymous approach (line 6).

The protocol is composed of three states (line 7): `n1`, `n2` e `n3`, where `n1` is the initial state and `n3` is the final state.

[1]Organizational concepts are explained in more details in [5], [11].

[2]Environment concepts are explained in the A&A meta-model introduced in [12].

[3]Due the lack of space we will only present the most important parts of the language.

Protocol 1 Election protocol.

```
1. protocol election {
2.   description: "Do an election";
3.   goals: "electLeader";
4.   participants:
5.     playerElector agent "elector" all;
6.     artBallotBox artifact "artifacts.BallotBox";
7.   states: n1 initial; n2; n3 final;
8.   transitions:
9.     n1 - n2 # playerElector -- action "vote(X)" -> artBallotBox
                 : ".string(X) & .is_agent(X)";
10.    n1 - n2 # timeout 30000;
11.    n2 - n3 # artBallotBox -- event "winner(Y)" -> playerElector;
12. }
```

Protocol 2 Attending protocol.

```
1. protocol attending {
2.   description: "Serve a costumer";
3.   goals: "chooseSeller";
4.   participants:
5.     playerCustomer agent "client";
6.     playerSeller agent "seller" all;
7.   states: k1 initial; k2; k3 final;
8.   transitions:
9.     k1 - k2 # playerCustomer -- message[tell] "needSeller" -> playerSeller;
10.    k2 - k3 # import "election.ptl" mapping { playerSeller playerElector; };
11. }
```

Protocol 3 Composition between the attending and election protocols.

```
1. protocol attending {
2.   description: "Serve a customer";
3.   goals: "chooseSeller";
4.   participants:
5.     playerCustomer agent "client";
6.     playerSeller agent "seller" all;
7.     artBallotBox artifact "artifacts.BallotBox";
8.   states: k1 initial; k2; n2[k2]; k3 final;
9.   transitions:
10.    k1 - k2 # playerCustomer -- message[tell] "needSeller" -> playerSeller;
11.    k2 - n2[k2] # playerSeller -- action "vote(X)" -> artBallotBox
                     : ".string(X) & .is_agent(X)";
12.    k2 - n2[k2] # timeout 30000;
13.    n2[k2] - k3 # artBallotBox -- event "winner(Y)" -> playerSeller;
14. }
```

The available transitions from state `n1` are those defined in lines 9 and 10. The first one can be triggered only by agents participating as `playerElector` in the protocol by doing the action `vote(X)` on the artifact `artBallotBox` (the ballot box). Moreover, when the protocol is in the state `n1` an obligation to perform the action `vote(X)` is created for the agents playing `elector`. Although created from a fact in the interaction component, this obligation exists in the organizational component of the MAS.

Note that a transition between `n1` and `n2` is defined with a `timeout` (line 10). The `timeout` is important in situations where the temporal constraints are fundamental, such as the time that an agent must wait for the proposals of the others in an auction. Moreover, the liveness in the protocol can be improved by means of a timeout, that is, the protocol will always achieve a final state.

The last transition (line 11) of the protocol defines that the participant `artBallotBox` must count the votes and emits an observable signal named `winner(Y)`, where `Y` is the winner name. With the successful termination of the protocol, the goal `electLeader` is achieved in the organization (line 3).

A second example of protocol (Protocol 2) describes the situation where a virtual agent decides to buy something in a website. The new protocol has three states (line 7): `k1`, `k2`, and `k3`, where `k1` is the initial state and `k3` is the final state. The first transition (line 9) defines that the agent that is playing the participant `playerCustomer` must send a message to the agents that are playing the participant `playerSeller` telling them that it needs some seller. The next transition (line 10) defines that the sellers must perform an election to decided which one will attend to the client. This transition has an `import` directive, which allows the composition of protocols. The address of the sub-protocol and a mapping between the participants of both protocols are necessary and since the election protocol was already specified before, then the composition allows reusing it.

The transition with the import directive (line 10) notifies the interpreter to establish a link between the state `k2` and the initial state of the election protocol. All final states of the election protocol are mapped to the state `k3`. The other states of the election protocol are just renamed to avoid the clash of identifiers. For example, the state `n2` of the election protocol is renamed to `[k2]` because the state `n2` is the result of the composition between the attending protocol and the election protocol by means of the state `k2`. The state names should be chosen carefully to keep the protocol understandable.

Another element that exists in the `import` directive is a `mapping` between the participants that exist in the attending protocol onto the participants that exist in the election protocol. In the example, the `playerElector` in the election protocol will be replaced by the participant `playerSeller` of the attending protocol while the participant `artBallotBox` is preserved. The description and goals of the election protocol are discarded due to the composition. The result of the composition between both protocols is presented in Protocol 3.

The language also provides two different kinds of cardinality: the participant cardinality and the transition cardinality. The former is related to the number of necessary entities to play some participant in the protocol. The latter is related to the number of entities that are necessary to perform the duty specified in some transition.

The participant cardinality is represented by means of the non-terminal symbol `partCardinality`. Broadly speaking, it defines the minimum and the maximum number of entities that must play some participant. While the minimum cardinality is represented by a number, the maximum cardinality also can be a number or even the `all` or + directives. The `all` directive informs the interpreter that all agents that are playing some organizational role or all artifacts that are of some type must play the participant. The + directive informs the interpreter that the number of required entities must be between one and the total of entities that the `all` directive calculates.

In contrast to the participant cardinality, the transition

cardinality allows just numeric values. Its aim is to define a minimum number of entities that must perform or undergo some occurrence specified in the transition. The non-terminal `pCardOccur`, next to the transition participant (between square brackets), represents the transition cardinality. By default, the omission of the transition cardinality results in the use of the same value that exists in the participant cardinality, meaning the participant cardinality and the transition cardinality are the same. There are four situations that can be represented by means of the transition cardinality. The first situation (`player1[1] -- action "foo" -> player2`) informs that at least *one* agent (left side) needs to execute some action in *all* artifacts (right side), that is, *all* artifacts must undergo at least *one* action. The second one (`player1 -- action "foo" -> player2[1]`) informs that *all* agents must execute an action in at least *one* artifact. The third one (`player1[1] -- action "foo" -> player2[1]`) informs that at least *one* agent is necessary to execute an action in at least *one* artifact. The latter (`player1 -- action "foo" -> player2`) informs that *all* agents must execute an action in at least *one* artifact and *all* artifacts must undergo at least *one* action.

Finally, the non-terminal `duty` defines what must be performed to fire the transitions and each transition may have several different verifications (represented by the non-terminal `trigger`) to make sure whether the occurrence is valid to fire it. The non-terminal `trigger` is composed of an expression to evaluate the occurrence pattern (represented by the non-terminal `pattern`) and an expression to evaluate the occurrence content (represented by the non-terminal `content`). If the `pattern` is omitted, the expression defined in the non-terminal `duty` will be considered as the `pattern`. For example, consider the action `vote(X)` presented in Protocol 1. The agent receives this obligation and it has to perform the action `vote`. As the pattern is omitted, the expression specified in the `duty` (i.e. `vote(X)`) is used as the pattern. Next to the symbol `:` (line 9), it is defined the expression to evaluate the content of the action. Suppose the agent tries to execute something like `vote("Ana",22)`. This action is not valid because it does not unify with the pattern `vote(X)`, then the action is discarded. However, suppose that the agent performs the action `vote(22)`. This action follows the pattern because it unifies the pattern (with `X = 22`), however the action is invalid because 22 is not a String as required by the `content`. Finally, suppose the agent tries to execute the action `vote("Ana")`. In this case, we have `X = "Ana"` and `"Ana"` is a `String`. In the case where Ana is also an agent, the action is valid to fire the transition.

IV. Sketch of the Dynamic of Execution

In this section, we briefly present the dynamic of execution. In an MAS with organization, the agents usually adopt a role and start working to accomplish the organizational goals related to its role. In the case where the goal has a protocol specified, the agent can instantiate it in order to achieve the goal. After the instantiation, the agent must ask the other agents to join the scene and must add the artifacts that will attend the scene. For example, in a scene of the Protocol 1, it is necessary the definition of the agents that will play the participant `playerElector` and also the artifact the will play the participant `artBallotBox`. Once the participating artifacts and agents are defined, the agent can start the execution of the scene. When the scene starts, it enables the transitions of the initial state and also creates the related obligations into the organization. Afterwards, the agent can follow the obligations in order to accomplish the protocol. For example, an obligation is created into the organization for all electors that are attending the scene of the Protocol 1 to obligate them to perform their votes when the scene starts.

Throughout the MAS execution, several occurrences (messages, actions, events) are intercepted and sent to the scenes. Then, the occurrences are processed in order to check whether they are valid to fire some enabled transition. For example, the agent Bob can execute the action `vote("Ana")` and this action must be intercepted and added in a queue to be processed afterwards.

The evaluation process occurs as following. While the invalid occurrences must be discarded, the valid ones must be added in a set and when the occurrences satisfy the cardinality of some enabled transition, the transition can fire and make the scene achieve a new state. The cardinality of a transition is satisfied when two conditions are true: (i) the number of valid occurrences that have the source entities that are playing the responsible participant of the transition is greater or equal to the cardinality specified for the responsible participant of the transition; (ii) the number of valid occurrences that have the target entities that are playing the target participant of the transition is greater or equal to the cardinality specified for the target participant of the transition.

For example, in a scene that executes the election protocol (Protocol 1), all electors are obligated to perform their votes. Thus, the agents must execute the action `vote` on the ballot box artifact. Each `vote` action that was performed is processed according to their informations, such as the agent that performed the vote, the description of the action vote with its parameters, and the ballot box that was used. Suppose the vote actions of all electors were processed and they were considered valid actions, however the vote action performed by the agent Bob still needs to be processed. Considering the same action `vote("Ana")` used in section III, note that it is valid to fire the transition between the states `n1` and `n2`, because it respects the validation expression defined in the transition. Also, suppose it has the responsible participant as an elector in the scene, the target participant is a ballot box in

the scene, and this action is not repeated. As this is the only action that was lacking to complete the cardinality defined in the transition (*all* electors must execute an vote action in *one* ballot box), considering this action, the transition `n1-n2` can fire and make the scene achieve the next state (`n2`).

A different situation can happen when some occurrence is valid but no transition has the cardinality satisfied to fire. For example, this situation could happen in some election with more than one elector. Even if some agent executes the action `vote("Ana")`, which is a valid vote action, it will be necessary to wait for all vote actions from the other electors to fire the transition between the states `n1` and `n2`.

Another way to achieve a new state is when some timeout happens. If no transition is fired in time, then a timeout can happen and the next state is pointed by the timeout transition. For example, in the election protocol (Protocol 1), if not all electors vote in 30 seconds, defined by the second transition between the states `n1` and `n2`, then a timeout happens and the scene achieves the state `n2`, pointed by the timeout transition.

Finally, when some transition is fired, the protocol can achieve a final state and then the organizational goals related to the protocol must be satisfied. For example, when some scene of the election protocol (Protocol 1) achieves the state `n3`, which is stated as final state, the goal `electLeader` must be satisfied in the organization.

V. Results and Discussion

In order to validate our approach we have integrated it into JaCaMo platform [9]. JaCaMo is a project that aims to permit the developer to consider each one of the MAS components as first class abstractions. Although the agent, environment, and organization components are already considered by this platform, the interaction component was not properly integrated.

Our main contribution is a programming language to specify interaction protocols considering the agents, environment, and organization. The aim is to institutionalize how the agents must interact with the different elements in an MAS to achieve the organizational goals by means of protocols. When an agent adopts some role in the organization, the agent receives the list of goals that it needs to achieve. In order to achieve them, the agent may look at the interaction component for a protocol that achieve them. It is useful since, sometimes, the agents may not know how to proceed to achieve their goals, then the protocol helps the agents in this situation by means of a well-defined sequence of steps.

The integration with the organization also helps the agents to search for partners to cooperate. They can do it reading the roles in the organization and the agents that are playing each role. Other important organization concept are the obligations that are useful to help the agents to follow the protocols. The obligations facilitate the agent programming and allow the agents to reason about them, specially whether the agents already can handle with organizational obligations. Moreover, such organizational mechanisms allow us to create punishment and reward mechanisms to prevent malicious behavior and reward the agents with good performances.

However, even in simple and closed systems, where the organization may not be necessary, our model needs an organization, which could require more time to develop the MAS. Indeed, our proposal is focused in more complex MAS, composed of agents, environment, and organization. Our aim is to integrate these components by means of the interaction and explore the advantages of this kind of MAS.

Other differential of our proposal is to regard the interaction with the environment, which allows the representation of more scenarios. For example, the language allows the specification about how the agents have to proceed to interact with the artifacts by means of actions and events.

The language also provides features like composition, cardinality, timeout, which allow the representation of several scenarios. These features allow improvements to reuse code, because it is possible to build more complex protocols by combining simple protocols; to represent real-time situations, because of the timeout mechanism; to comprehend the protocol, since the language is suited to conceive interaction protocols and it is written as a whole; etc.

The model can also be used in open systems where the agents can be heterogeneous. The separation of the interaction component of the other MAS parts endows the system with this capacity. The agents can join the MAS, play some role and read the interaction protocols in runtime. In order to do it, the agent must know how to follow protocols by means of obligations and then, the agent is able to learn new protocols and to interact with other agents or even with the environment. In addition, we can change the protocol specification without change the agent code. Indeed, it would be necessary for an agent to communicate with other agents if it does not understand what is the meaning of some protocol step that was modified. Finally, even in the case of open and heterogeneous MAS, a global behavior can be defined for the overall system.

A. Related Work

As already presented in [10], the interaction in MAS has several different approaches. Most of approaches do not consider the interaction between the agents and the environment or between the agents and the organization [3], [4], [11], [13]–[18], but some of them already try to conceive an interaction model that handles the interaction with the other components [19]–[23]. However, their aim is different than ours. For example, their interaction specification is conceived to be handled by humans during the MAS design and do not allow the agents to read it (or eventually to change it) at run-time. Furthermore, in [21], [22], the organization is considered in a simplified version, just with

roles, because their focus is to deploy a different way to use the environment during the interactions.

The MERCURIO framework [23], a very similar work to ours, focuses on interaction regarding agents and environment. The environment conception considers the actions performed by the agents and the event that the agents may sense. The limitation of MERCURIO is related to the organizational component. The roles in the interaction are not strongly connected with the roles existing in the organization. The existence of the other organizational concepts is not considered either. Indeed, the aim of MERCURIO is to deploy the interaction with the environment.

On the other hand, MAS-ML [19] and O-MaSE [20] are a modeling language and a methodology, respectively, that consider the interaction integration with the three other components. Both approaches are conceived for the specification phase, not regarding the implementation and execution phases. For example, these methodologies do not provide a feature to generate the interaction code. Therefore, our work contributes to fill the gap between this specification and the implementation.

REFERENCES

[1] Y. Demazeau, "From interactions to collective behaviour in agent-based systems," in *Proc. of EuroCogSci, Saint-Malo*, 1995, pp. 117–132.

[2] A. Ricci, M. Viroli, and A. Omicini, "CArtAgO: An infrastructure for engineering computational environments in MAS," in *Proc. of E4MAS*, D. Weyns, H. V. D. Parunak, and F. Michel, Eds., AAMAS 2006, Hakodate, Japan, 2006, pp. 102–119.

[3] J. Ferber, O. Gutknecht, and F. Michel, "From agents to organizations: An organizational view of multi-agent systems," in *Proc. of AOSE*. Springer, 2003, pp. 214–230.

[4] M. Esteva, B. Rosell, J. A. Rodriguez-Aguilar, and J. L. Arcos, "Ameli: An agent-based middleware for electronic institutions," in *Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, ser. Proc. of AAMAS. Washington, DC, USA: IEEE Computer Society, 2004, pp. 236–243.

[5] J. F. Hübner, J. S. Sichman, and O. Boissier, "A model for the structural, functional, and deontic specification of organizations in multiagent systems," in *Proc. of SBIA*. London, UK: Springer, 2002, pp. 118–128.

[6] K. V. Hindriks, "Programming rational agents in GOAL," *Multi-Agent Programming: Languages and Tools and Applications*, pp. 119–157, 2009.

[7] L. Braubach, E. Pokahr, and W. Lamersdorf, "Jadex: A bdi agent system combining middleware and reasoning," in *Ch. of Software Agent-Based Applications, Platforms and Development Kits*. Birkhaeuser, 2005, pp. 143–168.

[8] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*. Liverpool: Wiley, 2007.

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

[10] M. R. Zatelli and J. F. Hübner, "A unified interaction model with agent, organization, and environment," in *Anais do IX Encontro Nacional de Inteligência Artificial (ENIA@BRACIS)*, Curitiba, Brazil, 2012.

[11] V. Dignum, J. Vázquez-salceda, and F. Dignum, "Omni: Introducing social structure, norms and ontologies into agent organizations," in *Proc. of PROMAS*. Springer, 2004, pp. 181–198.

[12] A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the A&A meta-model for multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 17, pp. 432–456, 2008.

[13] E. Platon, N. Sabouret, and S. Honiden, "Overhearing and direct interactions: point of view of an active environment, a preliminary study," in *Proc. of E4MAS*. Springer, 2005, pp. 121–138.

[14] D. Keil and D. Q. Goldin, "Indirect interaction in environments for multi-agent systems," in *Proc. of E4MAS*, 2005, pp. 68–87.

[15] J. Saunier and F. Balbo, "Regulated multi-party communications and context awareness through the environment," *Multiagent Grid Syst.*, pp. 75–91, 2009.

[16] O. Boissier, F. Balbo, and F. Badeig, "Controlling multi-party interaction within normative multi-agent organizations," in *Proc. of MALLOW*, 2010, pp. 17–32.

[17] A. Hübner, G. P. Dimuro, A. C. R. Costa, and V. L. D. Mattos, "A dialogic dimension for the Moise+ organization model," in *Proc. of MALLOW*, 2010, pp. 21–26.

[18] M. P. Singh, "Information-driven interaction-oriented programming: BSPL, the blindingly simple protocol language," in *Proc. of AAMAS*, 2011, pp. 491–598.

[19] V. T. Silva, R. Choren, and C. J. P. de Lucena, "A uml based approach for modeling and implementing multi-agent systems," in *Proc. of AAMAS*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 914–921.

[20] S. A. DeLoach and J. L. Valenzuela, "An agent-environment interaction model," in *Proc. of AOSE*. Berlin, Heidelberg: Springer, 2006, pp. 1–18.

[21] E. Oliva, M. Viroli, A. Omicini, and P. Mcburney, "Argumentation and artifact for dialogue support," in *Argumentation in Multi-Agent Systems*, ser. LNAI. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 107–121.

[22] Y. Kubera, P. Mathieu, and S. Picault, "Interaction-oriented agent simulations: From theory to implementation," in *Proc. of ECAI*. Patras, Greece: IOS Press, 2008, pp. 383–387.

[23] M. Baldoni, C. Baroglio, F. Bergenti, E. Marengo, V. Mascardi, V. Patti, A. Ricci, and A. Santi, "An interaction-oriented agent framework for open environments," in *Proc. of AI*IA*. Berlin, Heidelberg: Springer, 2011, pp. 68–79.

Extending deontic interpreted systems with action logic

Raquel de Miranda Barbosa, Antônio Carlos da Rocha Costa
Programa de Pós-Graduação em Modelagem Computacional - PPGMC
Programa de Pós-Graduação em Computação - PPGCOMP
Universidade Federal do Rio Grande (FURG)
Rio Grande - RS, Brasil
{raq.mbarbosa,ac.rocha.costa}@gmail.com

***Abstract*—This paper presents an extension to deontic interpreted systems with the use of action logic for the specification of normative aspects in multiagent systems. The paper presents a preliminary formalization of operators required and describes a simple example of the application of this formalization, using the segregation model (a simulation model available in NetLogo platform) through the formalization and proof of some properties of this system.**

Keywords*—*Deontic Interpreted Systems; Action Logic; Multiagent Systems; Norms

I. INTRODUÇÃO

Aspectos formais são frequentemente considerados importantes no processo de desenvolvimento de software, visto que permitem a descrição de especificações não ambíguas e verificações e correções em diferentes etapas do desenvolvimento, entre outros aspectos. Quando se trata do desenvolvimento de sistemas baseados em agentes, não é diferente. Há uma grande preocupação na formalização de diferentes aspectos do sistema. Existem diversas teorias formais na área, as mais comuns baseadas em lógica, porém nem sempre fica claro o que tais teorias devem representar, como são aplicadas ao problema em questão e como podem ser integradas em um processo de especificação e verificação de sistemas.

Quando pensamos em sistemas multiagentes, devemos considerar dois aspectos importantes: i) seus aspectos organizacionais, que refletem a estrutura e funcionamento dos sistemas através da descrição de papéis de agentes, grupos, instituições/organizações e seus relacionamentos; e ii) a questão de normas (características regulatórias do sistema), que ditam como os agentes devem se comportar no sistema, o que eles devem ou não fazer e até mesmo casos nos quais os padrões são violados para se atingir um objetivo mais importante (cf. [1]). Um sistema multiagente que usa normas (ou baseado em normas) é chamado sistema multiagente normativo, conforme apresentado em [2]. De acordo com Boella [2], sistemas normativos são um exemplo de uso de teorias sociológicas em sistemas multiagentes, i.e., o relacionamento entre teoria de agentes e ciências sociais como Sociologia, Economia e Ciência Jurídica. Estes conceitos são importantes em sistemas multiagentes sociais, visto que existe o interesse no comportamento dos agentes tanto no nível individual como no social.

A representação de normas em SMA normalmente é feita através de fórmulas da lógica deôntica, utilizando-se operadores de obrigação (O-*obligation*), permissão (P-*permission*) e proibição (F-*forbidden*) e conceitos relacionados [3].

Um dos problemas encontrados é que a maioria das linguagens de especificação não usa operadores deônticos, dificultando o processo de verificação de compatibilidade do sistema em relação às suas normas.

Em [4] é apresentada uma formalização para a noção de conhecimento em sistemas distribuídos, onde, embora o foco seja sistemas distribuídos de processadores, estes "processadores" podem ser pensados como pessoas ou agentes, por exemplo. A ideia está baseada no modelo clássico de mundos possíveis (cf. [5]), onde diz-se que um agente conhece um fato φ se φ é verdadeiro em todos os mundos que ele considera possíveis; porém o autor salienta que em sistemas distribuídos não se tem um conhecimento global do sistema. Em contrapartida, é apresentada uma interpretação concreta para a noção de mundos possíveis, identificando o sistema como um conjunto de *runs*, onde um *run* é uma descrição completa do que acontece no sistema ao longo do tempo. São definidos pontos como sendo um par (r, t), consistindo de um *run* e um tempo e estes pontos do sistema são vistos como mundos possíveis. Em qualquer ponto o sistema está em algum estado global o qual pode ser visto apenas como uma tupla contendo os estados locais de cada processador.

Esta formalização deu origem à ideia de sistemas interpretados, (cf. [6] e [7]), onde adiciona-se uma função de interpretação para proposições que descrevem fatos do sistema e, posteriormente, foi estendida por Lomuscio [8] com a introdução de conceitos deônticos, permitindo a utilização de fórmulas com um operador similar ao de obrigação, mas que representa a ideia de um comportamento ideal/correto de um agente.

Neste contexto, este artigo apresenta uma aplicação destes operadores deônticos, definidos por Lomuscio e Sergot para a descrição de aspectos normativos em sistemas multiagentes, estendendo-os através do uso de lógica de ações. O artigo apresenta uma formalização preliminar dos operadores necessários e descreve um simples exemplo de aplicação desta formalização em um modelo de Segregação (cf. e.g [9]) através da prova de algumas propriedades deste sistema.

O modelo de Segregação é um modelo de simulação disponível na ferramenta NetLogo [10], inspirado nos artigos de Thomas Schelling sobre sistemas sociais [9]. Este modelo apresenta dois tipos de agentes que se dão bem uns com os

outros em um determinado ambiente. Cada um deles quer ter certeza de que vive próximo de algum outro do mesmo tipo. A simulação mostra o que acontece em uma população com estas características [11].

O artigo está estruturado conforme descrito a seguir. Na Seção 2 é apresentada a teoria de sistemas deônticos interpretados, na qual este trabalho baseia-se. A Seção 3 descreve a extensão proposta para tratar sistemas multiagentes normativos. Na Seção 4 é apresentado um exemplo de utilização desta proposta, seguida pelas conclusões e trabalhos futuros.

II. Sistemas Deônticos Interpretados

A noção básica de sistemas interpretados foi introduzida em [7], onde um sistema é visto como um conjunto de *runs*. Se observarmos o sistema em qualquer ponto no tempo, cada um dos agentes está em algum estado (chamado estado local do agente), que encapsula toda a informação a qual o agente tem acesso. O sistema é dividido conceitualmente em dois componentes: agentes e ambiente (que representa tudo o que é relevante). O estado global de um sistema descreve o sistema em um dado ponto no tempo. O estado global com n agentes é uma tupla (n+1) da forma $(s_e, s_1, ..., s_n)$ onde s_e é o estado do ambiente e $s_1, ..., s_n$ são os estados locais do agente i. Considerando-se que os sistemas não são estáticos, define-se *run* como a descrição completa de como o estado global do sistema evolui no tempo (por exemplo, *r(0)* representa o estado global do sistema em uma possível execução 0).

Em [8], [12] descreve-se a adaptação desta noção para fornecer um fundamento básico para questões deônticas.

Para a caracterização de sistemas interpretados, assume-se um conjunto P de átomos proposicionais e Ag = $\{1, .., n\}$ de agentes.

A linguagem $\mathcal{L}$ é definida como segue:

φ ::= **false** | qualquer elemento de P | $\neg\varphi$ | $\varphi \wedge \varphi$ | $\mathcal{O}_i\varphi (i \in Ag)$

O operador modal indexado $\mathcal{O}_i$ é usado para representar circunstâncias de **funcionamento correto** do agente i onde: $\mathcal{O}_i\varphi$ pode ser lido como "em todas as alternativas de funcionamento corretamente possíveis do agente i, φ é o caso", ou "quando o agente i está funcionando corretamente (em relação a algum protocolo ou especificação), φ é o caso".

A fórmula φ pode se referir tanto a propriedades locais quanto globais ou a ambas ao mesmo tempo.

O operador modal $\mathcal{P}_i$ é o dual de $\mathcal{O}_i$ de forma que $\mathcal{P}_i\varphi =_{def} \neg\mathcal{O}_i\neg\varphi$. $\mathcal{P}_i\varphi$ pode ser lido como "em alguns dos estados nos quais o agente i opera corretamente φ ocorre", ou "φ acontece em algumas das alternativas de funcionamento correto do agente i".

A escolha do operador $\mathcal{O}$ deve-se ao fato de que sua semântica é similar àquela do operador de obrigação da lógica deôntica padrão. No entanto, não é correto ler $\mathcal{O}_i\varphi$ como "é obrigatório para o agente i que φ".

No exemplo da Segregação, podemos pensar que o comportamento correto do agente é "ser feliz", ou seja, ele está vivendo perto de, no mínimo, alguns vizinhos do mesmo tipo (se o agente não está feliz, ele deve mover-se para uma nova posição). O conhecimento que um agente pode ter em um determinado estado, por exemplo, é sua posição corrente, se ele está sozinho e se está feliz. Utilizando o operador $\mathcal{O}_i$ podemos escrever $\mathcal{O}_1 happy$ para representar a proposição "em todas as alternativas possíveis de funcionamento correto do agente 1, ele está feliz".

Sistemas Deônticos Interpretados de Estados Globais (*IDS - Deontic Interpreted Systems*) são definidos em [8], [12], da seguinte forma:

Um IDS para n agentes é um par $IDS = (DS, \pi)$, onde DS é um *sistema deôntico de estados globais* e π é uma *interpretação* para os átomos. Os autores não tratam a noção de tempo, desconsiderando *runs* e trabalhando apenas com estados.

Sistemas deônticos de estados globais são definidos, assumindo-se que, para cada agente, seu conjunto de estados locais pode ser dividido em estados permitidos e proibidos (chamados de estados verdes e vermelhos, respectivamente). Desta forma, dados n agentes e $n + 1$ conjuntos mutuamente disjuntos e não vazios G_e, G_1, ..., G_n, um *sistema deôntico de estados globais* é qualquer sistema de estados globais definido em conjuntos quaisquer $L_e \supseteq G_e$, $L_1 \supseteq G_1$, ..., $L_n \supseteq G_n$.

Um sistema de estados globais para n agentes S é um subconjunto não vazio de um produto cartesiano $L_e \times L_1 \times ... \times Ln$, sendo $L_1, ..., L_n$ os conjuntos de estados locais para cada agente do sistema e L_e o conjunto de estados para o ambiente.

G_e é chamado o conjunto de *estados verdes* para o ambiente e, para cada agente i, G_i é chamado o conjunto de estados verdes para o agente i. O complemento de G_e em relação a L_e (respectivamente, G_i em relação à L_i) é chamado o conjunto de *estados vermelhos* para o ambiente (respectivamente para o agente i).

Uma coleção de estados verdes e vermelhos identifica uma classe de estados globais. A classe de sistemas deônticos de estados globais é denotada por $\mathcal{DS}$.

No caso da Segregação, os estados verdes seriam aqueles em que o agente está sozinho em uma determinada posição e está feliz, enquanto os demais seriam considerados vermelhos.

Esta ideia de sistemas deônticos interpretados também é utilizada em outros artigos que descrevem pesquisas sobre a representação e raciocínio sobre estados de funcionamento correto e incorreto dos agentes e o sistema como um todo [8], verificação automática de sistemas deônticos interpretados através de *model checking* ([13], [14]), um sistema de *tableaux* para sistemas deônticos interpretados [15], entre outros.

III. Estendendo Sistemas Deônticos Interpretados com Lógica de Ações

Em trabalhos anteriores [16], foi investigada a possibilidade de utilização de métodos formais tradicionais de engenharia de software para a especificação de organizações de sistemas multiagentes (mais especificamente o método RAISE e sua linguagem de especificação RSL) e observou-se que esta linguagem, bem como as demais linguagens de especificação tradicionais não oferecem suporte ao uso de operadores deônticos para descrever características de regulação dos sistemas. No caso particular de sistemas multiagentes, é

necessário (ou desejável) formalizar aspectos, tais como: o que o agente sabe em um determinado estado, o que ele pode fazer, o que acontece quando ele executa certas ações, entre outras coisas.

Para tratar estas questões, utilizando-se a noção de sistemas deônticos interpretados, é necessária uma extensão das definições propostas por [17], de forma a tratar os aspectos relacionados à realização de ações pelos agentes, permitindo a representação de situações do tipo "Em todas as alternativas de funcionamento correto do agente i, se ele executar a ação α, ele vai para um estado verde".

O modelo M é definido como uma tupla

$M = (G, \pi, \mathcal{R}_i^K, \mathcal{R}_i^O, \mathcal{R}_i^\alpha, \Sigma_A, P)$, sendo A um conjunto de ações e i = 1..n (agentes), onde:

- G é um conjunto de estados globais (i.e. g = $\langle l_e, l_1, ..., l_n \rangle$)
- π é uma interpretação que associa a cada estado em G uma atribuição de verdade para as proposições primitivas de P (i.e., $\pi(g) : P \rightarrow \{true, false\}$ para cada estado $g \in G$).
- $\mathcal{R}_i^K$ é uma relação binária sobre G (i.e., $\mathcal{R}_i^K \subseteq G \times G$), tal que para todo $g \in G$ existe pelo menos um $g' \in G$ com $g \rightarrow g'$, ou seja, $(g, g') \in \mathcal{R}_i^K$.
- $\mathcal{R}_i^O$ é uma relação binária sobre G (i.e., $\mathcal{R}_i^O \subseteq G \times G$), tal que para $g \in G$, $g\mathcal{R}_i^O g'$, se $l_i' \in \mathbf{G}_i$ (estados verdes do agente i).
- $\mathcal{R}_i^\alpha$ (onde $\alpha \in A$) é uma relação binária (de transições rotuladas) sobre $G \times G$ (i.e., $\mathcal{R}_i^\alpha \subseteq G \times G$), tal que para todo $g \in G$, $g\mathcal{R}_i^\alpha g'$ (escrita como g $\stackrel{\alpha_i}{\rightarrow}$ g').
- Σ_A é um conjunto de ações ou rótulos.
- P é um conjunto de proposições primitivas.

Considerando-se P um conjunto de proposições atômicas, A um conjunto de ações e $i \in \{1..n\}$ (agentes), o conjunto $\mathcal{L}$ é definido por:

1) $\varphi \in P$ implica $\varphi \in \mathcal{L}$
2) $\alpha \in A$ implica $\alpha \in \mathcal{L}_{Act}$
3) φ e $\psi \in \mathcal{L}$ implica $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi \in \mathcal{L}$.
4) $\varphi \in \mathcal{L}$ implica $\mathcal{O}_i\varphi$, $\mathcal{P}_i\varphi$, $K_i\varphi \in \mathcal{L}$
5) $\mathbf{g}_i \in \mathcal{L}$
6) $\varphi \in \mathcal{L}$, $\alpha \in \mathcal{L}_{Act}$ implica $[\alpha_i]\varphi$, $\langle\alpha_i\rangle\,\varphi \in \mathcal{L}$
7) $\varphi \in \mathcal{L}$ implica $\varphi? \in \mathcal{L}_{Act}$
8) $\alpha, \beta \in \mathcal{L}_{Act}$ implica $\alpha; \beta\ , \alpha + \beta,\ \alpha* \in \mathcal{L}_{Act}$

Note que o operador ; significa composição sequencial, o operador + representa uma escolha não-determinística e o operador * é usado para uma representação finita arbitrária. A construção desta linguagem está baseada em [18].

A interpretação para os operadores $\mathcal{O}_i, \mathcal{P}_i, \mathbf{g}_i, K_i, [\alpha_i]$ e $\langle\alpha_i\rangle$ é dada da seguinte maneira:

- $\mathcal{O}_i\varphi$: em todas as alternativas possíveis de funcionamento correto do agente i, φ é o caso.
- $\mathcal{P}_i\varphi$: em alguns dos estados nos quais o agente i opera corretamente φ ocorre.
- $\mathbf{g}_i$: o agente i está em um estado local de funcionamento correto de acordo com seu protocolo.
- $K_i\varphi$: o agente i conhece φ.
- $[\alpha_i]\ \varphi$: após a ação α ser executada pelo agente i, necessariamente se obtém φ.
- $\langle\alpha_i\rangle\,\varphi$: após a ação α ser executada pelo agente i, possivelmente se obtenha φ.

Para verificar a satisfação deste sistema deôntico interpretado de estados globais deve-se provar que:

$$M \models_g \varphi,$$

para todo $\varphi \in \mathcal{L}$.

Desta forma, tem-se que:

1) $M \models_g$ **true**
2) $M \models_g p$ se $g \in \pi(p)$
3) $M \models_g \neg\varphi$ se $\neg\ M \models_g \varphi$
4) $M \models_g \varphi \wedge \psi$ se $(M \models_g \varphi)$ e $(M \models_g \psi)$
5) $M \models_g \varphi \vee \psi$ se $(M \models_g \varphi)$ ou $(M \models_g \psi)$
6) $M \models_g \varphi \rightarrow \psi$ se (não $M \models_g \varphi$) ou $(M \models_g \psi)$
7) $M \models_g \varphi \leftrightarrow \psi$ se $(M \models_g \varphi$ implica $M \models_g \psi)$ e $(M \models_g \psi$ implica $M \models_g \varphi)$
8) $M \models_g \mathcal{O}_i\varphi$ se, $\forall g'$, temos que $gR_i^O g'$ implica $M \models_{g'} \varphi$
9) $M \models_g \mathcal{P}_i\varphi$ se, $\exists g'$, tal que $gR_i^O g'$ implica $M \models_{g'} \varphi$
10) $M \models_g \mathbf{g}_i$, se $g \in R_i^O(g)$ $(i \in A)$
11) $M \models_g K_i\varphi$, se para todo g' temos que $gR_i^K g'$ implica $M \models_{g'} \varphi$
12) $M \models_g [\alpha_i]\varphi$, se $\forall g'$ temos que $gR_i^\alpha g'$ implica $M \models_{g'} \varphi$
13) $M \models_g \langle\alpha_i\rangle\,\varphi$, se $\exists g'$ tal que $gR_i^\alpha g'$ implica $M \models_{g'} \varphi$
14) $M \models_g [\alpha_i; \beta_j]\varphi$, se $\forall g', g''$ temos que $gR_i^\alpha g'$ e $g'R_j^\beta g''$ implica $M \models_{g''} \varphi$
15) $M \models_g [\alpha_i + \beta_j]\varphi$, se $\forall g'$ temos que $gR_i^\alpha g'$ implica $M \models_{g'} \varphi$ ou $gR_j^\beta g'$ implica $M \models_{g'} \varphi$
16) $M \models_g [\alpha_i^*]\varphi$, se $\forall g'$ temos que $gR_i^\alpha g'$ implica $M \models_{g'} [\alpha_i^*]\varphi$
17) $M \models_g \langle\alpha_i; \beta_j\rangle\,\varphi$, se $\exists g', g''$ tal que $gR_i^\alpha g'$ e $g'R_j^\beta g''$ implica $M \models_{g''} \varphi$
18) $M \models_g \langle\alpha_i + \beta_j\rangle\,\varphi$, se $\exists g'$ tal que $gR_i^\alpha g'$ implica $M \models_{g'} \varphi$ ou $gR_j^\beta g'$ implica $M \models_{g'} \varphi$
19) $M \models_g \langle\alpha_i^*\rangle\,\varphi$, se $\exists g'$ tal que $gR_i^\alpha g'$ implica $M \models_{g'} [\alpha_i^*]\varphi$

A fim de ilustrar esta formalização em um sistema multiagente é utilizado o Modelo de Segregação. Na Seção IV, este modelo é especificado como um sistema deôntico interpretado e algumas propriedades são formalizadas e verificadas.

IV. Exemplo: o Modelo de Segregação

O modelo de Segregação, inspirado nos artigos de Thomas Schelling sobre sistemas sociais [9] e disponível na ferramenta de simulação NetLogo [10], apresenta dois tipos de agentes (representados na simulação por tartarugas verdes e vermelhas) que convivem em um determinado ambiente. Cada um deles quer ter certeza de que vive próximo de algum outro do mesmo

tipo (mesma cor). A simulação mostra o que acontece em uma população com estas características [11].

Este modelo pode ser formalizado como um sistema deôntico interpretado. Para isto deve-se fazer escolhas a respeito de como modelar os estados locais de cada agente e do ambiente. Aqui optou-se pelas seguintes alternativas (observa-se que as cores das tartarugas, definidas no modelo como verdes e vermelhas, foram alteradas para evitar a confusão com os estados verdes (desejáveis) e vermelhos (indesejáveis)):

- O estado local de cada agente é representado por uma tupla `(i,p,c,sn,on,sw)`, onde `i` indica o índice do agente, `p` indica a posição representada pelo par (linha,coluna), `c` a cor da tartaruga: branca (W) ou laranja (O), `sn` (*similar nearby*) indica o número de vizinhos de mesma cor, `on` (*other nearby*), indica o número de vizinhos de cor diferente e `sw` (*similar wanted*) o percentual de similaridade desejado para o agente.
- O estado local do ambiente é representado pela tupla `(tp,ta,OP,FP)`, onde `tp` (*total positions*) indica o número de posições existentes no ambiente, `ta` (*total agents*), indica o número de agentes da simulação, `OP` o conjunto de posições ocupadas e `FP` o conjunto de posições livres.

O conjunto de proposições atômicas que podem ser observadas neste exemplo é representado por:

P = {position$_i$(x,y), happy$_i$, alone$_i$, occupied(x,y) }

O conjunto de ações que os agentes podem realizar em cada estado é representado por:

Act = {ϵ, move$_i$(x,y) },

onde ϵ representa a falta de movimentação do agente, ou seja, ele permanece na mesma posição, e move$_i$(x,y), representa a movimentação do agente i para a posição (x,y) no ambiente.

Um estado correto para o agente i ($\mathbf{g}_i$) é aquele em que o agente está feliz (i.e. com o percentual de vizinhos da mesma cor de acordo com o desejado) e sozinho na posição que ocupa.

O comportamento correto de um agente nesta simulação segue o seguinte protocolo:

$$\boxed{\text{enquanto } \neg(happy)_i : move_i(x,y)}$$

Isto significa que os agentes que ainda não estão em uma posição com o percentual desejado de vizinhos da mesma cor, devem saltar para outra posição aleatória até que fiquem "felizes". Uma alternativa de funcionamento correto, neste caso, é aquela em que, partindo do estado inicial, após alguns saltos, o agente chega a um estado correto (verde).

Considere o exemplo mostrado na Tabela I, onde o ambiente possui 9 posições e 4 agentes. Cada agente está representado pela sua cor (W-*white* ou O-*orange*) e um índice que o identifica.

Neste contexto, o estado Global Inicial do Sistema ($L_e \times L_i \times ... \times L_n$) é descrito como:

TABLE I. EXEMPLO DE SEGREGAÇÃO

W_1	W_2	
	O_3	
O_4		

g$_0$ = ((9,4, {(1,1),(1,2),(2,2),(3,1)}, {(1,3),(2,1),(2,3),(3,2),(3,3)}), (1,(1,1),W,1,1,50), (2,(1,2),W,1,1,50), (3,(2,2),O,1,2,50), (4,(3,1),O,1,0,50)).

Os possíveis estados locais de cada agente i podem ser divididos em estados desejáveis - verdes (G_i) e não desejáveis - vermelhos (R_i) e são representados como:

$G_i = \{(i,(x,y),sn,on,50)$ onde x e $y \in \{1..3\}\}$, sn e $on \leq 3$, para $i = 1..4$

$R_i = \{(i,(x,y),sn,on,50)$ onde x e $y \in \{1..3\}\}$, sn e $on \leq 3$, para $i = 1..4$

Uma possível configuração final do ambiente pode ser vista na Tabela II, cujo estado global seria representado por:

g$_n$ = ((9,4, {(1,1),(1,2),(3,1),(3,2)}, {(1,3),(2,1),(2,2),(2,3),(3,3)}), (1,(1,1),W,1,1,50), (2,(1,2),W,1,1,50), (3,(3,2),O,0,1,50), (4,(3,1),O,1,0,50))

onde pode-se observar que nos estados locais de todos os agentes o número de vizinhos de mesma cor é maior ou igual ao número de vizinhos de cor diferente, ou seja, todos estão felizes e, assim, não precisam deslocar-se para outro ponto.

TABLE II. EXEMPLO DE SEGREGAÇÃO

W_1	W_2	
O_4	O_3	

Para exemplificar este sistema deôntico, observe a Figura 1.

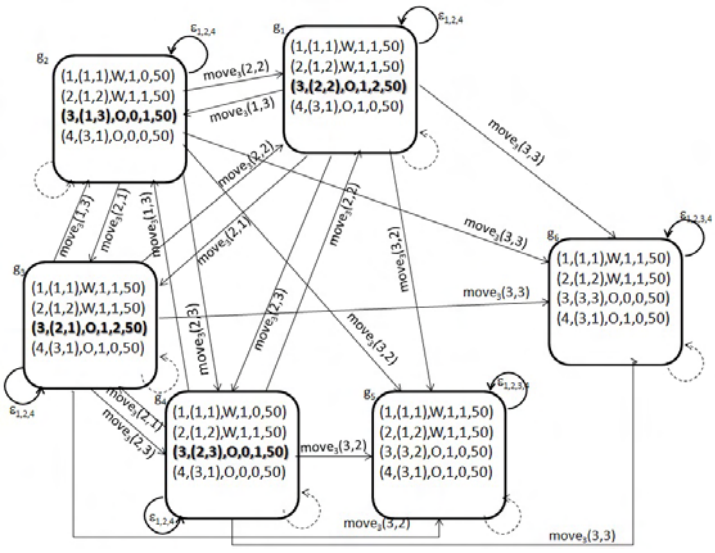

Fig. 1. Frame gerado a partir do sistema deôntico global.

Nesta figura o ambiente não é considerado e os estados locais para os agentes estão identificados dentro de cada estado global $g_1, ..., g_6$.

A Figura apresenta um subconjunto de SD que apresenta configurações para os agentes 1, 2, 3 e 4, respectivamente. Os links rotulados indicam as relações R_i^{α} (para i $\in \{1,2,3,4\}$), onde os rótulos indicam as ações executadas. Neste exemplo as ações de movimentação são apenas para o agente 3, visto que ele é o único que precisa mudar de posição para chegar a um estado final correto.

As setas tracejadas representam as relações de conhecimento R_i^K (para i $\in \{1,2,3,4\}$). Na figura, não estão representados os conhecimentos de cada agente, porém cabe salientar que em cada um dos estados globais, este refere-se a fatos como sua posição, cor, felicidade e se o agente está sozinho nesta posição. Por exemplo, considerando-se o agente 1 no estado global g_1 pode-se afirmar que ele conhece os fatos $position_1(1,1), color_1(W), happy_1$ e $alone_1$.

A relação $\mathrm{R}_i^{\mathcal{O}}$ não está explícita na figura, porém pode ser explicada observando-se os estados locais dos agentes. Os agentes 1, 2 e 4 já estão em estados corretos em todos os estados globais analisados $(g_1, ..., g_6)$, portanto os estados g' alcançáveis para os agentes 1, 2 e 4, a partir de $g \in \{g_1, ..., g_6\}$, são eles mesmos (e.g. $g_1 R_i^{\mathcal{O}} g_1$, para $i \in 1, 2, 4$). O agente 3 não está em estado correto nos estados g_1, g_2, g_3 e g_4, representado em negrito na figura. Por exemplo, no estado g_1 (configuração inicial do sistema apresentada na Tabela I, o agente 3 tem dois vizinhos W e apenas um vizinho O). Este agente somente vai obter um estado correto quando, após alguns movimentos, chegar a um dos estados g_5 ou g_6 (e.g $g_1 R_3^{\mathcal{O}} g_5$).

A. Propriedades

Com base no sistema deôntico especificado, é possível verificar algumas propriedades, tais como:

1) $M \models_{g_1} \mathcal{O}_1\, happy_1$: em todas as alternativas de funcionamento correto do agente 1, partindo do estado g_1, ele está feliz.

$$M \models_{g_1} \mathcal{O}_1\, happy_1$$
$$|$$
$$M \models_{g_1} happy_1$$
$$|$$
$$g_1 \in \pi(happy_1)$$
$$g_1 \in \{g_1, g_2, g_3, g_4, g_5, g_6\}\ \checkmark$$

2) $M \models_{g_1} \mathcal{O}_1(\mathbf{g}_1 \rightarrow happy_1 \wedge alone_1)$: em todas as alternativas de funcionamento correto do agente 1, partindo do estado g_1, se ele está em um estado verde, então ele está feliz e sozinho.

$$M \models_{g_1} \mathcal{O}_1(\mathbf{g}_1 \rightarrow happy_1 \wedge alone_1)$$
$$|$$
$$M \models_{g_1} (\mathbf{g}_1 \rightarrow happy_1 \wedge alone_1)$$
$$|$$
$$\neg M \models_{g_1} \mathbf{g}_1 \ \vee\ M \models_{g_1} (happy_1 \wedge alone_1)$$
$$|$$
$$g_1 \in \pi(happy_1 \wedge alone_1)$$
$$g_1 \in \{g_1, g_2, g_3, g_4, g_5, g_6\}\ \checkmark$$

3) $M \models_{g_1} K_1(position_1(1,1))$: o agente 1, no estado g_1, sabe que está na posição (1,1).

$$M \models_{g_1} K_1(position_1(1,1))$$
$$|$$
$$M \models_{g_1} (position_1(1,1))$$
$$|$$
$$g_1 \in \pi(position(1,1))$$
$$g_1 \in \{g_1, g_2, g_3, g_4, g_5, g_6\}\ \checkmark$$

V. Conclusão

Este artigo apresentou uma extensão à abordagem de sistemas deônticos interpretados para a especificação de sistemas multiagentes, incluindo operadores relacionados às ações executadas pelos agentes. A ideia é que se possa utilizá-la para especificar organizações de sistemas multiagentes onde existem regras que regem o comportamento dos seus elementos.

Outros aspectos ainda necessitam ser desenvolvidos como, por exemplo, um método de cálculo para a validação de propriedades. Estão sendo desenvolvidas, pelos autores, regras para um sistema de tableaux que envolvam estes novos operadores de forma que se possa demonstrar a prova destas propriedades. Pretende-se aplicar estas definições em outros exemplos mais complexos e com aspectos organizacionais bem definidos e verificar a possibilidade de inclusão destes operadores em uma linguagem de especificação formal.

Agradecimentos

O trabalho conta com apoio financeiro do CNPq - Edital PDI, através do Projeto MSPP - Modelagem e Simulação de Políticas Públicas. Raquel de Miranda Barbosa é bolsista PosDoc/CAPES junto ao PPGMC/FURG.

Título em Português: Estendendo sistemas deônticos interpretados com lógica de ações.

References

[1] F. Dignum, "Autonomous agents with norms," *Artificial Intelligence and Law*, vol. 7, pp. 69–79, 1999.

[2] G. Boella and L. V. D. Torre, "Introduction to normative multiagent systems," *Computational and Mathematical Organization Theory*, vol. 12, pp. 71–79, 2006.

[3] G. H. von Wright, "Deontic logic," in *Mind*. Oxford University Press, 1951, vol. 60, no. 237, pp. 1–15.

[4] J. Y. Halpern, "Using reasoning about knowledge to analyze distributed systems," *Annual Review of Computer Science*, vol. 2, no. 1, pp. 37–68, 1987. [Online]. Available: http://www.annualreviews.org/doi/abs/10.1146/annurev.cs.02.060187.000345

[5] J. Hintikka, *Knowledge and Belief*. Ithaca: Cornell Univ., 1962.

[6] J. Y. Halpern, "Reasoning about knowledge: A survey," in *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford University Press, 1995, pp. 1–34.

[7] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi, *Reasoning about Knowledge*. The MIT Press, Cambridge Massachusetts, 1995.

[8] A. Lomuscio and M. Sergot, "On multi-agent systems specification via deontic logic," in *Proceedings of ATAL 2001*. Springer Verlag, 2001.

[9] T. C. Schelling, *Micromotives and Macrobehavior*. New York: Norton, 1978, see also a recent Atlantic article: Rauch, J. (2002). Seeing Around Corners; *The Atlantic Monthly*; April 2002;Volume 289, No. 4; 35-48. http://www.theatlantic.com/issues/2002/04/rauch.htm.

[10] U. Wilensky, "Netlogo," http://ccl.northwestern.edu/netlogo/, Evanston,IL, 1999.

[11] ——, "Netlogo segregation model," http://ccl.northwestern.edu/netlogo/models/Segregation, Evanston,IL, 1997.

[12] A. Lomuscio and M. Sergot, "Extending interpreting systems with some deontic concepts," in *Proceedings of TARK 2001*. Morgan Kauffman, 2001, pp. 207–218.

[13] F. Raimondi and A. Lomuscio, "Automatic verification of deontic interpreted systems by model checking via obdd's," 2004.

[14] B. Wozna, A. Lomuscio, W. Penczek, and W. Penczek, "Bounded model checking for deontic interpreted systems," in *Proc. of the 2nd Workshop on Logic and Communication in Multi-Agent Systems (LCMAS 04)*. Elsevier, 2004, pp. 93–114.

[15] G. Governatori, A. Lomuscio, and M. J. Sergot, "A tableux system for deontic interpreted systems," in *AI 2003: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, T. Gedeon and L. Fung, Eds., vol. 2903. Berlin, Springer, 2003, pp. 339–350.

[16] R. M. Barbosa, "Especificação formal de organizações de sistemas multiagentes." Ph.D. dissertation, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, RS, 2011.

[17] A. Lomuscio and M. Sergot, "Deontic interpreted systems," *Studia Logica*, vol. 75, no. 1, pp. 63–92, 2003. [Online]. Available: http://dx.doi.org/10.1023/A%3A1026176900459

[18] J.-J. Meyer, "Dynamic logic reasoning about actions and agents," in *LogicBased Artificial Intelligence*. Kluwer Academic Publishers, 2000, pp. 281–311.

Application of Workflow in Multi-Agent System Organization

José R. F. Neri, Jomi F. Hübner
Departamento de Automação e Sistemas (DAS)
Universidade Federal de Santa Catarina (UFSC)
Florianópolis, SC, Brasil
jrf.neri@gmail.com, jomi@das.ufsc.br

Carlos H. F. Santos
Grupo de Pesquisa em Robótica (GPR)
Universidade Estadual do Oeste do Paraná (UNIOESTE)
Foz do Iguaçu, PR, Brasil
cfh.santos@uol.com.br

Abstract— This paper presents a proposal for a model to integrate a workflow system into a multi-agent organizational model. Three alternatives are evaluated and the best alternative is integrated into the Moise organizational framework. An example of the chosen model is shown and a comparison based on workflow patterns is presented. The advantages of using this model are also discussed.

Keywords—workflow; agents; organization

I. INTRODUÇÃO

De modo geral, uma organização é composta por grupos de agentes que se relacionam entre si a fim de alcançarem objetivos comuns. Estrutura organizacional é frequentemente vista como um meio de gerenciar dinâmicas complexas em sociedades. Isto implica que abordagens para modelagens organizacionais devem incorporar ambos os aspectos estruturais e dinâmicos de tais sociedades [5], [6]. As características que tornam o estudo das organizações um desafio bastante pesquisado é que elas são sistemas complexos, dinâmicos e adaptativos que evoluem [9].

Em [2] os autores citam quatro dimensões que são utilizadas na maioria dos modelos organizacionais. Estas dimensões são: Estrutural: A dimensão estrutural está ligada à especificação de papéis, grupos e relacionamentos entre estes, que podem ou não ser definidos a partir de objetivos organizacionais; Dialógica: A modelagem dialógica caracteriza-se pela especificação de estruturas de interação direta entre papéis por troca de mensagens tendo em vista a realização de objetivos organizacionais, diálogos, cenas e protocolos; Funcional: A dimensão funcional caracteriza-se pela especificação e decomposição de metas e a relação entre essas metas; Normativa: Na dimensão normativa são definidas as normas que interrelacionam e regulamentam elementos funcionais, estruturais e dialógicos.

Desta forma, a especificação funcional de alguns dos modelos organizacionais como o Moise [7], [8], TEAM [3], STEAM [10], [11] e Opera [4], [5] é formada a partir da decomposição de um objetivo em uma estrutura de árvore, onde a raiz é a meta global e as folhas são as metas locais. Essa estrutura utilizada por esses modelos possuem algumas limitações, tais como:

- Não permite que haja um encadeamento condicional: utilizado para modelar uma escolha entre duas ou mais alternativas, por exemplo: if e then da lógica de programação;
- Não permite que haja um encadeamento iterativo: algumas vezes é necessário executar a mesma tarefa, ou o mesmo grupo de tarefas, múltiplas vezes até que uma dada condição seja alcançada;
- Não possui tratamento de exceção: onde podem ser previstas as situações excepcionais que acontecem durante a execução de uma tarefa.

Considerando-se essas limitações apresentadas por alguns dos modelos organizacionais existentes, o que se propõe neste trabalho é a utilização de um modelo conceitual para integração de um sistema de workflow com um modelo organizacional de sistemas multiagentes. Isto tem o objetivo de utilizar o workflow para controlar o fluxo das metas da dimensão funcional de uma organização, tratando as limitações apresentadas acima e adicionando outras propriedades importantes que sistemas de workflow podem oferecer.

Sistemas de workflow são uma tecnologia capaz de coordenar e sincronizar a maneira com que as atividades de uma organização são executadas para a realização de uma determinada tarefa. Workflow é definido pela WfMC (Workflow Management Coalition) como “a automação total ou parcial de um processo de negócio, durante a qual documentos, informações e tarefas são passadas entre os participantes do processo” [13].

Segundo a WfMC, um processo é "um conjunto coordenado de tarefas (sequenciais ou paralelas) que são interligadas com o objetivo de alcançar um meta comum", sendo tarefa conceituada como "uma descrição de um fragmento de trabalho que contribui para o cumprimento de um processo" [13].

Este documento está dividido em sete seções. Na segunda seção, comenta-se sobre o modelo organizacional utilizado neste trabalho, o Moise. Na terceira seção são apresentadas três propostas e realizado um comparativo entre elas. Na quarta seção, descreve-se o modelo conceitual escolhido em maiores detalhes. Em seguida na quinta seção é mostrado um exemplo completo da utilização do modelo escolhido. Na sexta seção,

são apresentados os resultados e na sétima seção, apresentam-se as conclusões e algumas ideias que podem servir como direções para pesquisas futuras.

II. Moise

Dentre os vários modelos organizacionais existentes, neste projeto iremos focar no modelo Moise [8], [9], que é um modelo organizacional para projeto de sistemas multiagentes baseado em noções como papéis, grupos e missões. Para o Moise, a organização é vista como um meio para reduzir a complexidade do problema a ser resolvido, através do esclarecimento e da divisão de tarefas entre os agentes e da definição de relações entre eles. Moise define uma especificação explícita da organização utilizada pelos agentes para raciocinar sobre a organização e também utilizada como uma plataforma que impõe aos agentes uma especificação a ser seguida.

No Moise, a organização possui três dimensões: a estrutura (papéis), o funcionamento (planos globais) e as normas (obrigações). No aspecto estrutural, o Moise define como os papéis estão relacionados a outros elementos da organização, inclusive a outros papéis.

A especificação funcional do Moise, foco deste trabalho, é formada a partir da especificação de esquemas sociais que são compostos de planos e missões que visam atingir uma meta global. Uma meta global representa um estado do mundo desejado pela organização, enquanto uma meta local é um objetivo de um único agente. Planos determinam a coordenação da realização das metas. Uma missão é um conjunto de metas locais que pode ser atribuído a um agente através de seus papéis, sendo que este agente é responsável pela satisfação de todas as metas da missão.

O aspecto normativo liga os aspectos funcionais e aspectos estruturais, indicando quais as responsabilidades dos agentes nos planos globais.

III. Proposta de Solução

Esta seção apresenta três propostas de soluções para integrar sistemas de workflow a um modelo organizacional. É feito um comparativo entre os modelos propostos e explicado com maiores detalhes o modelo conceitual escolhido. As três propostas apresentadas abaixo são: modelo híbrido, modelo workflow com missões e modelo workflow.

A. Modelo Híbrido

O modelo híbrido é composto por esquemas sociais, planos, metas, missões e um sistema de workflow. Essa proposta foi concebida com o intuito de utilizar tudo que existe atualmente no Moise, nada do Moise é excluído.

As modificações para esse modelo consistem na adição de um novo tipo de meta que não utiliza os planos fornecidos pelo esquema social do Moise, passando a utilizar o controle de fluxo fornecido por um sistema de worfklow. Metas com fluxo controlado pelo motor de workflow passam a se chamar tarefas, para ficar de acordo com os conceitos de workflows.

Para cada meta global do tipo workflow existe um motor de workflow para controlar o fluxo das tarefas executadas pelos agentes, funcionando de forma semelhante a um processo. A meta tipo workflow torna-se satisfeita quando todas as tarefas do processo estão satisfeitas e as tarefas do processo são alocadas de acordo com as missões dos agentes.

Por exemplo, na figura 1 as metas globais Destruir Inimigo e Defesa possuem um plano paralelo, a meta global Ataque, possui plano sequencial e podem ser cumpridas sem uma ordem pré-determinada, as metas Ataque e Defesa são usuais do Moise, entretanto a meta Captar Recurso é do tipo workflow, esse tipo de meta controla o fluxo das tarefas com um motor de workflow.

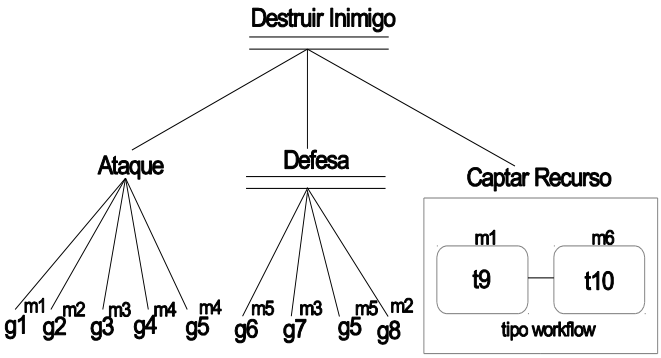

Figura 1: Modelo Híbrido.

B. Modelo Workflow com Missões

O modelo workflow com missões é composto por um sistema workflow e missões do Moise, os esquemas sociais são substituídos por processos. Os papéis dos agentes continuam possuindo obrigações com missões, o que significa que os agentes devem se comprometer com as tarefas do workflow que estão associadas às suas missões. As missões ainda permanecem com o intuito de não precisar alterar a dimensão normativa do Moise. Não existem mais planos, o fluxo das tarefas é controlado apenas pelo sistema de workflow.

Na figura 2 é possível observar que cada tarefa possui uma missão associada. O agente que se compromete com a missão m1 deve cumprir as tarefas task1 e task3, e o agente que se compromete com a missão m2 deve cumprir as tarefas task2 e task4.

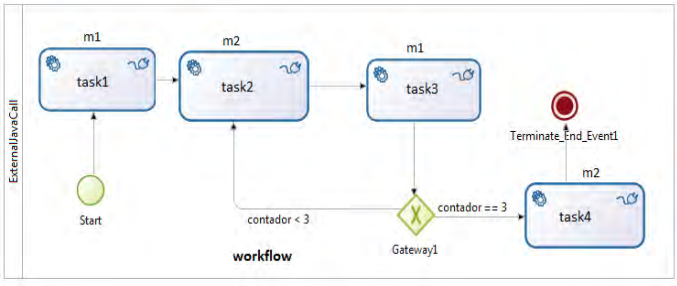

Figura 2: Modelo Workflow com Missões.

C. Modelo Workflow

O modelo workflow é muito semelhante ao modelo workflow com missões, a diferença está no fato do modelo de workflow não utilizar mais missões, um agente se compromete

com uma tarefa de acordo com o seu papel no grupo, não existe mais a necessidade de assumir uma missão para só então assumir as tarefas dessa missão. Nesse caso a dimensão normativa no Moise precisa ser alterada. Por exemplo, no lugar de determinar obrigações para com missões, irá determinar obrigações para tarefas.

Esse modelo foi concebido pensando em ser o mais parecido possível com a teoria de workflows. Uma pessoa que já trabalha com sistemas de workflows não precisa aprender novos conceitos para começar a utilizá-lo.

D. Comparativo entre os modelos

Foi realizado um comparativo entre as vantagens e desvantagens apresentadas pelos três modelos propostos (tabela 1). Os contextos escolhidos para essa avalição foram: usuários Moise, usuários workflow, aplicações antigas, desempenho, encadeamento iterativo e dimensão normativa. Cada item possui uma pontuação de acordo com o seu modelo, a pontuação vai de + a +++, sendo que + atende muito pouco ao item e +++ atende totalmente. A seguir são detalhados cada item do comparativo.

TABELA 1: QUADRO COMPARATIVO DAS VANTAGENS E DESVANTAGENS APRESENTADAS PELOS TRÊS MODELOS.

Contextos	**Híbrido**	***Workflow* c/ Missões**	***Workflow***
1 - Usuários Moise	++	+	+
2 - Usuários *Workflow*	+	++	+++
3 - Novos usuários	+	++	+++
4 - Desempenho	+	+++	+++
5 - Encadeamento iterativo	+	+	+++
6 - Dimensão normativa	+++	+++	+

1. Usuários Moise. Diz respeito a usuários antigos que já utilizam o Moise, o modelo híbrido ganhou a maior nota nesse item, pois possui a vantagem de utilizar tudo que existe atualmente na dimensão funcional do Moise. Um usuário habituado ao Moise, não precisa aprender todos os conceitos de workflow, podendo se adaptar aos poucos, pois esses conceitos somente serão utilizados quando a meta é do tipo workflow.
2. Usuários workflow. Usuários de um sistema workflow teriam maiores dificuldades para se adaptarem ao modelo híbrido, pois precisariam aprender conceitos de esquema social. O modelo workflow com missões ganhou uma nota inferior ao modelo workflow por utilizar missões, o que poderia confundir usuários de worflow.
3. Novos usuários. São usuários que nunca utilizaram o Moise e não possuem conhecimento sobre sistemas de workflow. O modelo híbrido leva desvantagem nesse quesito em relação aos demais, pois usuários levariam mais tempo aprendendo conceitos (tanto de workflows quanto do Moise) para utilizar todos os recursos do modelo hibrido.
4. Desempenho. Em relação ao desempenho o modelo híbrido ganhou a menor nota nesse quesito por possuir a necessidade de utilizar varias instâncias do motor de workflow, uma para cada meta do tipo workflow, causando problemas de desempenho e escalabilidade. Nos modelos workflow com missões e workflow é necessário apenas um motor de workflow para a dimensão funcional do Moise.
5. Encadeamento iterativo. O modelo híbrido e workflow com missões possuem dificuldades de implementação de um encadeamento iterativo, onde uma determinada meta ou um grupo de metas podem ser alcançados várias vezes. Isso ocorre por causa do uso de missões, pois quando um agente cumpre todas as metas de sua missão, ele deixa a missão, não se comprometendo mais com as metas dessa missão. Entretanto caso o agente precise cumprir novamente uma meta dessa missão, isso não será possível, pois ele já abandonou a missão. O modelo workflow resolve esse problema de implementação, onde as tarefas são assumidas diretamente pelos agentes e o agente nunca deixa uma tarefa.
6. Dimensão normativa. Como o modelo workflow não utiliza missões, a dimensão normativa precisa ser reimplementada no modelo. O modelo híbrido e workflow com missões utilizam missões, não necessitando modificações na dimensão normativa.

Foi escolhido o modelo workflow para utilização nesse trabalho, porque ele conseguiu oferecer maiores vantagens em relação aos outros modelos apresentados. Pela tabela 1 é possível observar que o modelo workflow conseguiu pontuação máxima em quatro dos seis itens, enquanto o modelo workflow e modelo híbrido conseguiram apenas duas e uma pontuação máxima respectivamente. Os quesitos considerados mais importantes e que definiram essa escolha foram: usuários novos e encadeamento iterativo. Encadeamento iterativo por facilitar a próxima etapa desse projeto que é o desenvolvimento da proposta, e usuários novos, pois como não existem muitas aplicações com o Moise atual, não existe necessidade de focar num modelo para usuários Moise.

IV. AGREGAÇÃO DO WORKFLOW AO MOISE

Esta seção descreve o modelo conceitual escolhido para incorporação de um sistema de workflow ao modelo organizacional Moise. O modelo em uma visão geral é dividido em três componentes: a organização, o ambiente e os agentes.

O componente organização engloba o motor de workflow e o estado organizacional. O estado organizacional é um elemento importante na integração de um sistema de workflow ao Moise, nele estão contidos a especificação organizacional, a entidade organizacional, os fatos, regras e normas que os agentes têm obrigação ou permissão de se comprometerem. O estado organizacional informa ao motor de workflow quando uma tarefa foi reativada e avisa das possíveis exceções geradas por uma tarefa, três tipos de exceções são possíveis: exceção de tempo, exceção de recursos e exceção de remoção. Uma exceção de tempo ocorre quando o tempo determinado para executar uma tarefa foi ultrapassado, uma exceção de recursos acontece quando não há recursos suficientes para a execução de uma tarefa e uma exceção de remoção ocorre quando uma tarefa é removida por um agente.

O ambiente segue a proposta de Agents & Artifacts [12], onde o ambiente é formado por vários artefatos, um artefato pode ser qualquer objeto do ambiente, como por exemplo, os recursos necessários para a execução de uma tarefa. O ambiente informa ao sistema de workflow quando uma tarefa foi executada por um agente e informa ao estado organizacional os recursos disponíveis atualmente no ambiente.

Os agentes atuam sobre o ambiente e sobre a organização, são responsáveis por cumprir as obrigações designadas pela organização, tais como: participar de um determinado grupo, assumir um papel na organização ou executar uma tarefa. Um agente pode receber e dar ordem para outros agentes, pode monitorar o fluxo das tarefas através de métricas de processo fornecidas pelo workflow ou pelo estado organizacional, pode pensar sobre processos, verificar gargalos, atrasos, delegar tarefas, criar novas tarefas e cancelar ou remover tarefas. Caso uma tarefa não possua nenhum papel associado na organização, ela pode ser executada por um agente externo ou por um humano.

A figura 3 apresenta o modelo conceitual de forma detalhada. O componente motor de workflow é constituído por processos e pelo estado das tarefas. Em um processo, existe um conjunto de tarefas interligadas de forma sequencial ou paralela, visando alcançar um objetivo comum da organização. O estado das tarefas armazena as exceções geradas pelo estado organizacional, receber as tarefas executadas pelos agentes e informar aos processos e ao estado organizacional as mudanças nos estados das tarefas.

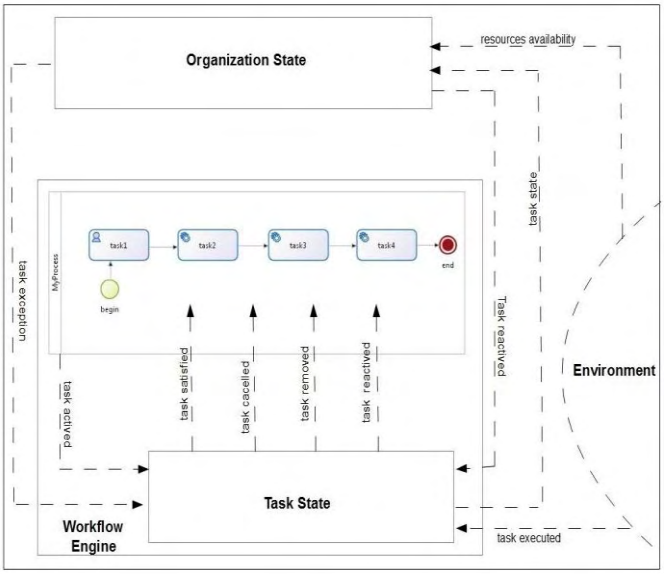

Figura 3: Modelo Conceitual.

Os estados possíveis de uma tarefa (figura 4) são: inativa, ativa, satisfeita, cancelada ou removida. Todas as tarefas de um processo iniciam-se no estado inativa, caso os pré-requisitos para tonar uma tarefa habilitada sejam atingidos, a tarefa se torna ativa e pronta para ser executada. O fluxo do processo é interrompido até que uma tarefa ativa seja cancelada ou satisfeita. Para uma tarefa torna-se satisfeita sua cardinalidade precisa ser alcançada, ou seja, número de vezes que a tarefa precisa ser executada para se tornar satisfeita.

Uma tarefa ativa é cancelada, caso ocorram uma exceção de recurso ou uma exceção de tempo, caso uma exceção de recurso aconteça, o fluxo do processo pode tomar um caminho alternativo, onde tarefas para captação de recursos serão ativadas, uma vez que os recursos necessários para a execução da tarefa foram captados, a tarefa que causou a exceção será reativada, tornando-se ativa novamente. Uma tarefa cancelada por exceção de tempo fica cancelada por prazo indeterminado até que um agente de monitoração decida reativar a tarefa.

Uma tarefa satisfeita pode ser ativada novamente, caso os seus pré-requisitos de habilitação sejam atingidos. Se uma exceção de remoção acontecer, a tarefa é excluída do processo e ao contrário de uma tarefa cancelada uma tarefa removida não pode mais se tornar ativa. Quando o processo termina todas as tarefas são finalizadas.

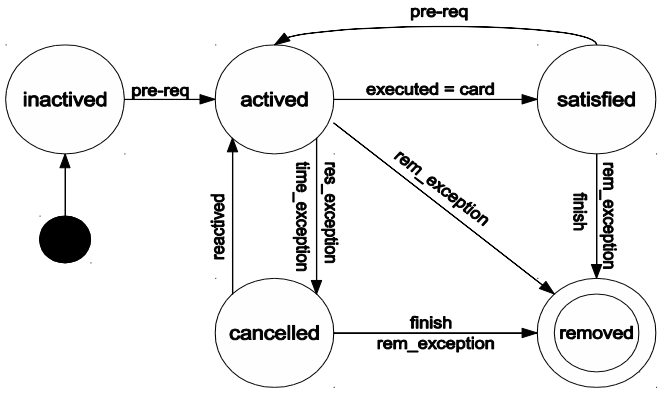

Figura 4: Estados possíveis de uma tarefa.

Através das obrigações geradas pelo estado organizacional é que um agente fica ciente do momento que deve executar suas tarefas. Uma vez que a tarefa é executada a obrigação do agente é cumprida. O modelo proposto não utiliza planos, o fluxo de tarefas é controlado por um motor de workflow, permitindo trabalhar com fluxos complexos, não possível na versão do Moise sem a integração.

V. Estudo de Caso

Esta seção apresenta um exemplo completo da utilização do modelo escolhido e mostra um comparativo entre a dimensão funcional com e sem agregação de um sistema de workflow. Conforme pode ser visto na figura 5, na especificação estrutural foi definido o grupo laticínio composto pelos papéis fazendeiro, entregador e operário. Os papéis fazendeiro e operário possuem cardinalidade de 1 a 5, ou seja, no mínimo 1 e no máximo 5 agentes podem assumir esses papéis. O papel entregar possui cardinalidade de 1 a 3 e fazendeiro e o papel operário possui autoridade sobre o papel entregador.

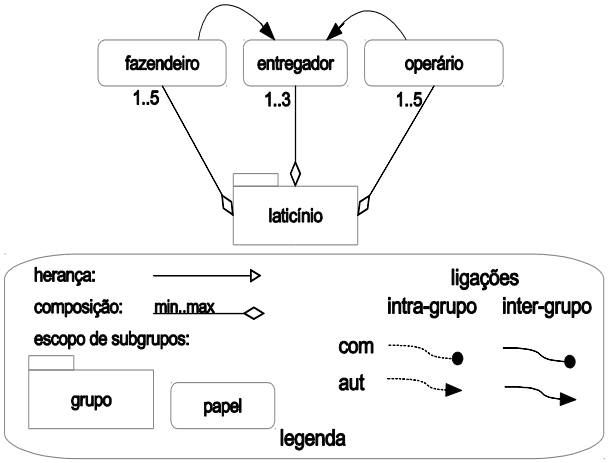

Figura 5: Dimensão Estrtural.

A figura 6 apresenta um processo definindo a especificação funcional de fabricação de queijo. Um laticínio precisa entregar uma encomenda de 1000 peças de queijo para um supermercado. Para cumprir esse objetivo são utilizados três papéis como descrito na especificação estrutural: o papel fazendeiro, o papel entregador e o papel operário.

As tarefas são assumidas pelos agentes no momento em que estes assumem um papel no grupo laticínio. Conforme especificado na tabela 2, o agente que assume o papel de fazendeiro é responsável por tirar o leite das vacas, o entregador é encarregado de levar o leite ao laticínio e entregar a encomenda de 1000 peças de queijos ao supermercado e o agente operário é encarregado de fabricar o queijo.

O fluxo do processo inicia-se com a tarefa ordenhar leite da vaca, em seguida o leite é entregue ao laticínio para que produção do queijo possa ser iniciada. O queijo é produzido até o leite acabar ou até que a meta de 1000 peças de queijos seja alcançada. Caso o leite termine o fluxo do processo é reiniciado, voltando para a tarefa de tirar o leite da vaca.

Esse fluxo de processo não seria possível na dimensão funcional do Moise sem integração do workflow, pois os planos do Moise não suportam encadeamento condicional e nem o encadeamento iterativo. Sem encadeamento iterativo, uma meta satisfeita não ficaria mais ativa para ser satisfeita novamente. Sem encadeamento condicional, o fluxo do processo não conseguiria decidir qual caminho seguir depois da tarefa fazer queijo se tornar satisfeita, ele ativaria as tarefas entregar encomenda, tirar leite da vaca e fazer queijo.

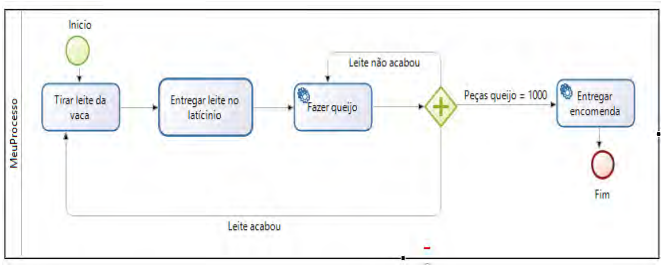

Figura 6: Dimensão Funcional.

A relação entre a especificação estrutural e a especificação funcional é feita pela especificação normativa. Na especificação normativa são descritas as tarefas com as quais um papel tem permissão ou obrigação de se comprometer.

TABELA 2 : DIMENSÃO NORMATIVA.

PAPEL	RELAÇÃO DEÔNTICA	TAREFA
FAZENDEIRO	OBRIGAÇÃO	TIRAR LEITE DA VACA
ENTREGADOR	OBRIGAÇÃO	ENTREGAR LEITE NO LATICINIO, ENTREGADOR ENCOMENDA
OPERÁRIO	OBRIGAÇÃO	FAZER O QUEIJO

VI. RESULTADOS E DISCUSSÕES

A. *Comparativo entre os modelos*

Padrões de workflow têm atraído a atenção de pesquisadores e da indústria de software devido as suas potenciais vantagens. Em [1] são descritos 21 padrões de workflow para controle de fluxo (e.g., sequencial, paralelo, condicional). Tais padrões são úteis tanto para a definição de workflows, como para validar o poder de expressão das linguagens e ferramentas de workflow. Os padrões descrevem diversas maneiras, através das quais, dados podem ser representados em definições de workflow.

A tabela a seguir sumariza os resultados da pesquisa em termos de suporte aos padrões de workflow propostos. Para cada combinação, é indicado se o modelo suporta ou não o padrão. Como convenção, atribui-se "suportado" para padrões suportados direta e/ou indiretamente pela ferramenta, e "não suportado" para padrões não suportados pela ferramenta.

TABELA 3 : PADRÕES DE FLUXOS.

Padrão	Moise	Moise + *Workflow*
1 *(Sequence)*	Suportado	Suportado
2 (*Parallel Split*)	Suportado	Suportado
3 *Synchronization*	Suportado	Suportado
4 (*Exclusive Choice*	Suportado	Suportado
5 *Simple Merge*	Suportado	Suportado
6 *Multi-choice*	Suportado	Suportado
7 *Synchronizing Merge*	Não suportado	Suportado
8 *Multi-merge*	Não suportado	Não suportado
9 *Discriminator*	Não suportado	Não suportado
10 *Arbitrary Cycles*	Não suportado	Suportado
11 *Implicit Termination*	Não suportado	Não suportado
12 *MI Without Synchronization*	Não suportado	Suportado
13 *MI With a Priori Design Time Knowledge*	Não suportado	Suportado

Pela tabela 3 é possível notar que o Moise com a agregação do workflow suporta 10 dos 13 padrões apresentados. Enquanto que o Moise sem a integração suporta apenas 6 dos 13 padrões. Com suporte ao padrão Arbitrary Cycles uma ou mais tarefas podem repetir ciclicamente, permitindo 'loop'. Isso demonstra vantagem de agregar workflow ao Moise, pois é possível resolver o problema de encadeamento iterativo e condicional.

B. Vantagens Encontradas

Abaixo são apresentadas algumas vantagens encontradas na utilização de workflow em organização de sistemas multiagentes:

Padrões de Fluxos Prontos: Na dimensão funcional do Moise sem agregação do workflow, a modelagem dos fluxos é difícil, pouco intuitiva e não há padrões de fluxos prontos a serem seguidos como em sistemas de workflow. Algumas vantagens de utilizar padrões de fluxos prontos são:

- Eles foram provados. Os padrões refletem a experiência, conhecimento e soluções dos desenvolvedores que tiveram sucesso usando esses padrões em seus trabalhos;
- São reusáveis. Os padrões provêm uma solução pronta que pode ser aplicada a diferentes problemas;
- São expressíveis. Os padrões provêm um vocabulário comum de soluções que podem expressar muitas soluções, sucintamente.

Interface gráfica intuitiva: Ao contrário do Moise que não utiliza uma ferramenta para modelagem dos fluxos de metas da organização, a maioria das ferramentas de sistemas de workflow possuem um editor gráfico para modelagem dos fluxos de tarefas no padrão BPMN (Business Process Modeling Notation), trata-se de uma notação padrão para o desenho de fluxogramas em processos de negócios.

Agentes podem raciocinar sobre workflow: Utilizando um componente BAM (Business Activity Monitoring) um agente monitora o workflow, verifica gargalos, atrasos, pensa sobre o processo e cria novos processos em tempo de execução. Caso um agente se recuse a executar uma tarefa ele pode delegar a tarefa para outro agente ou mudar o papel do agente, realizando assim o refinamento e ajuste de processos.

Utilização de um Motor de regras: Um componente BRM (Business Rules Management) é uma tecnologia que permite aos usuários finais definirem regras de negócio de forma declarativa. Regras de negócio ficam em um repositório separado, e podem ser consultados pelos agentes em tempo de execução. Isso facilita a institucionalização das regras de negócio na organização, a transparência sobre as regras existentes e redução do esforço de manutenção de sistema.

Agentes podem utilizar dos conectores do workflow: Agentes podem utilizar de um banco de dados, acessar webservices, enviar e-mail, entre outras possibilidades que conectores do workflow oferecem.

Tarefas podem ser executadas por humanos ou por agentes: Alguns fluxos de tarefas podem ser totalmente automatizados, sendo todas as tarefas executadas por agentes. Em outros fluxos de tarefas, algumas são executadas por humanos e outras por agentes, aumentando a interação entre esses dois.

VII. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresenta uma proposta de agregação de um sistema de workflow ao modelo organizacional Moise. São avaliadas três alternativas para o modelo de integração e a alternativa escolhida foi apresentada com maiores detalhes.
Com o estu- do de caso, foi ilustrado o uso do encadeamento condicional e iterativo na dimensão funcional do Moise. Isso foi possível graças a utilização de um motor de workflow no controle do fluxo das tarefas.

Os próximos passos do trabalho são a avaliação do modelo através da definição e desenvolvimento da arquitetura de implementação e a comparação com as outras propostas de aplicações de sistemas multiagentes que utilizam sistemas de workflow. Na comparação serão apresentados os componentes atendidos e não atendidos pelas propostas.

REFERENCIAS

[1] Aalst, W.M.P. van der; Hee, K. van. (2002) "Workflow Management": models, methods, and systems. London: The MIT Press.

[2] Coutinho, L. R., Sichman, J. S., Boissier, O. (2006). Organizational Modeling Dimensions in Multiagent Systems. In: IBERAGENTS, Ribeirao Preto, SP, Brasil. Anais.

[3] Decker, K.; Lesser, V. Task environment centered design of organization. In: AAI Spring Symposium on Computational Organization Design. Menlo Park: AAAI, 1994.

[4] Dignum, V. A model for organization interaction: based on agents, founded in logic. Tese (doutorado) – Utrecht University, 2004. SIKS Dissertation Series No. 2004-1.

[5] Dignum, V.; Meyer, J.; Weigand, H,; Dignum, F. An organizational-oriented model for agent societies. In: (RASTA '02), at AAMAS, Bologna, Italy, 16 July 2002.

[6] Ferber, J.; Michel, F.; Báez-Barranco, J. Agre: Integrating environments with organizations. In: Environments for Multi-Agent Systems. Berlin, Heidelberg: Springer-Verlag, 2005. (Lecture Notes in Computer Science, v. 3374), p. 48-56.

[7] Hannoun, M., Boissier, Oliver ans Sichman, J. S., e Sayettat, C. (1999). Moise: Un modèle organisationnel pour la conception de systèmes multi-agents. Em Acts des tèmes Journèes Francophones Intelligence Artificielle Distribuée & Systèmes Multi-Agents. Hermès Science Publications.

[8] Hübner, J. F. (2003). Um Modelo de Reorganização de Sistemas Multiagentes. Tese de Doutorado, Escola Politécnica da Universidade de São Paulo.

[9] Okuyama, F. Y. (2008). Modelo MAS-SOC: Integrando Ambientes e Organizações para Simulações Baseadas em Sistemas Multiagentes Situados, Tese. UFRGS-RS.

[10] Tambe, M. Towards flexible teamwork. Journal of Artificial Intelligence Research, v. 7, p, 83-124, 1997.

[11] Tambe, M,; Adibi, J,; Al-Onaizan, Y,; Erdem, A,; Kaminka, G. A.; Marsella, S. C.; Muslea, I. Building agente teams using a explicit teamwork model and learning. Artificial Intelligence, v. 110, p. 215-239, 199.

[12] Omicini, A.; Ricci, A.; Viroli, M. Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers, Hingham, MA, USA, v. 17, p. 432-456, 2008.

[13] WfMC (2008). "Workflow Management Coalition Terminology & Glossary", Document Number WFMC-TC-1011, Document Status - Issue 3.0.

A Normative and Self-Organizing Piloting Model for Virtual Network Management

Carolina Valadares, Manoel T. Abreu Netto, and Carlos J. P. de Lucena
Department of Informatics
Pontifícia Universidade Católica do Rio de Janeiro
Rio de Janeiro, Brazil
{cvaladares, mnetto, lucena}@inf.puc-rio.br

***Abstract*—The Internet has become an essential role in the society, serving every day billions of users spread all over the world. It is a complex network that holds an extensive range of services, applications and technologies. Its model, however, makes it difficult to solve structural problems such as management and maintaining. Network virtualization has been proposed to tackle this issue. In this paper, we use the concept of multi-agent system, norms and self-* properties to propose and validate an autonomic self-organizing model for virtual network management. As our proof-of-concept, we show that our system, which is composed of a virtual network of virtual machines capable of self-organizing themselves in a totally decentralized way across a physical infrastructure in order to cope with environment changes, satisfies its main goal of efficiently re-organize itself with no central control.**

Keywords—multi-agent; selg-organazing; virtual network management; norms; autonomic network;

I. INTRODUCTION

The Internet is a complex network that servers billions of users spread all over the world. It caries an extensive range of services, technologies, applications and has also enabled a variety of forms of human interactions and information exchange. Even through its architecture facilitates the deployment of new applications, due to its transparency, its model makes it difficult to solve structural problems such as scalability, management, mobility and security [3]. The Internet is a large-scale network and a trivial approach for its management, which involves human being interference, becomes costly and flawed as its size increases.

Autonomic network virtualization has been pushed forward by its proponents to tackle the Internet ossification problem. It represents a new approach that has recently received substantial attention from academia, whereas it is able to run multiple virtual networks simultaneously on the top of a single physical substrate. We intend to deal with the complexity aggregated to the new concept of virtual network by enabling the self-management behavior. This self-* capability represents a specific area of autonomic computing [11], a term coined by IBM, to deal with such complexity by enabling systems to self-manage themselves. The main key behind autonomic network visualization is, therefore, the building of flexible networks capable of managing themselves in order to deal with external changes and interferences from the environment.

Virtual networks can support simultaneous independents network experiments, services and architectures over a shared substrate network [7][9]. Each virtual network is capable of running its own protocols, routing process, services and management solutions, in a way of totally isolation and independency, although they share the same infrastructure. It is composed of a set of physical and virtual resources, as depicted in Figure 1, in which physical resources (substrate node) consist of devices such as router, access points, and are able to embed many virtual nodes. These virtual nodes are connected together by virtual links, which is also embedded on physical resources. Both virtual node and virtual link belong to a dedicated virtual network, that supports a specific service or protocol [5], in which every substrate and virtual node has a self-organizing piloting agent embedded, responsible for handling local decisions and actions, which characterizes it as a decentralized model.

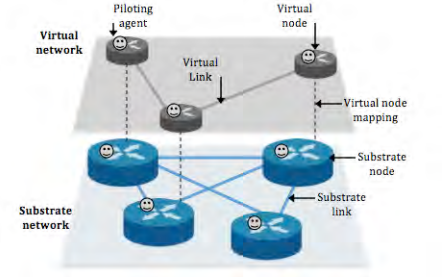

Figure 1 Virtual Network Model

The piloting agent itself is what leads the virtual network to emerge a self-organizing and is in charge of handling local behaviors to enable a proper control and management of the virtual network, its components, and the network flows, in order to maintain the efficient use of substrate resources on network virtualization. It represents the adaptive method running inside of each substrate and virtual node, which is responsible for adapting and managing the network resources in order to meet quality policies and users requirements in case of environment changes. Moreover, it is composed of high-level norms and a self-organizing control loop to retrieve

local knowledge to support the decision making on whether to self-organize the substrate network to cope with changes on either traffic loads or resources availability.

As our proof-of-concept, we implemented and validated a piloting system for virtual network management, in which the piloting system acts upon a network failure by either creating a new virtual router capable of handling the traffic demand or migrating an existent virtual router to a distinct host. The main goal of our research is to offer a scalable and robust way to evaluate the effectiveness of our piloting system, also its ability to self-configure its virtual resources on specific scenarios.

The remainder of this paper is structured as follows. Section 2 summarizes the related work on adaptive provisioning. Section 3 brings an overview of the concepts applied in this work. In section 4, we describe the self-organizing piloting model itself, under the multi-agent system (MAS) perspective. We evaluate testbed setup and experimental results in section 5. Finally, section 6 concludes this paper and presents on going and future works.

II. Related Work

The problem of virtual network management can be divided into two main sub-problems. Firstly, there is the Virtual Network Mapping problem [1], which tackles the problem of mapping virtual resources in the physical infrastructure, concerning about the efficient resource mapping while dealing with the simultaneous optimization of the placement of virtual nodes and links on a substrate network. Secondly, assuming that the virtual network has been provisioned, the adaptive maintenance itself comes into play in order to deal with dynamic changes from the variations in the substrates and virtual networks, also related to failures, mobility, migration and maintenance needs. The idea behind the adaptive provisioning is to maintain the original topology and service levels agreements during the virtual network lifetime. The virtual network provisioning involves virtual routers and links management, such as live migration, and virtual router allocation.

Although there are in the literature substantial amount of work dealing with Virtual Network Mapping, from the Network perspective, to the best of our knowledge, there are few studies on adaptive provisioning of instantiated virtual networks to cope with dynamic changes in service demands and resource availability, mainly from the MAS perspective.

We have not dealt with Virtual Network Mapping problem yet, as this paper cover mainly the adaptive provisioning, in which it maintains the virtual network running as efficiently as possible during the virtual network lifetime. In order to solve the virtual network provisioning problem, many approaches have been suggested dealing mostly with (i) virtual node live migration to a distinct host and (ii) virtual link reassignment and setup to preserve the virtual network topology.

For instance, the authors of [13] proposed an autonomic system called Violin, which manages a virtual environment, composed of virtual nodes capable of live migration across a multi-domain physical infrastructure. Moreover, in [14], the authors proposed an adaptive virtual resource provisioning, which brings substrate node agents to cope with failures and severe performance degradation in network virtualization. Furthermore, [15] proposes a distributed self-organizing model to manage the substrate network resources. There also exists approaches dealing with virtual link reassignment, as the system proposed in [16], in which it changes the mapping of virtual links if the load of specific physical links increases more than a certain threshold.

We note that these approaches have treated the virtual network management from a semi-decentralized way, in which the autonomic entities are spread only over substrate nodes. Differently from those highlighted research, the self-organizing model proposed in this paper addresses the management of substrate and virtual resources by taking advantage of the total distribution of the autonomic entities spread all over the network, including virtual networks rather than only substrate nodes.

III. Concepts

Recent research has pointed out network virtualization as a promising technique to deploy future networks that meet current and future users requirements [7][3][10]. The main idea behind network virtualization is of slicing (sharing) physical resources to create multiple virtual networks capable of running its own protocols, services and management solutions. Hence, its main concept relies on the fact that it adjusts the network flow and routes, in an autonomic way, dismissing any kind of central/external control. It aims to maintain the quality of service (QoS) defined in the SLA by controlling the agent's behaviors through norms. Each substrate and virtual node has a piloting agent embedded, which is responsible for capturing local information, reasoning about the collected data by translating simple measurements into significant knowledge, exchanging the acquired knowledge and supporting the instantiation and management of virtual resources. Therefore, it is in charge of managing virtual resources already existent by replacing or migrating overloaded virtual routers, or creating and instantiating new ones.

The main characteristics of the architecture for virtualization, in which a virtual network represents a collection of virtual routers connected together by a set of virtual links to form a virtual topology, which is essentially a subset mapped on the top of the underlying physical network, is regarding to (i) virtual node, and (ii) substrate node. Virtual nodes are hosted on a particular substrate node, in other words, it is a slice of its physical host, comprising CPU, memory RAM, storage capacity, etc. The substrate node, usually composed of physical resources, the resource manager, virtual nodes and virtual links, consists of devices such as router, access points or physical links, and are able to embed many virtual nodes. Further details regarding to virtual

network architecture can be found in projects like 4AWARD [4].

A. *Network Virtualization Management*

Network virtualization management, therefore, involves operations such as instantiating, deleting, monitoring, migrating virtual networks elements and setting its resource-allocation parameters. Such functionalities are what make our piloting system a suitable model for creating and managing multiple virtual networks and, as a consequence, for supporting the pluralist approach for the Future Internet, since it is able to create multiple customized virtual networks at the same time it exhibits a flexible management and a real-time control [17]. An important challenge on network virtualization is the efficient allocation of the physical resources at virtual network mapping and adaptive provisioning stage. To accomplish such efficient use the management of the physical resources should be frequently executed at runtime in order to deal with the variation on the load requests of different users.

B. *Multi-Agent System*

Recent research has pointed out that providing a distributed self-organized approach for the management of virtual networks is a viable solution to deal with the increase of complexity that network virtualization has brought. We strongly believe that such complexity could be handled by autonomic computing together with the concept of Multi-Agent System (MAS), Norms and Self-* properties.

We propose a virtual network architecture applying the MAS paradigm as a modeling foundation. We have chosen such paradigm mostly because it seems to be particularly suitable to build automatic system, due to some properties of agents, such as autonomy, proactivity, adaptability, cooperating, and mobility. Moreover, the notions of agents and organizations and their decentralized and pro-active nature match well the requirements of large-scale autonomic computing environments. In the other hand, Self-* brings to the piloting model the ability to self-manage its own resources in order to meet polices and user's requirements.

Accordingly, this paper provides the design and evaluation of a distributed, autonomic and self-organizing system based on MAS and Self-Organizing approaches [13] to ensure distributed negotiation and synchronization between the substrate nodes and virtual resources, so that the virtual and physical nodes are able to handle autonomous and intelligent agents, which exchange messages and cooperate to each other to carry out the distributed virtual network management. We apply such concepts to enable communications between the substrate and virtual agents to gain performance and scalability results of the distributed and autonomic virtual network manager, in order to tackle the virtual network adaptive provisioning challenge.

IV. SELF-ORGANIZING MODEL

Our normative and self-organizing piloting system is based on a distributed algorithm, which embeds an autonomic agent inside every virtual and physical node, disseminated all over the substrate and virtual network. The agents monitor, capture and reason about local information, communicate with each other, cooperating, in order to exchange their local knowledge and decisions feedback, so that each piloting agent turns into an autonomic entity capable of inferring about the global network state and, as a consequence, supporting the core of the self-organizing model to trigger adaptation plans depending on the local knowledge, global inferring and environment condition.

The control loop consists of four main behaviors: collector, analyzer, decision maker and executor, which are executed every so often. Firstly, relevant data from the measurement of availability of resources and network load is acquired by the behavior known as Collector, which is also responsible for storing this local information. Secondly, the Analyzer behavior comes into play in order to translate the measured information and the exchanged knowledge into local knowledge, it also checks if the translated data is in accordance with the quality of service and policies requirements by verifying whether a re-organization of virtual components is required. Afterwards, the Decision Maker thought the knowledge analyzed, might active a self-organization by running an adaptive plan. Such reorganization is activated by the identification of both network overload or lack of resources inside the substrate and virtual node.

The Decision Maker represents the core of our piloting system, since it is responsible for having the virtual network as stable as possible, avoiding any kind of bottleneck, overloaded link, and keeping high levels of quality of Service. It is in charge of translating the analyzed data into an action that might prevent future critical scenarios. The adaptive plans, re-organize the virtual network resources for a more efficient use of them. Such re-organization might be triggered by the detection of an (i) overloaded substrate node, which triggers the replacement of virtual node behavior, responsible for replacing a virtual router by a new one capable of handling the actual demand, and (ii) the identification of lack of physical resources in the virtual node, which causes, in this case, an increasing of the virtual node capacity.

The following components give us a better description of the self-organizing architecture itself:

A. *Self-organizing behaviors and the cognition loop*

We have designed supporting behaviors, which encompass the ones responsible for the communication task, knowledge sharing, event trigger, and environment sensors and adaptive plans behaviors, which represent the actions taken by the decision-maker. The later has used ontologies for a proper understanding while exchanging actions requests between agents. Such behaviors are divided in three distinct categories: Adaptive plans, Environment sensors and Control loop.

1) Adaptive Plans.

Create Virtual Router: Creating a virtual node can impact the virtual network in two distinct ways. In the first

case, if an existent virtual node is running multiple flows from different users request, its piloting agent might trigger an adaptive plan that will tackle the instantiation of a new virtual router to balance the requested flows. From the instantiation on, the flows get balanced between those two virtual nodes and the new virtual router starts to respond some of the requested flows. The second case occurs when it supports the Replace virtual node adaptive plan, explained bellow.

Replace Virtual Router: The replace virtual node plan is triggered in a specific scenario where a virtual router suffers from anomalies and failures such as lack of resource, link overload or when it gets unresponsive. Together with the Create Virtual Router plan, they create a new virtual router, capable of handling the current demand and users requests, and the new virtual router takes place of the failed virtual router. All services and flows must be kept running inside the new virtual router.

Migrate Virtual Router: Migrating virtual nodes across distinct physical hosts is an important functionality of our virtual network manager through the piloting system: It facilitates fault management and load balancing, since we can migrate virtual nodes aiming a better distribution of network load usage. Whether the piloting agent detects a future critical scenario regarding to physical resources, it might trigger the migrate adaptive plan, which is responsible for migrating a running virtual node to a different substrate node, maintaining the same virtual topology and running process.

2) Environment Sensors.

Monitor: Monitors are behaviors coupled to the Collector behavior to handle the different types of data collection; they are responsible for measuring specific information from links and physical resources, which will be later filtered in the Collector actions. The Monitors are composed of (i) devices monitors, which is in charge of monitoring Ethernet and virtual devices, (ii) routers monitors, that monitors the router tables running inside a virtual node and (iii) resources monitor, which will monitor physical and virtual resources such as memory RAM, CPU, IO load, etc.

Informers: Such behaviors are responsible for the communication between autonomic entities- virtual and physical. It sends request of data update, when a virtual node has out dated information related to its neighbors, it caries out knowledge sharing and inform the running events, such as the execution of an adaptive plan.

3) Control Loop

The core of our self-organizing model is composed of an autonomic control loop, in which four behaviors run frequently. Such behaviors come in to action as a machine state, where there exists distinct transitions between the behaviors depending on the state of the piloting agent. The components of the main control loop are:

Collector: The collector is responsible for obtaining information, supervising, monitoring and storing necessary measurement from network links, physical and virtual resources that are of significance to the self-properties of the underlying network. It captures data from both substrate and virtual nodes and also from the neighborhood.

Analyzer: The acquired data, from the Collector, is translated into knowledge by the Analyzer behavior that checks whether they are in accordance with the quality of services and required policies. Accordingly, it verifies the current performance, predicts future critical scenarios and detects events, such as link overload. It is also in charge of activating the decision-making, in case of adaptation need.

Decision Maker: The core of our self-organizing piloting model makes decision according to the knowledge retrieved by the Collector and Analyzer, also from the knowledge exchanged between neighbors nodes. Such decisions depend essentially on the virtual network state, the local knowledge and the norms undertaken. The decision itself is based on the choice of adaptation plans previously designed, such as (i) activating the creation or the delete of a virtual node, (ii) tuning the amount of virtual resources allocated to a specific virtual node, and (iii) migrating a virtual node to a different physical host.

Executor: The executor actually performs the decision previously made by reconfiguring the managed component and communicating with other autonomic managers.

B. Norms

In order to provide a controlled autonomy to the virtual devices, restricting its behaviors to prevent malfunctions and undesirable behaviors, we apply the concept of normative agents. Thus, the proposed model is responsible for adjusting the network flow and routes by controlling the agents' behaviors through norms. Such norms are what makes the piloting model aware of the required polices, quality of services and user's requirements (SLA and QoS).

The normative regulation system is divided into two groups, virtual and physical, in which the piloting agents might (i) abide by the norm, and (ii) violate the norm, it also can restrict access to the network to those agents that violate such norms by delaying their control-loop execution. Such norms are checked every so often through the piloting control loop and it informs the agent's neighbors all norms that have been complied or violated.

C. Self-Organizing Piloting Communication

In order to validate the multi-agent self-organizing piloting system based on a distributed algorithm, which consists of autonomic entities spread all over the virtual network, we first need to evaluate the best mean of communication between the piloting system components. Such evaluation represents an important sub-task of our work, since it supports describing a proper self-organizing model for the context of autonomic virtual network management.

The use of multi-agent communication to represent the piloting system model is essential since our piloting system makes use of this autonomic communication between agents

of different substrate nodes to gain advantages over traditional approaches to manage virtual networks. In developing our piloting model and implementing it we have had to address two key issues regarding to the agent communication:

1) *Where do we host the virtual piloting agents?*
2) *How to ensure the reliability in such distribution?*

It has been necessary to simulate different scenarios for different methodologies bearing in mind mainly the efficiency of the piloting prototype. The major advantage the piloting system can offer through the communication approach itself lies in the fact that through the communication between piloting agents from different virtual and physical machines we can ensure that the virtual nodes of the network are in accordance with the rules and policies of the model.

The first issue we addressed by simulating different scenarios where we host the piloting agent under the substrate node and under the virtual node, in which we could be able to automatically reproduce adaptive plans for a given adaptive scenario. The second issue we addressed by defining a suitable set of norms and adaptive plans that incorporates both anomalies from the surrounding environment, and failures of distributed communication cases. By avoiding having to develop extra requirement norms for the case in which the piloting agent is hosted under the virtual node, we reduce the amount of norms, and are able to ensure that the self-organizing model and its adaptive plans are more reliable.

V. PROOF OF CONCEPT AND INITIAL EXPERIMENTS

The initial experiments correspond to scenarios where virtual and physical resources, allocated to the virtual network, suffer from anomalies such as substrate/virtual node overload. The piloting system maintains the virtual network topology by selecting new virtual or physical resource to replace or compensate for the affected resource. Two resource failure scenarios are discussed in this paper: (i) virtual node overload, and (ii) substrate node overload. We aim, through the proof of concept, show that MAS and self-* capabilities are feasible approaches to deal with virtual network complexity. Thus, we present adaptive plans to deal with both critical scenarios in order to measure the effectiveness of the proposed piloting system and its ability to self-manage its virtual resources under critical scenarios, by dynamically binding and allocating new resources to maintain the virtual network.

Virtual node overload: When the piloting agent detects that its supported virtual node is about to get overload, it must either request its substrate node it belongs to allocate, at runtime, resources for the virtual node or, if not possible, re-instantiate a new virtual node in the same substrate node to take place of the failed node. The virtual links associated with the affected virtual node should also be reallocated if necessary.

Substrate node overload: When a substrate node, that hosts multiple virtual nodes, fails, gets overloaded or gets unresponsive, all agents hosted on its virtual nodes can detect such failure through keep-alive messages exchanged periodically. Only substrate node agents that belong to the same neighborhood are allowed to collaborate in order to choose alternative hosts where the affected virtual nodes as well as their associated links will be migrated or allocated. Thus, the distributed adaptive migrate router plan is executed for each virtual node hosted inside the substrate node in which the failure was detected.

A. Experimental Setup

We carried out preliminary experiments in which the virtual network topology and the initial mapping of the virtual network allocated on the top of the computers A, B and C, are arranged as depicted in Figure 2. The virtual network itself contains two virtual nodes, Va and Vb, hosted inside the substrate nodes A and B respectively. Two sets of flows are running inside the virtual network. Although the simplicity of this setup, it has enabled us to evaluate two different scenarios:

Case 1: For the first scenario we set up a virtual network containing two virtual nodes, Va and Vb, with 256MB of memory RAM and a limited network bandwidth of 5MB/s each. The virtual network runs a data flow associated to the user's request, in which the packets are transmitted over the virtual link starting at the virtual node Va and arriving at the node Vb. The experiment itself consist of generating a large amount of data flow and forwarding them to the virtual node Va, hosted inside the substrate node A, in order to force a network performance degradation. In the meantime, when the virtual node Va is about to dismiss the QoS, due to the overload caused by the traffic generator, its supporting piloting agent detects a possible future failure and acts upon it by creating a new virtual node, capable of handling the current data flow as it has larger network bandwidth, and replacing the affected router with the new one, maintaining the same network configuration and data flow settings. In matter of a few seconds, after a small interference (~3 seconds), the virtual network and the virtual link get stable again, obeying the required norms and, as consequence, the user's request no longer gets fuzzy. This experiment, despite being simple, simulates a scenario in which agents located inside virtual nodes are able to detect high utilization either of the virtual links or virtual resources and decide to update or replace the affected virtual node.

Case 2: The second scenario differs from the first one in that it handles live virtual node migration instead or node replacement. Similar to the previous experiment, we set up a virtual network, maintaining the same topology and capacity, also responding to a user's request. Unlike the first scenario, the purpose of this one is to generate a large amount of data flow and forward the generated packets to the substrate node A, in order to overload the substrate node A instead of the virtual node. After a short while, when the substrate node is

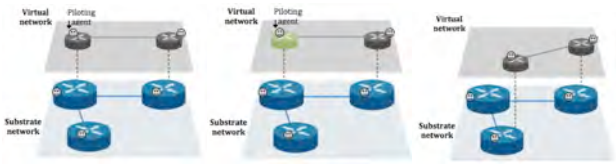

Figure 2. a) The initial Virtual network setup; b) Case1 result; c) Case 2 result.

about to get overloaded due to the traffic generator and, as consequence, next to dismiss the QoS, its supporting piloting agent, in accordance with the required norms, triggers the adaptation plan responsible for the virtual node live migration. The piloting agent responsible for supporting the affected physical machine is then in charge of applying the live migration algorithm on all virtual nodes the affected node hosts. The algorithm itself considers only physical routers from the neighbourhood to support the decision on where to migrate, and the closest one with enough resource availability is the one chosen as destination. In matter of a few seconds, the virtual node is then hosted inside a different substrate node from the neighbourhood, in this case, the substrate node C. This experiment simulates a scenario where agents located inside substrate nodes detect a high utilization of the physical resources and decide to migrate the virtual nodes from affected physical routers.

VI. Conclusion and Future Work

We analyzed the impact and the effectiveness of the self-organizing behavior emerged from our proposed piloting model, in which it is able to control and manage virtual resources in order to address the complexity of an autonomic virtual network management. Through this research we have proposed and validates an autonomic piloting model from the multi-agent system perspective. The experimental results of this paper showed us that it satisfies the model's main goal of automatically reconfigure itself, in order to meet the quality requirements and to improve the network performance whenever it is exposed to a critical scenario.

Through our piloting system, we show that it is possible to design an autonomic virtual network manager by applying MAS approach together with self-* capabilities in order to distribute the responsibility to maintain the virtual network running in accordance with the policies and requirements. Although our current work has focused on piloting system designing, modeling and agent communication, we believe that this general model will certainly support the development of more complex network structure, which will be able to perform live migration of virtual routers supported by agent reputation, virtual link management, normative approach to support the policies and requirements, all those from the MAS perspective. We also highlight that, besides the topics above, virtual link management, process of knowledge acquiring /sharing and live migration, considering agent reputation, are important points that deserve our attention in a near future investigations.

References

[1] J. Nogueira, et. al., "Virtual network mapping into heterogeneous substrate networks", in ISCC 2011, June 2011.

[2] A. Fischer, J. F. Botero, M. Duelli, D. Schlosser, X. Hesselbach, and H. De Meer. "ALEVIN - a framework to develop, compare, and analyze virtual network embedding algorithms", In: Electronic Communications of the EASST, 2011, vol. 37, pp. 1–12.

[3] D. Clark, R. Braden, K. Sollins, J. Wroclawski, D. Katabi, J. Kulik, X. Yang, T. Faber, A. Falk, V. Pingali, M. Handley, and N. Chiappa. "New Arch: Future generation Internet architecture", Technical report, MIT Laboratory for Computer Science and International Computer Science Institute (ICSI), 2011.

[4] "4WARD FP7 project." [Online]. Available: http://www.4ward-project.eu/

[5] I. Fajjari, M. Ayari, G. Pujolle and Hubert Zimmermann. r"Towards an Autonomic Piloting Virtual Network Architecture", In: IFIP International Conference on New Technologies, Mobility and Security - NTMS, IEEE XPlore, 2011, Paris, France.

[6] M. A. Netto, B. S. Neto, E. Cirilo, C. Lucena. "A Self-Organizing and Normative Piloting System". Technical repot, Pontifícia Universidade Católica do Rio de Janeiro, 2013, 05/13.

[7] M. S. Blumenthal and D. D. Clark. "Rethinking the design of the Internet: the end-to-end arguments vs. the brave new world", In: ACM Transactions on Internet Technology, 2001, 1(1):70–109.

[8] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, and T. Schooley. "Evaluating Xen for router virtualization", In: International Workshop on Performance Modeling and Evaluation (PMECT), 2007.

[9] N. Feamster, L. Gao, and J. Rexford. "How to lease the Internet in your spare time", In: SIGCOMM Comput. Commun. Rev., vol. 37, no. 1, pp. 61-64, 2007.

[10] N. Fernandes, M. Moreira, I. Moraes, L. Ferraz, R. Couto, H. Carvalho, M. Campista, L. Costa, and O. Duarte. "Virtual networks: isolation, performance, and trends", In: Annals of Telecommunications, 2011, vol. 66, pp. 339–355.

[11] P. Horn "Autonomic computing: IBM's perspective on the state of information technology, also known as IBM's Autonomic Computing", 2001.

[12] T. Anderson, L. Peterson, S. Shenker, and J. Turner. "Overcoming the Internet impasse through virtualization", In: IEEE Computer Magazine, 2005, vol. 38, no. 4, pp. 34-4.

[13] P. Ruth, J. Rhee, D. Xu, R. Kennell and S. Goasguen. "Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure", In: Proc. IEEE ICAC, 2006, pp.5 -14 .

[14] Ines Houidi, Wajdi Louati, Djamal Zeghlache, Panagiotis Papadimitriou, Laurent Mathy "Adaptive virtual network provisioning", In: Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures, 2010, New Delhi, India .

[15] C. Marquezan, L. Granville, G. Nunzi, and M. Brunner. "Distributed autonomic resource management for network virtualization," In: Network Operations and Management Symposium (NOMS) IEEE, 2010, pp. 463-470.

[16] C. Senna, M. Soares, D. Batista, E. Madeira, and N. Fonseca. "Experiments with a self-management system for virtual networks," in II Workshop de Pesquisa Experimental da Internet do Futuro (WPEIF), 2011, Campo Grande.

[17] N. C. Fernandes, M. D. D. Moreira, I. M. Moraes, L. H. G. Ferraz, R. S. Couto, H. E. T. Carvalho, M. E. M. Campista, L. H. M. K. Costa, and O. C. M. B. Duarte, "Virtual networks: Isolation, performance, and trends," To be published in the Annals of Telecommunications, 2010.

A Multiagent System for Urban Traffic Control

Antonio de Abreu Batista Júnior
Núcleo de Tecnologia da Informação
Universidade Federal do Maranhão
São Luís, Maranhão, Brasil
junior2004@pop.com.br

Luciano Reis Coutinho
Departamento de Informática
Universidade Federal do Maranhão
São Luís, Maranhão, Brasil
lrc@deinf.ufma.br

***Abstract*— In this paper we propose a multiagent system (MAS) – designed as a social organization – to operate in the control of successive signals along a section of an avenue that combines green wave and adaptive control in a dynamic way. The members of this organization represent traffic agents that communicate among themselve via a specific purpose interaction protocol. On the one hand, the coordination of successive signals is achieved dynamically through standardized communication between organizational roles. On the other hand, the adaptive control is achieved by the independent and autonomous actions of the agents that make up the organization. We test our approach in simulation. The results show that our approach result in good performance, achieving both local control at the intersections as well as coordination of successive intersections.**

***Index Terms*— multiagent coordination, intelligent traffic control, traffic simulation, social organization.**

I. INTRODUÇÃO

O problema do congestionamento é uma questão séria na vida urbana causando transtornos sociais tais como atrasos, perdas econômicas e poluição ambiental [1]. Há muitos aspectos técnicos e sociais com relação a mobilidade que afetam o trânsito e necessitam de melhoras, um deles é como os semáforos regulam os fluxos de veículos nas interseções. Com relação a esse aspecto, uma abordagem usada é o controle adaptativo, onde se busca otimizar o desempenho de cada semáforo de uma forma isolada. Todavia, a suposição de interseção isolada simplifica o problema de otimização da rede viária por não considerar os conflitos que surgem entre interseções adjacentes. Uma outra abordagem empregada é a sincronização de sucessivos sinais para alcançar a chamada onda verde. No entanto, a onda verde tem seus benefícios reduzidos justamente por não exibir o controle adaptativo.

Neste artigo, nós propomos um sistema multiagente (SMA) – concebido como uma organização social – para operar no controle de sucessivos sinais ao longo de uma seção de uma avenida que combina onda verde e controle adaptativo de uma forma dinâmica. Os membros dessa organização representam agentes de trânsito que se comunicam entre si por meio de um protocolo de uso específico. Por um lado, a coordenação de sucessivos sinais é alcançada dinamicamente através da comunicação padronizada entre papéis organizacionais. Por outro lado, o controle adaptativo é alcançado pelas ações autônomas e independente dos agentes que compõem a organização.

Este trabalho é inspirado na atuação dos agentes humanos no controle do tráfego urbano. A imitação do comportamento autônomo dos agentes e da ação conjunta deles em uma organização social são as bases para este trabalho.

Este artigo está organizado como segue. Na seção II nós apresentamos alguns trabalhos relacionados. Na seção III, nós propomos uma nova arquitetura multiagente para o controle do tráfego urbano que imita uma organização social. A seguir, na seção IV discutimos os detalhes de implementação e os resultados de simulação. Finalmente, na seção V concluímos o artigo e apontamos as direções das pequisas futuras.

II. TRABALHOS RELACIONADOS

Em [1] é proposta uma abordagem baseada em agentes, onde cada interseção é controlada por um agente auto interessado operando com uma visão limitada da chegada de veículos. A questão central dessa abordagem é uma representação agregada dos fluxos de tráfego como padrões agregados críticos de filas e pelotões antecipados. Esses padrões agregados fornecem a base para políticas de controle de sinais em tempo real que incorpora uma visão antecipada do fluxo de veículos. Os autores projetaram duas políticas baseadas em pelotões com o objetivo de decidir se estende ou não a fase através de períodos ociosos sem tráfego, a fim de servir tráfego futuro, com o objetivo de promover coordenação indireta entre sucessivos sinais e o estabelecimento de ondas verdes.

Em [2] é apresentada uma abordagem onde cada semáforo se comporta como um inseto social, tendo planos coordenados de sinais como tarefas a serem executadas. O modelo utiliza um mecanismo de comunicação restrita e grupos coordenados são formados de uma maneira dinâmica. A abordagem pretende combinar as vantagens da descentralização, via *swarm intelligence* e a formação dinâmica de grupos. A principal vantagem desta abordagem é a adaptação às mudanças no tráfego. As mudanças são percebidas e os agentes reagem a essas mudanças de forma rápida e independente, sem qualquer organização hierárquica.

Em [3] é descrito uma abordagem onde cada semáforo é modelado como um agente. Cada um possui planos prédefinidos para sincronização/coordenação com agentes adjacentes em diferentes direções de acordo com a situação do tráfego. Essa abordagem utiliza técnicas de teoria dos jogos evolucionária, tendo como principais benefícios: os agentes podem criar subgrupos de sincronização para melhor

atender as necessidades do fluxo em alguma direção, não há necessidade de um controle central e não há comunicação nem negociação direta entre os agentes.

In [4] é considerado uma abordagem que mistura redes neurais, aprendizagem por reforço e lógica *fuzzy* para aprender em tempo real os comportamentos dos sinais de trânsitos. Porém a parte que interessa a esse trabalho é a maneira como o problema da coordenação entre sinais adjacentes foi modelada. A coordenação é feita usando agentes de controle e o conceito de teoria dos jogos. Um algoritmo foi projetado para resolver o problema da coordenação diretamente, sem que nenhuma negociação acontecesse.

Em [5] é proposto um framework multiagente para o controle de sinais que combina coordenação indireta e direta. A reação a fluxo de tráfego dinâmico é atendido pela coordenação indireta e a formação de onda verde é atendida pela coordenação direta. Em fluxo de tráfego diário normal, cada agente opera no modo de coordenação indireta. Porém, quando o fluxo de tráfego colapsa próximo a certos agentes, os agentes mudam seu modo de coordenação para o modo de coordenação direta para formar um grupo organizado que crie onda verde.

Em [6] é desenvolvido um modelo de fluxo de tráfego, composto pelos seguintes elementos: estradas, cruzamentos, semáforos e volumes de veículos. Para garantir os resultados esperados, um algoritmo de formigas foi aplicado no qual os veículos em movimento nas estradas do modelo deixam feromônio dependendo de quanto tempo eles iriam esperar para atravessar um cruzamento particular. Para estradas congestionadas com sequências de semáforos mal coordenados, um nível adequado de feromônio depositado incentivaria o tráfego a ser direcionado em uma direção apropriada. Esta relação entre o tempo de espera e a concentração de feromônio também funcionaria como um sinal para iniciar mudanças em sequências de semáforos nas interseções de modo que os veículos poderiam sair mais rapidamente da área congestionada.

Os trabalhos citados são muito atrativos e apresentam bons resultados, porém as vantagens de se combinar o controle adaptativo e a sincronização de sucessivos sinais ainda não foi completamente explorado. Nós acreditamos que podemos conseguir resultados ainda melhores. Neste artigo nós propomos um sistema multiagente para o controle do tráfego urbano que explora essas características. O sistema é capaz de sincronizar sucessivos sinais de uma forma dinâmica e localmente exibir controle adaptativo.

III. UMA ARQUITETURA ORGANIZACIONAL PARA O CONTROLE DO TRÁFEGO URBANO

A Fig. 1 mostra uma visão geral da arquitetura. Aqui, a camada organizacional estabelece um conjunto de restrições que moldam ou restringem a atividade conjunta dentro da organização tendo em vista o propósito de combinar onda verde e controle adaptativo sobre a via arterial. A segunda camada é a camada sobre a qual a organização de agentes executa. Essa camada fornece toda infraestrutura necessária para implementação distribuída dos agentes. A terceira camada é a camada sobre a qual os agentes operam. O ambiente é suposto ser o ambiente real, porém para propósito de teste, como descrito neste trabalho, o ambiente pode ser um simulador de tráfego.

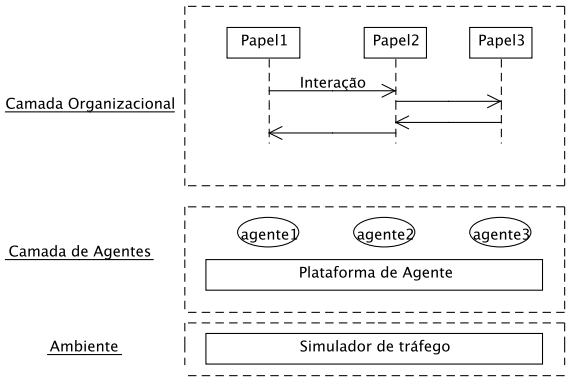

Fig. 1. Visão geral da Arquitetura em Camadas.

A seguir definimos um agente de trânsito desta organização social.

A. Agente de Trânsito BDI

São entidades de software que estão situadas em um ambiente dinâmico, que recebem continuamente informações sobre ele e tomam ações para modificá-lo, tudo baseado em seu estado mental interno. Crenças, Desejos e Intenções são as três atitudes mentais e elas capturam os componentes informacionais, motivacionais, e de decisões do agente, respectivamente.

1) Crenças: As crenças C de um agente j são usadas para determinar que pré-condições de planos da biblioteca de planos do agente são satisfeitas. Por meio de sensores distribuídos sobre as pistas e da interação com outros agentes adjacentes, o agente j captura uma coleção de informações, sobre o cruzamento j que ele controla, criando um mundo virtual que ele acredita existir e sobre o qual gera desejos. Na tabela I essas crenças são apresentadas.

TABLE I
MUNDO VIRTUAL CRIADO PELO AGENTE J.

Base de Crenças do Agente j
Plano semafórico i em uso no cruzamento j (PS_j^i)
Grau de saturação do link crítico de cada fase i do cruzamento j (Sat_j^i)
Número de carros na fila n

Onde o grau de saturação de um link crítico é dado pela fórmula,

$$Sat_j^i = \frac{VecIn}{VecOut} \quad (1)$$

Sendo que $VecIn$ é o número de veículos que deseja atravessar a área de retenção do cruzamento j e $VecOut$

é o número de veículos que pode atravessar, ambos medidos em um certa quantidade de ciclos.

Em termos práticos, o grau de saturação de um link reflete seu nível de carregamento. Por exemplo, se o grau de saturação de um link é igual a 50% ($Sat_j^i = 0.5$), significa que poderia passar duas vezes o número de veículos durante o tempo de verde adotado. Um outro exemplo, se o grau de saturação de um link é igual a 150% ($Sat_j^i = 1.5$), isto significa que dois terços dos veículos cruzariam a área de retenção durante o tempo de verde adotado enquanto um terço seria retido para o próximo período.

2) Desejos: Os desejos de um agente são as pós-condições de planos da biblioteca de planos do agente. Os desejos de um agente j são um conjunto de pares $\langle d, Cond_{dj} \rangle$ tal que, d é um desejo e $Cond_{dj}$ é uma atribuição a uma proposição p. A interpretação é que j acredita que o desejo d é satisfeito se $Cond_{dj}$ é satisfeita com relação a C_j. A TABLE II mostra o modelo de objetivos do agente j fictício. O agente controla um cruzamento com duas fases $i = \{1, 2\}$. Ele pode acionar 5 planos semafóricos. Ao perceber o modelo do mundo, o agente decidirá se aciona um plano semafórico ou mantém o atual.

TABLE II
MODELO DE OBJETIVOS DO AGENTE j.

d	$Cond_{dj}$
PS_j^1	$PS_j^2 \bigwedge Sat_j^1 < 1.0 \bigwedge Sat_j^2 < 1.0$
PS_j^2	$(\forall PS_j^i \neq PS_j^2) \bigwedge Sat_j^1 > 1.5 \bigwedge Sat_j^2 < 1.0$
PS_j^3	$(\forall PS_j^i \neq PS_j^3) \bigwedge Sat_j^1 > 1.5 \bigwedge Sat_j^2 > 1.5$
PS_j^4	$(\forall PS_j^i \neq PS_j^4) \bigwedge Sat_j^1 < 1.0 \bigwedge Sat_j^2 > 1.5$
PS_j^5	$PS_j^4 \bigwedge Sat_j^1 < 1.0 \bigwedge Sat_j^2 < 1.5$

3) Intenções: As intenções de um agente representam os desejos $\langle d, Cond_{dj} \rangle$ que ele se compromete a alcançar. Esse comprometimento, do agente com um objetivo, se dá por dois motivos:

- Ou porque ele faz parte de uma sociedade, e assim, ele resolve atender um pedido de outro agente para alcançar um objetivo em prol do funcionamento coerente da comunidade; ou
- Porque após a análise do seu modelo de mundo atual, ele deseja algo diferente. Ele acredita que a execução de um determinado plano pode levar ao modelo de mundo pretendido.

A seguir é definido o protocolo de interação usado pelos agentes de trânsito. Ele foi criado com o propósito de direcionar as ações dos agentes para criar ondas verdes dinamicamente, sobre a artéria na direção de fluxo mais carregado. Porém mantendo as preocupações locais de cada agente, tais como, não deixar criar filas em nenhuma das pistas dos cruzamentos que eles controlam.

B. Protocolo de Onda Verde Adaptativa (POVA)

Este protocolo usa a linguagem de comunicação KQML [7] usada pelo interpretador Jason. Sua semântica é baseada na teoria dos atos da fala definida em [8], onde as mensagens estão associadas a atos performativos que representam a vontade do agente sobre a informação contida na mensagem. A seguir é listado os atos performativos usados neste protocolo, r é o emissor da mensagem e s o receptor:

- *askOne* é usado quando um agente r quer saber se o conteúdo da mensagem é verdadeiro para um outro agente s;
- *tell* é usado quando um agente r quer que um agente s acredite que ele acredita que o conteúdo da mensagem é verdadeiro;
- *achieve* é usado quando um agente r quer que o agente s tente e alcance um estado de coisas onde a literal no conteúdo da mensagem é verdadeira.

Nas próximas seções definimos a parte estática e dinâmica do protocolo.

1) Estrutura Estática: Neste artigo, os agentes de trânsito são vistos como membros de uma organização social. O modelo MOISE+ [9], na Fig. 3, é usado para a especificação estrutural da organização. Nesta especificação, os *Ag Trânsitos* são responsáveis por controlar a sequência de semáforos ao longo da artéria. São definidos três papéis, *Ag Externo*, *Ag Meio*, *Ag Base*. Os *Ag Externos* comandam os *Ag Meios* e controlam as interseções das extremidades. Um *Ag Base* é o *Ag Externo* que comanda a organização em um determinado momento. Um *Ag Externo* é um *Ag Base* se ele possui um objeto indicador que indica que ele é o *Ag Base*. Os *Ag Meios* são responsáveis por controlarem as interseções intermediárias e colaborarem com os *Ag Externos*.

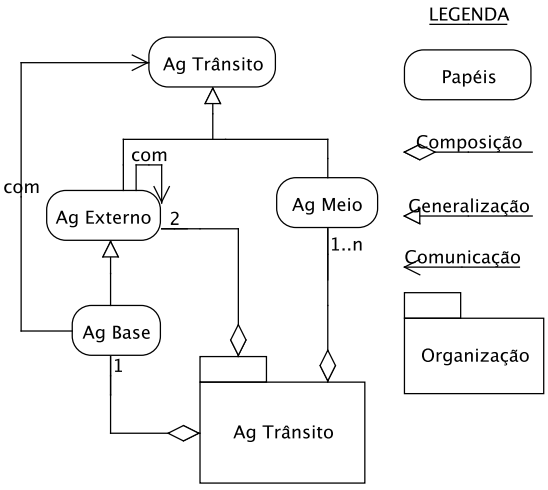

Fig. 2. Estrutura organizacional do SMA, em MOISE+.

2) Estrutura Dinâmica: Nesta seção descrevemos a dinâmica de funcionamento da organização de agentes. Por

meio de diagramas AUML [10], são ilustradas as sequências de mensagens trocadas entre os papéis e os tipos de mensagens que podem ser enviadas e recebidas por cada papel.

O protocolo prevê três situações - (1) que agente inicia o protocolo, (2) o que um agente base sem o objeto de negociação deve fazer para obtê-lo e, (3) de posse do objeto de negociação, o que deve fazer o agente base quando a situação do trânsito mudar.

Iniciando o protocolo: No diagrama AUML da Fig. 3 são ilustradas as sequências e as mensagens trocadas para iniciar o protocolo. Após o término da inicialização do sistema, qualquer $Ag\ Externo$ pode iniciar o protocolo. Um $Ag\ Externo$, ao receber uma mensagem de inicialização deve responder, indicando que ele tomou conhecimento e que ele sabe a quem requisitar o objeto indicador. O $Ag\ Externo$ que iniciou o sistema, após receber a mensagem de resposta, deve informar aos $Ag\ Meios$ que ele é o $Ag\ Base$. Em seguida, ele deve informá-los o objetivo que eles devem alcançar $achieve(objetivo)$.

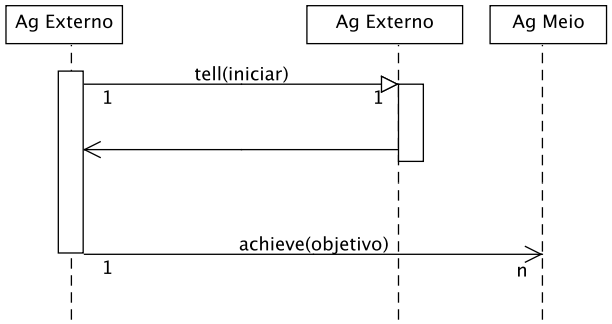

Fig. 3. Inicialização do protocolo.

Adquirindo o objeto de negociação: A Fig. 4 ilustra a troca de mensagens entre os papéis para adquirir o direito de ser um $Ag\ Base$ o objeto indicador. Sobre a artéria, somente um $Ag\ Base$ possuirá o direito de controlar o sentido da onda verde, sendo este o responsável pela decisão da liberação do direito a outro $Ag\ Externo$. No entanto, assim que um $Ag\ Externo$ analisa as condições de sua via e verifica que necessita do direito de controlar o sentido da onda verde, ele realiza um pedido do objeto ao $Ag\ Base$ que o possui. Assim que um $Ag\ Base$ recebe um pedido do objeto, ele analisa as condições de sua via e decide, através da análise de suas crenças e objetivos, por passar ou não o objeto de negociação. Caso resolva não passar o objeto, o $Ag\ Externo$ aguarda alguns segundos e tenta novamente. Desta forma o objeto indicador passa de $Ag\ Externo$ para outro alternando o sentido da onda verde de acordo com o sentido mais carregado da via arterial.

No diagrama também é mostrado que de posse do objeto indicador o $Ag\ Base$ requisita aos $Ag\ Meios$ para alcançar um objetivo. Um desejo d (plano semafórico) e uma condição $Cond_{dj}$ (situação do trânsito que deve ser mantida para utilização do plano semafórico) são passados para os $Ag\ Meios$. O alinhamento dos planos sobre a artéria se dá quando cada Ag de trânsito individualmente alcança seus objetivos (planos), uma vez que a defasagem entre os planos foi previamente configurada.

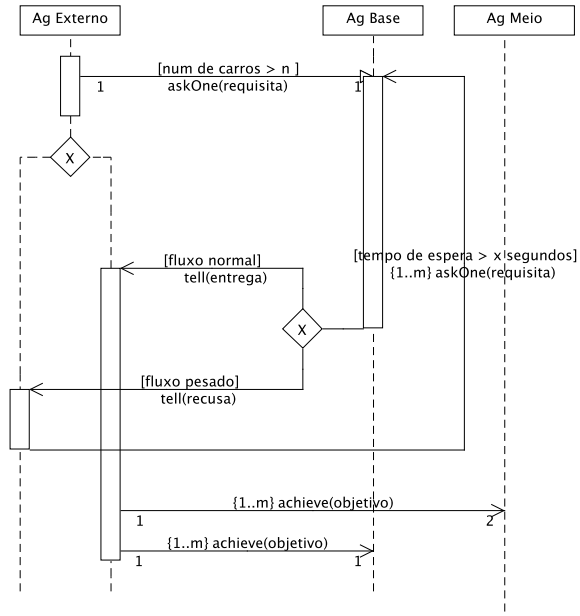

Fig. 4. Adquirindo o objeto de negociação.

Mudando os objetivos: O diagrama AUML da Fig. 5 ilustra a situação em que a atualização de crenças leva a novos objetivos. O $Ag\ Base$ de posse do objeto pode mudar o seu objetivo e reenviar um novo objetivo aos $Ag\ Meios$.

O controle Adaptativo é alcançado através da autonomia dos agentes à medida que eles devem manter um estado de coisas apropriado (situação do trânsito que deve ser mantida para utilização do plano semafórico perseguido). De um modo geral, os $Ag\ de$ trânsitos mantêm o desejo de implementar o plano semafórico alvo e, para satisfazer suas vontades, modifica as condições do ambiente para que as pré-condições do plano alvo sejam satisfeitas para que possam adotá-lo. Após implementada a sua vontade, eles observam continuamente as pós-condições do plano alvo de modo a tomar ações para corrigi-lo, caso ele não esteja de acordo com suas crenças.

IV. IMPLEMENTAÇÃO DO MODELO

Nesta seção, nós discutimos a implementação e testes para validação do modelo.

A. Construção do Ambiente de Simulação

A Fig. 6 mostra uma visão geral do ambiente de simulação. O interpretador Jason [11] é usado para criar os agentes. O simulador de tráfego SUMO (Simulation of Urban Mobility) [12] torna-se o ambiente onde os agentes atuam. O Jason é integrado ao SUMO por meio da XTraCI API[1].

[1]http://www.cs.cmu.edu/ xfxie/download/xtraci1.0.tar.gz

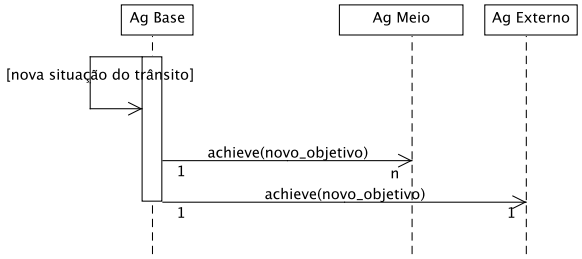

Fig. 5. Informando novos objetivos.

Cada agente pode perceber e modificar a condição atual do tráfego no cruzamento sob sua responsabilidade. Quando um agente decide adotar um plano semafórico, um processo é executado para modificar o plano semafórico atual, no ambiente do simulador, para refletir a vontade do agente. Da mesma forma, a percepção dos agentes é obtida através de sensores espalhados pelo ambiente do simulador.

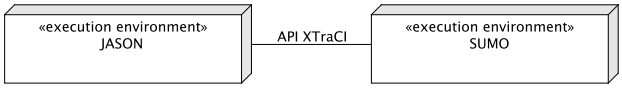

Fig. 6. Integrando Jason ao SUMO.

B. Cenário de Simulação

O experimento acontece sobre o cenário simples abstrato da Fig. 7 e, tem dois objetivos:

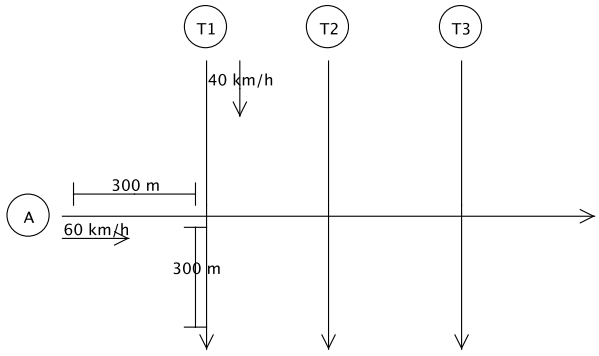

Fig. 7. Arterial network.

1) *Medir a capacidade de adaptabilidade em tempo real dos agentes ao enfrentar problemas locais; e*
2) *Medir a habilidade dos agentes de se auto organizarem criando onda verde, em um sentido programado, à medida que as condições em seus cruzamentos forem jugadas adequadas por eles.*

Cada agente foi equipado com um conjunto de planos previamente configurados e, apresentados na TABLE III.

TABLE III
PLANOS DOS AGENTES.

Planos	Cíclo	Fase1	Fase2	Objetivo
plan 1	80	50	30	servir rapidamente
plan 2	120	80	40	Aumentar atendimento
plan 3	120	30	90	priorizar fase2

A velocidade máxima permitida na via arterial (A) é de 60 km/h e nas transversais (T_i) é de 40 km/h. Todos os segmentos de pista têm o mesmo comprimento L=300 m. A fim de medir os objetivos 1 e 2, foram reproduzidas situações de tráfego onde essas propriedades pudessem ser avaliadas. Nós simulamos tráfego aleatório de um único tipo de veículo. Os veículos partem das fontes P1, T 1, T 2 e T 3 de acordo com um processo de Poisson aproximado aqui por uma distribuição binomial.

Resultados e Discussões: A Fig. IV simula uma possível situação de fluxo intenso nas transversais T2=3/16 e T3=4/16, nos primeiros 20 minutos da simulação e, de fluxo normal nos 40 minutos finais. Em contrapartida, na artéria, há um aumento gradual do fluxo de veículos ao longo de uma hora. Nos últimos 20 minutos, o fluxo é intenso sobre ela. Com o propósito de medir o objetivo 1 o parâmetro TD_{T_i}

TABLE IV
NÚMERO DE VEÍCULOS/PERÍODO DE TEMPO GERADO EM CADA FONTE.

Cenário 1	T1	T2	T3	P4
1-20 min	1/30	3/16	3/16	1/30
21-40 min	1/30	1/30	1/30	4/16
41-60 min	1/30	1/30	1/30	1/16

foi usado para medir a adaptabilidade. O TD_{T_i} representa o atraso experienciado pelos veículos nas transversais T_i. O cenário 1 é usado para verificar esse objetivo.

A TABLE V mostra o resultado das medições. A diferença do atraso medido, TD_{T_i}, entre um agente capaz de perceber e adaptar-se as variações de fluxo nas transversais e uma estratégia de plano fixo foi bastante expressiva e, mostra o ganho obtido quando controlado por um agente. Cada agente individualmente é capaz de decidir o plano correto de acordo com a situação percebida no seu cruzamento.

TABLE V
O ATRASO EXPERIENCIADO PELOS VEÍCULOS NAS TRANSVERSAIS

Forma de controle	TD_{T_1}	TD_{T_2}	TD_{T_3}	$\sum_{i=1}^{3} TD_{Ti}$
Plano fixo	50,8	6028,9	6425,3	12505,0
Agente	41,8	458,5	315,2	815,5

A Tab. VI reproduz uma situação de fluxo intenso em T2 nos primeiros 20 minutos da simulação. Nos próximos 20 minutos T3 exibe fluxo intenso, enquanto a situação em T2 se normaliza. Nos últimos 20 minutos o fluxo é normal em T2 e T3. Por outro lado, na artéria, há uma diminuição gradual do fluxo de veículos.

Com o propósito de medir a habilidade do SMA de criar onda verde sobre a artéria, o parâmetro TD_{A_i} foi medido. O

TABLE VI
QUANTIDADE DE VEÍCULOS/SEGUNDO PARTINDO DAS FONTES GERADORAS.

Cenário 2	T1	T2	T3	P4
1-20 min	$1/30$	$3/16$	$1/16$	$2/16$
21-40 min	$1/30$	$1/30$	$3/16$	$2/16$
41-60 min	$1/30$	$1/30$	$1/30$	$1/16$

TD_{A_i} representa o atraso experienciado pelos veículos sobre o segmento i da artéria.

Na TABLE VII é mostrado os resultados de simulação do cenário 2. Os resultados mostram que nos primeiros 20 minutos da simulação há uma sobrecarga $T2 = 3/16$ sobre a pista transversal do cruzamento 2. Diante dessa situação, o agente 2 adota planos para trazer a situação do seu cruzamento para o estado normal. Nesse intervalo, a quebra de sincronismo devido à necessidade de adaptação local, causa atrasos nas pistas arteriais TD_{A_2} e TD_{A_3}. No entanto, passados os primeiros 20 minutos a situação na transversal do cruzamento 2 se normaliza $T2 = 1/30$. Essa condição é percebida pelo agente 2 que imediatamente volta a adota o plano de sincronismo. O valor de $TD_{A_2} = 0$, medido nos últimos 30 minutos da simulação, indica que há sincronismo dos cruzamentos 1 e 2.

TABLE VII
O TD_{A_i} SOBRE OS CRUZAMENTOS 2 AND 3 MEDIDOS EM UMA HORA DE SIMULAÇÃO.

Primeiros 30 min		**útimos 30 min**		**últimos 15 min**	
TD_{A_2}	TD_{A_3}	TD_{A_2}	TD_{A_3}	TD_{A_2}	TD_{A_3}
191	134	0	113	0	0

A mesma situação pode ser observada sobre o cruzamento 3, no intervalo de 20 à 40, o agente precisa constantemente trocar de plano para normalizar as condições do tráfego. Porém passado o intervalo, a situação volta a se normalizar $T3 = 1/30$ e o agente adota o plano de sincronismo. Novamente, essa situação pode ser vista na TABLE VII, nos 15 minutos finais o valor dos parâmetros $TD_{A_2} = 0$ e $TD_{A_3} = 0$ é nulo, o que indica que o sistema está sincronizado pela cruzamento 1.

Dessa forma, comprova-se a habilidade dos agentes de se auto organizar para gerar fluxo suave quando as condições são favoráveis. O mais comum é fluxo mais intenso sobre a artéria, o que justifica o esforço dos agentes para manter o sincronismo sobre ela.

V. CONCLUSÃO

Neste artigo, nós propomos um sistema multiagente – que imita uma organização social – para operar no controle do tráfego urbano combinando adaptação local e coordenação de sucessivos semáforos. Um protocolo de uso específico foi usado internamente pela organização de agentes para alinhar uma sequência de semáforos de uma forma dinâmica. O controle adaptativo requerido para otimização local foi alcançado pelas ações independentes e autônomas dos agentes localmente. A arquitetura BDI foi usada para modelar a parte cognitiva dos agentes de trânsito e um simulador de tráfego foi usado para o propósito de teste e validação do funcionamento do sistema.

Nós testamos nossa abordagem em simulação. Os resultados mostraram que nossa abordagem resultou em uma boa performance, alcançando tanto controle local quanto coordenação de sucessivos sinais. A melhora na performance é atribuída à autonomia dos agentes e à eficiência do protocolo de onda verde adaptativa.

No entanto, ainda há muito a ser investigado. Como trabalhos futuros pretendemos dar aos agentes à habilidade de aprender novos planos, e desta forma, tornar o sistema proposto independente das peculiaridades de cada sequência de semáforos. Nós também pretendemos adicionar ao protocolo mecanismos de tolerância a falhas e melhorar a capacidade de percepção dos agentes.

REFERENCES

[1] X. Xie, G.J. Barlow, S.F. Smith, Z.B. Rubinstein, "Platoon-Based Self-Scheduling for Real-Time Traffic Signal Control," in IEEE International Conference on Intelligent Transportation Systems (ITSC), pp. 879–884, 2011.

[2] D. Oliveira, A.L.C Bazzan, "Swarm Intelligence Applied to Traffic Lights Group Formation," in VI Encontro Nacional de Inteligência Artificial (ENIA 2007), pp. 1003-1012, 2007.

[3] A.L.C Bazzan, "A Distributed Approach for Coordination of Traffic Signal Agents," in Autonomous Agents and Multi-Agent Systems, Vol. 10, pp. 131–164, 2005.

[4] S. Shamshirband, "A Distributed Approach for Coordination Between Traffic Lights Based on Game Theory," The International Arab Journal of Information Technology, pp. 148–152, 2012.

[5] T. Shirai, Y. Konaka, J. Yano, S. Nishimura, K. Kagawa, T. Morita, M. Numao and S. Kurihara, "Multi-agent traffic light control framework based on direct and indirect coordination," in Proceedings of the 7th International Workshop on Agents in Traffic and Transportation, pp. 9–17, 2012.

[6] D. Król, M. Mrozek, "Swarm-based Multi-agent Simulation: a Case Study of Urban Traffic Flow in the city of Wroclaw," ICCCI 2011, LNAI 6923, Springer, pp. 191–200, 2011.

[7] T. Finin, R. Fritzson, D. McKay, R. McEntire, "Kqml as an agent communication language," in Proceedings of the 3rd International Conference on Information and Knowledge Management, pp. 456–463, 1994.

[8] J. Searle,"Speech Acts: An Essay in the Philosophy of Language," Cambridge University Press, 1969.

[9] J. F. Hübner, J. S. Sichman, O. Boissier, "S-moise+: A middleware for developing organised multiagent systems," in International Workshop on Organizations in Multi-Agent Systems: From Organizations to Organization Oriented Programming, pp. 107–120, 2005.

[10] B. Bauer, "Extending uml for the specification of interaction protocols," in submission for the 6th Call for Proposal of FIPA and revised version part of FIPA, 1999.

[11] R. H. Bordini, J. F. Hubner, M. Wooldridge, "Programming Multi-Agent Systems in AgentSpeak using Jason," John Wiley & Sons, London, UK, 2007.

[12] M. Behrisch, L. Bieker, J. Erdmann and D. Krajzewicz, "SUMO - Simulation of Urban MObility: An Overview," in SIMUL 2011, The Third International Conference on Advances in System Simulation, pp. 63–68, 2011.

Multiagent System to search and contracting Tourism services

João Ferreira de Santanna Filho, Scheila Nair Costa, Camila Pontes B. da Costa,
Charbel Szymanski, João Eduardo Hornburg
PPGEAS – Department of Automation and Systems
Universidade Federal de Santa Catarina
Florianópolis, Brasil
{joaosantanna,Camila,charbel}@das.ufsc.br, {nc.scheila, joao.hornburg}@gmail.com

***Abstract*— This paper presents a multi-agent system (MAS) to integrate the service offerings of numerous companies of the tourism segment. This idea was motivated by world sporting events that will happen in the coming years in Brazil (FIFA World Cup and Olympics). The paper demonstrates that MAS's can be a platform for integrating distributed systems resulting in more flexible systems with embedded intelligence.**

***Keywords*— multi-agent system ; distributed systems; Tourism services.**

I. INTRODUÇÃO

Nos anos de 2014 e 2016 o Brasil será sede de dois grandes eventos esportivos de nível mundial, a Copa do mundo de Futebol e as Olimpíadas. De acordo com o SEBRAE [1], a realização desses eventos proporciona uma chance de promoção e atração de investimentos, alavancando significativamente a economia brasileira. Estima-se que o fluxo seja em torno de 600 mil turistas internacionais e 3.100 mil nacionais. É necessário que o setor de serviços se prepare para esse aumento de demanda, em especial o setor Hoteleiro.

Todo esse montante de turistas deve utilizar algum meio para programar e contratar serviços relacionados às viagens. Nesse contexto, a utilização da Internet para o planejamento e compra de pacotes de viagens tem se mostrado uma opção cada vez mais popular. Várias empresas já dispõem de portais para contratação de serviços tais como hotéis, locadoras de carros, restaurantes, bares, serviços de *concierge*, etc. Todavia, esses serviços encontram-se distribuídos cada qual com seu próprio sitio na internet. Dessa maneira, o cliente se vê obrigado a fazer uma pesquisa extensa na internet em busca de melhores preços e melhores ofertas.

Segundo Zagheni and Luna [2], no Brasil, empresas privadas e também do governo não utilizam todo o potencial que a Internet oferece, ou seja, ela é vista apenas como um canal de divulgação, podendo ser comparada com uma versão eletrônica das listas telefônicas. Para que os clientes possam usufruir dos benefícios das novas tecnologias, é necessário que as empresas busquem formas mais simples, ágeis e eficientes de tentar integrar esses serviços, facilitando a busca dos clientes.

A tecnologia de Sistemas Multiagentes (SMA) pode surgir como uma solução para o cenário apresentado. No presente artigo se desenvolve um sistema multiagente (SMA) para a contratação de serviços turísticos. Os agentes representam os vários atores envolvidos no cenário proposto.

Por motivos de ordem prática, o SMA abordou apenas os serviços de Hotelaria e Locação de Automóveis. Porém, visando explorar os diversos recursos disponíveis em Sistemas Multiagentes, foram utilizadas as tecnologias de ontologias, sistemas especialistas e agentes do tipo reativo e cognitivo (Agentes BDI). Para a comunicação entre os agentes foram utilizados dois protocolos diferentes de transação, um baseado em barganha (utilizado pelos agentes hotéis) e o *Contract Net*, utilizado pelas locadoras de carros. O uso de todas essas tecnologias em um único sistema visa demonstrar que um SMA desse tipo pode atender a requisitos diferentes demandados pelas empresas prestadoras de serviço.

O objetivo principal deste trabalho é demonstrar a viabilidade de um SMA agregar serviços distribuídos pela internet visando oferecer uma ferramenta de busca e contratação centralizada que englobe vários serviços.

II. DEFINIÇÃO DO ESCOPO DO SMA

A partir da motivação apresentada, foram implementados no SMA agentes para representar os serviços de Hotelaria, os serviços de locação de automóveis e agentes intermediadores de negociação.

Fig. 1. : Escopo do SMA.

O escopo deste trabalho foi definido como ilustrado na Figura 1. Um Agente de Viagens interage com um agente Gestor de Hotel para encontrar agentes hotéis, e com um agente Gestor de Locadora de Automóveis para encontrar agentes locadoras de Automóveis.

Cada um dos elementos presentes na Figura 1 é implementado na prática por um agente. Dessa forma, temos: 1 agente de viagens, 1 agente gestor de Hotéis, 1 agente gestor de locadoras, *n* agentes de hotéis e *n* agentes de locadoras de automóveis.

O cliente faz algumas escolhas quanto ao serviço. Com base nessas escolhas, o agente classifica o tipo de Hotel e o tipo de automóvel. Depois dessa classificação, o agente de viagens requisita ajuda a outros dois agentes (Gestor de Hotel e Gestor de Locadora de Automóveis) para realizar buscas por hotéis e automóveis segundo as escolhas feitas pelo cliente.

Por fim, será apresentado ao cliente um pacote contendo: a reserva de hospedagem em um hotel e uma locadora com automóvel reservado.

III. CONSIDERAÇÕES SOBRE A ESCOLHA DAS FERRAMENTAS DE IMPLEMENTAÇÃO DOS AGENTES

A plataforma JADE (*Java Agent Development Framework*) foi escolhida como base do SMA e grande parte dos agentes que compõe o SMA foi implementado usando esse *middleware*. Segundo Bellifemine, et al. [3], Jade é um *middleware* que facilita o desenvolvimento de SMA's e já vem sendo usado com sucesso em aplicações de diversos setores como gestão de cadeia de suprimentos, gestão de frotas, leilões e turismo.

Algumas características levaram à opção pelo uso do JADE, principalmente o total suporte ao padrão FIPA (*Foundation for Intelligent Physical Agents*) para a comunicação entre os agentes [3]. O uso desse padrão de comunicação permitiu que, em um segundo momento, os agentes JADE pudessem conversar com os agentes Jason (*A Java-based AgentSpeak Interpreter Used with Saci For Multi-Agent Distribution Over the Net*) [4] utilizando o JADE como plataforma de comunicação. Outro fator que levou à adoção da plataforma JADE no projeto foi o fato dele ser implementado na linguagem de programação Java. O *middleware* foi desenvolvido para prover um conjunto de API's que são independentes do tipo de rede e da versão de Java utilizada. Dessa forma, o SMA pode ser facilmente estendido no futuro e ter agentes rodando a partir de dispositivos móveis como celulares que suportem a plataforma Java.

Além do JADE, parte do sistema foi desenvolvido usando Jason. A motivação para o uso do interpretador Jason foi poder utilizar agentes cognitivos baseados na arquitetura BDI (B*elief, Desires and Intentions*) [5]. Diferentes dos agentes JADE, que são reativos, os agentes Jason possuem crenças, objetivos, planos e intenções, sendo a programação realizada na linguagem AgentSpeak, e a comunicação entre agentes baseada na teoria de atos de fala[6]. Em agentes Jason os planos executados e as crenças podem mudar dinamicamente conforme os agentes vão interagindo no SMA. Dessa forma, um agente que esteja participando de uma negociação pode facilmente mudar de estratégia, desde que suas crenças sejam alteradas. Apesar dos agentes Jason apresentarem uma arquitetura mais rica que os agentes JADE, no presente trabalho eles só foram utilizados para negociação do tipo *Contract NET*, onde cada agente Jason representava uma locadora de automóveis. No presente artigo só integramos agentes com arquiteturas diferentes em um mesmo SMA. As características de agentes BDI podem ser melhor exploradas em um futuro trabalho.

IV. DESENVOLVIMENTO DO SMA

Atualmente existe um grande número de metodologias na área de Sistemas Multiagentes. Segundo Akbari [7], de 1990 até 2010 foram desenvolvidas cerca de 75 metodologias diferentes. A metodologia selecionada para a modelagem do sistema foi a metodologia "Tropos", em razão da mesma estar bem documentada e apresentar ferramentas computacionais de apoio na plataforma em que o SMA estava sendo desenvolvido.

Tropos é uma metodologia de desenvolvimento de *software* orientada a agentes. Foi originalmente desenvolvida na Universidade de Toronto, no Canadá. Tropos tem como foco principal a análise de requisitos e adota o paradigma i* Mylopoulos, et al. [8] como base. O i* oferece conceitos como atores, objetivos e dependências destinadas para modelar as estruturas sociais e descrever relações entre eles. A metodologia Tropos é dividida em 5 fases de desenvolvimento: (1) requisitos iniciais; (2) requisitos finais; (3) projeto arquitetural; (4) projeto detalhado e (5) implementação.

A. Requisitos iniciais do SMA

Utilizando o escopo definido, podemos identificar as partes interessadas, suas intenções (*goals*) e seus relacionamentos. Além dos agentes especificados no escopo, foi criado um novo agente categorizador de hotéis. A tarefa desse agente é receber as características escolhidas pelos clientes e, a partir dessas características, ranquear o hotel em uma categoria: 3, 4 ou 5 estrelas.

Após definidos os interessados e suas intenções, uma nova análise foi realizada para identificar quais planos os atores deverão realizar para atingir suas intenções. Também foram identificados os relacionamentos entre os atores para a realização dos planos.

Utilizando a ferramenta TAOM4E [9] , as intenções foram transcritas como metas ou meta-soft. Temos ainda, a partir da análise inicial, os planos e as dependências entre os atores, além dos recursos identificados que foram gerados por eles.

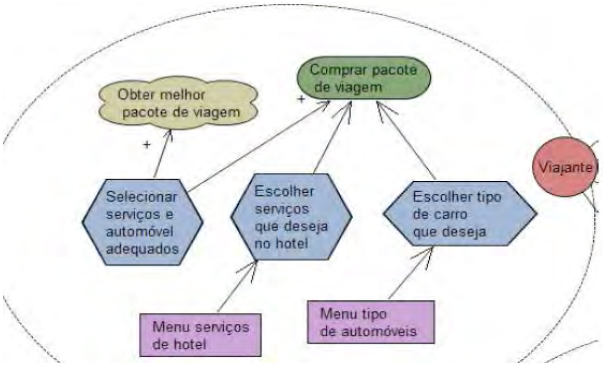

Fig. 2. Diagrama de metas do ator viajante.

Depois de feita a modelagem do Diagrama de Atores, foi modelado o Diagrama de Metas. Essa tarefa consiste em verificar a possibilidade de decompor cada meta que um ator possui em sub-metas, podendo surgir novas metas e novos planos. Depois foi realizada a análise meio-fim, definindo quais planos são os meios para alcançar as metas. A Figura 2 apresenta o diagrama de metas do ator viajante.

B. Requisitos Finais do SMA

O primeiro passo nessa fase foi inserir um ator que representa a interface com o usuário. Esse ator foi denominado SAV (Sistema de Apoio a Viagens). Depois realizou-se uma nova análise em relação as intenções, planos e relacionamentos, incluindo esse ator. O resultado pode ser conferido na Tabela 1.

TABELA I. INTENÇÕES, PLANOS E RELACIONAMENTO ENTRE ATORES

Ator	Intenções (*goals*)	Plano	Relacionamentos
Viajante	- Comprar pacote de viagem - Obter o melhor Pacote de viagem	- Escolher serviço que deseja no hotel - Escolher tipo de carro que deseja	SAV
SAV	- Vender pacote de viagem - Categorizar Hotel - Interagir com Viajante	- Requisitar reserva de hotel - Requisitar locação de automóvel - Analisar serviços escolhidos para categorizar hotéis	- Gestor Hotel - Gestor Locadora - Viajante
Gestor Hotel	Fechar negócio com Hotel	Negociar reserva	Hotéis
Gestor Locadora de Automóveis	Fechar negócio com Locadora	Negociar locação	Locadoras
Hotel	- Obter hóspede - Fornecer serviços	Barganhar proposta	Gestor Hotel
Locadora de Automóveis	Obter Locatário	- Registrar serviços - Informar valor	Gestor Locadora

A partir da Tabela 1 foi possível gerar o Diagrama de Atores da fase de Requisitos Finais e, em seguida, realizar a modelagem do diagrama de metas para o ator SAV.

C. Projeto Arquitetural

Segundo Silva [10], existem um conjunto de estilos arquiteturais já definidos para SMA's. Nesse projeto, identificamos que parte da estrutura é Flat e parte é Hierárquica.

Analisando o diagrama de metas do ator SAV, identificamos cinco papéis. Cada papel foi especificado segundo seus objetivos, responsabilidades, colaboradores, habilidades e normas, que são representadas na Tabela 2.

TABELA 2. ESPECIFICAÇÃO DE PAPÉIS

Agente de viagem	Objetivos:	Montar pacote de viagens
	Responsabilidades:	Procurar Gestor de hotel e gestor de locadora, Categorizar Hotel, Requisitar reserva em Hotel e locação de Automóvel, Publicar pacote montado.
	Colaboradores:	Gerenciador de Hotéis e Gerenciador de Locadoras
	Habilidades:	Conhecer os Hotéis selecionados pelo viajante
	Normas:	Acessar a base de serviços
Gestor de Hotéis	Objetivos:	Negociar reserva em um Hotel
	Responsabilidades:	Procurar Hotéis de uma categoria específica em uma lista, Enviar mensagem requisitando preço para Hotéis, Selecionar melhor oferta.
	Colaboradores:	Hotéis e Gerenciador de Viagens (SAV)
	Habilidades:	Conhecer a categoria do hotel procurado e as propostas do hotéis envolvidos na negociação
	Normas:	Acessar a base de serviços
Gestor de Locadoras	Objetivos:	Negociar locação de automóvel
	Responsabilidades:	Enviar mensagens para locadoras de automóveis, Selecionar melhor oferta.
	Colaboradores:	Locadoras de automóveis e Gerenciador de viagens (SAV)
	Habilidades:	Realizar a *Contract Net* com as locadoras encontradas
	Normas:	Acessar a lista de locadoras disponíveis
Hotel	Objetivos:	Vender a reserva em Hotel
	Responsabilidades:	Responder mensagem com valor
	Colaboradores:	Gerenciador de Hotéis
	Habilidades:	Conhecer ou descobrir sua categoria
	Normas:	Conhecer sua categoria. Caso contrário, deve acessar uma ontologia para descobrir em que categoria se encaixa.
Locadora	Objetivos:	Alugar Automóvel
	Responsabilidades:	Se registrar junto a Gerenciador de Locadoras, Responder mensagem com valor do respectivo automóvel.
	Colaboradores:	Gerenciador de Locadoras
	Habilidades:	Conhecer o tipo de carro desejado
	Normas:	Acesso ao tipo de carro desejado

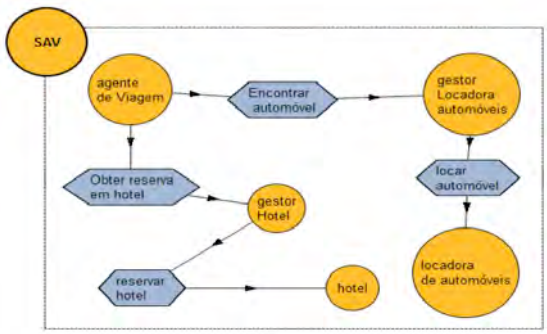

Fig. 3. Diagrama Arquitetural do Sistema.

Na Figura 3 é apresentado o Diagrama Arquitetural do sistema, resultante do mapeamento em conformidade com o estilo arquitetural escolhido.

Finalizou-se a fase de Projeto Arquitetural com a elaboração do diagrama arquitetural e a definição dos papéis.

D. Projeto Detalhado

Nessa fase foi descrita a estrutura interna de cada papel e detalhes de projeto.

Analisando o diagrama de metas do ator SAV (Figura 3), identificamos cinco papéis. Cada papel foi especificado segundo seus objetivos, responsabilidades, colaboradores, habilidades e normas, que são representadas na Tabela 2.

1) Funcionamento do protocolo de Barganha (Gestor Hotel X Hotel)

O protocolo de negociação implementado funciona utilizando barganha em 4 rodadas de negociação. No primeiro *round* o agente Gestor de Hotéis recebe uma requisição para buscar por um hotel com determinada categoria (3, 4 ou 5 estrelas). Baseado nessa requisição, o gestor consulta o serviço de páginas amarelas e recebe uma lista com agentes que representam os hotéis da categoria procurada. Usando a lista, o agente gestor envia pedidos para todos os hotéis utilizando a performativa *request* com o seguinte conteúdo na mensagem (1, 2 ,0). O 1 informa aos agentes que eles estão participando do protocolo de barganha e que estão na primeira rodada de negociação. O 2 representa que a busca é para quartos da 2ª semana do ano. No projeto do protocolo definimos o segundo campo como uma informação de data, mas essa informação não foi utilizada para tomada de decisão. Se assumiu que na data escolhida todos os Hotéis terão vaga. O 0 (zero) da mensagem é o campo que indica a melhor oferta atual. Como os agentes ainda não ofertaram preços, se envia preço 0 como indicador da primeira rodada. Os agentes recebem essa mensagem e, de acordo com suas próprias estratégias, ofertam um valor para o serviço.

Na segunda rodada o agente gestor processa as ofertas que chegam e decide como melhor preço a menor oferta enviada pelos agentes. A seguir o gestor envia uma mensagem com performativa *inform* com o seguinte conteúdo (2, 2, ***melhor Preço***). Como no primeiro envio, o primeiro campo de valor 2 representa que estamos na 2º rodada de barganha, o segundo 2 continua representando a semana de locação do quarto, e o último campo carrega o melhor preço escolhido do conjunto de preços da rodada anterior. Genericamente podemos descrever que o conteúdo das mensagens segue o seguinte formato: **(*n*,*x*,*y*)**, sendo: *n* (número da rodada), *x* (campo de data para reserva do quarto) e *y* (menor preço da última rodada de negociação).

O processo continua até a 4ª rodada. Na rodada final o agente gestor finaliza a negociação enviando uma mensagem com a performativa *Accept Proposal* para o agente do hotel vencedor que deve, nesse momento, reservar o quarto. O restante dos hotéis não recebe mensagem nenhuma e, como o sistema foi projetado para que os agentes dos hotéis não reservem quartos antes da 4ª rodada, não ocorre prejuízo para os hotéis perdedores.

Os agentes do SMA relacionados a hotéis foram todos feitos usando a plataforma JADE[11]. O agente Gestor de Hotéis foi projetado para controlar o protocolo de barganha utilizando um *Tick Behaviour* temporizado para 6 segundos. Nesse meio tempo existe um *Cyclic Behaviour* que atualiza a lista de hotéis continuamente. O agente gestor envia mensagens para esse conjunto atualizado de agentes. Dessa forma, o protocolo permite que um hotel que não participou na 1ª rodada possa participar da 2ª rodada ou da 3ª. Outra escolha de projeto foi que os agentes hotéis não respondam quando não puderem ofertar um preço melhor que o barganhado. Isso foi feito para diminuir a quantidade de mensagens circulando na rede.

2) *Estratégia dos Agentes Hotéis*

Como escolha de projeto desenhamos três agentes com estratégias distintas para participar da barganha de preços. Para essas três classes de agentes com comportamentos distintos foram projetadas mais três classes dessas para cada categoria de hotel (3, 4 e 5 estrelas) com diferentes preços do serviço segundo a categoria dos hotéis . Os três tipos de hotéis são os seguintes:

- Hotel Escada: Oferta 3 preços fixos, um para cada rodada sem variar.
- Hotel Turco: Oferta um preço randômico próximo ao primeiro preço do hotel escada e, a partir daí, nas outras rodadas, sempre que receber do agente Gestor de Hotéis um preço, diminui em R$ 1 (um real, moeda brasileira) a melhor oferta e envia essa oferta ao gestor.
- Hotel Randômico: Como o turco, sua primeira oferta é gerada por um número randômico próximo ao do hotel escada. A partir daí, nas próximas rodadas ele recebe o melhor preço, gera um número randômico e diminui do melhor preço e faz sua oferta com o resultado.

A partir desses hotéis são criados tipos específicos para cada categoria. Cada um desses hotéis com seu próprio agente. Dessa forma temos: Hotel Escada 3, 4 e 5 estrelas; Hotel Turco 3, 4 e 5 estrelas; Hotel Randômico 3, 4 e 5 estrelas. A única alteração feita entre esses hotéis é a precificação do serviço em cada *round*.

3) *Uso de ontologias para um Hotel que não sabe a que classe pertence*

Foi propositadamente desenvolvido no SMA um agente Hotel que não tem conhecimento da sua categoria e utiliza a ontologia criada para descobrir em que categoria se encaixa. Esse agente foi criado prevendo a participação de estabelecimentos não classificados previamente, mas que poderiam participar da negociação. Dessa forma, foi definida uma ontologia para representar os atributos de um hotel. Esta ontologia define as classes *Service*, Hotel e *Room*. A classe *Service* possui como subclasses os serviços que um hotel pode oferecer. A classe Hotel possui como subclasses as categorias de hotéis (“estrelagem”). A classe *Room* define o quarto de um hotel.

Utilizando a ferramenta Protégé[12], temos a hierarquia completa de todas as classes. Foram definidas algumas propriedades para cada classe, tais como: *hasRoom, isRoomOf, hasService e isServiceOf.* A propriedade *hasRoom* possui como domínio a classe Hotel e como *range* a classe *Room.* A propriedade *isRoomOf* é a inversa de *hasRoom.* A propriedade *hasService* também possui como domínio a classe Hotel, e possui como *range* a classe *Service.* Já *isServiceOf* é a propriedade inversa de *hasService*. Além

disso, as subclasses de Hotel possuem restrições (superclasses anônimas), na forma de *hasService some xxx*, que associam cada categoria de hotel aos serviços que eles oferecem.

O agente hotel que utiliza a ontologia inicializa aleatoriamente uma lista de serviços, simulando um hotel não ranqueado tentando participar da negociação. A seguir, o agente carrega a ontologia e a partir da sua lista de serviços realiza um *reasoning* para determinar qual é a sua categoria (3Star, 4Star ou 5Star). O agente também pode concluir que o conjunto de serviços por ele ofertados não atendem os requisitos de nenhuma das categorias. Se o agente se enquadra em alguma categoria, ele se registra nas páginas amarelas para aquela categoria, caso contrário, ele não se registra e não participa de negociações, sua estratégia de negociação é igual à do Hotel Escada.

4) *Sistema Especialista*

Um sistema especialista embutido no agente que interage com usuário (SAV) foi desenvolvido utilizando a ferramenta Jess [13] para fazer a classificação do tipo de hotel que o usuário deseja a partir dos serviços escolhidos. Dessa forma, o agente pode requisitar ao Gestor de Hotéis um hotel com uma classificação com relação às estrelas (3, 4 ou 5 estrelas).

Para definir os requisitos básicos de cada categoria de hotel, foram utilizadas informações do Sistema Brasileiro de Classificação de Meios de Hospedagem [14] A partir dessa classificação, selecionamos alguns serviços que hotéis devem possuir para obter determinada classificação.

Para inserir como um fato os serviços básicos de cada hotel no sistema especialista, foi usada a palavra-chave *deffacts*, que insere os fatos quando é dado um (reset) no programa, conforme mostra o código exemplo a seguir:

(deffacts hoteis)
(hotel3 Internet Frigobar Estacionamento Cafe)
(hotel4 Internet Frigobar Estacionamento Cafe CafeQuarto ServicoQuarto Manobrista)
(hotel5 Internet Frigobar Estacionamento Cafe CafeQuarto ServicoQuarto Manobrista SalaoEventos Banheira Concierge)

O sistema de viagem executa o sistema especialista acrescentando um fato tipo lista com cabeçalho “serviços”com os serviços solicitados pelo usuário. Por exemplo, se um usuário solicitou Banheira e Internet, o Sistema de Viagem acrescentaria o seguinte fato no SE: (serviços Banheira Internet).

5) Funcionamento do agente gestor de locadora de Automóveis

Utilizando o Jason, foi criado um agente BDI do tipo Gestor que utiliza uma Rede de Contrato (*Contract Net*) como suporte para a coordenação da negociação, que tem como estratégia escolher a empresa que oferece a menor diária para o carro solicitado pelo cliente.

Como crença inicial, o agente tem que todas as respostas foram recebidas se o número de participantes for igual ao número de propostas somado ao número de recusas. O agente Gestor de Locadora possui como objetivo inicial se registrar nas páginas amarelas, para que o sistema de viagem possa encontrá-lo. Para atingir esse objetivo, ele tem um plano que chama uma ação interna “jadedf.register”, que o registra como provedor de serviços do tipo “locação-carros” e com o nome “Gestor-LocadoraCarros”. Existe um segundo plano para o caso de ter sido adicionado à sua base de crenças o recebimento de uma mensagem KQML do tipo CFP (*Call for Proposal*). Nesse plano é armazenado como uma crença o remetente da mensagem (agente que solicitou um carro) e então é adicionado um novo objetivo, o de iniciar a rede de contrato *(“!iniciaContractNet”).*

Para atingir o objetivo de iniciar a rede de contrato, existe um plano de esperar 2 segundos para os participantes se registrarem, guardar o estado da RdC (rede de contrato) como fase de “proposta”, procurar por todos os participantes que se registraram e enviar para cada um deles um pedido de proposta para o carro solicitado na CFP recebida. Em seguida ele espera mais 4 segundos e adiciona o plano de “contratar”. Enquanto isso, o agente gestor recebe as propostas e recusas dos participantes. Quando todas as propostas foram recebidas, ele adiciona o objetivo de “contratar”.

O plano para contratar é executado somente se o estado da RdC atual é “proposta”. Ele consiste de alterar o estado da RdC de “proposta” para “contratar”, encontrar todas as ofertas dos participantes, calcular o menor valor de oferta e então determinar o ofertante desse valor como Agente Vencedor. Então, ele adiciona como novo objetivo anunciar o resultado (avisa ao agente vencedor que ele ganhou e aos outros que a proposta foi recusada) e envia para o remetente a empresa de locação que venceu a RdC, juntamente com sua oferta vencedora. O estado da RdC é alterado para “terminado” e a crença das propostas recebidas e da mensagem KQML daquele remetente são apagadas.

6) Funcionamento dos agentes Locadoras de Automóveis

As locadoras de automóveis foram implementadas como agentes Jason. Foram criados 4 agentes locadoras para fins de simulação, cada um com estoque de carros diferente. Os agentes possuem como crenças iniciais os carros com os quais ele trabalha ou não trabalha, e um preço randômico para ofertar. Os agentes possuem um plano inicial de enviar uma mensagem para o Gestor de Locadora, se registrando para participar das RdC gerenciadas por ele.

Caso um agente tenha uma crença de uma CFP vinda do Gestor solicitando um carro, e ele tenha um carro da categoria disponível no estoque. O agente executa o plano de responder à essa CFP, que reserva um carro do tipo solicitado, adiciona uma crença com a proposta feita e envia a proposta para o Gestor. Se o agente receber uma crença do tipo “aceitar proposta” do Gestor de Locadora, significa que o Gestor está avisando que ele ganhou a RdC. Então ele mantém a reserva do carro para o cliente e apaga as crenças com relação àquela RdC, finalizando a contratação. Caso o agente tenha perdido uma RdC, ele receberá uma crença “recusa proposta” do Gestor de Locadora. Então o agente executará o plano de atualizar o estoque, reinserindo o carro que havia sido reservado, e apagará as crenças com relação àquela RdC.

Finalmente, caso o agente não trabalhe com o carro solicitado ou o carro tenha acabado no estoque, é executado um segundo plano onde o agente envia uma mensagem de recusa para o Gestor, informando que ele não participará da RdC

E. Implementação

O SMA foi implementado na plataforma JADE. Os agentes Gestor de Locadora de Automóveis e Locadora de Automóveis são agentes BDI feitos em Jason. O Gestor de Hotéis e os Hotéis são agentes reativos feitos em Java na plataforma JADE.

O agente SAV foi codificado em Java/JADE [11]. Embutido nesse agente, temos um sistema especialista feito em Jess [15] (anteriormente mencionado) para classificar os hotéis segundo escolhas do cliente.

Fig. 4. Solicitação ao SMA de Hotel e Carro econômico com Ar condicionado

Adicionalmente foi implementado uma GUI (*Graphical User Interface*) usando Java/Swing para servir de interface com o usuário (Figura 4). Essa interface captura os serviços do hotel que o usuário necessita, bem como o tipo de carro, e passa essa lista para o "Agente de Viagens", que por sua vez inicia todo o processo de busca no SMA.

V. CONSIDERAÇÕES FINAIS

O potencial de um Sistema Multiagentes vai muito além do que foi apresentado nesse trabalho. SMA's representam um novo paradigma na área de desenvolvimento de *software* para construção de sistemas distribuídos e na integração de sistemas. O SMA desenvolvido nesse artigo demonstra que tal tecnologia pode ser uma alternativa viável para a construção de uma ferramenta para atender ao cenário proposto.

Por praticidade, os agentes do SMA foram implementados de forma semelhante. Em um sistema real, cada empresa poderia implementar seu(s) agente(s) de acordo com o seu modelo e políticas de negócio, usando diversos protocolos de negociação em um mesmo agente. Dessa forma, um agente poderia participar simultaneamente em negociações do tipo barganha, *Contract Net* e leilão, bastando que fosse programado para isso. Nesse caso seria essencial o seguinte: os agentes intermediários (Intermediários na negociação entre o cliente e os prestadores de serviço), aqui representados pelos agentes Gestor de Hotel e Gestor de Locadora, teriam que informar que tipo de protocolo de negociação está sendo utilizado antes de iniciar a negociação propriamente dita. Além disso, tais protocolos teriam que ser padronizados e de conhecimento dos demais agentes. Dessa forma, um agente poderia participar de vários tipos de negociações diferentes, desde que utilizasse os protocolos padronizados.

Para os agentes intermediários também é valido o uso de vários protocolos de negociação. Tais agentes poderiam ser implementados pelas Secretarias de Turismo locais. Um agente desse tipo poderia mudar de protocolo de negociação segundo a demanda por determinado serviço, como por exemplo, utilizar um protocolo que maximize o lucro em época de maior demanda pelo serviço.

Diferente da implementação do SMA apresentada nesse artigo, um único agente intermediário poderia lidar com vários tipos de serviços diferentes, como por exemplo, um mesmo agente intermediário servir para negociações com hotéis, bares e restaurantes, etc. Dessa maneira é possível perceber que a tecnologia de SMA's pode ser utilizada para a construção de SMA's muito mais complexos que o demonstrado no artigo. Porém, deve-se sempre levar em conta que um agente intermediário lida com dezenas de agentes simultaneamente e o fator escala sempre deve ser levado em conta para não sobrecarregar a infraestrutura de comunicação e a capacidade de processamento do agente.

Outra característica importante desse tipo de sistema é que a complexidade de um SMA pode crescer, mas isso fica transparente para os usuários. Um cliente procurando serviços não precisa saber dos detalhes de implementação dos agentes intermediários, basta que os agentes envolvidos utilizem protocolos padrão de negociação e uma mesma ontologia de serviços (mesma definição para nomes e detalhes dos serviços). Essa transparência se estende ao longo do SMA, uma vez que cada um dos atores (Secretaria de Turismo, hotéis, restaurantes, locadoras de automóveis, bares, etc.) vai cuidar da implementação do seu próprio agente sem precisar conhecer os detalhes de funcionamento do restante dos agentes do SMA.

No SMA de exemplo, o sistema ficou centralizado nos dois agentes intermediários (agente Gestor de Hotel e agente Gestor de Locadora de Automóveis), mas não necessariamente precisa ser dessa forma. Um SMA mais completo poderia contar com diversos gestores diferentes representando regiões distintas geograficamente. Dessa maneira seria possível que um agente de busca mais especializado procurasse por um conjunto de serviços e informasse ao cliente em qual região/estado seria mais vantajoso passar as férias. A plataforma JADE permite a comunicação com diversos SMA's distribuídos.

Dentro do cenário proposto nesse artigo, as Secretarias de Turismo poderiam fomentar a criação de um catálogo eletrônico de serviços turísticos que abrangesse todos seus associados. Dessa maneira, agentes poderiam integrar seus serviços em um único/ou vários SMA's de serviços turísticos.

Sistemas mais sofisticados lidam com APIs e tecnologias que abstraem máquinas de inferência, protocolos de interação e estruturas de informações. Algumas delas foram utilizadas nesse trabalho. Porém, tais ferramentas apresentam um conjunto bem maior de funcionalidades habilitando SMA para lidar com cenários bem mais complexos que o apresentado aqui.

REFERENCES

[1] SEBRAE, "Copa 2014 - oportunidades e desafios," 2011.

[2] E. S. d. S. Zagheni and M. M. M. Luna, "Canais de distribuição do turismo e as tecnologias de informação: um panorama da realidade nacional," *Revista Produção Online*, vol. 11, pp. 476-502, 2011.

[3] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, "Title," unpublished|.

[4] R. H. BORDINI, J. F. HUBNER, and M. WOOLDRIDGE, *Programming Multi-Agent Systems in AgentSpeak using Jason*. University of Liverpool, UK, 2007.

[5] A. S. Rao and M. P. Georgeff, "An Abstract Architecture for Rational Agents," in *Principles of knowledge representation and reasoning: proceedings of the third international conference (KR'92)*, 1992, p. 439.

[6] R. H. Bordini and R. Vieira, "Linguagens de Programação Orientadas a Agentes: uma introdução baseada em AgentSpeak (L)," *Revista de informática teórica e aplicada. Porto Alegre. Vol. 10, n. 1 (2003), p. 7-38*, 2003.

[7] O. Z. Akbari, "A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance," *J. Comput. Engg. Res*, vol. 1, pp. 14-28, 2010.

[8] J. Mylopoulos, M. Kolp, and J. Castro, "UML for agent-oriented software development: The Tropos proposal," in *≪ UML≫ 2001—The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, ed: Springer, 2001, pp. 422-441.

[9] D. Bertolini, A. Novikau, A. Susi, and A. Perini, "TAOM4E: an Eclipse ready tool for Agent-Oriented Modeling. Issue on the development process," *University of Trento, Trento, Italy*, 2006.

[10] M. Silva, "U-TROPOS: uma proposta de processo unificado para apoiar o desenvolvimento de software orientado a agentes," Dissertação. Recife, 2008-Universidade Federal de Pernambuco, 2008.

[11] F. Bellifemine, G. Caire, and T. Trucco, "Jade Programmer's Guide. Java Agent Development Framework (2010)," ed.

[12] N. F. Noy, M. Crubézy, R. W. Fergerson, H. Knublauch, S. W. Tu, J. Vendetti, *et al.*, "Protege-2000: an open-source ontology-development and knowledge-acquisition environment," in *AMIA Annu Symp Proc*, 2003, p. 953.

[13] E. Friedman-Hill, "Jess, the rule engine for the java platform," ed, 2003.

[14] D. L. Candeia, "Classificação dos meios de hospedagem," 2004.

[15] E. J. Friedman-Hill, "Jess, the java expert system shell," *Distributed Computing Systems, Sandia National Laboratories, USA*, 1997.

A multiagent approach for detecting and mitigating DDoS attacks

João P. A. Pereira, Marcos A. Simplicio Jr.,Anarosa A. F. Brandão
Dept. of Computer and Digital Systems Engineering
Escola Politecnica, Universidade de São Paulo (USP), São Paulo/SP 05508–010
Email: {raijoma,mjunior}@larc.usp.br, anarosa.brandao@poli.usp.br

***Abstract*—This paper describes Arquitena, a multiagent approach for detecting and mitigating DDoS attacks in Internet Services Providers (ISP) networks. Arquitena's main property is to identify situations that characterize attack scenarios, such as a large stream of packets directed to a network service or equipment. This is accomplished by using a virtual network of agents that mirrors the actual network infrastructure, which tends to facilitate the detection of attack routes, identification of malicious traffic and protection of hypothetical victims. Together with the system's description and rationale, we describe our preliminary prototype that will be employed for its evaluation.**

***Keywords*—multiagent systems; distributed denial of service attacks; detection; mitigation; internet service providers.**

I. Introduction

Modern society is increasingly dependent on the Internet services, including the speed of the links and data availability. Internet access by companies and final users is usually done indirectly, through the infrastructure of an Internet Service Provider (ISP) that charges for this service. This dependence puts great responsibility upon the ISP, since any network unavailability can seriously affect the transactions of many of the ISP's clients (e.g., causing financial and image losses). One example of threat against network availability are the so-called Distributed Denial of Service (DDoS). Albeit diverse in nature (for a taxonomy, see [12]), such attacks usually involve a large number of entities that, by directing a large volume of traffic to a single target, compromise its capability of providing services to legitimate users.

DDoS attacks are currently among the most serious threats to the infrastructure of critical services from an ISP [1]. This is especially troubling considering that they have been gradually growing in number and volume in the last few decades [1]. In addition, among the many forms of DDoS techniques, very few can be handled by the victim alone. Aiming to tackle this issue, many detection and mitigating strategies against DDoS attacks have been conceived. This includes strategies that try to create a defense perimeter around the target [5] or rely on statistical analyzes to detect DDoS attacks [11]. A more recent technique to address this challenging scenario, however, is to employ Multiagent Systems (MAS). The advantage of using MAS is related to the difficulty of blocking DDoS attacks without using a highly distributed mechanism that covers all regions of the ISP's network [10]. In this context, using MAS becomes an interesting approach to deal with such problem, since it naturally deals with intensely dynamic environments.

The goal of this study is, thus, to employ a multiagent approach in the design of a system able to detect, mitigate and block DDoS attacks. The agents' intelligence allows them to react to changes perceived in their surrounding environment, such as a large stream of packets going to a same target (packet flooding). The resulting MAS solution consists of a self-coordinated group of agents that represent the ISP's network elements. These agents communicate among themselves and also with the underlying equipment, detecting unusual network patterns and acting accordingly. Among the characteristics aimed by Arquitena, the most fundamental are its ability to: (i) provide fast detection, (ii) identify most types of DDoS attacks, (iii) provide service continuity for legitimate traffic, (iv) achieve low false-positive and false-negative rates, and (v) impose a low operation overhead. The system runs on a simulation environment, called Delos, that is created according to the real network topology of the target ISP.

The rest of this document is organized as follows. Section II provides a brief overview of DDoS attacks. Section III presents in details the Arquitena system and the Delos simulation environment. The performance metrics to be used in Arquitena's evaluation are covered in section IV. Section V discusses the related work. Finally, section VI concludes the discussion and suggest ideas for future work.

II. Background: DDoS Attacks

A Distributed Denial of Service (DDoS) attack corresponds to an attempt to compromise the availability of some service, hindering or blocking completely its ability to handle legitimate users' requests [14, Chapter 7]. Specifically, such attacks attempt to exhaust some critical resource upon which the target service depends. For example, an attacker can use many machines to direct a large amount of spurious requests addressed to a specific target, effectively flooding the target's equipment (e.g., a server or an access router) with packets. The target equipment, unable to handle so many requests in such a short interval of time, may either collapse completely or start dropping a large portion of the received packets. Whichever the case, this affects the service provider's ability to respond to requests from legitimate users in a timely manner, thus preventing most (or all) legitimate users from accessing the service.

As shown in Figure 1, DDoS attacks are often accomplished by a large number of machines, which are recruited and controlled by some malicious entity and work in synchronization to attack a particular service or device. The participating machines typically do not belong to the attacker him/herself, but are controlled after its security is compromised (e.g., by

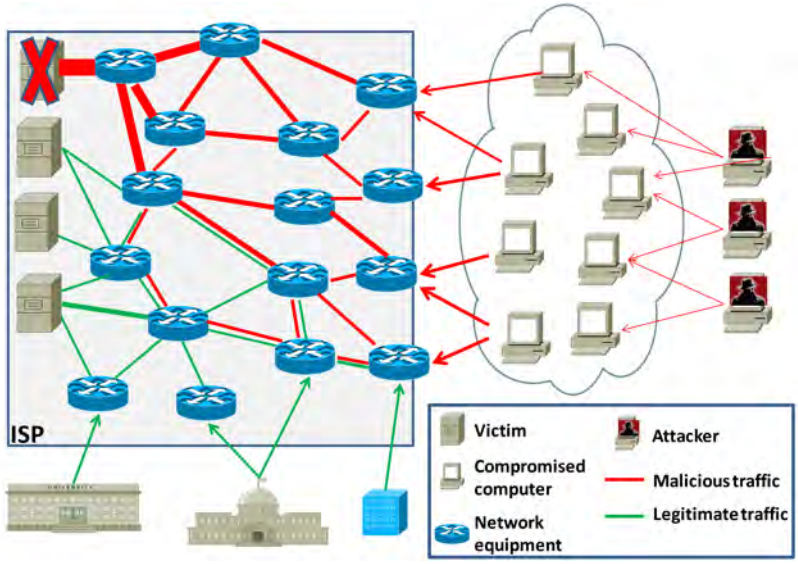

Fig. 1. A botnet-based DDoS attack. Thicker lines indicate a higher volume of traffic.

a virus or trojan horse), forming what is called a botnet [14, Chapter 6]. The higher the number of machines involved in the attack, the larger the traffic that reaches the target and, thus, the higher the chances of collapsing the service.

Preventing DDoS attacks is a challenging task, especially because it is difficult to differentiate legitimate from malicious packet flows. After all, in both cases the packets carry the IP address of the victim as their destination and they are all directed to a valid service; especially in the case of botnets, they also carry a valid IP as their source address. The main difference is, thus, in the *number* of packets in each flow, as an attacker is bound to send a huge amount of traffic to the victim in a short period of time.

Even if one is able to identify malicious packets, blocking their source while preserving legitimate traffic (e.g., by means of authentication policies, packet filtering and application of firewalls) remains a challenging issue. This happens because a malicious packet flow can only be effectively blocked if this is done near its point of origin, where the volume of traffic is still small enough to be handled with small impact over other (legitimate) packets. Otherwise, any piece of equipment trying to block the traffic would have to deal with the large amount of traffic that accumulate in the links near the intended victim, becoming itself vulnerable to a service failure due to excessive processing. Therefore, it is necessary to determine the path of malicious packet from the network's edge to the possible victim (the so-called "attack route"), applying filtering mechanisms on the relevant network equipment in a dynamic manner. Addressing this issue in a efficient manner is the main goal of this article, which proposes an intelligent orchestration mechanism that relies on holistic and updated information about the state of the network for preventing malicious packets from reaching their target.

III. A MAS FOR DETECTING, MITIGATING AND BLOCKING DDOS ATTACKS: ARQUITENA

Arquitena is our proposal for dealing with DDoS attacks using a MAS approach. The underlying idea is to benefit from cooperation and coordination among Arquitena agents to monitor an ISP network, detect DDoS attacks, and mitigate such attacks by blocking the attackers. In order to do that, Arquitena adopts two basic types of agents, the external agent and the internal agent, whose instances run in a virtual environment (called Delos) that simulates the underlying network topology. In this section we describe the Arquitena agents, the way they interact and the environment where such interactions occur.

A. Arquitena agents

As already said, an Arquitena agent can be of one of two types: an external agent or an internal agent. The choice for their names is related to their position inside the ISP's network. The external agent has knowledge about the network topology and available resources, such as routers, firewalls, servers and load balancers, and is responsible for providing traffic information related to different pieces of network equipment to internal agents. Internal agents are responsible for monitoring network equipment while dealing with traffic information received from external agents. Such monitoring is what provides internal agents with the knowledge required for detecting and mitigating DDoS attacks. In addition, internal agents are further specialized into Minos, Adamantos and Eacos agents (see figure 2).

Minos agents are responsible for monitoring and protecting potential victims, i.e., equipment whose IP addresses are enclosed within the packets sent by attackers. Common examples are servers hosting services such as Web or DNS (Domain Name System), being connected to the wider Internet by the ISP. Adamantos agents are responsible for monitoring and protecting critical equipment at the network's core and also the network's entry points at the edge of the ISP's network. A piece of equipment is considered critical if it is

usually responsible for large volumes of traffic (e.g., because it is located at a point of confluence), meaning that (1) it concentrates a lot of information about the network conditions and (2) applying filters to it can considerably reduce the traffic load at the victim. The importance of the network's entry points, on the other hand, comes from the fact that they are expected to become the final and long-living location of the filtering mechanisms applied by Arquitena. Although critical equipment at the network core might also be considered potential victims, Minos agents differ from Adamantos agents because the latter are also responsible for building attack routes and applying filtering mechanisms whenever necessary. Finally, Eacos agent behave similarly to Adamantos agents, but are responsible for less critical equipment at the network's core and, for this reason, are designed to be inactive most of the time. This "sleeping behavior" is part of an strategy to avoid communication and processing overhead while ensuring detection efficiency.

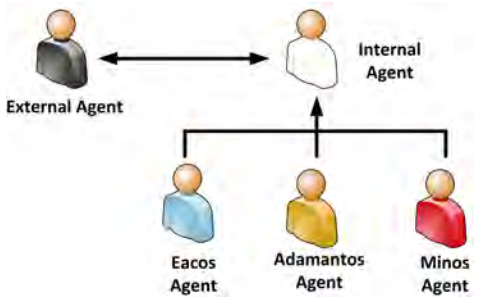

Fig. 2. Arquitena Agents

B. Interaction within Arquitena

Interaction between Arquitena agents provides coordination and learning within the system. External agents communicate with everyone in the system. They send information related to the network traffic to internal agents in order to update their knowledge about it. Therefore, internal agents learn about legitimate and malicious traffic based on information received from external agents, identifying high volumes of traffic addressed to a same target. Also, active internal agents periodically request such information from external agents, characterizing a two-way communication channel.

The language of communication employed is the Knowledge Query and Manipulation Language (KQML) [6], while the Arquitena system itself is developmed using Jason [3] and Java. The interoperability of the system with the (non-Jason) external agent can be resolved by developing an Internal Action with Java and Jason as well. This allows the creation of a wrapper for encapsulating the external agent, enabling it to communicate via KQML.

The learning algorithm adopted in Arquitena is the Random Forest [4], which operates by constructing a large number of decision trees and provides outputs generated by individual trees. This algorithm does not require data normalization, allows lots of data to be analyzed with little computational resources and enable model validation through statistical tests. Its application in the system should allow agents them to learn the average rate of packets (FPav) that nornally pass through the monitored equipment. In this manner, the Arquitena system can be considered truly intelligent, since its agents contantly learn about the traffic characteristics and this knowledge becomes mote accurate with time. The decision and learning models applied are consistent with the characteristics of each agent, which stores data and predict suitable actions.

Interaction between internal agents occur in two ways: (i) Minos agents interact with every internal agent, since it must inform the IP address of the equipment under its responsibility; and (ii) other internal agents interact with their neighbors. Since each internal agent is responsible for a specific equipment, internal agents A and B are said to be neighbors if there is a direct network link connecting the two pieces of equipment monitored by them. Such interaction provide neighbors with information related to their location and, more importantly, their current traffic flow. In addition to such information, Eacos agents must be activated whenever one of their neighbors perceive any possibility of malicious traffic flow coming from its neighborhood.

Coordination is also achieved based in the information exchanged during interaction among agents. In special, each internal agent periodically queries the external agents for traffic statistics corresponding the equipment being monitored by it. This allows internal agents to update the value of a threshold for each potential victim, accommodating natural changes on network traffic, and also to detect when the threshold is exceeded. If that happens often enough, this may indicate a DDoS attack and, thus, internal agents activates all Eacos agents in their neighborhood. Eacos agents activated in this manner may fall back to the inactive state if there is not enough traffic passing by the equipment monitored by it, or may activate other neighboring agents in a chain reaction. Eventually, Adamantos agents near attackers are triggered at the network's edge, meaning that the attack route is complete and, hence, the DDoS attack is fully detected.

Simultaneously to the above detection process, the agents also cooperatively follow specific plans aiming to prevent an excessive amount of traffic from reaching the victim during the attack, thus avoiding the disruption of that equipment's service. This is accomplished by having the agents configure (temporary) filtering mechanisms in the monitored equipments, blocking network traffic directed to the victim, something that could not be easily done manually. As these filters move toward the edge of the network where the attackers actually are, the legitimate traffic becomes less and less affected. As a final remark, we notice that the system must be manually initialized with some basic information that cannot be learned otherwise. This includes the maximum flow of packets (FPmax) that each device can support, a security parameter that depends on the specific equipment being protected. Another example is the position of the agents, the connections between them and the target equipment each of them monitor, which depend on the actual layout of the network where the Arquitena system is deployed and which are the nodes that need protection. Since this layout is not expected to change often, in principle Arquitena does not need to include mechanisms to deal with the dynamic entrance and exit of agents. Therefore, the agents are placed in fixed and permanent positions. Nonetheless, Arquitena shows some dynamism in the sense that any active internal agent may, depending on the state of the environment,

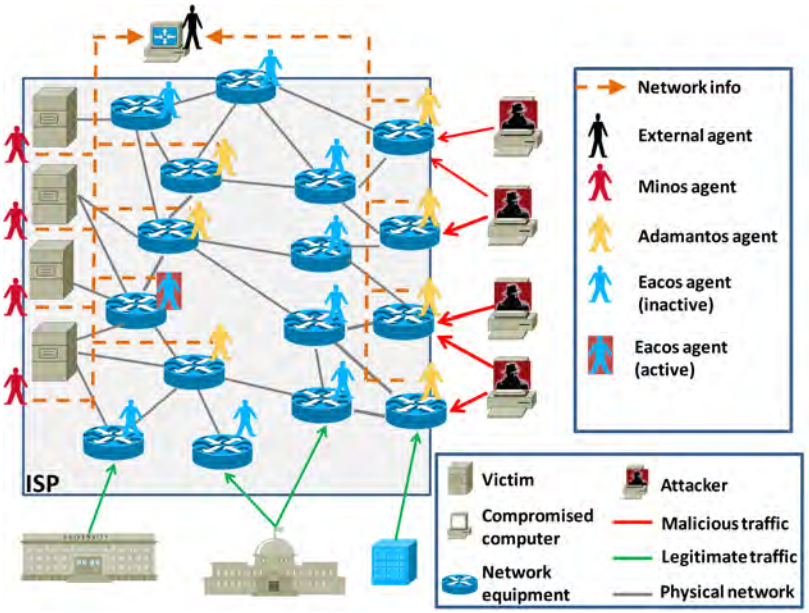

Fig. 3. Arquitena system and agents

enable Eacos agents during the system's mitigation process.

C. The Delos environment

Although Arquitena could in principle be implemented in a completely distributed manner, the platform being used for implementing the system is a centralized hardware (disk, CPU and memory) where all agents are executed. This facilitates the communication between agents and, at the same time, does not require modifications on the existing network equipment being monitored.

The network itself is simulated using a simulation environment, called Delos, which handles the agents' requests and message exchanges, as well as the dynamic changes in their surroundings. Delos implements a grid that maps the ISP topology and, thus, must be configured using information from the position of the network elements and their interconnections. Aiming to facilitate interpretation by humans, Delos provides a visual interface with a color system that identifies different network elements and their current status (see Figure 4): green for inactive agents (equipment not being monitored); blue for active agents monitoring a piece of equipment under normal traffic; red for active agents monitoring equipment with anomalous traffic; gray for active agents monitoring equipment that have an active access control list applied to them due to anomalous traffic.

In its current state, Delos is still a quite simple prototype that allows for communication between agents and a few types of network equipment. Its final implementation, however, should give support to the following components, some of which are depicted in Figure 4: (i) routers (oval form); (ii) firewalls (hexagon form); (iii) servers (diamond form); (iv) load balancers (triangle form); (v) Minos, Eacos and Adamantos agents associated with these objects. Network elements that have no connections with their neighbors in the grid are separated by black squares (empty cells in a grid).

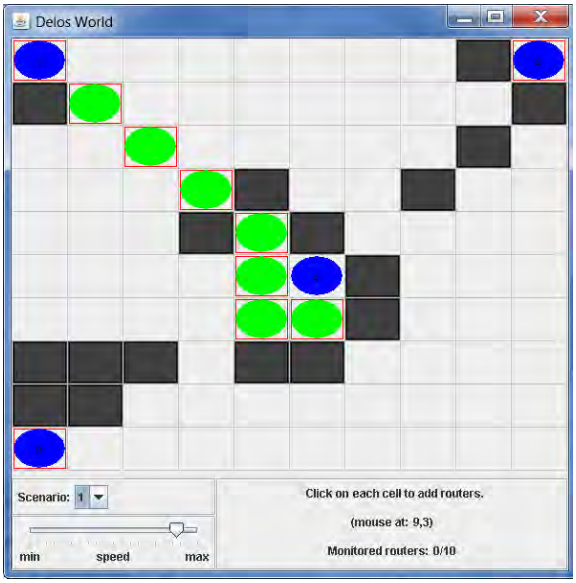

Fig. 4. Simulation Environment: Delos

Like Arquitena, Delos is developed using Jason [3] and Java.

IV. Analysis

The main functional requirements of the Arquitena system are: (1) the ability to accurately detect a situation of DDoS attack; (2) mitigate the attack after its detection, blocking enough malicious traffic flow along the network so that the amount of malicious packets reaching the target are not enough to compromise its service; and (3) preserve as much as possible the legitimate network traffic, whether or not it is directed to the victim of the DDoS attack.

These requirements are expected to be fulfilled as a direct consequence of the sparse distribution of agents monitoring the environment in a ubiquitous manner, covering diverse choices of agents and victims. As the agents learn the traffic distribution and normal thresholds of the protected network, they can notice abnormal behavior caused by different types of DDoS attacks. When that happens, they can act accordingly, building attack routes and placing barriers for the malicious traffic on the relevant network elements. Furthermore, the fact that the system allows some agents to remain inactive enables the network administrator to balance the solution's usage of computational resources, such as CPU time and memory.

It is important to notice that, as any general purpose solution that independs on the protocols employed by the attacker, Arquitena is unable to mitigate attacks that explore specific vulnerabilities in the target system. Nonetheless, if desired, the system can used in combination with solutions for preventing speficic attacks on the service-side, such as Intrusion Detection/Prevention Systems (IDS/IPS). The advantage of using Arquitena in this scenario is that it should prevent most of the malicious packets from reaching their intended victims, reducing the overall load of the companion IDS/IPS.

The experimental evaluation of Arquitena's effectiveness is still an ongoing work. Among the metrics to be considered, we can cite: the number of packets that arrive at the victim per time interval, which indicates how well the system blocks malicious traffic and allows legitimate packets through; the time required for the system to detect an attack situation; the time required for the system to block the attack, inserting the appropriate filtering mechanisms at the network's edge; the rate of false-positives, i.e., situations in which a (large) growth of legitimate traffic is misinterpreted as an attack; and the rate of false-negatives, in which an actual attack is not detected. The benchmark scenario beign built is composed by emulated routers and injector package software. In this manner, we will be able to create controlled DDoS attacks and assess the efficiency of the Arquitena system in detecting and mitigating them.

V. Related Work

The application of MAS for the detection and mitigation of DDoS attacks is not something new in the literature. One example is the work proposed in [2], which employs multiple specific agents having a single function: to detect DDoS attacks by means of pattern recognition mechanisms. The result is a solution with a high assertiveness rate in the detection of DDoS attacks. The cost to be paid, however, is high communication overhead depending on the amount of agents involved, since those agents communicate constantly to outline the attack flow. In comparison, Arquitena tries to keep active only the essential agents for the detection of a DDoS attack, dynamically activating relevant agents when (and only when) necessary for further mitigation. This reduces the overall system's burden in terms of processing and communication. A related solution also appears in [7], which describes a MAS-based learning system focused on intrusion detection, covering, among others threats, DDoS attacks. The learning process relies on different sources of raw data, including network traffic, operating system and application activities, which are then combined. One shortcoming of this approach is that the origin of this data is both distributed and heterogeneous, which may lead to inconsistencies in different parts of the system. Arquitena tries to overcome this issue by making an external agent responsible for all data collection, providing a more homogeneous picture of the real network traffic as required by each different agent. Another important limitation of [7] when compared to Arquitena is that the former only takes advantage of the multiple agents for identifying DDoS attacks, while no mitigation mechanism is provided. The combination of detection and mitigation of DDoS attacks using MAS is explored by Juneja et al. [8] and by Singh et al. [13]. Both frameworks use the multiple agents dispersed all over the network to verify a packet's authenticity via source IP validation, a process by which the system determines whether its source corresponds to a real machine. Packets detected as forged are then blocked. However, the very fact that they rely on source IP validation for discerning legitimate traffic is probably the main limitation of this strategy, since this method is ineffective against attacks based on botnets, which are quite common nowadays [9]. The system proposed in [13] is also limited to mitigate attacks from one type of transport protocol (namely, UDP) and provides no mechanism for tracing the route taken by forged packets. Even though [8] does allow attack routes to be traced, potentially admitting more effective mitigation, its own authors admit that the number of agents required to get optimal results is something that needs further investigation [13]. Arquitena, on the other hand, was designed to be generic enough to detect attacks based on any transport-layer protocol, relying on network patterns rather than IP addresses for determining the attack routes.

VI. Conclusions

DDoS attacks have become a major threat to ISPs in the last few decades [1].

Aiming to tackle this issue, we described Arquitena, a system for detection and mitigation of DDoS attacks based on large amounts of traffic. The system is composed of multiple collaborative agents that, without manual intervention of the ISP network's administrators, detect the attack route and gradually deploys barriers to the malicious packets, pushing the defense perimeter toward the network's edge (nearer to the attackers). By covering all regions of the ISP network and constantly learning the traffic patterns in that network, Arquitena can be used as a powerful tool against DDoS attacks, ensuring the availability of the ISP network's resources to real users.

Arquitena is currently under development, and its effectiveness in preventing DDoS attacks will be evaluated both individually and in association with traditional network security solutions (e.g., Intrusion Detection/Prevention Systems). Our goal is to feed the agents with real traffic traces collected from an ISP and then simulate different attack scenarios and assess the system's behavior in each of them.

Acknowledgment

This work was in part supported by the São Paulo Research Foundation (FAPESP) under grants 2010/20620-5 and 2011/21592-8.

REFERENCES

[1] Arbor. Worldwide infrastructure security report. http://www.arbornetworks.com/research/infrastructure-security-report, 2012.

[2] Z. Baig and K. Salah. Multi-agent pattern recognition mechanism for detecting distributed denial of service attacks. *IET Information Security*, 4(4):333–343, 2009.

[3] R. Bordini, J. Hubner, and M. Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*. Wiley, 1th edition, 2007.

[4] L. Breiman. Random forests. In *Machine Learning Journal, Vol. 45, Issue 1*, pages 5–32. Kluwer Academic Publishers, 2001.

[5] S. Chen and Q. Song. Perimeter-based defense against high bandwidth ddos attacks. *IEEE Trans. Parallel Distrib. Syst.*, 6(16):526–537, 2005.

[6] Tim Finin, Jay Weber, Gio Wiederhold, Mike Genesereth, Rich Fritzson, Don McKay, Stu Shapiro, Jim McGuire, Richard Pelavin, and Chris Beck. Specification of the kqml agent-communication language, 1994.

[7] V. Gorodetski, I. Kotenko, and O. Karsaev. Multi-agent technologies for computer network security: Attack simulation, intrusion detection and intrusion detection learning. *Int. J. of Computer Science Systems Science & Engineering*, 2003.

[8] D. Juneja, R. Chawla, and A. Singh. An agent-based framework to counterattack ddos attacks. *Int. J. of Wireless Networks and Communications*, 1(2):193–200, 2009.

[9] Z. Mao, Vyas Sekar, Oliver Spatscheck, Jacobus van der Merwe, and Rangarajan Vasudevan. Analyzing large ddos attacks using multiple data sources. In *Proc. of the 2006 SIGCOMM workshop on Large-scale attack defense*, LSAD '06, pages 161–168, NY, USA, 2006. ACM.

[10] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher. *Internet Denial of Service - Attack and defense mechanisms*. Prentice Hall, 1th edition, 2004.

[11] Y. Ohsita, S. Ata, and M. Murata. Detecting distributed denial-of-service attacks by analyzing TCP SYN packets statistically. In *IEEE GLOBECOM'04*, volume 4, pages 2043–2049, 2004.

[12] Riorey. Riorey taxonomy of ddosattacks. http://www.riorey.com/x-resources/2011/RioRey_Taxonomy_DDoS_Attacks_2.2_2011.pdf, 2011.

[13] A. Singh and D. Juneja. Agent based preventive measure for UDP flood attack in DDoS attacks. *Int. J. of Engineering Science and Technology*, 2:3405–3411, 2010.

[14] W. Stallings and L. Brown. *Computer Security: Principles and Practice (2nd ed.)*. Pearson, 2011.

A BDI-Fuzzy Agent Model for Exchanges of Non-Economic Services based on the Social Exchange Theory

Giovani P. Farias, Graçaliz P. Dimuro
Programa de Pós-Graduação em Computação
Universidade Federal do Rio Grande
Rio Grande, Brasil, Email: giovanifarias@gmail.com

Glenda Dimuro, Esteban de Manuel Jerez
Depto de Expresión Gráfica Arquitectónica
Universidad de Sevilla
Sevilla, Espanha, Email: glenda.dimuro@gmail.com

***Abstract*—The purpose of this work is to develop a BDI-Fuzzy agent model for the Jason platform, with abilities to assess qualitatively, subjectively the social exchanges values originated in the provision and in the receipt of non-economic services, based on Piaget's theory of social exchanges. An application to the simulation os exchange processes in a social organization, namely, the urban vegetable garden San Jerónimo (Seville, Spain) is presented.**

***Keywords*—Multiagent Systems; Social Exchanges; BDI Architecture; Fuzzy Logic;**

I. Introdução

Este trabalho situa-se no contexto das áreas de Sistemas Multiagentes (SMA) [1] e de Simulação Social [2] baseada em agentes, explorando temas relacionados à arquitetura de agentes BDI (*Beliefs, Desires and Intentions*) [3] e à Lógica Fuzzy [4], com o objetivo geral de experimentar um modelo de agente BDI-Fuzzy, implementado na plataforma de agentes Jason [5], para trocas de serviços não-econômicos baseados na Teoria das Trocas Sociais de Piaget [6].

A teoria das trocas sociais de Piaget tem sido utilizada como base para a análise de interações em sistemas multiagente, onde estas interações são compreendidas como processos de trocas de serviços, entre pares de agentes, seguidas da avaliação destes serviços por parte dos agentes envolvidos, produzindo valores de trocas sociais, cuja natureza é qualitativa, subjetiva, imprecisa, vaga, ambígua, incompleta.

A arquitetura de agentes BDI é baseada em estados mentais, tendo sua origem na teoria de raciocínio prático humano. O caráter intencional do modelo BDI mostra-se adequado para o problema abordado neste trabalho. Entretanto, observa-se que o tratamento da incerteza gerada em ambientes/problemas de informação imprecisa envolvidas em trocas de serviços não-econômicos não está previsto, em geral, na arquitetura BDI.

Por outro lado, a Lógica Fuzzy, utilizada para a modelagem de raciocínio aproximado e vago, permite descrever de forma efetiva as características de sistemas complexos ou que não podem ser definidos de forma exata, pois, na Lógica Fuzzy, os relacionamentos entre elementos e conjuntos seguem uma transição entre pertinência e não pertinência que é gradual, representados por valores de pertinência intermediários entre o verdadeiro e o falso da lógica clássica.

Neste trabalho, com base na teoria das trocas sociais de Piaget, desenvolveu-se um modelo híbrido de agente BDI-Fuzzy, na plataforma de agentes Jason, com habilidades para avaliar de forma qualitativa/subjetiva os valores de trocas sociais originados na prestação e no recebimento de serviços não-econômicos, onde considera-se que os agentes assumem diferentes critérios com relação aos atributos que vão utilizar na avaliação do serviço (seja por parte de quem o realiza ou de quem o recebe), o que os induz a diferentes *"atitudes de avaliação de serviço"* e os agentes também podem assumir diferentes estratégias referentes às trocas que optam por realizar.

II. Trabalhos Relacionados

Na literatura existem várias referências ao uso da Lógica Fuzzy para possibilitar aos agentes mecanismos de decisão mais adaptáveis à realidade, podendo ter maior flexibilidade em ambientes complexos e dinâmicos. Nos trabalhos de simulação social baseada em agentes em [7], [8], facetas e traços de personalidades humanas foram especificadas como regras condicionais em agentes fuzzy (que são capazes de executar raciocínio aproximado qualitativo), para realizar simulação do comportamento humano. Já em [9], a Lógica Fuzzy foi utilizada para avaliação de trocas sociais entre agentes baseados em personalidades, propondo a análise das interações entre agentes com base na noção de equilíbrio fuzzy em trocas de serviços entre agentes.

No contexto de agentes BDI, nos trabalhos em [10], [11] foi proposto um modelo geral para agentes BDI graduado, através de uma arquitetura baseada em sistemas multi-contextos, que admite atitudes mentais graduadas, em sentido similar ao da Lógica Fuzzy. Uma arquitetura BDI Fuzzy para agentes sociais foi proposta em [12], com uma proposta inicial para a modelagem de sociedades cooperativas de agentes, apontando para as condições sociais necessárias para os agentes formarem intenções e ações conjuntas. Outra extensão ao modelo BDI com características fuzzy é o modelo *Agent Fuzzy Decision-Making* (AFDM) [13], que permite que agentes BDI possam tomar decisões com base em julgamentos quantificados de forma fuzzy.

Finalmente, com relação a Teoria dos Valores de Trocas Sociais proposta em [6] que têm sido utilizada como base para aplicações em Sistemas Multiagente [14], onde, utiliza-se valores de natureza qualitativa, que representam conceitos subjetivos, porém a representação computacional destes valores

de trocas, ou de qualquer critério subjetivo, não é trivial. Os trabalhos [15], [16] incluem uma metodologia para avaliação de serviços em processos de trocas sociais. Em [9] foi proposta uma abordagem baseada na Lógica Fuzzy para a avaliação dos valores de trocas materiais (investimento e satisfação) gerados nos dois estágios de trocas sociais, com aplicação em sistemas multiagentes baseados em personalidades. Para tanto, introduziu-se uma definição de serviço para cada um dos agentes envolvidos na troca. Através da definição de fatores de personalidades, adotou-se um critério que possibilitou obter agentes com vários traços de personalidades, apesar de no trabalho constar uma generalização para apenas três traços de personalidades.

III. O Modelo de Trocas Sociais

Segundo a teoria de Piaget, uma troca social entre dois agentes α e β é executada em dois tipos de estágios. Nos estágios de tipo $\mathrm{I}_{\alpha\beta}$, o agente α realiza um serviço para o agente β. Os valores de troca envolvidos neste tipo de estágio são os seguintes:

- $r_{\mathrm{I}_{\alpha\beta}}$: investimento realizado por α para a realização de um serviço para β;
- $s_{\mathrm{I}_{\beta\alpha}}$: satisfação de β com o serviço realizado por α;
- $t_{\mathrm{I}_{\beta\alpha}}$: débito de β para com α por sua satisfação com o serviços realizado por α;
- $v_{\mathrm{I}_{\alpha\beta}}$: crédito que α adquire de β por ter realizado o serviço.

Nos estágios de tipo $\mathrm{II}_{\alpha\beta}$, o agente α solicita a β a realização de serviço em pagamento pelo serviço realizado anteriormente (no caso em que α tem créditos), e os valores relacionados com este estágio de troca são análogos aos dos estágios de tipo $\mathrm{I}_{\alpha\beta}$. Os valores $r_{\mathrm{I}_{\alpha\beta}}$, $s_{\mathrm{I}_{\beta\alpha}}$, $r_{\mathrm{II}_{\beta\alpha}}$ e $s_{\mathrm{II}_{\alpha\beta}}$ associados imediatamente à uma troca realizada, são denominados de *valores materiais*. Os valores associados às trocas postergadas $t_{\mathrm{I}_{\beta\alpha}}$, $v_{\mathrm{I}_{\alpha\beta}}$, $t_{\mathrm{II}_{\beta\alpha}}$ e $v_{\mathrm{II}_{\alpha\beta}}$ são conhecidos como *valores virtuais*.

Em resumo, uma interação entre dois agentes/indivíduos pode ser interpretada como trocas de serviços. Quando um serviço é executado então é possível que este seja avaliado, tanto pelo prestador (valor de seu investimento), como pelo receptor (valor de sua satisfação). Posteriormente, são gerados valores de débito (do receptor) e de crédito (do prestador), que são levados em conta em trocas futuras.

A. Avaliação Fuzzy de Serviços

A realização de um serviço pelo agente α para o agente β implica na geração imediata dos valores materiais de investimento $r_{\alpha\beta}$ (de α que realizou o serviço à β) e de satisfação $s_{\beta\alpha}$(de β que recebeu o serviço de α).

Definição 1: Em um processo de troca social, um serviço é definido como uma tupla $\mathbb{S} = (a_1; \ldots; a_n)$, onde cada a_i, com $i \in \mathbb{N}$, é um atributo que representa um aspecto do serviço, a ser analisado no processo de avaliação dos valores materiais gerados pela realização de $\mathbb{S}$. Se o processo de avaliação envolve a análise do valor de investimento realizado por um agente α, então utiliza-se a notação $\mathbb{S}_r(\alpha)$. Se o processo de avaliação envolve a análise do valor de satisfação de um agente β, então utiliza-se a notação $\mathbb{S}_s(\beta)$.

O conjunto de atributos é dependente de uma aplicação específica, e pode variar se for considerada a avaliação do valor do investimento do agente que presta o serviço ou a satisfação do agente que recebe o serviço. A avaliação fuzzy de um serviço é realizada através da composição da avaliação de cada atributo que pertence a este serviço. Os atributos são representados por variáveis linguísticas, cujo valor é expresso qualitativamente por um termo linguístico e quantitativamente por uma função de pertinência.

Uma escala com termos linguísticos $T_1; \ldots; T_m$ é denotada por $T = \langle T_1; \ldots; T_m \rangle$, com $m \in \mathbb{N}$. Denota-se $T_k \in T$ para significar que o termo T_k está na escala T, ou seja $1 \leq k \leq m$. Para a avaliação de um atributo a utilizando uma escala é necessário proceder a um processo de normalização.

Definição 2: Seja $V(a)$ o valor medido do atributo a, N o limite superior de uma escala decrescente e $\max$ o valor limite tolerável para o atributo a, de acordo com o senso comum. Então, o valor normalizado do atributo a é denotado por $V_{nor}(a)$ e definido como:

$$V_{nor}(a) = \min\{N, V^{'}(a)\}, \;\; onde \;\; V^{'}(a) = \frac{V(a) \times N}{\max} \quad (1)$$

O valor normalizado do atributo é então avaliado em uma escala de valores fuzzy, obtendo então avaliação fuzzy do atributo, denotada por $\mu(a)$.

Considerando um serviço $\mathbb{S}_r = (a_1; \ldots; a_n)$ (ou $\mathbb{S}_s = (b_1; \ldots; b_n)$), então é possível obter um conjunto de regras condicionais através do cruzamento de resultados das avaliações fuzzy individuais de seus atributos, utilizando a regra de inferência MAX-MIN [17].

Seja $T^i = \langle T^i_1; \ldots; T^i_k \rangle$ uma escala para avaliação de um atributo a_i de um serviço $\mathbb{S}_r(\alpha) = (a_1; \ldots; a_n)$ onde o agente α presta um serviço para um agente β. Seja $T^r = \langle T^r_1; \ldots; T^r_m \rangle$ a escala para avaliação fuzzy do investimento $r_{\alpha\beta}$ por parte de α. Então a avaliação fuzzy do valor de investimento $r_{\alpha\beta}$ é determinada pela regra de inferência MAX-MIN aplicada sobre uma base de regras do tipo "**IF** ... **THEN**" do tipo:

IF a_1 **is** T^1_j **AND** a_2 **is** T^2_l **AND** $\ldots a_n$ **is** T^n_p **THEN** $r^{'}_{\alpha\beta}$ **is** $T^{'r}_q$

onde $T^1_j \in T^1; T^2_l \in T^2 \ldots T^n_p \in T^n; T^r_q \in T^r$. Na avaliação de uma regra, primeiramente avalia-se cada condição do tipo a_i **is** T^i_j, com $i = 1; \ldots; n$, como sendo $\mu_i(V_{nor}(a_i))$. A partir desses valores obtém-se a avaliação de $r^{'}_{\alpha\beta}$ **is** $T^{'r}_q$ como sendo $\min\{\mu_1(V_{nor}(a_1)); \ldots; \mu_n(V_{nor}(a_n))\}$.

O valor fuzzy de investimento $r_{\alpha\beta}$ é calculado a partir das avaliações de todas as regras deste tipo. Para cada termo T^r_v , com $v = 1; \ldots; m$, calcula-se o valor $\max\{T^{'r}_v, T^{''r}_v, \ldots, T^{\omega r}_v\}$, onde $\omega \leq k_1 \times \ldots \times k_n$, com k_i, sendo a cardinalidade da escala T^i de avaliação de cada atributo a_i. Estes valores provocam um corte no termo linguístico T^r_v e portanto uma região fuzzy em T^r. Nessa região é aplicado um método de defuzzificação, por exemplo o centróide [17], para se obter o valor fuzzy de investimento $r_{\alpha\beta}$. De forma análoga se obtém o valor fuzzy da satisfação $s_{\beta\alpha}$ do agente β pelo recebimento do serviço realizado por α.

IV. Modelo BDI-Fuzzy Genérico

O modelo BDI-Fuzzy pode ser descrito como um Sistema Baseado em Regras Fuzzy (SBRF) acoplado a base de crenças do agente BDI, conforme apresentado na Figura 1.

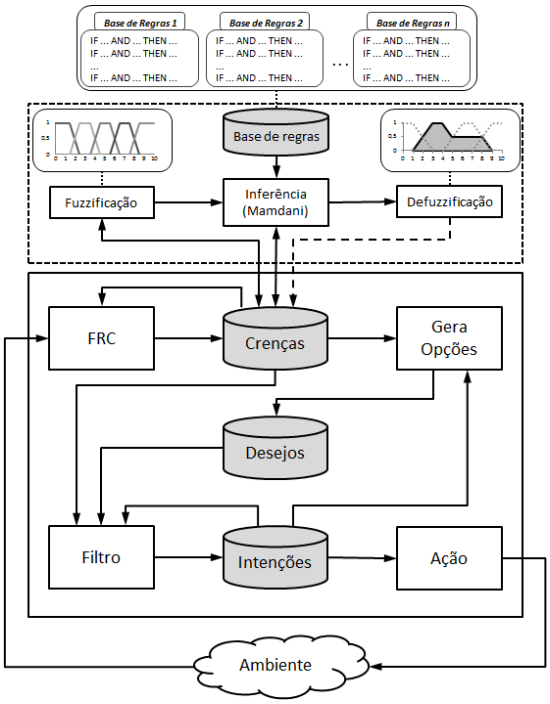

Fig. 1. Arquitetura BDI-Fuzzy Genérica.

O processo de **fuzzificação**, neste caso, utiliza funções de pertinência pré-estabelecidas, através das quais as entradas do sistema (presentes na base de crenças do agente BDI) são traduzidas em conjuntos fuzzy em seus respectivos domínios, ou seja, o fuzzificador mapeia cada variável de entrada do sistema em graus de pertinência de algum conjunto fuzzy que representa a variável em questão. Neste modelo, os valores fuzzificados podem simplesmente serem acrescentados à base de crenças do agente, para posteriormente serem utilizados em tomadas de decisões ou na máquina de inferência fuzzy para obter novas proposições fuzzy.

A **base de regras** é composta por uma coleção de proposições fuzzy na forma **IF**...**THEN**... descrevendo assim, as relações entre as variáveis linguísticas, para serem utilizadas na máquina de inferência fuzzy. Este componente, juntamente com a máquina de inferência, pode ser considerado o núcleo dos sistemas baseados em regras fuzzy, o qual, pode ser utilizado para acrescentar diferentes características entre os agentes BDI-Fuzzy, principalmente relacionadas a processos de avaliação e decisão.

A **máquina de inferência fuzzy** por meio das técnicas de raciocínio aproximado, traduz matematicamente cada proposição fuzzy, sendo de fundamental importância para o sucesso do sistema fuzzy, já que fornece a saída a partir de cada entrada fuzzy e da relação definida pela base de regras. Neste modelo, os agentes BDI-Fuzzy utilizam o método de inferência de Mamdani, onde: uma regra IF (antecedente) THEN (consequente) é definida pelo produto cartesiano fuzzy dos conjuntos fuzzy que compõem o antecedente e o consequente da regra. O método de Mamdani agrega as regras através do operador lógico "or", que é modelado pelo operador máximo (t-conorma $\vee$) e, em cada regra, o operador lógico "and" é modelado pelo operador mínimo (t-norma $\wedge$). Conforme as seguintes regras:

$$R^{(1)}\text{: } \textbf{IF } x \textbf{ is } A_1 \textbf{ and } y \textbf{ is } B_1 \textbf{ THEN } z \textbf{ is } C_1$$
$$R^{(2)}\text{: } \textbf{IF } x \textbf{ is } A_2 \textbf{ and } y \textbf{ is } B_2 \textbf{ THEN } z \textbf{ is } C_2$$

A **defuzzificação** é um processo de se representar um conjunto fuzzy por um número real. Em sistemas fuzzy, em geral a saída é um conjunto fuzzy. Assim, devemos escolher um método para defuzzificar a saída e obter um número real que a represente. Neste modelo, o agente BDI-Fuzzy e capaz de lidar com as duas saídas do sistema, sendo ela um conjunto fuzzy ou um número real. Desta forma, o processo de defuzzificação não é obrigatório, podendo ser utilizado para agregar informações à base de crenças do agente.

A. Atitudes de Avaliação de Serviço

A avaliação fuzzy de um serviço por um agente é realizada através da composição da avaliação de um, dois ou mais atributos que pertencem a esse serviço. Os atributos do serviço são representados por variáveis linguísticas, cujo valor é expresso qualitativamente por um termo linguístico e quantitativamente por uma função de pertinência.

A Figura 2 apresenta a arquitetura do sistema de avaliação fuzzy de serviço do agente BDI-Fuzzy, onde este obtém os atributos de um serviço $\mathbb{S}(a_1, \ldots, a_n)$, esses atributos passam por um processo de normalização, para se adequarem a uma escala padrão, sendo então escolhidos os atributos a serem considerados na avaliação fuzzy do serviço. Os atributos selecionados passam por um processo de fuzzificação para obter a sua representação qualitativa, através dos seus termos linguísticos. As combinações dos termos linguísticos ativam uma dentre as várias bases de regras disponíveis, as quais são baseadas no método de inferência de Mamdani. Como resultado final, obtém-se o valor fuzzy do serviço (valor fuzzy do *investimento* ou da *satisfação*), o qual pode ser defuzzificado.

B. Balanço Material e Virtual

Em uma sociedade, toda ação ou reação de um indivíduo, avaliado segundo sua escala pessoal, repercute necessariamente sobre os outros indivíduos. Logo, ocorre uma troca na qual cada ação (real ou virtual) do primeiro provocará uma ação de volta (real ou virtual) por parte dos outros indivíduos [18]. A dinâmica da troca provoca uma variação nos valores dos indivíduos, que pode ser positiva (i.e., satisfação, lucro), negativa (i.e., prejuízo) ou nula (i.e., equilíbrio).

O ***balanço material*** de um agente α, no modelo BDI-Fuzzy, é obtido através de uma avaliação fuzzy (qualitativa) influenciada pelos valores materiais de *investimento* (r_α) e *satisfação* (s_α), avaliados segundo a escala pessoal de α, gerados na troca de serviços com um outro agente. Deste modo, o *balanço material* representa uma avaliação pessoal do agente α, baseada nos valores materiais (r_α) e (s_α), a qual indica se a dinâmica da troca provoca uma variação *"possitiva"* (α recebe um serviço cuja *satisfação* é significativa), *"negativa"* (α realiza um serviço cujo *investimento* é significativo) ou *"nula"* (*investimento*/*satisfação* de α na realização/recebimento de um serviço é insignificante) nos valores materiais de α.

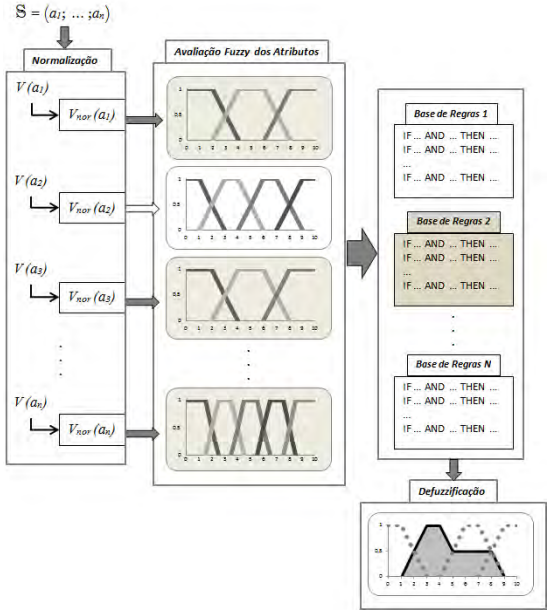

Fig. 2. Arq. do sistema de avaliação fuzzy de serviço do agente BDI-Fuzzy

O ***balanço virtual*** de um agente α, no modelo BDI-Fuzzy, é obtido através de uma avaliação fuzzy (qualitativa) influenciada pelos valores virtuais de *crédito* (v_α) e *débito* (t_α), gerados na troca de serviços com um outro agente. Deste modo, o balanço virtual representa uma avaliação do agente α, baseada nos valores virtuais (v_α) e (t_α), a qual indica se a dinâmica da troca provoca uma variação *"positiva"* (α realiza um serviço e adquire crédito significativo), *"negativa"* (α recebe um serviço cujo *débito* é significativo) ou *"nula"* (crédito/débito de α na realização/recebimento de um serviço é insignificante) nos valores virtuais de α.

V. Estudo de Casos

As simulações foram realizadas em uma ambiente de uma **horta** urbana, com objetivo de analisar o comportamento dos diferentes tipos de agentes hortelãos[1] BDI-Fuzzy, que apresentam características distintas, na maneira como realizam um determinado serviço, nas *"atitudes de avaliação de serviço"* e nas suas *"estratégias de troca"*. Considere que cada agente hortelão possui a sua parcela de terra e pode realizar três tipos de serviços: plantar, irrigar e colher. Onde esses serviços podem ser *"trocados"* entre os agentes. Neste ambiente, o hortelão α, *"prestador"* de um determinado serviço a β, pode avaliar o investimento $r_{\alpha\beta}$ necessário para a realização do serviço $\mathbb{S}$ de acordo com os seguites atributos: $\mathbb{S}_r$ = (*dificuldade*, *custo*, *tempo*). Já o hortelão β, *"recebedor"* de um serviço de α, pode avaliar a sua satisfação $s_{\beta\alpha}$ de acordo com os atributos: $\mathbb{S}_s$ = (*qualidade*, *preço*, *tempo*).

O **hortelão** é o agente capaz de realizar os serviços de plantar, irrigar e colher no ambiente da horta. Cada hortelão apresenta características específicas para a realização de cada um dos serviços, essas características afetam os atributos dos serviços realizados pelo agente, os quais podem influenciar nas interações com os outros agentes. O agente hortelão pode apresentar três características diferentes para escolher o agente com o qual irá interagir, chamadas de **"estratégias de trocas de serviços"**.

A primeira consiste na escolha baseada somente na *"satisfação esperada"*, ou seja, após o agente receber todas as *"propostas"* (valores dos atributos *qualidade*, *preço* e *tempo*) dos agentes presentes na horta para a realização de um serviço, o hortelão, de acordo com a sua *"atitude de avaliação de serviço"*, escolhe o agente que pode lhe proporcionar uma maior *satisfação* (s) com o serviço prestado.

A segunda consiste na escolha baseada na *"satisfação esperada"* e no *balanço virtual*, neste caso, o hortelão após receber todas as *"propostas"*, verifica a *"satisfação esperada"* e o *balanço virtual* com cada um dos agentes, sendo a combinação entre esses dois valores que determinará a *"chance de escolha"* do agente para a realização do serviço. Neste caso o hortelão prefere trocar serviços com os agentes que lhe enviaram uma melhor proposta e com os quais ele possui um *balanço virtual "positivo"* (*crédito* $v >$ *débito* t).

A terceira consiste na escolha baseada na *"satisfação esperada"* e no *balanço material*, neste caso, o hortelão após receber todas as *"propostas"*, verifica a *"satisfação esperada"* e o *balanço material* com cada um dos agentes, sendo a combinação entre esses dois valores que determinará a *"chance de escolha"* do agente para a realização do serviço. Assim, o hortelão prefere trocar serviços com os agentes que lhe enviaram uma melhor proposta e com os quais ele possui um *balanço material "negativo"* (*investimento* $r >$ *satisfação* s).

A. Análise das Simulações

As simulações representam o ambiente da Horta, no qual três agentes do tipo Hortelão interagem. Os agentes são denominados de *Hortelão 1*, *Hortelão 2* e *Hortelão 3*, respectivamente, e apresentam características distintas na maneira como realizam um determinado serviço e nas suas *atitudes de avaliação de serviço*. Os hortelãos podem realizar três tipos de serviços na horta: plantar, irrigar e colher. Os intervalos dos possíveis valores gerados para cada atributo dos serviços prestados pelos agentes *Hortelão 1, 2* e *3*, são apresentados na Tabela I, através da qual pode-se observar que:

- *Hortelão 1* – realiza serviços de baixa qualidade, com preço alto e com um tempo elevado;
- *Hortelão 2* – realiza serviços de média qualidade, com preço médio e com um tempo intermediário;
- *Hortelão 3* – realiza serviços de qualidade média a alta, com preços que variam de baixo a médio e com um tempo curto a médio.

A Tabela II apresenta os intervalos de valores, para cada atributo do serviço, utilizados para o cálculo do *investimento* (r) do *Hortelão 1, 2* e *3* na realização de um serviço. Através dos quais, pode-se observar que:

- *Hortelão 1* – considera que os três serviços da Horta são *"fáceis"* de realizar, ou seja, apresentam uma dificuldade baixa, porém lhe consomem muito tempo e recurso;

[1]Estes estudos foram realizados em um projeto de cooperação com a Universidad de Sevilla, para o desenvolvimento de ferramentas para simulação de processos de produção e gestão social de ecossistemas urbanos [19], mais especificamente, a horta urbana de San Jerónimo, (Sevilla, Espanha).[20], [21]

TABLE I. VALORES DOS ATRIBUTOS DOS SERVIÇOS REALIZADOS PELOS AGENTES *Hortelão* 1, 2 E 3

Serviços	Hortelão 1			Hortelão 2			Hortelão 3		
	qualidade	preço	tempo	qualidade	preço	tempo	qualidade	preço	tempo
plantar	0 a 2.5	75 a 100	60 a 90	2.5 a 7.5	25 a 75	30 a 60	5 a 10	0 a 50	1 a 45
irrigar	0 a 2.5	75 a 100	60 a 90	2.5 a 7.5	25 a 75	30 a 60	5 a 10	0 a 50	1 a 45
colher	0 a 2.5	75 a 100	60 a 90	2.5 a 7.5	25 a 75	30 a 60	5 a 10	0 a 50	1 a 45

- *Hortelão 2* – apresenta valores intermediários para os três atributos dos serviços da Horta;
- *Hortelão 3* – apresenta uma dificuldade elevada para executar os três serviços da Horta, porém estes não lhe consomem muito tempo e recurso.

Cada hortelão, quando necessita de um serviço, envia uma requisição de *"proposta"* de serviço a todos os outros hortelãos presentes na Horta. Após o recebimento de todas as *"propostas"* o hortelão escolhe o agente com o qual irá trocar serviço, de acordo com a sua *"estratégia de troca"*.

Nas simulações realizadas, cada hortelão executa 100 requisições de cada um dos serviços da horta, gerando um total de 300 requisições por agente. Essas interações seguem uma ordem pré-estabelecida, como descrita a seguir:

Hortelão 1 requisita serviço ... serviço terminado ↷
Hortelão 2 requisita serviço ... serviço terminado ↷
Hortelão 3 requisita serviço ... serviço terminado ↷
Hortelão 1 requisita serviço ... serviço terminado ↷
...

Essa sequência evita que um único agente requisite diversas propostas de serviços seguidas, impossibilitando que os outros agentes realizem suas requisições.

Nas simulações realizadas o *Hortelão 1* avalia o *investimento* gerado na realização de um serviço da Horta, levando em consideração somente o valor do atributo *"dificuldade"* e avalia a *satisfação* obtida no recebimento de um serviço levando em consideração somente o valor do atributo *"qualidade"*. O *Hortelão 2* avalia o *investimento* levando em consideração os valores dos atributos *"dificuldade"* e *"tempo"* e avalia a *satisfação* levando em consideração os valores dos atributos *"qualidade"* e *"tempo"*. O *Hortelão 3* avalia o *investimento* levando em consideração os três atributos do serviço *"dificuldade"*, *"custo"* e *"tempo"* e avalia a *satisfação*, também, levando em consideração os três atributos do serviço *"qualidade"*, *"preço"* e *"tempo"*.

B. Simulação com Estratégia de Troca Baseada na Satisfação

Nesta simulação, são avaliados os valores defuzzificados dos balanços materiais e virtuais dos três agentes hortelão (Hortelão 1, Hortelão 2 e Hortelão 3) nas diversas trocas de serviços realizadas. Neste caso, os três hortelãos utilizam a ***estratégia de troca baseada na satisfação***. Essa estratégia de troca combinada com as outras características dos hortelãos resultou nos seguintes números de serviços realizados por cada agente:

- O *Hortelão 1* não foi escolhido por nenhum dos agentes para realizar serviço;
- O *Hortelão 2* realizou 28 serviços para o *Hortelão 1* e realizou 300 serviços para o *Hortelão 3*;
- O *Hortelão 3* realizou 272 serviços para o *Hortelão 1* e realizou 300 serviços para o *Hortelão 2*.

Assim, pode-se somar o número de serviços prestados entre dois agentes, para obter o total de trocas realizadas entre esses agentes durante a simulação. Neste caso, ocorreram:

- 28 trocas entre os hortelãos 1 e 2;
- 272 trocas entre os hortelãos 1 e 3;
- 600 trocas entre os hortelãos 2 e 3.

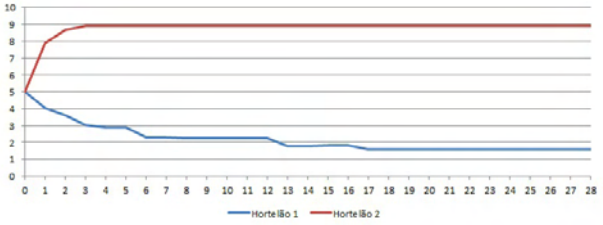

Fig. 3. Balanços virtuais dos hortelãos 1 e 2 com estratégia de troca baseada na satisfação

A Figura 3 mostra os balanços virtuais dos hortelãos 1 e 2, onde pode-se observar que, pelo fato do *Hortelão 1* só ter recebido serviços, este acumula dívidas com o *Hortelão 2*, o qual, por sua vez, acumula créditos por ter sido o único a realizar serviços nas 28 interações que teve com o *Hortelão 1*.

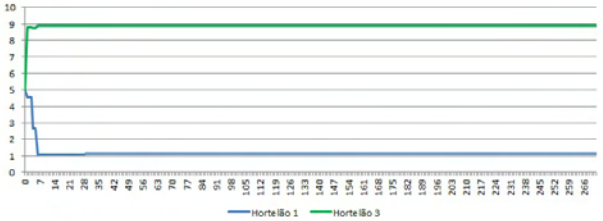

Fig. 4. Balanços virtuais dos hortelãos 1 e 3 com estratégia de troca baseada na satisfação

A Figura 4 mostra os balanços virtuais dos hortelãos 1 e 3, onde observa-se uma certa semelhança com a Figura 3, porém neste caso ocorreu um número muito maior de interações entre os agentes, onde nessas 272 interações, somente o *Hortelão 3* realizou serviços, resultando assim num balanço virtual *"negativo"* para o *Hortelão 1*, que acumulou dívidas.

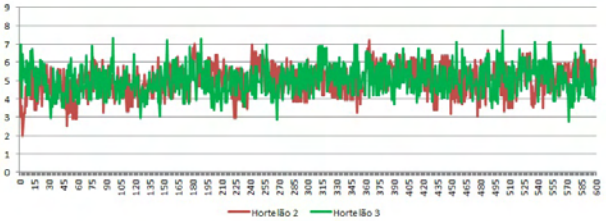

Fig. 5. Balanços virtuais dos hortelãos 2 e 3 com estratégia de troca baseada na satisfação

A Figura 5 mostra os balanços virtuais dos hortelãos 2 e 3, neste caso os valores dos balanços apresentam oscilações em torno de 5, sendo (5.030900671) a média do balanço virtual

TABLE II. VALORES DOS ATRIBUTOS DOS SERVIÇOS PARA O CÁLCULO DO *investimento* REALIZADO PELOS AGENTES *Hortelão* 1, 2 E 3

Serviços	Hortelão 1			Hortelão 2			Hortelão 3		
	dificuldade	custo	tempo	dificuldade	custo	tempo	dificuldade	custo	tempo
plantar	0 a 2.5	75 a 100	60 a 90	2.5 a 7.5	25 a 75	30 a 60	5 a 10	0 a 50	1 a 45
irrigar	0 a 2.5	75 a 100	60 a 90	2.5 a 7.5	25 a 75	30 a 60	5 a 10	0 a 50	1 a 45
colher	0 a 2.5	75 a 100	60 a 90	2.5 a 7.5	25 a 75	30 a 60	5 a 10	0 a 50	1 a 45

do *Hortelão 2* e (5.032719712) a média do balanço virtual do *Hortelão 3*, o que demonstra um equilíbrio virtual entre os agentes nas 600 interações, onde cada agente realizou a mesma quantia de serviços (300), fato esse que pode ter contribuído para o equilíbrio dos balanços virtuais.

Analisando os valores e os gráficos presentes, observa-se que a estratégia de troca baseada somente na satisfação favorece a interação (troca de serviços) entre os agentes que apresentam melhores características na realização do serviço, ou seja, esses agentes têm uma chance maior de serem escolhidos para realizarem o serviço, pois, apresentam atributos melhores para a realização do mesmo e consequentemente causam uma satisfação maior para o agente que recebe o serviço. Esse tipo de estratégia de troca não leva em consideração os balanços materias e virtuais entre os agentes, permitindo, neste caso, que um determinado agente somente receba ou realize um serviço em diversas interações (como o ocorrido com o *Hortelão 1* que não foi escolhido por nenhum dos agentes para realizar serviço).

VI. CONCLUSÃO

Neste trabalho foi desenvolvido um modelo híbrido de agente BDI-Fuzzy para a plataforma de agentes Jason, com habilidades de avaliar de forma qualitativa os valores de trocas sociais originados na prestação e no recebimento de serviços não-econômicos. Um serviço é definido como um conjunto de atributos que são utilizados na sua avaliação (e.g, qualidade, dificuldade, tempo, custo, etc). O agente pode considerar um, dois ou mais atributos do serviço na sua avaliação (tanto na prestação como no recebimento de um serviço), configurando as diferentes atitudes de avaliação de serviço. Os agentes também podem assumir diferentes atitudes com relação às trocas de serviços que optam por realizar, podendo levar em consideração os valores dos balanços materiais e virtuais na escolha dos agentes com os quais irão interagir.

ACKNOWLEDGMENTS

This work was supported by CNPq (Proc. 560118/10-4, 305131/2010-9, 476234/2011-5) , FAPERGS (Proc. 11/0872-3) and Projeto RS-SOC (FAPERGS Proc. 10/0049-7).

REFERENCES

[1] M. Wooldridge, *An Introduction to MultiAgent Systems*. Chichester: Wiley, 2002.

[2] N. Gilbert, *Agent-based Models*. Los Angeles: SAGE, 2008.

[3] A. S. Rao and M. P. Georgeff, "An abstract architecture for rational agents," in *Proc. of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*. Morgan Kaufmann, oct 1992, pp. 439–449.

[4] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[5] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason*. University of Liverpoll: Wiley, 2007.

[6] J. Piaget, *Sociological Studies*. London: Routlege, 1995.

[7] N. Ghasem-Aghaee and T. I. Ören, "Towards fuzzy agents with dynamic personality for human behavior simulation," in *Proc. of the 2003 Summer Computer Simulation Conference, Montreal, July 20-24, 2003*. San Diego: SCS, 2003, pp. 3–10.

[8] T. I. Ören and N. Ghasem-Aghaee, "Personality representation processable in fuzzy logic for human behavior simulation," in *Proc. of the 2003 Summer Computer Simulation Conference, Montreal, July 20-24, 2003*. San Diego: SCS, 2003, pp. 11–18.

[9] G. P. Dimuro, A. V. Santos, G. P. Bedregal, and A. C. R. Costa, "Fuzzy evaluation of social exchanges between personality-based agents," in *New Trends In Artificial Intelligence, Proc. of 14th Portuguese Conference on Artificial Intelligence, EPIA'2009*, L. S. Lopes, N. Lau, P. Mariano, and L. M. Rocha, Eds. Aveiro: APIA/Universidade de Aveiro, 2009, pp. 451–462.

[10] A. Casali, L. Godo, and C. Sierra, "Graded BDI models for agent architectures," in *In 5th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA V*, ser. LNAI, vol. 3487. Springer, 2004, pp. 126–143.

[11] ——, "Modelos BDI graduados para arquitecturas de agentes," *Revista Iberoamericana de Inteligencia Artificial*, vol. 9, no. 26, pp. 67–75, 2005.

[12] S. A. Long and A. C. Esterline, "Fuzzy BDI architecture for social agents," in *Proceedings of the IEEE Southeastcon 2000*, N. R. Pal, N. Kasabov, R. K. Mudi, S. Pal, and S. K. Parui, Eds. Los Alamitos: IEEE, 2000, pp. 68–74.

[13] S. Shen, G. M. P. O'Hare, and R. Collier, "Decision-making of BDI agents, a fuzzy approach," in *Proceedings of The Fourth International Conference on Computer and Information Technology*. Washington: IEEE, 2004, pp. 1022 – 1027.

[14] G. P. Dimuro, A. C. R. Costa, and L. A. M. Palazzo, "Systems of exchange values as tools for multi-agent organizations," *Journal of the Brazilian Computer Society*, vol. 11, no. 1, pp. 31–50, 2005, (Special Issue on Agents' Organizations).

[15] M. R. Rodrigues, A. C. R. Costa, and R. Bordini, "A system of exchange values to support social interactions in artificial societes," in *Proc. II Intl Conf. on Autonomous Agents and Multiag. Systems, AAMAS'03*. Melbourne: ACM Press, 2003, pp. 81–88.

[16] M. R. Rodrigues and A. C. R. Costa, "Using qualitative exchange values to improve the modelling of social interactions," in *Proc. of IV Work. on Agent Based Simulations, MABS'03, Melbourne, 2003*, ser. LNAI, D. Hales, B. Edmonds, E. Norling, and J. Rouchier, Eds., no. 2927. Berlin: Springer, 2004, pp. 57–72.

[17] T. J. Ross, *Fuzzy Logic with Engineering Applications*. New Mexico - USA: Wiley, 2004.

[18] J. Piaget, *Estudos Sociológicos*. Rio de Janeiro: Forense, 1973.

[19] G. Dimuro and E. M. Jerez, "La comunidad como escala de trabajo en los ecosistemas urbanos," *Revista Ciencia y Tecnología*, vol. 10, pp. 101–116, 2011.

[20] F. C. P. Santos, T. F. Rodrigues, G. Dimuro, D. F. Adamatti, G. P. Dimuro, A. C. R. Costa, and E. De Manuel Jerez, "Modeling role interactions in a social organization for the simulation of the social production and management of urban ecosystems: the case of San Jerónimo vegetable garden of Seville, Spain," in *2012 Brazilian Workshop on Social Simulation, Advances in Social Simulation II*. Los Alamitos: IEEE, 2012, pp. 136–139.

[21] I. S. Santos, T. Rodrigues, G. P. Dimuro, A. R. Costa, G. Dimuro, and E. de Manuel, "Towards the modeling of the social organization of an experiment of social management of urban vegetable gardens," in *Proceedings of 2011 Workshop and School of Agent Systems, their Environment and Applications*. Los Alamitos: IEEE, 2011, pp. 98 –101.

Integrating CartAgO Artifacts for the Simulation of the Social Production and Management of Urban Ecosystems: the case of San Jerónimo Vegetable Garden of Seville, Spain

Flávia Santos, Henrique Rodrigues, Thiago Rodrigues
Diana F. Adamatti, Graçaliz P. Dimuro
Programa de Pos-Graduação em Modelagem Computacional
Universidade Federal de Rio Grande, FURG
Rio Grande, Brazil
Email: faflasan@gmail.com

Glenda Dimuro, Esteban de Manuel Jerez
Depto de Expresión Gráfica y Arquitectónica,
Universidad de Sevilla
Sevilla, Spain
Email: glenda.dimuro@gmail.com

***Abstract*—The paper presents some new results obtained with the modeling of a multi-agent system for the simulation of the social production and management processes of an urban ecosystem, namely, the urban vegetable garden San Jerónimo (HSJ) of Seville, Spain. The modeling was obtained by the application of several tools related to the JaCaMo platform, as the MOISE+ model for the organization modeling, the CArtAgO and MSPP frameworks for the construction of normative and communication artifacts. The aim is to provide tools for the modeling of the interactions between organizational roles when performing periodic actions in the HSJ's social system routines. Furthermore, we show an initial proposal for integration between JaCaMo tools in the context of HSJ.**

***Keywords*—Periodic routines, JaCaMo, Artifacts; multiagent systems; Social Systems;**

I. Introdução

Para reparar danos ambientais é necessário solucionar questões sociais e econômicas, que implicam mudanças de mentalidades e comportamentos, ampliando a participação e implicação de cidadãos na defesa do seu entorno. É neste ponto que se faz a conexão entre a ecologia urbana e a produção e gestão social do hábitat [1], [2], [3], [4]. Transpor a sustentabilidade da teoria à prática significa conceber o ser humano e o território onde a maioria da espécie se desenvolve – as cidades – como parte da natureza, sob o conceito de "ecossistemas urbanos" [5], [6]. Um ecossistema urbano não é uma simples agregação de espaços aleatórios, mas um todo conectado com redes dentro de redes com causas e efeitos; um hábitat com uma estrutura coerente com os paradigmas culturais e necessidades específicas de um determinado grupo e contexto; um processo de incremento incessante de informações; um território fisicamente fechado, mas aberto a fluxos de energia e recursos.

O conceito de produção e gestão social de ecossistemas urbanos pode ser compreendido como a geração de novas situações, físicas ou relacionais, mediante a construção, transformação ou eliminação de objetos físicos ou de objetos relacionais com objetivo de assegurar, nas novas situações produzidas, o cumprimento de suas funções sociais e ambientais [2], [3]. Isto inclui a participação cidadã nos processos de planejamento e transformação urbana, articulando distintos agentes envolvidos (governo, instituições, técnicos, cidadãos), formando uma rede estruturada e apoiada em mecanismos e ferramentas que possibilitem a distribuição igualitária de poder na tomada de decisões, de modo que todos agentes possam participar, dialogar ativamente em todo processo de determinado projeto, desde sua planificação até a gestão. A produção gestão social de ecossistemas urbanos contribuem ao fortalecimento de práticas comunitárias, ao aumento da responsabilidade por um projeto coletivo, ao exercício da democracia, ao desenvolvimento de ações mais solidárias, incluindo tanto temas produtivos e econômicos, como ambientais.

Este trabalho apresenta alguns resultados preliminares obtidos na modelagem de um sistema multiagente para a simulação da produção e gestão social de um ecossistema urbano - um esforço conjunto para a inter-relação do conhecimento, buscando interpretações coletivas, adotando como estudo de caso a tendência atual de (re) aproximar o campo à cidade através de hortas urbanas. A organização escolhida é o projeto de hortas sociais realizado no Parque San Jerónimo (Sevilha/Espanha), impulsionada pela ONG Ecologistas em Acción.

O objetivo geral do Projeto onde se insere este trabalho é desenvolver uma ferramenta de simulação baseada em SMA para a análise da realidade atual do projeto, permitindo discussões sobre os processos de gestão social adotadas, e também para investigar como possíveis mudanças nas ações, comportamentos e papéis assumidos pelos agentes na organização, especialmente do ponto de vista de sua participação nos processos de tomada de decisões, podem transformar esta realidade, desde o ponto de vista social, ambiental e econômico, e contribuir para a sustentabilidade do projeto HSJ.

A ferramenta adotada neste Projeto é a plataforma JaCaMo (http://jacamo.sourceforge.net/) [7], que é um *framework* para programação multiagente que combina três tecnologias distintas, ou seja, Jason (para programação de agentes) [8], Cartago

(para artefatos do ambiente de programação) [9] e MOISE+ (para modelagem da organização multiagentes) [10], [11].

Em trabalhos anteriores [12], [13], apresentou-se a primeira fase da modelagem da organização do SMA, desenvolvida usando o modelo organizacional MOISE+, identificando os papéis organizacionais da horta San Jerónimo e suas rotinas, as interações sociais, as normas reguladoras e constitutivas.

No entanto, verificou-se que, embora o modelo MOISE+ possibilite a visualização estrutural do sistema organizacional, bem como seus schemes, permitindo facilmente ver, por exemplo, a ordem em que os objetivos devem ser atingidos, as interações entre os papéis organizacionais, não é possível concretizar a modelagem de um sistema social que envolva o aspecto de "periodicidade" em suas rotinas. Observe que a organização social da HSJ é baseada na realização das rotinas periódicas pelos papéis e também no cumprimento das normas que regulam seus comportamentos periódicos. Assim, para contornar esse problema, optou-se por explorar mais a plataforma JaCaMo e utilizar outra ferramenta relacionada, a saber, o *framework* MSPP (Modelagem e Simulação de Políticas Públicas) [14], que mostrou-se adequado, permitindo especificar a periodicidade das rotinas de papéis para o sistema da HSJ.

Então, no presente artigo, propõe-se a integração entre diversos tipos de Artefatos (Organizacionais, Normativos, de Comunicação e Físicos) e o ambiente para o domínio da HSJ através do *framework* CArtAgO na plataforma Jason. Para construir os artefatos normativos levaram-se em consideração as normas da HSJ e utilizou-se o *framework* MSPP (Modelagem e Simulação de Políticas Públicas).

Diagramas de atividades da UML foram utilizados inicialmente para identificar como ocorrem as interações entre os papéis, com objetivo de definir uma abordagem para lidar com a comunicação de papéis em interações usando artefatos CArtAgO, visando afastar dos agentes que desempenham papéis organizacionais a lógica de troca de mensagens utilizando um determinado protocolo, obtendo uma abordagem mais modular de comunicação do agente. Para isso, foram criados Artefatos de Comunicação divididos em dois grupos "artefatos de atos de fala" e "artefatos de protocolo", implementados através do *framework* CArtAgO [15].

O artigo está organizado da seguinte forma. Na Seção II, identificam-se os papéis que compõem a organização da HSJ e suas rotinas, com base em normas constitutivas e regulativas, construídas a partir do regulamento da HSJ. Na Seção III, apresenta-se o estudo sobre a plataforma JaCaMo e suas três tecnologias Jason, CartAgO e MOISE+ e o *framework* MSPP para modelagem de rotinas de papéis na HSJ.A Seção IV descreve uma breve introdução sobre a criação e uso dos artefatos para o estudo de caso da HSJ. A Seção V apresenta exemplos de uso dos artefatos no contexto da horta. A Seção VI é a Conclusão.

II. Projeto Hortas de Ócio (huertos de Ócio)

A Horta San Jerónimo (HSJ) é uma iniciativa da ONG Ecologistas em Acción com o objetivo de fomentar a participação social em práticas de agricultura orgânica, mediante o uso e desfrute de hortas de *lazer*, e realização de atividades vinculadas com a educação ambiental. Atualmente a zona de hortas ocupa uma área de mais ou menos 1,5 hectares do Parque de São Jerônimo, com um total de 42 parcelas (dimensões entre $75m^2$ e $150m^2$) que se distribuem em hortas de ócio (principalmente cuidadas por hortelãos aposentados, mas não exclusivamente), hortas escolares (dedicadas aos alunos do ensino fundamental das escolas do bairro) e em parcelas cedidas a outras associações para experimentos científicos (como a Rede Andaluzia de Sementes e a Plataforma Andaluzia Livre de Transgênicos). As hortas são designadas aos hortelãos e o direito a utilização (e não a propriedade) da parcela ocorre por um prazo de dois anos prorrogáveis – sempre que cumpram as normas e regras estabelecidas no regulamento definido pela ONG. Suas principais características são o fato de ser uma horta social sem fins lucrativos, ou seja, a produção é dedicada para o autoconsumo daqueles que as cultivam (a venda é ilegal) e ser apoiada economicamente por financiamento municipal e colaboração dos participantes.

A HJS é regida por normas e a ONG é quem verifica o cumprimento destas normas. Quando alguma norma é desobedecida, deve ser aplicada uma punição/sanção de acordo com o regulamento da horta, podendo o hortelão responsável ser expulso da parcela sob sua responsabilidade. A Figura 1 mostra exemplos destas regras, discutidas em maior detalhe em[12]. Por exemplo, durante o uso da horta, o agente hortelão precisa permissão junto a organização para "plantar árvores com ciclo maior que dois anos", que pode ou não ser autorizada e não tem uma sanção como resultado. Por outro lado, tem-se a norma "proibido vender produto cultivados na horta", que é considerada uma falta grave e se o hortelão acumular três faltas graves, uma assembléia é convocada e é decidido por votação dos hortelãos que possuem parcelas na HSJ se o "hortelão infrator" permanece ou será expulso do projeto.

Tipo da Norma	Situação em que se aplica	Ação Normatizada					Verificador da ação	Sanções (Punições, Recompensas)
		Pré-condição	Normatização	Id da ação	Resultados	Papel realizador		
Regulativa	Durante o uso da horta	Possuir uma parcela	Proibição	Vender produtos da horta	Continuar na horta OUX Sair da horta (depende do numero de faltas graves)	Hortelão	ONG (Técnicos)	Punição: Falta grave (cumulativa)
Regulativa	Durante o uso da horta, mensalmente	Possuir uma parcela	Obrigação	Pagar mensalidade	Ganha auxílio de o cultivo E Continua na horta; Não ganha auxílio de cultivo E Sai da horta (depende do número de faltas graves)	Hortelão	ONG (Secretaria)	Punição: Falta grave (cumulativa)
Constitutiva	Durante o uso da horta	Possuir uma parcela	Permissão	Plantar árvores com ciclos maiores 2 anos	Continuar na horta; Sair da horta (depende do número de faltas graves)	Hortelão	ONG (Técnicos)	-
Regulativa	Durante o uso da horta	Possuir uma parcela	Proibição	Modificação do desenho da horta	Continuar na horta; Sair da horta (depende do número de faltas graves)	Hortelão	ONG (Técnicos)	Punição: Falta grave (cumulativa)

Fig. 1. Parte da tabela de normas da HSJ

III. A Plataforma JaCaMo e MSPP *framework* para modelagem de rotinas de papéis na HSJ

No estudo realizado na Horta San Jerónimo (HSJ) foram identificados alguns papéis, os quais possuem rotinas seguidas periodicamente, ou seja, atividades realizadas pelos hortelãos que se repetem em intervalos regulares. Na JaCaMo, a modelagem de rotinas periódicas como estas não pode ser feita facilmente, pois não há ferramentas nativas na plataforma que permitam esta especificação. Atualmente, os processos permitidos no sistema multiagente i.e. os objetivos que devem ser atingidos, podem ser descritos por meio do modelo MOISE+. Este modelo permite um bom nível de abstração para especificação destas unidades, bem como a definição de uma hierarquia entre os mesmos. Entretanto, uma rotina envolve a satisfação de objetivos periódicos (períodos de um

mês, uma semana, um dia) e no modelo não há estruturas para isto. Quando um objetivo é atingido ele é considerado na dimensão Deôntica/Normativa como satisfeito, ou seja, não gera objetivos de manutenção para que possa ser retomado novamente de uma rotina periódica.

A satisfação destes objetivos estão relacionadas diretamente ao modelo Moise, ou seja, quando um objetivo é atingido ele é considerado na dimensão Deôntica/Normativa como satisfeito, ou seja, não gera objetivos de manutenção para que possa ser retomado novamente de uma rotina periódica.

Em sistemas sociais, há também muitas situações onde ocorre a aplicação de normas, que impõem sanções sob ações realizadas pelos agentes. Na HSJ, foram identificadas diversas ações normatizadas desta forma, como "vender produtos da horta", "utilização de mangueiras na regagem" e "uso de produtos químicos no cultivo das hortas".

Da mesma forma como não há estruturas na plataforma para definição de periodicidades (de ações, de satisfação de objetivos), não há meio para definir normas, seus atributos básicos (nome, período, papel que a aplica) e as suas sanções.

Objetivando oferecer uma forma modular para descrição de normas periódicas, de modo a facilitar a modelagem do sistema social compreendido pela HSJ, utiliza-se *framework* MSPP, que é implementado, não como extensão, mas como complemento ao suporte da JaCaMo através do MOISE+ pela dimensão Deôntica/Normativa. O mesmo complementa o modelo MOISE+, oferecendo mais uma camada de abstração. Nela, rotinas são modeladas e sofrem ação de normas definidas no *framework*, enquanto que em uma camada inferior, no modelo MOISE+, são especificadas as ações normatizadas, e que constituem as rotinas.

A. Jason

Jason é um interpretador para a linguagem *AgentSpeak-L* e provê uma plataforma para desenvolvimento de sistemas multiagentes, incluindo comunicação entre agentes baseada na teoria dos atos de fala. Utilizando o *SACI* (Simple Agent Communication Infrastructure), um SMA desenvolvido em Jason pode ser distribuído em uma rede de computadores sem muito esforço. Existem muitas implementações *ad hoc* de sistemas BDI, contudo uma característica importante do *AgentSpeak-L* é sua fundamentação teórica. Outra característica importante do Jason em comparação com outros sistemas BDI é que ele é implementado em Java (portanto multi-plataforma) e é disponível como *Open Source* sob a licença GNU LGPL [7].

B. CArtAgo

CArtAgo (Common ARTifact infrastructure for AGents Open environments) é um *framework* para a programação e execução de ambientes virtuais para sistemas multiagentes. Através do *framework* CArtAgO é possível implementar ambientes virtuais, onde implementa-se o ambiente como uma camada computacional que encapsula as funcionalidades e os serviços não-autônomos que os agentes podem explorar em tempo de execução.

CartAgo é baseado no meta-modelo Agents & Artefacts (A & A) [9] para modelar sistemas multiagentes. Este modelo introduz uma metáfora de alto nível retirada da ideia de que humanos trabalham de forma cooperativa com o seu ambiente: agentes são como entidades computacionais que realizam algum tipo de tarefa orientada para alcançar um objetivo (em analogia aos trabalhadores humanos), e artefatos são como recursos e ferramentas dinamicamente construídas, manipuladas e compartilhadas pelos agentes para dar suporte e realizarem suas atividades individuais e coletivas (como artefatos no contexto humano).

Assim, é possível desenvolver artefatos os quais são instanciados no ambiente e podem prover serviços para os agentes, podendo inclusive realizar uma comunicação com serviços externos do tipo web-services, contribuindo para programação de agentes. CArtAgo é uma tecnologia Open Source, disponível em [9], e inclui uma API baseada na linguagem Java para programar os artefatos.

C. MOISE$^+$

O modelo organizacional MOISE$^+$ [11] foi desenvolvido para modelar a organização de SMA e 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.

Assim, MOISE+ é um modelo de organização para sistemas multiagentes baseado em noções como papéis, grupos e missões. Isso permite que o sistema tenha sua organização explícita e que seja usada uma plataforma que faça os agentes cumprirem com suas obrigações da organização.

A Especificação Funcional é constituída por um conjunto de Esquemas Sociais (schemes sociais), que é um conjunto de metas estruturado por meio de planos. Esquema Social: o conjunto de todos os esquemas sociais é denotado por SCH e um esquema sch é representado pela tupla sch = (G;P;M; mo;nm) onde

- $\mathcal{G}$ é o conjunto de metas do ES sch ;
- $\mathcal{P}$ é o conjunto planos (constrói a árvore de decomposição de metas);
- $\mathcal{M}$ é o conjunto de missões, ou seja, um conjunto de metas globais que pode ser atribuído a um agente através de um de seus papéis(ligam os agentes aos planos);
- mo: $\mathcal{M} \mapsto \mathbb{P}$ (G) é uma função que determina o conjunto de metas de cada missão;
- nm: $\mathcal{M} \mapsto \mathbb{N} \times \mathbb{N}$ determina o número (mínimo e máximo) de agentes que devem se comprometer em cada missão.

Um ES é uma árvore de decomposição de metas globais onde a raiz é a meta do ES e a decomposição é feita por meio de planos (denotados pelo operador =) que indicam uma forma de satisfazer uma meta. Por exemplo, no plano $g0 = g1, g2, g3$, a meta $g0$ é decomposta em três planos indicando que ela será satisfeita somente se os planos $g1, g2, g3$ também serem satisfeitos.

D. framework *MSPP (Modelagem e Simulação de Políticas Públicas) framework*

O *framework* MSPP (Modelagem e Simulação de Políticas Públicas) foi desenvolvido para modelagem e simulação de políticas públicas e conforme Seção IV (Artefatos Normativos) foram feitas adaptações para o sistema da HSJ. O MSPP concretiza-se no formato de artefatos no modelo CArtAgO para modelar e projetar sistemas multiagente. Neste *framework* estão incluídos dois tipos de artefatos normativos que são: NormObrig e NormPrb, modelando normas de obrigação e proibição, respectivamente.

Além dos artefatos, estão previamente inseridos agentes para executar/verificar tais normas. São eles: o agente governamental (responsável por emitir normas), os agentes sociais, que estão submetidos às normatizações e buscam atingir objetos próprios e os agentes governamentais detectores e/ou efetores, responsáveis por detectar o cumprimento das normas de uma política emitida pelo agente governamental, como também regulamentar recursos do ambiente e aplicar possíveis sanções a ações que caracterizarem o descumprimento das normas.

O *framework* pressupõe adotar estes quatro tipos de agentes interagindo para promover o ciclo de política, para que seja criada uma política, que os agentes sociais como também os governamentais tomem conhecimento desta e, por fim que todos passem a ter seus objetivos orientados seguindo aquilo que foi estipulado e de acordo com seus papéis. As normas previamente inseridas estão estruturadas da seguinte maneira em [14]:

- Id: o identificador da norma;
- Destinatário: especifica o papel ao qual a norma se aplica;
- Ação: especifica uma ação a ser realizada pelo agente que assume o papel ao qual a norma foi endereçada;
- Condição: especifica uma condição contextual necessária para a aplicação da norma;
- Periodicidade: especifica o evento que deve ocorrer (mês, semana, ou uma ação específica) para que se verifique a condição;
- Exceção: especifica uma condição na qual a norma não se aplica;
- Sanção: especifica a sanção a ser aplicada no caso da violação da norma.

Os agentes sociais e também os agentes efetuadores e detectores estão constantemente observando as normas como também tomam conhecimento de uma eventual modificação ou exclusão delas do sistema. O conhecimento destes sobre as normas é adicionado através de crenças onde se define que uma ação qualquer é proibida, obrigatória, ou se necessita ser obervado o estado atual da permissão ou se é resguardado o direito de executá-la, como apresentado no estudo de caso da HSJ. Uma vez cometida uma infração a essas normas, cabe ao agente detector e/ou efetuador buscar junto ao artefato a devida sanção e eventualmente aplicá-la. Os recursos públicos disponíveis no ambiente e até o mesmo devem estar disposto também na forma de um artefato CArtAgO a fim de estabelecer maior interação entre o sistema.

IV. Criação dos Artefatos para o estudo de caso da HSJ

A proposta deste artigo é a integração entre diversos tipos de Artefatos (Organizacionais, Normativos, de Comunicação e Físicos) para o domínio da HSJ. Nesta seção apresenta-se um breve roteiro de criação dos Artefatos Normativos e de Comunicação. Em seguida, mostra-se o uso dos artefatos através de exemplos criados a partir das rotinas da HSJ.

Os **Artefatos Organizacionais** são os artefatos que fazem a comunicação entre a organização definidas a partir do modelo criado no MOISE+ [12] e a população de agentes. A integração deste modelo com os outros componentes da plataforma JaCaMo ocorre por meio de dois artefatos existentes por padrão no framework CArtAgO: GroupBoard e SchemeBoard. Ambos pertencem ao pacote ora4mas.nopl, contido na distribuição padrão do *framework*.

Os **Artefatos de Comunicação** devem ter a função de mediar à comunicação, ou seja, encaminhar mensagens aos respectivos destinatários, fiscalizando a ordem de execução do envio destas. Os Artefatos de Comunicação são divididos em dois grupos [15]:

- artefatos de protocolo: são responsáveis pelo encapsulamento da lógica de algum protocolo de comunicação (tipo de comunicação que em geral é mais complexa);
- artefatos de atos de fala: executam atos de fala simples.

A comunicação é realizada através da invocação das operações disponíveis nos artefatos e após os procedimentos de inicialização e criação dos artefatos, obrigatoriamente a primeira ação que os agentes devem realizar é seu cadastro nos artefatos, através da operação *subscribe*. Em seguida, é necessária a execução da operação *focus* para cada artefato em uso. Dessa forma o agente perceberá quaisquer mudanças nas propriedades observáveis dos artefatos, mapeando-as para sua base de crenças.

Finalmente, a comunicação se dará pela chamada às operações disponíveis nos artefatos, parametrizadas com os identificadores da conversa(id), do agente destinatário, tipo de mensagem, a mensagem e o remetente. Cada operação de envio dispara um sinal, percebido pelo receptor. Isto lhe informa que há uma mensagem nova, bem como quem a mandou e qual o identificador da conversa. Os artefatos implementados no projeto são: *AcceptProposal, Agree, CallForProposals, Failure, Inform, Message, Propose, Reject e Request*. Contudo, para o contexto da HSJ e os testes executados, os artefatos *Request e Inform* tem suprido as necessidades do sistema até o presente.

Os **Artefatos Normativos** tem a função de demonstrar como ocorrem as relações e interações entre os papéis com relação às normas da HSJ e sendo esta uma organização social baseada no cumprimento das normas que regulamentam os comportamentos dos agentes, adaptou-se com alguns ajustes, à tabela de normas demonstrada na Seção II aos parâmetros do *framework* MSPP [14] descrito na Seção III, pois foram identificados na tabela que regulamenta a HSJ, além dos tipos de normas de obrigação e proibição, outras duas sendo elas as normas de permissão e direito, as quais foram inseridas ao *framework* com o objetivo de adequação ao sistema da

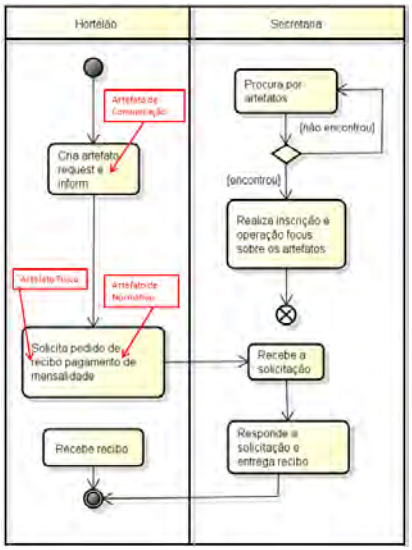

Fig. 2. Diagrama Atividades Pagamento Mensalidade-HSJ

organização da HSJ que possui estas normas explicitas no regulamento. A estrutura das normas implementadas mediante a tabela segue a seguinte forma:

Norma (Id; Tipo de norma: obrigatório, proibido, direito ou permissão; Ação; Sanção; Parâmetro de extensão da sanção). Nota-se portando a adição do parâmetro de extensão da sanção, isto pois as sanções previstas na HSJ estipulam que o agente social pode ser penalizado com uma falta leve, falta grave ou expulsão arbitrária, mas há casos em que as faltas são cumulativas, e que havendo reincidência destas, num total de três vezes é convocada uma assembleia para determinar a continuação desse "infrator" no projeto. Também foi acrescentado um parâmetro que informa o tipo de norma: obrigação, proibição, direito ou permissão. É previsto que ao integrar a estrutura organizacional implementada no MOISE+, seja utilizado os demais parâmetros propostos no *framework* MSPP : Destinatário e Periodicidade. Isto pois os agentes vão estar condicionados a um papel e que este por sua vez possui rotinas atreladas a periodicidade tais como "pagar mensalidade" por exemplo.

Assim, um exemplo de norma aplicada a HSJ ficaria estruturada da seguinte forma: Norma (n22, Hortelão , Obrigação, Pagar a Mensalidade, Mensal, Falta Grave, Cumulativa).

Os **Artefatos Físicos** são abstrações sobre o ambiente, simbolizando serviços e objetos físicos que os agentes utilizam para realizar suas tarefas. Estes artefatos são objetos do ambiente implementados no Cartago como por exemplo: pá, enxada, armário, semeadeira, parcela.

A Figura 2 mostra um exemplo de uso dos Artefatos Físico, Normativo e de Comunicação, onde um agente hortelão durante o uso da horta, tem a "obrigação" (artefato normativo) de efetuar o pagamento de mensalidade junto a secretaria da organização e para isso "faz a solicitação" (artefato de comunicação *request*) de um "recibo" (artefato físico). A secretaria recebe a solicitação e emite o recibo para o hortelão, nesse momento é emitido um *"inform"* ao hortelão.

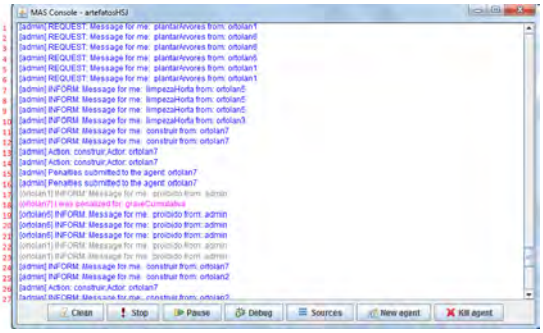

Fig. 3. Execução código no Jason - Uso de Artefatos HSJ

V. Cenários de Uso dos Artefatos na HSJ

Nesta seção apresenta-se dois cenários de uso na HSJ, onde agentes com papéis hortelão (é quem possui uma parcela de cultivo na HSJ), implementados na plataforma Jason como "ortolan" estabelecem comunicação com outros agentes, como admin (quem verifica as ações do hortelão) e ong (agente que cria artefatos normativos).

Os exemplos de cenários apresentados apenas testaram o cumprimento ou não das normas, definido randomicamente. Não existe uma comunicação direta entre os agentes para cumprir ou não as normas, onde haja algum tipo de negociação.

A. Cenário de Uso 1

Inicialmente, conforme Figura 3 o agente para utilizar o artefato deverá fazer sua inscrição no mesmo. Assim, os agentes *ortolan1, ortolan6, ortolan5 e admin* executam a operação focus sobre os artefatos e iniciarm a comunicação através de funções incluídas no artefatos de comunicação que tem como parâmetros o identificador da conversa *(id)*, o destinatário e a mensagem. O agente *ortolan1* através do artefato de comunicação *"request"* (linha 1) faz uma solicitação para "plantarArvores" para o agente admin, que envia uma resposta de que é proibido com um artefato de comunicação *"inform"* (linha 22). Uma outra comunicação é a do *"ortolan5"* (linha 7) que está usando um mensagem simples informando que está executando a "limpezaHorta", a qual não necessita uma resposta do destinatário.

B. Cenário de Uso 2

Neste cenário conforme Figura 4 observa-se uma situação da HSJ onde o hortelão (*"ortolan7"* - linha 1) ao descumprir uma norma (artefato normativo) de proibição recebe como sanção a penalidade de uma falta grave cumulativa (linha 16), e após o registro de três faltas (linhas 13, 14 e 15) é convocada a assembléia (linha 21) de decisão da permanência do mesmo no projeto da HSJ. Após a votação pelos agentes *"ortolan1, ortolan3, ortolan5, ortolan4, ortolan2 e ortolan6* (linhas 22, 23, 34, 25, 26 e 27), se o agente tiver metade ou mais de votos contra, o infrator é "expulso" $(.kill\_agent)$.

VI. Conclusão e Trabalhos Futuros

A plataforma JaCaMo possui recursos que a tornam uma plataforma fully-fledged para modelagem de sistemas multiagentes, inclusive de sistemas sociais. Entretanto, para o estudo

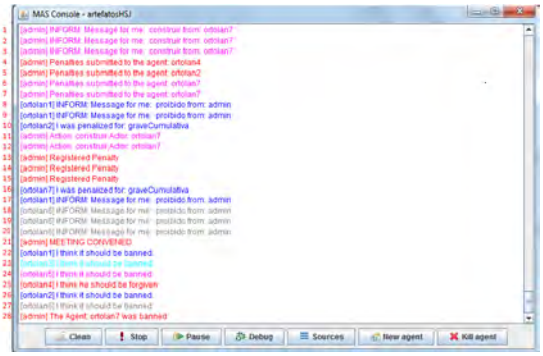

Fig. 4. Execucao codigo no Jason - Uso de Artefatos HSJ

de caso deste artigo, pode-se observar limitações com relação a modelagem de rotinas i.e. ações periódicas.

A especificação de rotinas não pode ser feita na JaCaMo nativamente, fazendo necessário o uso de outro recurso. Neste trabalho utilizou-se *framework* MSPP, que permitiu a especificação das rotinas da horta, tornando-a modular como os outros aspectos pertencentes a um SMA (organização, população e ambiente, já tratados na JaCaMo). Ressalta-se que as funcionalidades do *framework* MSSP vem a complementar aquelas providas pelo modelo MOISE+ (componente da JaCaMo), onde objetivos e ações podem ser descritos de forma hierárquica, mantendo as ações no MOISE+ e a descrição das periodicidades seriam feitas dentro do *framework* MSPP.

Por outro lado, pode-se explorar as facilidades providas pelo framework CArtAgO para criar uma infraestrutura de comunicação baseada em artefatos para a plataforma JaCaMo. Os Artefatos de Comunicação abordados neste artigo tem o objetivo de prover mais uma forma de comunicação para a plataforma, permitindo assim que agentes programados na linguagem *AgentSpeak-L* possam comunicar-se através de protocolos ou por simples atos de fala de uma forma que permita abstrair especificidades não relacionadas diretamente à comunicação, tais como a linguagem de implementação dos agentes e sua localização.

Assim, o presente artigo, apresentou os resultados preliminares da implementação na plataforma JaCaMo, utilizando o Jason e CArtAgO nesse primeiro momento, para demonstração do uso de artefatos para o domínio da HSJ.

A proposta para trabalho futuro é a integração dos Artefatos (Organizacionais, Normativos, de Comunicação e Físicos) e o ambiente para o domínio da HSJ através da plataforma JaCaMo (Jason, CArtAgO e MOISE) e o *framework* MSPP, onde pretende-se aplicar uma abordagem no cenário da simulação, onde os artefatos estejam ligados ao ambiente e que cada papel possa identificar nestes o que é necessário para a simulação da atividade da horta. Ao final da integração, simular o ambiente real da horta, e realizar a análise das ações no ambiente, com os 4 tipos de artefatos.

Acknowledgment

This work was supported by CNPq (Proc. 560118/10-4, 305131/2010-9, 476234/2011-5) , FAPERGS (Proc. 11/0872-3) and Projeto RS-SOC (FAPERGS Proc. 10/0049-7).

References

[1] C. G. Lobo, *Vivienda y Ciudad Posibles*. Bogotá: Escala, 1998.

[2] E. Ortiz, "Derecho a la ciudad, producción social y gestión participativa del hábitat. la promoción de iniciativas comunitarias incluyentes en la ciudad de méxico," *Hábitat y Sociedad*, vol. 1, pp. 55–70, 2010, disponível em http://habitatysociedad.us.es.

[3] V. Pelli, "La gestión de la producción social del hábitat," *Hábitat y Sociedad*, vol. 1, pp. 39–54, 2010, disponível em http://habitatysociedad.us.es.

[4] G. Romero, R. Mesías, M. Enet, R. Oliveras, L. García, M. Coipel, and D. Osorio, *La participación en el diseño urbano y arquitectónico en la producción social del hábitat*. Mexico: CYTED-HABYTED-Red XIV.F, 2004.

[5] J. Terradas, *Ecología urbana*. Barcelona: Rubes Editorial, 2001.

[6] G. Dimuro and E. M. Jerez, "La comunidad como escala de trabajo en los ecosistemas urbanos," *Revista Ciencia y Tecnologia*, vol. 10, pp. 101–116, 2011.

[7] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, and A. Santi, "Multi-agent oriented programming with jacamo," *Science of Computer Programming*, vol. 78, no. 6, pp. 747 – 761, 2013.

[8] R. H. Bordini, J. F. Hubner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason*. New Jersey: Wiley, 2007.

[9] A. Ricci, A. Santi, and M. Piunti, "CArtAgO (common atifact infrastructure for agents open environments)," 2013. [Online]. Available: http://apice.unibo.it/xwiki/bin/view/CARTAGO/

[10] J. F. Hübner, "Um modelo de reorganização de sistemas multiagentes," Ph.D. dissertation, Universidade de São Paulo, São Paulo, 2003.

[11] J. F. Hübner, J. S. Sichman, and O. Boissier, "A model for the structural, functional, and deontic specification of organizations in MAS," in *Brazilian Symposium on Artificial Intelligence - SBIA 2002, Porto de Galinhas*, ser. LNAI, no. 2507. Berlin: Springer, 2002, pp. 118–128.

[12] F. C. P. Santos, G. Dimuro, T. F. Rodrigues, D. F. Adamatti, G. P. Dimuro, and A. C. R. Costa, "Modelando a organização social de um SMA para simulação dos processos de produção e gestão social de um ecossistema urbano: o caso da Horta San Jerónimo da cidade de Sevilla, Espanha," in *Anais do WESAAC 2012*. Florianópolis: UFSC, 2012, pp. 93–104.

[13] F. C. P. Santos, T. F. Rodrigues, G. Dimuro, D. F. Adamatti, G. P. Dimuro, A. C. R. Costa, and E. De Manuel Jerez, "Modeling role interactions in a social organization for the simulation of the social production and management of urban ecosystems: the case of San Jerónimo vegetable garden of Seville, Spain," in *2012 (BWSS)*. Los Alamitos: IEEE, 2012, pp. 136–139.

[14] I. Santos and A. C. R. Rocha, "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*. Berlin: Springer, 2012, pp. 45–59.

[15] T. F. Rodrigues, A. C. R. Costa, and G. P. Dimuro, "A communication infrastructure based on artifacts for the jacamo platform," in *Proceedings of EMAS 2013 - 1st International Workshop on Engineering Multi-Agent Systems at AAMAS 2013*, M. Cossentino, A. E. F. Seghrouchni, and M. Winikoff, Eds. Saint Paul: IFAMAS, 2013, pp. 1–15.

A MAS for the Simulation of Normative Policies of the Urban Vegetable Garden of San Jerónimo, Seville, Spain

Henrique D. N. Rodrigues
Iverton Santos
Graçaliz P. Dimuro
Diana F. Adamatti
Universidade Federal do Rio Grande
Rio Grande, Rio Grande do Sul
Email: {henriquedonancio,iverton.santos,gracaliz,dianaada}@gmail.com

Glenda Dimuro
e Esteban de Manuel Jerez
Depto de Expresión Gráfica Arquitectónica
Universidad de Sevilla
Sevilha, Espanha

***Abstract*—This paper presents a multi-agent system for the modeling and simulation of normative policies of the social organization of the urban vegetable garden San Jerónimo, located in Seville, Spain. For that, we developed an adapted version of the framework MSPP for the Modeling and Simulation of Public Policies, in order to be used in the modeling and simulation of normative policies that are internal to a single social organization.**

I. Introdução

Os Sistemas Multiagentes (SMA) oferecem ambientes computacionais onde programas que possuem certo grau de autonomia (agentes) interagem uns com os outros, no cumprimento de objetivos particulares e coletivos. [1], [2]

Este trabalho é parte de um projeto que tem como objetivo o desenvolvimento de ferramentas SMA para simulação de processos de produção e gestão social de ecossistemas urbanos, em particular, o projeto social da Horta Urbana "San Jerónimo", localizada em Sevilha, Espanha, coordenado pela ONG "Ecologistas en Acción".

O projeto tem o intuito de fomentar a participação social em práticas de agricultura orgânica. Atualmente os beneficiados com este projeto são principalmente hortelãos aposentados, alunos do ensino fundamental das escolas do bairro onde está localizado o projeto e associações para experimentos científicos. [3]–[5]

Os hortelãos, uma vez incluídos no projeto, têm direito a utilização da parcela (área designada ao cultivo e manejo da horta) por um prazo de dois anos prorrogáveis, desde que cumpram as normas e regras estabelecidas no regulamento definido pela ONG.

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. Nele estão incluídos quatro diferentes tipos de normas que são: as normas de direito, que concedem ao agente o direito de determinada ação ser executada sem que haja restrições e desde que não infrinja outras normas , as normas de permissão as quais o agente necessita requisitar junto a outros agentes a possibilidade de executar a ação prevista, normas de obrigação que são aquelas onde o agente é em determinado período obrigado a executar a ação que nela consta e normas de proibição as quais restrigem ações que os agentes possam vir a executar.

Em trabalhos anteriores do grupo de pesquisa, vários aspectos relativos à modelagem da organização social da horta, modelagem de rotinas dos papéis desta organização, modelos especiais de agentes, dentre outros aspectos, foram introduzidos. [6]–[8] Estes trabalhos foram desenvolvidos utilizando as diversas ferramentas que integram a plataforma JaCaMo (Jason, CArtAgO e MOISE+) [9].

Este artigo apresenta um SMA para simular as políticas internas ou normas regulamentares da organização social da Horta San Jerónimo, utilizando a plataforma Jason [10], um interpretador da linguagem AgentSpeak(L), baseada na arquitetura BDI [11], [12], juntamente com o *framework* CArtAgO [13] e o *framework* para prover a simulação de políticas públicas MSPP (Modeling and Simulation of Public Policies) [14], [15].

O *framework* MSPP é uma API para inserção de políticas públicas em um SMA que modela conjuntos de normas orientadas a aplicação de proibições e obrigações. Nele estão incluídos previamente agentes para execução, detecção e efetuação de normas, além de planos para tratar eventos relacionados. [1]

O artigo está organizado como descrito a seguir: A Seção 2 apresenta uma síntese sobre Sistemas Multiagentes, a linguagem AgentSpeak(L), a plataforma Jason, o *framework* CArtAgO e estrututa organizacional MOISE+. A Seção 3 aborda características do *framework* MSPP adaptado para inserção das políticas normativas da organização. A Seção 4 apresenta a simulação da Horta San Jéronimo e sua implementação com o *framework* MSPP. A Seção 5 apresenta a conclusão e trabalhos futuros.

[1] Veja [16], para uma breve discussão sobre a diferença e relações entre políticas públicas e normas sociais.

II. Sistemas Multiagentes e a Plataforma JaCaMo

Os agentes, de acordo com [10], são capazes de sentir o ambiente amplamente ou parcialmente e tomar ações que possam modificá-lo. São dotados de certa autonomia para essas ações, diferentemente de programas processuais, além de se comunicarem e se organizarem, o que se pode chamar de *habilidade social*.

Já os Sistemas Multiagentes (SMA) são ambientes *habitados* por vários agentes que são capazes de interagir, trocar informações, são sensíveis a percepções, se adaptam às mudanças, tem conhecimentos sobre o ambiente (pleno ou parcial) e podem tomar ações (coordenadas entre si ou não) para modificá-lo dentro da chamada *esfera de influência*.

Os SMA podem ser classificados da seguinte forma: Sistemas Multiagentes Reativos e Sistemas Multiagentes Cognitivos, sendo o último o modelo adotado neste trabalho.

Um aspecto que geralmente pode diferir um SMA cognitivo de um reativo é o fato de o primeiro trabalhar usualmente com poucos agentes, já no segundo costuma-se usar populações de agentes, que podem alcançar a ordem dos milhares.

O que o caracteriza ser um SMA cognitivo é o fato dos agentes possuírem crenças, percepções, comunicação, se organizarem em grupos, ter objetivos a serem alcançados, planos para poder atingir tais objetivos, ou seja, interagem e possuem conhecimentos e tem ações para modificar o ambiente.

O modelo de agente BDI *(Believe, Desires and Intentions)* é caracterizado pelo aspecto cognitivo, apresentando crenças, desejos e intenções. Crenças representam a informação que o agente tem sobre outros agentes e sobre o ambiente, os desejos expressam os objetivos que esse agente tenciona atingir, já as intenções são metas que o agente se comprometeu a cumprir.

A plataforma JaCaMo [9] é um *framework* para programação de Sistemas Multiagentes constituída de três ferramentas (Jason, CArtAgO e MOISE+).

A. Jason

O Jason [10] é um interpretador da linguagem AgentSpeak(L), baseada na arquitetura BDI. Quando se inicia a simulação em Jason, o agente percebe e atualiza suas crenças, isso significa atualizar tudo o que ele acredita ser *verdade* sobre o ambiente através de suas percepções, consequentemente gerando um evento, que por sua vez também pode desencadear um plano.

Antes de um plano ser iniciado, precisa-se de um objetivo, que pode ser tratado como fato desencadeador de um plano. Pode-se, por exemplo, ter um objetivo do tipo "!viajar" (o símbolo de exclamação é usado em Jason para notação de objetivo) que fará o agente atingir as etapas necessárias para que isso se realize, ou seja, que o agente realize uma viagem. O plano por sua vez é na verdade etapa(s) que o agente terá que cumprir para concluir um objetivo.

Um aspecto a ressaltar é que os agentes podem trocar mensagens entre si, podendo inclusive alterar a base de crença de outro agente, criando assim planos a partir do conhecimento adicionado por uma crença com informações enviadas por outro agente.

B. CArtAgO

O *framework* CArtAgO (Common ARTifact infrastructure for AGents Open environments) [13] 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. O ambiente como também os recursos disponíveis no mesmo podem ser modelados na forma de um artefato CArtAgO como foram feitos no presente trabalho.

C. Moise+

O modelo organizacional MOISE+ [17] é uma ferramenta com intuito de modelar a organização de um 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.

III. Modelagem e Simulação de Políticas Públicas

O *framework* MSPP [14], [15], utilizado neste trabalho, adota como fundamentação conceitual de Política Pública, a abordagem definida em [18], [19] a qual em termos gerais aborda o conceito de Políticas Pública como um conjunto de ações para buscar soluções para problemas da sociedade, orientando práticas e resguardando direitos a fim de atender as demandas e garantindo o direito coletivo.

O processo de criação e aplicação de políticas públicas sequencial o qual esse *framework* se baseia é concebida como uma sequência de etapas a serem realizadas, a cada momento, por um dos diferente atores ou conjunto deles envolvidos no processo. O ciclo de etapas é da seguinte forma [14]:

1. Identificação e formulação do problema a ser resolvido através da emissão e implementação de uma política pública;

2. Formulação e análise comparativa das várias possíveis políticas alternativas capazes de resolver o problema;

3. Escolha de uma das políticas para a implementação dessas;

4. Implementação da política pública escolhida;

5. Avaliação dos efeitos da implementação da política pública, e eventual ajustamento da política, para melhorar os resultados e reduzir os efeitos negativos (retornando assim o processo para a etapa 1).

Destaca-se a preocupação dos idealizadores do *framework* MSPP na limitação do modelo cíclico na obtenção de modelos que operam baseados em políticas públicas.

O *framework* para inserção de políticas públicas concretiza-se no formato de artefatos no modelo CArtAgO. Estão incluídos neste *framework* dois tipos de artefatos normativos que são: NormObrig e NormPrb, modelando normas de obrigação e proibição respectivamente. Além dos artefatos, estão previamente inseridos agentes para executar/verificar tais normas.

São eles o agente governamental, responsável por emitir as normas, os agentes sociais que estão submetidos às normatizações e buscam atingir objetivos próprios, e também os agentes governamentais detectores/efetores responsáveis por detectar o cumprimento das normas da política como também características e recursos do ambiente, aplicar possíveis sanções a ações que caracterizarem o descumprimentos de normas e por fim regularizar os recursos disponíveis no ambiente.

O MSPP *framework* pressupõe adotar estes quatro tipos de agentes interagindo para promover o ciclo de política.

As normas implementadas estão estruturadas da seguinte maneira em [15]:

Id: o identificador da norma;

Destinatário: especifica o papel ao qual a norma se aplica;

Ação: especifica uma ação a ser realizada pelo agente que assume o papel ao qual a norma foi endereçada;

Condição: especifica uma condição contextual necessária para a aplicação da norma;

Periodicidade: especifica o evento que deve ocorrer (mês, semana, ou uma ação específica) para que se verifique a condição;

Exceção: especifica uma condição na qual a norma não se aplica;

Sanção: especifica a sanção a ser aplicada no caso da violação da norma.

Os agentes sociais e também os agentes efetuadores/detectores estão constantemente a observar as normas como também tomam conhecimento de uma eventual modificação ou exclusão delas do sistema. O conhecimento destes sobre as normas é adicionado através de crenças onde se define que uma ação qualquer é proibida, obrigatória, ou se necessita ser observado o estado atual da permissão ou é resguardado o direito de executá-la. Uma vez cometida uma infração a essas normas, cabe ao agente detector e efetuador buscar junto ao artefato a devida sanção.

Por último os recursos públicos disponíveis no ambiente e até o mesmo devem estar disposto também na forma de um artefato CArtAgO a fim de estabelecer interação entre o sistema.

IV. O MODELO: A HORTA SAN JERÓNIMO

A Horta San Jerónimo é um projeto social coordenado pela ONG *Ecologistas en Acción*, na cidade de Sevilha, Espanha, e foi escolhida para este trabalho por ter um regulamento próprio que busca o melhor convívio e participação entre os seus agentes, além de resguardar seus direitos e atribuir-lhes restrições. Neste ambiente verificou-se também papéis e suas respectivas rotinas, hierarquias, além da regulamentação através de uma tabela de normas para implementação de uma política [6] [7].

Nesse contexto, baseado em [15] foram identificados dentre os papéis do projeto social características que se adequam aos papéis de agentes sociais, governamentais e também o agente emissor de políticas públicas, conforme explicado na Seção 3. Os agentes sociais são entendidos como os hortelões,

ID	Tipo de norma (constitutiva/ reguladora)	Ação Normatizada					Verificador da aplicação da norma (papel)	Sanções (Punições, Recompensas)
		Pré-condição da ação	Normatização da ação	Id da ação	Resultado da ação	Possível realizador da ação		
N16	Regulativa	Possuir uma horta de cultivo	Proibição	Utilização de mangueira na regagem	Continuar na horta ou sair (dependendo do número de faltas graves)	Hortelão	ONG (técnicos)	Falta grave e cumulativa
N28	Constitutiva	Possuir horta/Requerimento de auxílio	Permissão	Pedir auxílio a técnicos especializados	Recebimento de auxílio	Hortelão	ONG (técnicos)	
N29	Constitutiva	Possuir uma horta de cultivo	Proibição	Trabalhar em mais de uma parcela	Falta leve	Hortelão	ONG	Falta leve
N31	Constitutiva	Possuir uma horta de cultivo	Direito	Possuir tonel para água	Regar a horta	Hortelão	ONG (técnicos)	
N35	Regulativa	Possuir uma horta de cultivo	Obrigação	Organizar turnos de irrigação	Continuar na horta ou sair (dependendo do número de faltas graves)	Hortelão	ONG (técnicos, hortelãos e assembléia)	Falta grave e cumulativa

Fig. 1: Parte da Tabela de Normas

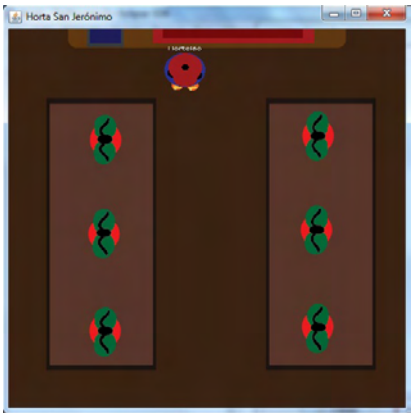

Fig. 2: Interface gráfica da Horta San Jerónimo

responsáveis pelo cultivo de suas parcelas (espaço destinado ao cultivo), já o agente emissor de políticas é entendido a ONG como um todo, e por último existem vários papéis governamentais tais como técnicos e secretaria e para este estudo englobou-se todos num único agente responsável pela administração.

A tabela de normas dessa organização conta um total de quarenta normas identificadas como tipos de proibição, obrigação, direito e permissão de acordo com [6]. Essa tabela regulamenta a entrada e saída de agentes do projeto, como também impõe obrigações e restrições ao uso dos recursos disponíveis no ambiente, resguarda direitos e define quais ações necessitam permissão para serem executadas.

A. O ambiente e sua visualização gráfica

O que chamamos de ambiente é um modelo na forma de um artefato CArtAgO que nada mais é que um aninhado de iterações simulando dias, meses e anos com condições referentes a época de plantio, do crescimento e colheita das hortaliças e aleatoriedade referentes ao clima. A cada evento gerado, a horta notifica os hortelões que a observam por *"signals"* uma propriedade do CArtAgO para inserção de crenças nos agentes. Quando isso acontece os hortelões adicionam essa nova crença em suas bases que desencadeiam planos, estes por sua vez são as ações que agentes governamentais devem verificar e que fazem parte (ou não) da rotina dos agentes.

Para a visualização gráfica foi usado o artefato horta integrando-o com propriedades do Java2D.

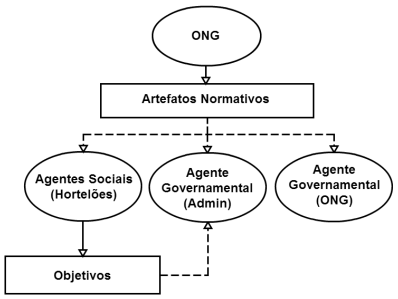

Fig. 3: O ciclo de ações e percepções dos agentes

A visualização gráfica permite observar o ciclo de crescimento das hortaliças e sua colheita. Trata-se de um trabalho em andamento e ainda escasso de recursos pois espera-se adicionar mais elementos na simulação.

B. Utilizando o framework MSPP no caso da Horta San Jerónimo

Verificou-se inicialmente que o *framework* MSPP poderia atender as necessidades normativas para o estudo de caso da Horta San Jerónimo, com exceção de algumas modificações.

A primeira é que foram identificados em [6] além dos tipos de normas de proibição e obrigação outros dois sendo eles as normas de direito e permissão para o caso. Normas de direito seriam aquelas que concedem ao agente social poder pleno de exercer aquela ação, sem haver restrições desde que não entre em conflito com as demais. Já as normas de permissão são aquelas que o agente necessita verificar o estado atual para poder executá-la. Por exemplo, consta na tabela de normas uma norma que determina que para o agente plantar árvores que tenham um ciclo maior que o de dois anos ele precisa de permissão.

Outra mudança foi o conteúdo que é passado para a criação da norma, ficando da seguinte forma: *Norma (Id; Tipo de norma: obrigatório, proibido, direito ou permissão; Ação; Sanção; Parâmetro de extensão da sanção)*. Dessa forma um exemplo de norma seria:

Norma (n08, proibido, venderProdutos, faltaGrave, graveCumulativa)

onde o agente ao vender produtos da horta que é uma ação proibida, receberia como execução uma falta grave e sua penalização seria o registro da mesma. Esse registro que também é um artefato, já consta no *framework* e para esse estudo de caso foi útil para convocar assembleias caso um agente reincida em executar uma ação cuja sanção é uma falta grave e cumulativa. Neste caso assumimos que sempre é convocada uma assembleia para decidir a permanência desse agente no sistema, onde os demais agentes se manifestam a favor da absolvição e consequentemente a permanência do hortelão, ou contrários optando pelo o afastamento, se metade ou mais dos agentes decidir que não, então ele é expulso, caso um empate aconteça é necessário a definição por parte de um agente governamental.

Destaca-se a omissão dos parâmetros inicialmente propostos pelo *framework*: Destinatário e Periodicidade. Estes dois parâmetros não foram incluídos inicialmente pois espera-se integrar funcionalidades disponíveis no modelo organizacional MOISE+ futuramente. Com a atribuição de papéis providas pelo modelo organizacional, torna-se ampla a designação de um destinatário. Já a periodicidade pelo mesmo motivo foi omitida, pois uma vez que os agentes estejam condicionados a papéis é possível estabelecer rotinas aos mesmos.

Para fins de testes foram inseridos um total de seis agentes com ações que explorassem funcionalidades chaves do MSPP *framework* como também ações que constassem no regulamento da horta. Como já dito, temos o agente ONG (agente governamental), o agente Admin (detector/efetuador), e as agentes sociais que são: Ortolan, OrtolanPoor, OrtolanGood, OrtolanLazy.

Para os agentes sociais foram atribuídas ações distintas para verificar a utilização dos quatro tipos de normas existentes (direito, permissão, obrigação, proibição), como também sanções cabíveis e condições possíveis.

Dessa forma consideramos o agente Ortolan neutro, podendo eventualmente infringir uma norma. O OrtolanPoor foi usado para tratar o caso de infração a uma norma proibida, e que a sanção seja uma falta grave e cumulativa, também foi usado normas que levassem a expulsão arbitrária. O OrtolanGood é aquele que jamais comete uma infração, sendo possível trabalhar com as normas de permissão e direito. Já o OrtolanLazy é aquele que descumpre obrigações.

Vale ressaltar que normas de obrigação muitas vezes são interpretadas como um caso de proibição. Por exemplo, é obrigação do agente social "pagar a mensalidade", e caso isso seja descumprido há uma sanção prevista.

Uma ressalva a se fazer é o modo como o *framework* identifica um infrator. Nos exemplos de implementação vistos o artefato que simula o ambiente envia um *"signal"*, e este por sua vez é acompanhado de um método proveniente do CArtAgO chamado *getOpUserName* que retorna o nome do agente que recebeu esse sinal. Já neste trabalho preferiu-se adotar outro método para a identificação das ações através da comunicação, ou seja, tudo o que o agente social faz ele comunica ao agente governamental responsável.

V. Conclusão

Este artigo apresenta um SMA para simular as políticas internas ou normas regulamentares da organização social da Horta San Jerónimo, utilizando a plataforma Jason juntamente com o *framework* CArtAgO e o *framework* para prover a simulação de políticas públicas MSPP (Modeling and Simulation of Public Policies).

Esperava-se que as normas previstas na Horta San Jerónimo pudessem ser modeladas e os agentes sejam estes sociais ou governamentais interagissem estando regidos por elas, podendo haver objetivos pessoais maiores que o cumprimento restrito destas normas e nesse caso que fosse identificado tal comportamento.

Observou-se que o *framework* MSPP atendeu as necessidades de modelagem de políticas normativas internas de um

sistema real, no caso o projeto social da Horta San Jerónimo. Tanto o ciclo de uma política adotado, quanto os agentes previstos para realizá-lo foram identificados no estudo de caso provendo resultados satisfatórios. No entanto foi necessária algumas modificações e adições a esta ferramenta para atender as necessidades desta modelagem como a criação dos artefatos normativos de direito e permissão, além da adequação dos parâmetros que constituem as normas.

Outro fator relevante foi a implementação do *framework* ter se dado como um artefato CArtAgO, proporcionando interação entre agentes e artefatos, artefatos e artefatos e artefatos e classes Java.

Ressalta-se a fácil adaptação tanto dos artefatos normativos quanto dos agentes previamente inseridos. Espera-se a partir desse trabalho torná-lo mais amplo e detalhista, inserindo novos recursos para que a simulação esteja mais próximo do modelo que a inspira.

A partir desse trabalho pretende-se ampliar o sistema a fim de verificar todas as ações com base à tabela de normas e consequentemente aumentar o ciclo de rotina dos agentes. Também se espera integrar o MOISE+ onde o modelo organizacional da Horta San Jerónimo já está explicitado em [6], a fim de criar um sistema aberto, onde agentes possam entrar e sair assumindo papéis definidos no modelo organizacional como também suas rotinas estejam previamente estipuladas. Ainda utilizando-se do modelo organizacional e também das hierarquias nele estabelecidas, serão usados artefatos de comunicação previstos em [8] como uma alternativa a comunicação.

Acknowledgment

This work was supported by CNPq (Proc. 560118/10-4, 305131/2010-9, 476234/2011-5) , FAPERGS (Proc. 11/0872-3) and Projeto RS-SOC (FAPERGS Proc. 10/0049-7).

References

[1] Wooldridge, M.: An Introduction to MultiAgent Systems. Wiley, Chichester (2002)

[2] Singh, M.P., Rao, A.S., Georgeff, M.P.: Formal methods in DAI: Logic-based representation and reasoning. In Weiss, G., ed.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press, Cambridge (1999) 331–376

[3] Dimuro, G., Jerez, E.M.: La comunidad como escala de trabajo en los ecosistemas urbanos. Revista Ciencia y Tecnología **10** (2011) 101–116

[4] Dimuro, G.: Sistemas urbanos: el estado de la cuestión y los ecosistemas como laboratorio. Arquitextos **124** (2010) 11

[5] Dimuro, G., Jerez, E.M.: Comunidades en transición: Hacia otras prácticas sostenibles en los ecosistemas urbanos. Cidades Comunidades e Territórios **20-21** (2010) 87–95

[6] Santos F.C.P., Rodrigues, T.D.G.A.D.D.G.R.A.e.J.E.M.: Modelando organização social de um sma para simulação dos processos de produção e gestão social de um ecossistema urbano: o caso da horta san jerónimo da cidade de sevilla, espanha. In H J.F., ed.: VI Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações - WESAAC 2012, FLORIANÓPOLIS, UFSC (2012) 93–104

[7] Santos, I., Rodrigues, T.F., Dimuro, G.P., Costa, A.C.R., Dimuro, G., Manuel, E.: Towards the modeling of the social organization of an experiment of social management of urban vegetable gardens. In Lugo, G., Hübner, J., eds.: 2011 Workshop and School of Agent Systems, their Environment and Applications (WESAAC) Proceedings, Los Alamitos, IEEE (2012) 98–101

[8] Rodrigues, T.F., Costa, A.C.R., Dimuro, G.P.: A communication infrastructure based on artifacts for the jacamo platform. In Cossentino, M., Seghrouchni, A.E.F., Winikoff, M., eds.: Proceedings of EMAS 2013 - 1st International Workshop on Engineering Multi-Agent Systems at AAMAS 2013, Saint Paul, IFAMAS (2013) 1–15

[9] Bordini, R.H., Hübner, J.F.: (JaCaMo project) Available at http://jacamo.sourceforge.net/, accessed in September 2012.

[10] Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley, New Jersey (2007)

[11] Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In van Hoe, R., ed.: Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World. Volume 1038 of LNCS. Springer, Berlin (1996) 42–55

[12] Rao, A.S., Georgeff, M.P.: An abstract architecture for rational agents. In Nebel, B., Rich, C., Swartout, W.R., eds.: Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92), Cambridge, MA, October 25–29, 1992, Morgan Kaufmann (1992) 439–449

[13] Ricci, A., Santi, A., Piunti, M.: CArtAgO (common atifact infrastructure for agents open environments) (2013)

[14] Santos, I., Rocha, A.C.R.: 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, Berlin, Springer (2012) 45–59

[15] Santos, I. A. S., M.F.P.C.A.C.R.e.D.G.P.: Um framework para simulação de políticas públicas aplicado ao caso da piracema, sob o olhar da teoria dos jogos. In: Anais do IX Encontro Nacional de Inteligência Artificial. (2012)

[16] Young, H.P.: Social norms and public policy (2007)

[17] Hübner, J.F.: Um Modelo de Reorganização de Sistemas Multiagentes. PhD thesis, Universidade de São Paulo, São Paulo (2003)

[18] Hill, M.: The Public Policy Process. 4th edn. Pearson Longman (2004)

[19] Easton, D.: A Framework for Political Analysis. Prentice-Hall (1965)

TrustE - An Emotional Trust Model for Agents

Guilherme K. Bitencourt
Departamento de Informática
e Estatística
Universidade Federal
de Santa Catarina
Email: bitencourt@inf.ufsc.br

Ricardo A. Silveira
Departamento de Informática
e Estatística
Universidade Federal
de Santa Catarina
Email: silveira@inf.ufsc.br

Jerusa Marchi
Departamento de Informática
e Estatística
Universidade Federal
de Santa Catarina
Email: jerusa@inf.ufsc.br

***Abstract*—Trust and Reputation has been proposed in the Multiagent System area as a way for assisting agents to select good partners in order to increese the well successed interactions between agents. As well as trust, agent emotions has been studied with the intention to turn actions and reactions of the agents more human like. In this paper, we present a trust emotional based model. This proposal is an hibrid model that congregates a mathematical and a symbolical models for capturing the complexity of the reasoning. Quantitative and qualitative evalutations are mixed throught the incorporation of some emotional flavors in the trust evaluation.**

I. Introdução

O estudo e modelagem da confiança tem atraído o interesse de pesquisadores em diversas áreas, tais como psicologia, sociologia, filosofia e economia, e possui grande importância nas relações sociais e comerciais [1]. Em Ciências da Computação esse interesse ocorre principalmente nos Sistemas Multiagente (SMA).

Sistemas Multiagente Abertos podem receber novos agentes a qualquer momento. Esta característica torna difícil, para um agente, verificar se outro recém ingressado no sistema é confiável, pois não existem informações suficientes referentes ao novo agente. Para amenizar essa dificuldade, vários modelos de confiança e reputação vem sendo desenvolvidos na área [2]–[4]. Considerando que os sistemas computacionais estão se tornando sistemas de larga escala, abertos, dinâmicos e distribuídos, contendo uma grande quantidade de agentes que agem por interesse próprio [3], a utilização da confiança a da reputação nesses sistemas torna-se fundamental para que haja uma efetiva interação entre os agentes [5].

Contudo, a maioria dos modelos de confiança e reputação existentes possui um enfoque essencialmente matemático [6], não levando em consideração a complexidade presente na maneira como nós, humanos, confiamos ou não uns nos outros. Para nós, as emoções influenciam diretamente o ato de confiar. Isto, de certa forma, impõe um caráter subjetivo à avaliação da confiança. Além disso, os modelos propostos na literatura, determinam, através de pesos arbitrários e/ou dependentes de funções, a relevância das informações, oriundas de outros agentes, utilizadas no cálculo de confiança pelo agente avaliador. Esse fato torna parte da ação de confiar independente do comportamento do agente e portanto dissociada do contexto no qual o agente está inserido.

Neste trabalho, é proposto o modelo **TrustE** de confiança que utiliza informações de natureza simbólica, que estão relacionadas diretamente com o contexto no qual o agente está inserido. Tais informações compõem sensações ou emoções que possibilitam ao agente associar às avaliações quantitativas ou racionais, avaliações qualitativas ou subjetivas, que são fruto da introspecção das situações vividas pelo agente.

O artigo está organizado da seguinte forma: na seção II, são apresentados os conceitos de confiança e reputação entre agentes. Na seção III será introduzido o conceito de emoções em agentes, apresentando o modelo OCC. O modelo proposto é apresentado na seção IV. Por fim, a seção V apresenta as considerações finais.

II. Modelos de Confiança

Os modelos de confiança existentes podem ser divididos em dois grupos: modelos baseados em confiança e modelos baseados em reputação. Embora os termos confiança e reputação possam se confundir, a principal diferença entre eles está na origem da informação. A confiança está relacionada a uma relação direta entre dois agentes a e b, e é gerada através de experiências e interações ocorridas entre eles. Portanto a ação de um agente a confiar ou não em um agente b depende somente da análise dessas interações pelo agente a (a origem da informação é o próprio agente a), não sendo considerada a reputação de b. A reputação, por sua vez, é uma confiança socializada (a origem da informação são os outros agentes), que são transmitidas entre os agentes, possibilitando que a confie em b, sem a necessidade de ambos terem tido alguma interação direta no passado. Assim, a reputação de um agente se constrói a partir de informações provenientes de vários agentes, e através dela, o agente é capaz de decidir se confia ou não em um outro agente [6].

São vários os modelos de confiança e reputação encontrados na literatura, dentre os quais destacam-se: o modelo de Marsh [7], que considera apenas a confiança de cada agente; o modelo SPORA [8], que considera somente a reputação dos agentes; os modelos REGRET [2], Referral Network [9] e TRAVOS [3] que levam em consideração tanto a confiança quanto a reputação, combinando esses valores para chegar a um resultado; e finalmente o modelo FIRE [5], que introduz dois conceitos adicionais, a confiança baseada em papéis e

a reputação certificada. A seguir, apresenta-se uma descrição sucinta destes modelos.

A. Modelo de Marsh

O modelo proposto por [7] foi um dos primeiros modelos desenvolvidos sobre confiança local (considera apenas a interação direta entre agentes para medir a confiança). Sua arquitetura é distribuída, uma vez que, cada agente é o responsável pelo cálculo da sua confiança perante os outros agentes.

Ele diferencia três tipos de confiança: *confiança básica* - representa a disposição de um agente confiar em outro; *confiança geral* - a confiança que um agente exerce sobre outro sem levar em conta qualquer situação específica; e *confiança situacional* - a confiança que um agente tem em relação a outro, levando em consideração uma situação específica.

Esse modelo, por levar em consideração apenas a confiança local (direta) entre agentes, é limitado em relação a capacidade de calcular a confiança quando os agentes nunca interagiram entre si. No entanto, sua citação é importante devido ao seu pioneirismo na área e a sua definição de confiança direta entre agentes.

B. TRAVOS

O modelo de confiança e reputação TRAVOS [3] foi desenvolvido para ser utilizado, principalmente, em SMA abertos e assume que o comportamento dos agentes não mudam com o tempo, no entanto, essa hipótese nem sempre é verdadeira. A sua principal característica - que também está presente nos modelos REGRET [2] e FIRE [5] - é a possibilidade de um agente avaliar a confiança em outro agente de forma direta, através de experiências passadas, ou através da reputação, quando tais experiências não existirem ou forem insuficientes. Ao estabelecer a confiança em outros agentes e escolher aquele que é mais confiável, o agente tem a capacidade de maximizar a probabilidade de que sua interação seja bem sucedida. Outra característica presente é a possibilidade de filtrar as opiniões imprecisas de outros agentes, permitindo que a reputação seja utilizada para aumentar significativamente o desempenho do sistema, já que as informações indesejáveis serão descartadas.

Neste modelo, um agente a_{tr} (*truster*) possui dois métodos para calcular a confiança em outro agente a_{te} (*trustee*) em um contexto específico. Primeiramente o agente a_{tr} faz a avaliação baseado nas interações diretas com o agente a_{te}, depois o agente a_{tr} avalia a confiabilidade de a_{te} através da reputação de a_{te}.

A reputação é necessária quando um agente a_{tr} quer fazer uma avaliação sobre o comportamento de um agente a_{te}, e possui poucas informações sobre ele ou a confiança entre ambos possui uma baixa confiabilidade. Assim, a reputação pode aumentar a precisão da confiabilidade do valor de confiança que a_{tr} tem em a_{te}, auxiliando a_{tr} na sua tomada de decisão.

A confiança é modelada através de uma abordagem probabilística, baseada nas experiências passadas de um agente sendo avaliado. Se um agente avaliador (a_{tr}), tem todas as informações sobre o agente sendo avaliado (a_{te}), então, de acordo com a_{tr}, a probabilidade de a_{te} cumprir com suas obrigações é expressa por $B_{a_{tr},a_{te}}$. Contudo, normalmente não se tem toda a informação necessária sobre a_{te}, logo, o melhor caminho a seguir é utilizar o *valor esperado* de $B_{a_{tr},a_{te}}$ dado o conhecimento (conjunto de todos os resultados das interações observadas) de a_{tr}.

C. FIRE

O modelo FIRE, proposto por [5], é um modelo de confiança e reputação integrado, com uma arquitetura de tomada de decisão distribuída entre os agentes, semelhante ao modelo REGRET [2]. Este modelo incorpora quatro fontes de informação: confiança por interação, reputação baseada em testemunho, confiança baseada em papéis e reputação certificada, as quais são combinadas para fornecer uma métrica de confiança em praticamente todas as circunstâncias. Essa variedade de fontes torna-se importante, visto que, em várias situações nem todas estarão prontamente disponíveis, além de permitir aos agentes combiná-las para lidar com as incertezas do ambiente.

D. REGRET

Neste modelo, a reputação é vista como uma opinião ou visão de um agente sobre algo, sendo formada e atualizada ao longo do tempo através das interações com os outros agentes do sistema. As interações fornecem como resultado *impressões* que são registradas pelos agentes e refletem como eles avaliam suas experiências com outros agentes, de acordo com o resultado de um diálogo (contrato inicial que estabelece os termos e condições de uma transação) firmado entre os agentes. Como cada agente possuiu uma opinião diferente dos demais, pode-se dizer que a reputação assume um caráter mais subjetivo.

Um resultado, o_b, de um diálogo relacionado a uma transação comercial, entre dois agentes a e b, do ponto de vista do agente comprador b, poderia ser:

$$o_b = (DataEntrega =_c 10/02 \wedge Preco =_c 2000 \wedge Qualidade =_c A \wedge DataEntrega = 15/02 \wedge Preco = 2000 \wedge Qualidade = C)$$

Nesse exemplo, as variáveis com o subscrito c representam o acordo inicial entre ambos os agentes. Assim o agente b esperava um produto com qualidade A (boa) porém recebeu um produto de qualidade C (ruim), além de ter recebido o produto com 5 dias de atraso.

O modelo utiliza o termo Reputação Individual (RI) para representar a confiança direta entre dois agentes, e Reputação Social (RS) para representar a reputação propriamente dita. A abordagem descentralizada implementada por este modelo permite a cada agente calcular a RI e a RS de outro agente, podendo utilizar ambas ou apenas uma delas para se chegar a um resultado final.

A RS leva em consideração três fontes de informação: a interação do agente a (avaliador) com os membros do grupo que o agente b (avaliado) pertence, expressa por $R_{a\to B}(\varphi)$, equação 1. O que os membros do grupo A (o grupo do agente a) pensam sobre o agente b, expressa por $R_{A\to b}(\varphi)$, equação 2. E o que os membros do grupo A pensam sobre o outro grupo B, expressa por $R_{A\to B}(\varphi)$, equação 3.

$$R_{a\to B}(\varphi) = \sum_{b_i \in B} \omega^{ab_i}.R_{a\to b_i}(\varphi) \quad (1)$$

$$R_{A\to b}(\varphi) = \sum_{a_i \in A} \omega^{a_i b}.R_{a_i\to b}(\varphi) \quad (2)$$

$$R_{A\to B}(\varphi) = \sum_{a_i \in A} \omega^{a_i B}.R_{a_i\to B}(\varphi) \quad (3)$$

Esses 3 valores provenientes dessas fontes de informação são combinados com a RI do agente, denotada por $R_{a\to b}(\varphi)$, para se chegar ao valor final de confiança, representado por $SR_{a\to b}(\varphi)$, equação 4.

$$SR_{a\to b}(\varphi) = \xi_{ab}.R_{a\to b}(\varphi) + \xi_{aB}.R_{a\to B}(\varphi) + \xi_{Ab}.R_{A\to b}(\varphi) + \xi_{AB}.R_{A\to B}(\varphi) \quad (4)$$

onde $\xi_{ab}+\xi_{aB}+\xi_{Ab}+\xi_{AB} = 1$, e representam a importância de cada uma das fontes de informação para o agente avaliador, sendo essa decisão dependente de aplicação.

As características deste modelo permitem que à ele sejam incorporadas avaliações simbólicas, na forma de emoções. Na seção seguinte é apresentado um modelo de emoções para agentes, que conjuntamente ao modelo REGRET, forma a base para a proposta do modelo *TrustE*.

III. Emoções em Agentes

O estudo das emoções está presente em várias disciplinas como psicologia, economia, neurociência cognitiva, e nos últimos anos esse estudo também está presente nas pesquisas em IA e Ciência da Computação. Tal estudo visa a criação de sistemas de interação emocional, como por exemplo, robôs com comportamento emocional e agentes virtuais para entretenimento [10].

O modelo psicológico de emoções, conhecido como *OCC*, proposto por [11], tem ganhado popularidade entre pesquisadores que desenvolvem sistemas de raciocínio sobre emoções ou que incorporam emoções em agentes artificias. O modelo classifica 22 tipos de emoções, sendo metade destas positivas (ex.: alegria e esperança) e metade negativas (ex.: tristeza e medo). Duas das emoções descritas no modelo OCC são:

- $ALEGRIA$ - *satisfeito* em relação a um evento desejável;
- $MEDO$ - *insatisfeito* em relação a um evento indesejável.

Há uma relação de temporalidade nas emoções, enquanto a $ALEGRIA$ está relacionada a algo que está acontecendo, o $MEDO$ refere-se a algo que poderá acontecer.

Quanto aos aspectos quantitativos das emoções, estes são descritos no modelo OCC em termos de potencialidades, limiares e intensidades. Para cada uma das 22 emoções, é fornecida uma lista de variáveis que afetam a intensidade da emoção e quais as condições necessárias para que a emoção ocorra.

Assim, intensidade de uma emoção é definida subtraindo-se o *limiar* de seu *potencial*. O modelo OCC não especifica como são calculados os *limiares* das emoções, porém acredita-se que eles dependam de variáveis globais que indicam o humor do agente [12]. Por exemplo, se um agente está de bom humor, os limiares das emoções negativas aumentam, causando uma diminuição na intensidade dessa emoções. Quando uma condição necessária para disparar uma emoção ocorre, mas o seu *potencial* está abaixo do seu *limiar*, um agente pode reconhecer que essa emoção foi desencadeada porém ela não o afetará. Por exemplo, "o humor de um agente estava tão bom que mesmo ele tendo praticado uma ação ruim, ele não foi afetado pela vergonha".

Para cada emoção, são necessárias 3 funções para o calculo da *intensidade*. São elas: *função de potencialidade*, *função do limiar* e *função de intensidade*. A maneira como essas 3 funções são calculados é dependente de aplicação, porém, de um modo geral, a *função de intensidade* de uma emoção, denotada por $I(P(E), L(E), t)$, que por simplificação será representada por $I(E)$, pode ser declarada como:

$$I(P(E), L(E), t) \to \mathbb{R}^+ \quad (5)$$

onde o parâmetro $P(E)$ representa a *função de potencialidade* da emoção E, $L(E)$ a *função do limiar* de E, e t o tempo corrente. Como resultado, a função retorna um real positivo, incluindo 0 (zero), que representa, de forma quantitativa, a intensidade da emoção. Segundo [12], o valor de $I(E)$ persiste ao longo do tempo, e tende a diminuir com o passar dele, diferentemente dos valores de $P(E)$ e $L(E)$ que são recalculados a cada vez que E é disparada novamente, portanto esses valores não persistem com o passar do tempo. Para suportar a temporalidade da intensidade cada agente deve possuir uma memória (MEA) que armazene os valores de cada uma das emoções sentidas por ele.

A. Nova hierarquia de emoções

Contudo, para que as emoções possam ser utilizadas em agentes inteligentes é necessário formalizá-las, possibilitando a sua implementação. Steunebrink et al. [13] propuseram uma revisão do modelo OCC, visando adequar o modelo a implementações computacionais. Neste trabalho são identificadas as ambiguidades existentes na estrutura lógica do OCC e são propostas alterações, conduzindo o modelo a uma estrutura baseada em herança (Figura 1), suportada por uma nova estrutura lógica, e novas especificações dos tipos de emoções. A partir dessas alterações, as 22 emoções do modelo OCC foram formalizadas, criando-se um modelo qualitativo de emoções, descrevendo precisamente quando as emoções são desencadeadas [14].

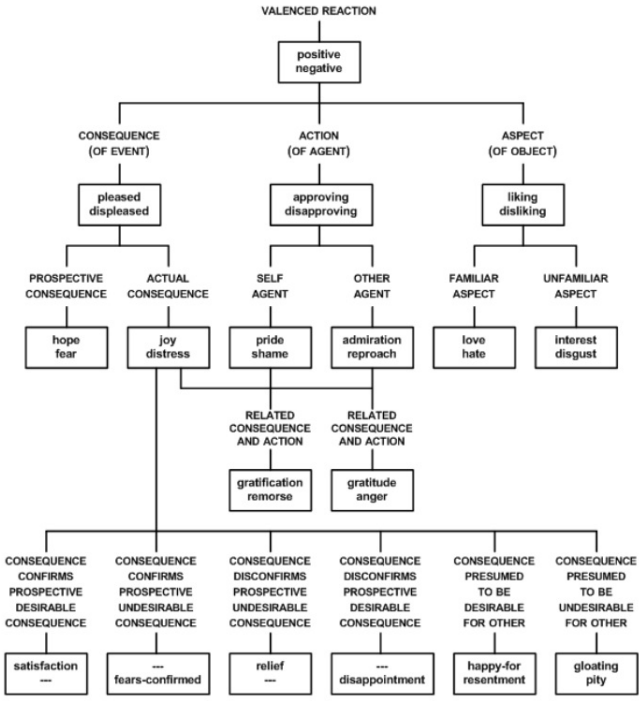

Figure 1. Hierarquia baseada em herança de emoções do modelo OCC modificado [13]

As principais alterações na estrutura do modelo foram: a introdução de herança explícita na composição hierárquica das emoções; o uso de rótulos em cada ponto da hierarquia e definição de nós filhos como superconjunto dos nós pais. A partir dessa nova estrutura foram criada novas especificações dos tipos de emoções, conforme a figura 2. Tais modificações possibilitam uma melhor avaliação das emoções pois torna o modelo mais formal e com uma maior linearidade na composição das emoções.

Como pode ser visto nas figuras 1 e 2, as emoções SATISFEITO (”PLEASED”), INSATISFEITO (”DISPLEASED”), APROVAÇÃO (”APPROVING”), DESAPROVAÇÃO (”DISAPPROVING”), GOSTO (”LIKING”) e DESGOSTO (”DISLIKING”) que no modelo original OCC fazem parte do processo de avaliação e não representam uma emoção propriamente dita, na alteração proposta por Steunebrink et al. [13], esses sentimentos são tratados como emoções e são a base para todas as outras emoções. A diferença entre a ALEGRIA e ESPERANÇA, por exemplo, é que a ALEGRIA está relacionada a uma consequência que está ou já aconteceu, enquanto a ESPERANÇA está relacionada a uma consequência que poderá acontecer. O mesmo raciocínio vale para TRISTEZA e MEDO. Por exemplo, ”O homem estava alegre porque seu time ganhou e tinha esperança de ver seu time ser campeão ao final do campeonato”.

positive and *negative* are valenced reactions (to “something”)
pleased is being *positive* about a consequence (of an event)
displeased is being *negative* about a consequence (of an event)
hope is being *pleased* about a prospective consequence (of an event)
fear is being *displeased* about a prospective consequence (of an event)
joy is being *pleased* about an actual consequence (of an event)
distress is being *displeased* about an actual consequence (of an event)
satisfaction is *joy* about the confirmation of a prospective desirable consequence
fears-confirmed is *distress* about the confirmation of a prospective undesirable consequence
relief is *joy* about the disconfirmation of a prospective undesirable consequence
disappointment is *distress* about the disconfirmation of a prospective desirable consequence
happy-for is *joy* about a consequence (of an event) presumed to be desirable for someone else
resentment is *distress* about a consequence (of an event) presumed to be desirable for someone else
gloating is *joy* about a consequence (of an event) presumed to be undesirable for someone else
pity is *distress* about a consequence (of an event) presumed to be undesirable for someone else
approving is being *positive* about an action (of an agent)
disapproving is being *negative* about an action (of an agent)
pride is *approving* of one’s own action
shame is *disapproving* of one’s own action
admiration is *approving* of someone else’s action
reproach is *disapproving* of someone else’s action
gratification is *pride* about an action and *joy* about a related consequence
remorse is *shame* about an action and *distress* about a related consequence
gratitude is *admiration* about an action and *joy* about a related consequence
anger is *reproach* about an action and *distress* about a related consequence
liking is being *positive* about an aspect (of an object)
disliking is being *negative* about an aspect (of an object)
love is *liking* a familiar aspect (of an object)
hate is *disliking* a familiar aspect (of an object)
interest is *liking* an unfamiliar aspect (of an object)
disgust is *disliking* an unfamiliar aspect (of an object)

Figure 2. Especificações dos tipos de emoções do modelo OCC modificado [13]

IV. TrustE: um modelo de Confiança baseado em Emoções

O modelo TrustE, proposta deste trabalho, agrega emoções ao cálculo da confiança e da reputação em agentes. Todos os modelos encontrados na literatura fazem uso de análise algébrica para determinar o ato de confiar, o que torna tais modelos essencialmente matemáticos e desconectados do histórico de agente. A proposta do modelo TrustE é permear o modelo de confiança com avaliações oriundas de raciocínio simbólico, tornando o ato de confiar mais dinâmico e dependente do histórico do agente.

A incorporação de emoções pode ser feita de várias formas, contudo optou-se por incorporar ao modelo REGRET, fatores emocionais, baseados nas *intensidades das emoções* do agente. Assim, os pesos definidos pelo programador, ξ e ω das equações do modelo REGRET (Subseção II-D) são substituídos por símbolos emocionais.

A. Formalização das emoções no TrustE

Embora o modelo OCC defina 22 tipos de emoções, para implementação do modelo TrustE foram utilizadas 4 tipos de emoções, *ALEGRIA*, *TRISTEZA*, *ADMIRAÇÃO* e *DESCONSIDERAÇÃO*. Forem escolhidas essas emoções pois as duas primeiras representam um tipo de emoção que não está ligada diretamente a ação de outro agente (ex. O agente estava feliz por ter atingido seus objetivos), enquanto as duas últimas são emoções ligadas a atitude de um determinado agente em relação a outro (ex. O agente comprador admirou a honestidade do agente vendedor). A seguir serão dadas algumas definições necessárias para a formalização dessas 4 emoções.

- Sejam a e b agentes, X um evento, Y uma ação.
- Se a confia em b então $CONF_a(b)$ senão $\neg CONF_a(b)$.
- $POS_a(Y)$ - uma ação Y positiva realizada por a.

- $NEG_a(Y)$ - uma ação Y negativa realizada por a.
- $EVENT(X)$ - um evento (fato) X já realizado.
- $SAT_a(X)$ - a $SATISFACAO$ de a em relação a consequência de $EVENT(X)$.
- $\neg SAT_a(X)$ - a $INSATISFACAO$ de a em relação a consequência de $EVENT(X)$.
- $AP_a(Y, b)$ - a $APROVACAO$ de a em relação a uma ação Y de b. O agente a irá ter uma *reação positiva* em relação a Y, quando b realizar uma *ação positiva* do ponto de vista do agente a.

$$AP_a(Y, b) \leftrightarrow POS_b(Y) \wedge \neg NEG_b(Y) \quad (6)$$

- $DES_a(Y, b)$ - a $DESAPROVACAO$ de a em relação a uma ação Y de b. O agente a irá ter uma *reação negativa* em relação a Y, quando b realizar uma *ação negativa* do ponto de vista do agente a.

$$DES_a(Y, b) \leftrightarrow NEG_b(Y) \wedge \neg POS_b(Y) \quad (7)$$

Logo será admitido que $AP_a(Y, b) \leftrightarrow \neg DES_a(Y, b)$. Assim será utilizado a notação $AP_a(Y, b)$ para $APROVACAO$ e $\neg AP_a(Y, b)$ para $DESAPROVACAO$.

A formalização e a condição de ocorrência de cada uma das 4 emoções são descritas a seguir:

Sendo $ADM_a(Y, b)$ a $ADMIRACAO$ de a em relação a uma ação Y realizada por b. O agente a irá ADMIRAR Y, quando a APROVAR Y.

$$ADM_a(Y, b) \leftrightarrow AP_a(Y, b) \quad (8)$$

Sendo $DES_a(Y, b)$ a $DESCONSIDERACAO$ de a em relação a uma ação Y realizada por b. O agente a irá DESCONSIDERAR Y, quando a REPROVAR Y.

$$DES_a(Y, b) \leftrightarrow \neg AP_a(Y, b) \quad (9)$$

Sendo $AL_a(X)$ o sentimento de $ALEGRIA$ do agente a em relação a consequência atual de um evento X. O agente a irá sentir-se ALEGRE quando ficar SATISFEITO em relação a X.

$$AL_a(X) \leftrightarrow SAT_a(X) \wedge EVENT(X) \quad (10)$$

Sendo $TR_a(X)$ o sentimento de $TRISTEZA$ do agente a em relação a consequência atual de um evento X. O agente a irá sentir-se TRISTE quando ficar INSATISFEITO em relação a X.

$$TR_a(X) \leftrightarrow \neg SAT_a(X) \wedge EVENT(X) \quad (11)$$

No escopo utilizado para a explicação do modelo TrustE, as ações e eventos importantes no processo do cálculo de confiança são:

1) *a ação Y (positiva) de um agente em cumprir os termos firmados em um diálogo*;
2) *a ação Y (negativa) de um agente em não cumprir os termos firmados em um diálogo*;
3) *a ação Y de um agente confiar em outro agente*;
4) *o fato X de um agente receber a confiança de outro agente*;
5) *o fato X de um agente ter cumprido os termos firmados em um diálogo*;
6) *o fato X de um agente não ter cumprido os termos firmados em um diálogo*;

De certa forma, os eventos (5 e 6) se confundem com as ações (1 e 2), no entanto essas ações, que são realizadas pelo agente avaliado, são as responsáveis por dispararem os eventos que produzem consequências sentidas pelo agente avaliador.

Utilizando-se o modelo em um escopo mais abrangente, como por exemplo, numa simulação de bolsa de valores, existiriam outras ações e/ou eventos que não estariam relacionados diretamente com o cálculo da confiança. Por exemplo, $POS_b(Y)$ poderia representar a ação de b indicar ao agente a, uma venda de ações com baixo preço, enquanto $EVENT(X)$ poderia representar o lucro de a por ter comprado essas ações. Como consequência, a sentiria-se ALEGRE e teria uma ADMIRACAO por b.

B. Detalhamento do Modelo TrustE

A figura 3 mostra uma visão conceitual do TrustE que demonstra o fluxo de ações dos agentes. Para explicar esse fluxo será utilizado um cenário no qual um agente quer comprar um produto, de boa qualidade, pela Internet. Seja a um agente comprador que irá escolher um agente vendedor b entre todos os vendedores existentes (etapa 1). Para saber se a deve ou não comprar o produto de b, ele irá calcular a confiança em b (etapa 2). Caso o valor da confiança seja baixo, a irá procurar outro agente vendedor, caso ela seja alta, a irá comprar o produto de b (etapa 3). Ao receber a confiança de a o agente b irá aumentar suas emoções positivas (etapa 4). Passado algum tempo, b irá entregar o produto ao agente a (etapa 5), esse por sua vez irá avaliar a qualidade do produto (etapa 6). Se ele considerar a qualidade boa, suas emoções positivas irão aumentar (etapa 7) e sua avaliação a respeito de b será positiva (etapa 8), caso contrário suas emoções negativas irão aumentar e sua avaliação a respeito de b será negativa.

A figura 4 mostra os dois módulos existentes no TrustE, o módulo de confiança (MC) e o módulo de emoções (ME). O primeiro é o responsável pelo cálculo da Reputação Individual (RI) e Reputação Social (RS), que juntas irão resultar no valor final da confiança (VFC). O segundo contém as funções de intensidades das emoções $I(E)$ e a memória das emoções dos agentes (MEA), sendo que cada agente possui sua própria MEA.

Conforme explicado anteriormente, o modelo REGRET possui 4 valores para formar a confiança final, representados na figura por $R_{a\rightarrow b}$, $R_{a\rightarrow B}$, $R_{A\rightarrow b}$ e $R_{A\rightarrow B}$. O primeiro peso utilizado no cálculo da RI $R_{a\rightarrow b}$ é o mesmo utilizado no REGRET, que atribui valores maiores às avaliações dadas recentemente. Já os primeiros pesos utilizados nas Reputações Sociais $R_{a\rightarrow B}$, $R_{A\rightarrow b}$ e $R_{A\rightarrow B}$, são calculados a partir das IEA. A escolha por esse grupo de emoções (*emoções-de-ações*) para esses pesos, se deve ao fato desse tipo de emoção estar relacionado a algum agente em específico, assim é possível que cada agente consiga avaliar os outros levando em consideração as emoções relacionadas a cada um deles.

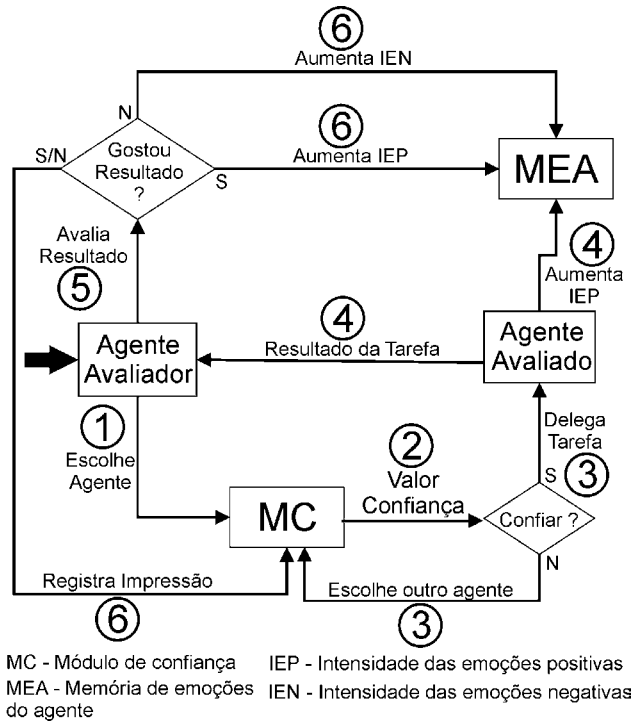

Figure 3. Visão Conceitual do Modelo TrustE

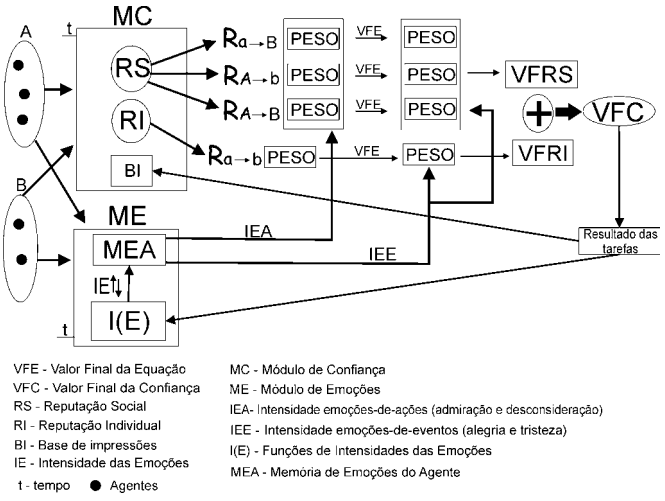

Figure 4. Modelo TrustE

Depois de calculados esses 4 valores, eles são combinados para formar o VCF. Os pesos utilizados nessa parte do cálculo levam em consideração as IEE, já que esse grupo de emoções (*emoções-de-eventos*) não está relacionado diretamente a nenhum agente em específico e sim a eventos ocorridos que modificam o estado emocional do agente. Devido ao fato de acreditar-se que quando estamos alegres temos uma maior tendência em confiar nos outros, quando um agente sentir alegria ele irá dar um peso maior para a RS, e quando sentir tristeza será dado um peso maior a RI.

V. Considerações Finais

Esse artigo apresentou o modelo híbrido TrustE, que é um modelo de confiança, baseado no REGRET, que utiliza as emoções do agente no mecanismo de cálculo da confiança, visando capturar a complexidade do raciocínio humano e flexibilizar as avaliações quantitativas, introduzindo elementos de natureza qualitativa ao modelo.

Acredita-se que a inserção de emoções e a utilização das suas intensidades possam propiciar um maior realismo ao modelo, pois a tomada de decisão do agente estará diretamente ligada ao seu estado emocional. Como próxima etapa, o modelo deverá ser implementado e validado em cenários de negociação entre agentes. O modelo originalmente proposto pode ainda ser expandido para comportar um maior número de emoções.

References

[1] E. Ostrom, "A behavioral approach to the rational choice theory of collective action: Presidential address, american political science association, 1997," *American Political Science Review*, pp. 1–22, 1998.

[2] J. Sabater and C. Sierra, "Regret: reputation in gregarious societies," in *Proceedings of the fifth international conference on Autonomous agents*. ACM, 2001, pp. 194–195.

[3] W. Teacy, J. Patel, N. Jennings, and M. Luck, "Travos: Trust and reputation in the context of inaccurate information sources," *Autonomous Agents and Multi-Agent Systems*, vol. 12, no. 2, pp. 183–198, 2006.

[4] J. Sabater and C. Sierra, "Review on computational trust and reputation models," *Artificial Intelligence Review*, vol. 24, no. 1, pp. 33–60, 2005.

[5] T. Dong-Huynha, N. Jennings, and N. Shadbolt, "Fire: An integrated trust and reputation model for open multi-agent systems," in *ECAI 2004: 16th European Conference on Artificial Intelligence, August 22-27, 2004, Valencia, Spain*, vol. 110. Ios Pr Inc, 2004, p. 18.

[6] G. Lu, J. Lu, S. Yao, and Y. Yip, "A review on computational trust models for multi-agent systems," *The Open Information Science Journal*, vol. 2, pp. 18–25, 2009.

[7] S. Marsh, "Formalising trust as a computational concept," Ph.D. dissertation, Department of Mathematics and Computer Science, University of Stirling, 1994.

[8] G. Zacharia, A. Moukas, and P. Maes, "Collaborative reputation mechanisms for electronic marketplaces," *Decision Support Systems*, vol. 29, no. 4, pp. 371–388, 2000.

[9] B. Yu and M. Singh, "Searching social networks," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. ACM, 2003, pp. 65–72.

[10] E. Lorini, "Agents with emotions: a logical perspective," *ALP Newsletter*, vol. 12, no. 2-3, 2008.

[11] A. Ortony, G. Clore, and A. Collins, *The cognitive structure of emotions*. Cambridge university press, 1990.

[12] B. R. Steunebrink, M. Dastani, and J.-J. C. Meyer, "Towards a quantitative model of emotions for intelligent agents," in *Proceedings of the 2nd Workshop on Emotion and Computing-Current Research and Future Impact, Osnabrück, Germany*, 2007.

[13] B. Steunebrink, M. Dastani, and J. Meyer, "The occ model revisited," in *Proceedings of the 4th Workshop on Emotion and Computing*, vol. 65, 2009, pp. 2047–2056.

[14] B. R. Steunebrink, M. Dastani, and J. Meyer, "A logic of emotions for intelligent agents," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 22, no. 1. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007, p. 142.

Using the JaCaMo framework to develop a SMA for the MAPC 2012 "Agents on Mars" scenario

Mariana Ramos Franco, Jaime Simão Sichman
Laboratório de Técnicas Inteligentes (LTI)
Escola Politécnica (EP)
Universidade de São Paulo (USP)
Email: mafranko@usp.br, jaime.sichman@poli.usp.br

***Abstract*—This paper describes the architecture and core ideas of the Multi-Agent System created by the LTI-USP team which participated in the 2012 edition of the Multi-Agent Programming Contest (MAPC 2012). The contest scenario represented a coordinated exploration on Mars. The team organization was based on the *Moise* organisational model, and its implementation used the *JaCaMo* multi-agent framework.**

***Keywords*—Multi-Agent System, Multi-Agent Programming, JaCaMo, Jason, CArtAgO, Moise.**

I. Introdução

Recentemente, tem havido um movimento rumo ao uso de organizações explícitas no projeto e desenvolvimento de Sistemas Multiagentes (SMA) [1], [2]. A organização ajuda a modelar melhor o problema em questão, e a aumentar a eficiência do sistema, definindo a estrutura do SMA e as regras que os agentes devem seguir para atingir seus objetivos.

Seguindo esta premissa, neste artigo descrevemos um SMA com organização explícita, desenvolvido para participação na edição de 2012 do *Multi-Agent Programming Contest*[1] (MAPC 2012).

O MAPC é uma competição realizada todos os anos com o propósito de estimular a pesquisa no campo da programação de SMA [3]. Em cada rodada do evento, dois times de agentes são situados no mesmo ambiente e competem diretamente num cenário definido pelos organizadores. Por se tratar de uma competição direta, é um cenário interessante para avaliar e comparar diferentes sistemas, identificar pontos fortes e fracos, e promover o desenvolvimento de todos os participantes.

Nosso time foi baseado no modelo organizacional *Moise*, e foi desenvolvido utilizando a infraestrutura fornecida pelo arcabouço *JaCaMo*.

Moise[2][4] é um modelo organizacional que decompõe a especificação da organização em três dimensões: estrutural, funcional e deôntica. O modelo permite que o projetista especifique explicitamente as restrições e os padrões de cooperação a serem impostas ao agentes pela organização.

JaCaMo[3] [5] é um arcabouço que cobre todos os níveis de abstração necessários para o desenvolvimento de SMA sofisticados, pois possibilita, além da programação dos agentes, a definição do ambiente e da organização do SMA. Cada um desses níveis de abstração (agente, ambiente e organização) são implementados através da combinação de três tecnologias distintas: *Jason*[4] [6], para programação dos agentes; *CArtAgO*[5] [7], para programação do ambiente; e *Moise*, para programação da organização.

A motivação principal para a participação na competição foi testar e analisar o arcabouço *JaCaMo* e sua camada organizacional, a fim de identificar os pontos fracos e fortes da plataforma, e suas possíveis limitações de desempenho.

A seguir, o cenário da competição é brevemente descrito na seção II, e em seguida o SMA desenvolvido é apresentado na seção III. Alguns resultados preliminares de análise de desempenho do arcabouço *JaCaMo* são mostrados na seção IV e as conclusões do trabalho apresentadas na seção V.

II. *Multi-Agent Programming Contest*

O MAPC é uma competição internacional realizada anualmente desde 2005 com o propósito de estimular a pesquisa na área de programação de SMA. Em 2011, o MAPC definiu como tema para a competição o cenário "*Agents on Mars*", no qual os concorrentes devem projetar uma equipe de 20 agentes para explorar e ocupar as melhores zonas de Marte.

Neste cenário, dois times de agentes competem para dominar os melhores poços de água do planeta. O ambiente é representado por um grafo ponderado, em que os vértices denotam poços e possíveis localizações para os agentes; e as arestas indicam a possibilidade de atravessar de um vértice para outro, com um custo em energia para o agente. Cada vértice possui um valor correspondente ao poço de água nele localizado, e esse valor é utilizado para o cálculo do valor das zonas ocupadas pelos agentes.

Uma zona é um sub-grafo coberto por uma equipe de acordo com um algoritmo de coloração baseado na noção de domínio [3]. Vários agentes podem estar em um único vértice, mas o seu domínio é dado ao time com maior número de agentes. Se um conjunto de agentes domina um vértice por número, este recebe a cor da equipe dominante. Um vértice sem cor que tem a maioria dos vizinhos com uma cor específica herda essa cor para si. Por fim, se o grafo geral contém um sub-grafo colorido que constitui uma fronteira, todos os nós que estão dentro desta fronteira também são coloridos. Isto

[1]Site da competição: http://multiagentcontest.org/.
[2]Disponível em http://moise.sourceforge.net/.
[3]Disponível em http://jacamo.sourceforge.net/.
[4]Disponível em http://jason.sourceforge.net/.
[5]Disponível em http://cartago.sourceforge.net/.

significa que os agentes podem colorir ou cobrir um sub-grafo que tem mais vértices do que o número de agentes. A Figura 1 mostra parte de um mapa com os sub-grafos coloridos.

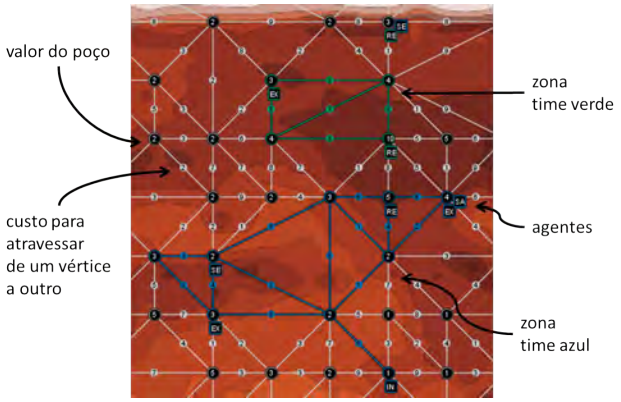

Figura 1. Cenário *"Agents on Mars"*.

O mapa é desconhecido dos agentes no começo da simulação. Assim, cada equipe precisa explorar o grafo antes de começar a conquistar as zonas, dado ainda que os agentes possuem visão limitada do mapa e só recebem percepções dos vértices próximos. Além disso, às vezes, uma equipe precisa sabotar o outro time para conseguir aumentar sua área, ou se defender para não perder zonas para o oponente.

Cada time é formado por 20 agentes e cinco papéis diferentes, sendo quatro agentes por papel. Os papéis, descritos na Tabela I, definem características próprias de cada agente, tais como nível de vida, energia máxima, força e visibilidade, e as ações que o agente pode realizar no ambiente. Por exemplo, os exploradores podem encontrar poços de água e ajudar a explorar o mapa, os sentinelas possuem sensores de longa distância e assim podem observar grandes áreas, os sabotadores podem atacar e desativar os inimigos, os inspetores podem espionar os adversários, e os reparadores podem consertar agentes danificados.

Se uma equipe atinge um marco importante, ela recebe uma recompensa em dinheiro. A recompensa ganha por uma equipe pode ser usada para equipar os agentes, aumentando, por exemplo, o máximo de energia ou a força de um agente. Existem diferentes marcos que podem ser atingidos durante uma competição, tais como ter zonas com valores fixos (por exemplo, 10 ou 20), número fixo de ataques bem sucedidos, ou número fixo de defesas realizadas com sucesso. Se não for usado, o dinheiro ganho é somado à pontuação total da equipe.

O objetivo do jogo é maximizar a pontuação do time, que é definida como a soma dos pontos obtidos pelas zonas ocupadas com o dinheiro ganho (e não gasto) em cada passo da simulação, como mostra a Equação 1:

$$pontuacao = \sum_{p=1}^{passos} (zonas_p + dinheiro_p) \qquad (1)$$

Tabela I. PAPÉIS E AÇÕES.

	explorador	reparador	sabotador	sentinela	inspetor
recarregar	x	x	x	x	x
atacar			x		
defender		x	x	x	
mover	x	x	x	x	x
sondar[6]	x				
examinar[7]	x	x	x	x	x
inspecionar[8]					x
comprar	x	x	x	x	x
reparar		x			

III. *LTI-USP Team*

A seguir, são apresentadas a arquitetura e as estratégias principais do SMA, chamado de "*LTI-USP Team*", que foi desenvolvido utilizando o arcabouço *JaCaMo* para participar do MAPC 2012.

O *JaCaMo* [5] é um arcabouço que cobre todos os níveis de abstração necessários para o desenvolvimento de sofisticados SMA, pois possibilita, além da programação dos agentes, a definição do ambiente e da organização do SMA. Cada um desses níveis de abstração (agente, ambiente e organização) são implementados através de um dos três componentes que compõem o arcabouço: *Jason*, *CArtAgO* e *Moise*.

A. *Arquitetura*

A arquitetura do SMA é apresentada na Figura 2 e baseia-se no modelo BDI [8]. Os agentes foram desenvolvidos através da plataforma *Jason*, e cada agente possui sua própria thread de controle, planos, uma base de crenças e modelo de mundo. Os planos são especificados em *AgentSpeak*[9], e em cada passo da simulação o agente decide qual plano será executado de acordo com as suas crenças e visão local do mundo.

O modelo de mundo consiste de um grafo desenvolvido em Java utilizando classes e estruturas de dados simples. Ele capta todos os detalhes recebidos do servidor do MAPC, tais como vértices e arestas exploradas, posição dos adversários, companheiros de equipe desativados, etc. Em cada passo da simulação, o modelo de mundo do agente é atualizado com as percepções recebidas do servidor e com as informações recebidas dos outros agentes. O agente pode acessar ou alterar o estado do seu modelo de mundo através de ações internas desenvolvidas em Java. Alguns exemplos de ações internas são:

- $closer\_repairer(Pos)$, que devolve ao agente a posição do agente reparador mais próximo,
- $move\_to\_target(Pos, Target, NextPos)$, que diz ao agente para qual vértice ele deve se mover a fim de alcançar determinada posição no grafo.

Algumas percepções recebidas do servidor são também armazenadas na base de crenças do agente, tais como seu

[6] A priori, os agentes não tem conhecimento sobre o valor dos poços de água. Apenas após sondar um vértice é que o time toma conhecimento do valor do poço.

[7] Inicialmente, os agentes não conhecem qual o custo de se atravessar de um vértice a outro. Para descobrir qual o valor de cada aresta, o agente precisa examiná-la antes.

[8] Esta ação coleta informações sobre os oponentes presentes em vértices vizinhos, tais como nível de vida, energia e papel.

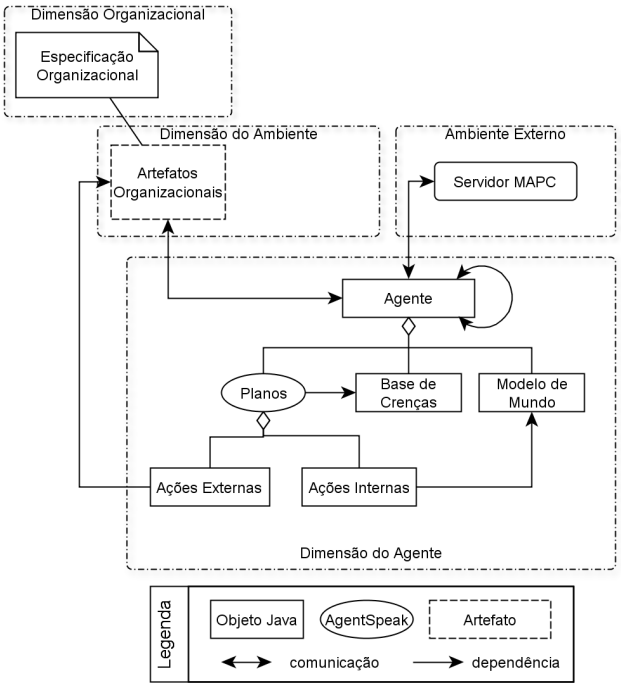

Figura 2. Arquitetura do *"LTI-USP Team"*.

papel, seu nível de energia e sua posição. Isto permite que o agente tenha acesso direto a essas informações sem ter que acessar o modelo de mundo. As percepções sobre vértices, arestas e outros agentes não são armazenadas na base de crenças para não comprometer o desempenho do agente, uma vez que poderia ser muito caro atualizar e acessar a base de crenças com tanta informação: o mapa de Marte utilizado na competição continha cerca de 400 vértices e centenas de arestas.

A comunicação entre os agentes ocorre através do *Jason* e, para reduzir a sobrecarga de comunicação, os agentes transmitem aos outros apenas as percepções novas, isto é, apenas as percepções recebidas do servidor que produzem alguma atualização no seu modelo de mundo. Por esta razão, existe uma forte troca de informação entre os agentes no início de uma partida, devido à comunicação sobre percepções novas, especialmente aquelas relacionadas à estrutura do mapa, como vértices e arestas. No entanto, a sobrecarga de comunicação diminui à medida que os agentes vão construindo um modelo de mundo mais completo. Os agentes se comunicam para:

(i) informar os outros agentes sobre a estrutura do mapa;

(ii) informar sobre agentes inimigos;

(iii) pedir que algum outro agente vá repará-lo;

(iv) pedir que um agente vá para determinado vértice.

Os agentes se comunicam com o servidor da competição através da interface EISMASSim incluída no pacote de software do simulador distribuído aos participantes. O EISMASSim se baseia no EIS[9][10], que é um padrão proposto para a interação agente-ambiente. Ele automaticamente estabelece e mantém conexões autenticadas com o servidor, além de abstrair a comunicação para simples chamadas de métodos em Java. A fim de utilizar essa interface, foi necessário estender a arquitetura padrão de agentes do *JaCaMo* para que fosse possível a um agente não só perceber e agir sobre os artefatos do *CArtAgO*, mas também interagir com o servidor.

CArtAgO é um arcabouço baseado no meta-modelo A&A (Agentes e Artefatos) [11] para o desenvolvimento e execução de ambientes em SMA. No *CArtAgO* o ambiente é projetado como um conjunto dinâmico de entidades computacionais chamadas de *artefatos*, que representam os serviços e ferramentas que os agentes são capazes de explorar em tempo de execução [5]. O conjunto de artefatos é organizado em um ou múltiplos *workspaces*, que podem ser distribuídos em vários nós de uma rede. Agentes podem entrar e sair dos *workspaces* e, dentro destes, artefatos podem ser criados e descartados dinamicamente pelos agentes [12]. Neste projeto não foi criado nenhum artefato novo, apenas se fez uso dos artefatos organizacionais fornecidos pelo *Moise*.

Moise é um modelo organizacional que decompõe a especificação da organização em três dimensões: estrutural, funcional e deôntica [4]. O modelo permite que o projetista especifique explicitamente a organização do SMA e suas restrições, além de também poder ser usado pelos agentes para raciocinar sobre sua organização. A especificação organizacional do time desenvolvido é descrita a seguir.

B. Estratégia

Podemos definir a estratégia adotada como uma combinação da estratégia organizacional, das estratégias específicas de cada papel, e da estratégia de coordenação.

1) Estratégia Organizacional: A estratégia principal do time foi dividir os agentes em três subgrupos: dois grupos responsáveis por ocupar as melhores zonas do mapa (subgrupos `zona1` e `zona2`), e um encarregado de sabotar o time inimigo (subgrupo `sabotagem`). Para organizar os agentes dessa maneira, foram utilizados os artefatos organizacionais fornecidos pelo *Moise*, e definidas as especificações estrutural (EE), funcional (EF) e deôntica (ED) do SMA.

A EE define os papéis, relações entre papéis, e grupos de uma organização. Cada agente pode assumir um ou mais papéis, e cada papel está relacionado com um conjunto de restrições comportamentais que um agente aceita ao entrar em um grupo. Como mostrado na Figura 3, na EE do *LTI-USP Team* foram definidos os três subgrupos citados anteriormente, e sete possíveis papéis: os cinco papéis especificados pelo cenário (explorador, sabotador, sentinela, reparador e inspetor), e mais dois papéis chamados *"marciano"* e *"coordenador"*. O coordenador é o responsável por conduzir os agentes a ocuparem as melhores zonas de Marte e, diferente dos outros agentes, o coordenador não se comunica com o servidor do MAPC. O marciano é o papel padrão adotado pelos outros agentes no início de uma partida, enquanto não recebem do servidor a informação sobre qual papel desempenhar.

A EF define como os objetivos globais devem ser alcançados, isto é, como esses objetivos são decompostos (em submetas) e distribuídos aos agentes (em missões). Ela também

[9]Disponível em http://sourceforge.net/projects/apleis/.

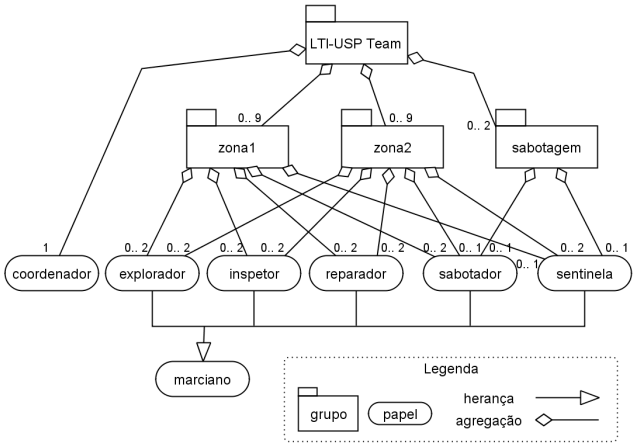

Figura 3. Especificação Estrutural do *"LTI-USP Team"*.

especifica como as sub-metas estão relacionadas, se devem ser alcançadas em paralelo ou em uma certa sequência. Desta forma, cada subgrupo anteriormente especificado possui um meta global associada, definida na EF. Na Figura 4, essas metas globais são representadas como raiz das árvores, que apresentam em suas folhas as sub-metas a serem atingidas pelos agentes. O rótulo que aparece logo acima de uma meta representa a missão ao qual o agente deve estar comprometido, a fim de atingir a meta relacionada. Para cada uma das missões existe um plano que o agente deve seguir.

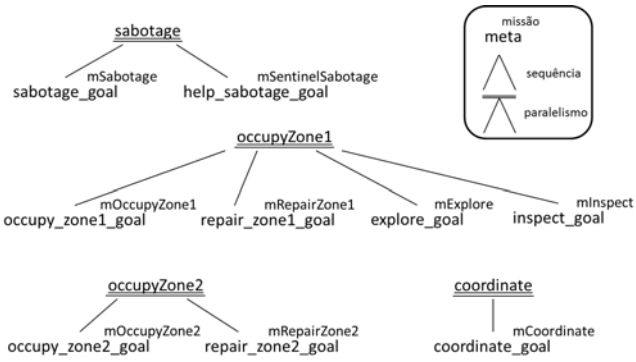

Figura 4. Especificação Funcional do *"LTI-USP Team"*.

As diferentes metas relacionadas a cada subgrupo são apresentadas a seguir:

- `occupyZone1`: Os agentes no grupo `zona1` devem ocupar a melhor zona no grafo seguindo as orientações dadas pelo agente coordenador. Além disso, um dos exploradores fica responsável por sondar os vértices do grafo, a fim de encontrar as melhores zonas a serem ocupadas. Por fim, um inspetor tem a missão de identificar o papel de cada agente no time inimigo. Apenas ao tomar conhecimento do papel de todos os oponentes é que o inspetor se junta ao resto do time na missão de ocupar a melhor zona do grafo.
- `occupyZone2`: Todos os agentes no grupo `zona2` tem a missão de ocupar a segunda melhor zona no grafo, ou ajudar o grupo `zona1` a formar uma área maior. Se este grupo deve ou não se juntar ao outro grupo é determinado pelo coordenador, que verifica a existência ou não de duas boas zonas dispersas no grafo.
- `sabotage`: O grupo `sabotagem` é formado por um sabotador e um sentinela. A missão do sabotador é atacar os oponentes que ocupam bons vértices; e a missão do sentinela é de ajudar na sabotagem, movendo-se para dentro da zona do time inimigo de forma a quebrá-la ou fazê-la diminuir.

Por fim, a ED liga as especificações estrutural e funcional definindo com quais missões um papel tem a obrigação ou a permissão de se comprometer. A ED do *LTI-USP Team* é representada na Tabela II.

Tabela II. ESPECIFICAÇÃO DEÔNTICA DO *"LTI-USP Team"*.

Papel	Missão	Relação Deôntica
explorador	mExplore, mOccupyZone1	permissão
explorador	mOccupyZone2	obrigação
reparador	mRepairZone1, mRepairZone2	obrigação
sabotador	mSabotage, mOccupyZone1, mOccupyZone2	obrigação
sentinela	mSentinelSabotage, mOccupyZone1, mOccupyZone2	obrigação
inspetor	mInspect, mOccupyZone1	permissão
inspetor	mOccupyZone2	obrigação
coordenador	mCoordinate	obrigação

2) Estratégias Dependentes do Papel: Para cada papel, foram desenvolvidas estratégias específicas. Por exemplo, os exploradores devem sondar e examinar todo vértice ou aresta no seu caminho, enquanto os inspetores devem sempre procurar inspecionar inimigos próximos.

Os agentes também se comportam diferentemente, dependendo do papel que desempenham, quando encontram um oponente no mesmo vértice. Os sabotadores sempre atacam o oponente, enquanto os sentinelas procuram se defender e os outros agentes fogem para um outro vértice.

3) Estratégia de Coordenação: A estratégia adotada é baseada na centralização da coordenação, isto é, um único agente (coordenador) é responsável por determinar quais as melhores zonas do mapa e, em seguida, conduzir os outros agentes a ocuparem essas zonas. A escolha da coordenação centralizada foi feita para permitir o rápido desenvolvimento da equipe, uma vez que a motivação principal era se concentrar no uso da plataforma *JaCaMo* e não nos aspectos de coordenação.

Desta forma, o coordenador constrói o seu modelo de mundo através das percepções difundidas pelos outros agentes e, sempre que o modelo de mundo é atualizado, calcula quais são as duas melhores zonas no grafo. O coordenador então pede para que os agentes (de cada um dos dois grupos responsáveis por ocuparem as zonas) se movam para dentro das zonas calculadas. Quando todos os agentes estiverem dentro das zonas, o coordenador passa então a olhar nos vértices vizinhos a estas, procurando posições para as quais os agentes possam se mover a fim de aumentar o tamanho da área conquistada.

IV. ANÁLISE DE DESEMPENHO

O grande obstáculo no desenvolvimento do time foi lidar com os problemas de desempenho relacionados com o uso concorrente dos artefatos organizacionais.

No início de cada partida, os agentes recebem quase que ao mesmo tempo do servidor do MAPC os papéis que irão desempenhar naquela partida para então tentar entrar em um dos três grupos especificados (`zona1`, `zona2` e `sabotagem`). Neste momento, é possível que, por exemplo, todos os quatro sabotadores queiram entrar no grupo `sabotagem`, mas, como mostrado na Figura 3, neste grupo só é permitido um sabotador. Neste caso, três sabotadores serão impedidos de entrar no grupo `sabotagem` e terão que tentar se juntar a outro grupo. Em testes realizados antes da competição, foi notado que a chamada de uma ação organizacional, tais como $adoptRole$ (para a adoção de um papel em um grupo) ou $commitMission$ (para se comprometer a uma missão), é muito cara, e que o número de tentativas feitas por um agente até finalmente conseguir entrar em um grupo podia ser muito alto. Isto fazia com que alguns agentes perdessem alguns passos no início da simulação, já que em cada passo os agentes tem um prazo limitado (1000 ms) para enviarem as suas ações.

Para corrigir esse problema, passamos para o agente coordenador a responsabilidade de distribuir entre os outros agentes os grupos e missões. Desta forma, foi evitada a ocorrência de colisões, como agentes tentando entrar no mesmo grupo ou se comprometer com a mesma missão. Isto melhorou o desempenho do SMA, mas mesmo assim foi possível observar durante a competição que alguns agentes continuavam perdendo passos antes de finalmente conseguir se comprometer com uma missão na organização.

Assim, após o término da competição foram realizados alguns experimentos para avaliar de forma preliminar o desempenho da operação $adoptRole$, em função do número de agentes tentando acessar a operação em concorrência.

No primeiro experimento, foram coletados para cada agente o tempo (em milissegundos) entre a chamada da operação $adoptRole$ e a atualização da base de crenças do agente com o papel adotado. Neste experimento, cada agente tenta assumir um papel diferente na organização, o que significa que não há risco da operação $adoptRole$ falhar. Para cada número de agentes (10, 20, 50 e 100), o experimento foi rodado 20 vezes, e o tempo médio de resposta para um agente adotar um papel é apresentado na Figura 5. É possível notar que o tempo de resposta da operação aumenta consideravelmente com o número de agentes acessando o artefato ao mesmo tempo.

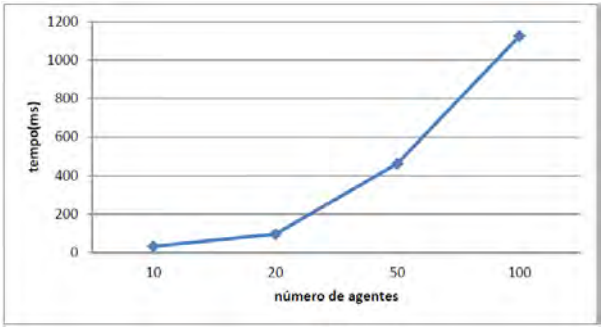

Figura 5. Tempo médio gasto para cada agente adotar um papel, em relação ao número de agentes executando a operação $adoptRole$ em concorrência.

Em outro experimento, novamente o número de papéis é igual ao número de agentes, mas inicialmente todos os agentes tentam adotar o mesmo papel. Como apenas um agente pode adotar cada papel, ocorre que a grande maioria dos agentes não consegue adotar o papel inicial, e então tentam adotar um segundo papel, e assim sucessivamente, até que todos os agentes tenham conseguido um papel na organização. Foram coletados neste experimento, o tempo médio gasto para cada agente entre a primeira chamada da operação $adoptRole$, e a atualização da base de crenças do agente com o papel adotado. Além disso, também foram coletados o número médio de tentativas feita por cada agente até finalmente ter sucesso em adotar um papel. Os resultados obtidos são mostrados na Figura 6.

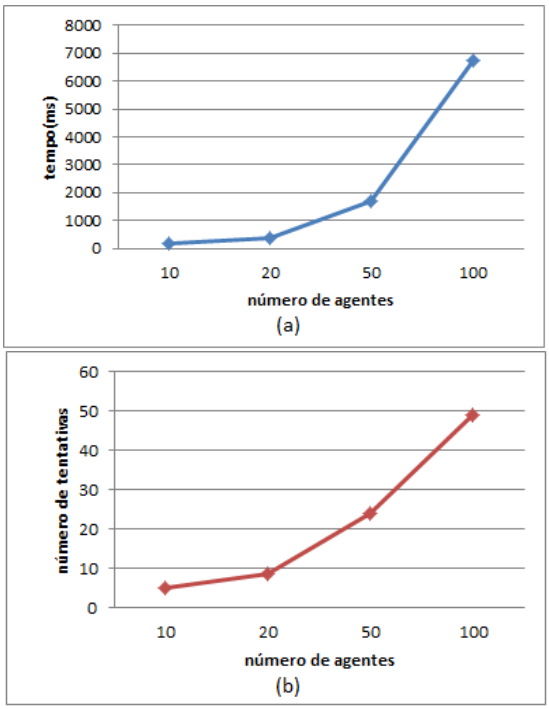

Figura 6. (a) Tempo médio gasto por cada agente para adotar um papel, em relação ao número de agentes executando a operação $adoptRole$ em concorrência; e (b) Número médio de chamadas a operação $adoptRole$ por agente.

Comparando os dois experimentos, é possível notar que o resultado obtido no primeiro, em que cada agente adota um papel diferente, é melhor do que os resultados obtidos no segundo experimento, em que inicialmente os agentes tentam adotar o mesmo papel. Isto reforça a nossa escolha de ter dado ao agente *coordenador* a responsabilidade de distribuir os papéis entre os outros agentes, a fim de reduzir o número de chamadas para a operação $adoptRole$, e assim melhorar o desempenho do SMA.

Além disso, é possível notar que o tempo aumenta consideravelmente com o número de agentes, principalmente para os valores acima de 20 agentes. Esse aumento no tempo de resposta da operação pode ser explicado pela arquitetura utilizada pelo CArtAgO, na qual cada *workspace* possui uma fila para onde vão as requisições dos agentes para a execução das operações nos artefatos, e de onde 20 threads ficam responsáveis por capturar as requisições e executá-las. Assim, quando se tem um número de requisições concorrentes maior

que o número de threads trabalhando para executá-las, é normal que as requisições "em excesso" tenham que ficar esperando mais tempo na fila até serem executadas.

V. Conclusões

A participação de nosso time no MAPC 2012 foi de grande valia no objetivo de conhecer melhor o funcionamento do arcabouço *JaCaMo*. Apesar dos esforços não terem sido direcionados na procura de uma melhor estratégia, o time desenvolvido acabou o torneio em quarto lugar em um total de sete equipes.

Nosso grande obstáculo no desenvolvimento do time foi lidar com os problemas de desempenho relacionados com o uso concorrente dos artefatos organizacionais. Em cenários onde o tempo é limitado, como o enfrentado nessa competição, o desempenho de uma plataforma é um fator muito importante. Assim, os problemas de desempenho encontrados podem acabar dificultando a adoção do *JaCaMo* em tais cenários. Consequentemente, como trabalho futuro pretende-se realizar uma avaliação completa do desempenho e gargalos no *JaCaMo*.

Apesar desses problemas de desempenho, o arcabouço *JaCaMo* provou ser bastante completo, fornecendo todas as ferramentas que precisávamos para desenvolver o SMA. Além disso, o uso do modelo organizacional *Moise* ajudou no projeto da equipe, uma vez que fornece uma forma de estruturar o SMA, não somente através da especificação de grupos e papéis, mas principalmente na definição de como os objetivos globais devem ser alcançados.

Agradecimentos

Jaime Simão Sichman é parcialmente financiado pelo CNPq e FAPESP/Brasil.

Referências

[1] J. Ferber, O. Gutknecht, and F. Michel, "From agents to organizations: an organizational view of multi-agent systems," *Agent-Oriented Software Engineering IV*, no. July 2003, pp. 214–230, 2004.

[2] O. Boissier, J. Hübner, and J. Sichman, "Organization oriented programming: From closed to open organizations," in *Engineering Societies in the Agents World VII*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, vol. 4457, pp. 86–105.

[3] T. Behrens, M. Köster, and F. Schlesinger, "The multi-agent programming contest 2011: a résumé," *Programming Multi-Agent Systems*, pp. 155–172, 2012.

[4] J. Hübner, J. Sichman, and O. Boissier, "Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels," *International Journal of Agent-Oriented Software Engineering*, pp. 1–27, 2007.

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

[6] R. Bordini, J. Hübner, and M. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*, 2007.

[7] A. Ricci, M. Piunti, and M. Viroli, "Environment programming in multi-agent systems: an artifact-based perspective," *Autonomous Agents and Multi-Agent Systems*, vol. 23, no. 2, pp. 158–192, Jun. 2010.

[8] M. Bratman, *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.

[9] A. S. Rao, "Agentspeak(l): Bdi agents speak out in a logical computable language," in *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : Agents Breaking Away*, ser. MAAMAW '96. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1996, pp. 42–55.

[10] T. M. Behrens, J. Dix, and K. V. Hindriks, "The Environment Interface Standard for Agent-Oriented Programming - Platform Integration Guide and Interface Implementation Guide," *Department of Informatics, Clausthal University of Technology, Technical Report*, vol. IfI-09-10, 2009.

[11] A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the a&a meta-model for multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 3, pp. 432–456, Dec. 2008.

[12] A. Ricci, M. Viroli, and A. Omicini, "CArtAgO: An infrastructure for engineering computational environments in MAS," *3rd Inter. Workshop "Environments for Multi-Agent Systems"(E4MAS)*, 2006.

An Experiment of Verification of Multi-agent Robotic Soccer Plans with Model Checking

Rui C. Botelho A. S.[1], Aline M. S. Andrade[2], Augusto Loureiro da Costa[3], Frederico J. R. Barboza[2]
[1]Post-graduation Program on Mechatronics – Master, [2]Distributed Systems Laboratory, [3]Robotics Laboratory
Federal University of Bahia, UFBA
Salvador, Brazil
{ruicbs,fred.barboza}@gmail.com,{aline,augusto.loureiro}@ufba.ba

***Abstract*—In this paper we present an experiment of model checking which consists of the verification of plans of a multi-agent system for simulated robot soccer. This system is of considerable complexity because it is concurrent, nondeterministic and with partial vision of the environment. Some solutions adopted relative to modeling and process of verification to circumvent state space explosion are reported.**

Keywords—Model Checking; Planning; Multi-agent Systems, Autonomous Agents

I. INTRODUCTION

In recent decades, research involving the development of Autonomous Agents - AAs and Multi-agent Systems - MAS has increased in academia and industry. This is due to the use of these systems, whether real or virtual entities, in various kinds of applications: autonomous robots, multi-robot systems, unmanned vehicles, control systems, flight plans of aircraft combat systems, air traffic control, simulation, matches of physical or simulated robots in software, softbots, among others.

The use of MAS and AAs are motivated by their inherent characteristics such as autonomy, collaboration, proactivity among others. The main activities of these entities are related to the achievement of objectives that are performed based on planning. It is necessary to ensure that AAs individually or MAS collectively have correct plans to guarantee that they do not behave unintendedly, and have desirable outcomes. Formal methods can be used in this context, particularly model checking, which has been used in several works published in this area [1, 2, 3, 4, 5, 6, 7].

In this paper we present the results of the verification of plans of robot soccer, the Mecateam [8], which uses the simulated environment Robocup [9]. Our interest in this team is mainly due to the fact that it is a multi-agent system with a multilayer architecture, which imposes a parallelism in the internal plans of the agents increasing the state space of the problem.

In robot soccer the environment is non-deterministic and players only have a partial vision of it. These features combined with the three-layer architecture of Mecateam add complexity to its planning, requiring care in both modeling and verification of the plans. With this in mind some solutions were carried out: abstractions were done to overcome the state space explosion; a decomposition of the plans based on the multilayer organization of the agents was considered; the process of verification was implemented in an incremental way considering an evolution of the models of the plans from the individual plans of the agents to their collective ones. These solutions together allowed the verification of a significant part of the state space of the problem.

For our experiment the UPPAAL model checker [10] was used. UPPAAL is a tool set applied in modeling and verification of systems which uses timed automata formalism to model the system and a subset of TCTL (Timed Computation Tree Logic)[10] for the specification of the system properties.

The contributions of this paper include solutions adopted in modeling and verification of a MAS complex real application with model checking which may be useful in similar multi-agent systems.

This paper is organized as follows: in the following section related works are presented; section 3 presents an introduction of the robot soccer team; section 4 presents the specification of the plans as automata; section 5 presents the plan model checking; and finally, in section 6, some conclusions are presented.

II. RELATED WORKS

In [1,2] the validation of plans for a multi-agent simulation environment for tactical fighter aircraft is considered. This is highly dynamic, non-deterministic and has partial vision. The multi-agent plans are modeled as a network of hybrid automata [3] and the agents have more reactive than cognitive behavior.

In [4,5] the plans verification of a controller and scheduler system that composes the remote control of the robot Deep Space 1 used in U.S. space agency missions (NASA - National Aeronautics Space Administration)was carried out using UPPAAL. This work deals with the reactive behavior of the individual agents without considering the interaction with other agents.

In [6] the potential use of hybrid automata using the HYTEC tool to model and verify plans of autonomous agents

is demonstrated considering an unmanned aerial reconnaissance case study. In this case the plans are almost always deterministic, the environment is non-dynamic and the agents do not interact with each other.

The work presented in [7] uses model checking for verification and simulation of soccer teams of robots at runtime. It considers a specific platform, the Ericson Company platform and the ERLANG specification language and McERLANG verifier. The verification of the agent's code is considered as a whole, which comprises planning activities, interface code, actions done in the environment, etc. which make it difficult to distinguish planning errors from other errors. In this work the team is modeled as a single component which prevents the analysis of agents in isolation. Finally, the verification is performed at runtime using the Soccerserver simulation environment to simulate the match.

The works presented above focus on different features of MAS, encompassing their several planning characteristics. However, none of them consider as many aspects together as our work does, namely: non-determinism, communication among agents, partial vision, structured planning in three-ties, verification of individual agents and collectively. These aspects together increase the complexity of the problem and require creative solutions to abstract the model and a systematization of the verification process in order to model and check the plans.

III. Robotic Soccer

Robot soccer is used as a platform for the study of a wide variety of problems inherent in AAs, robotics and cooperative MAS. Proposed by Robocup Federation, the robot soccer Robocup has emerged as a laboratory for the study of Distributed Artificial Intelligence (DSI) and its ramifications [9].

In a robot soccer match the robots must be able to: recognize their position and all references of location in the soccer field, represent the environment of the game, set objectives, plan and execute actions to achieve their goals. To deal with and provide consistent and intelligent behavior, agents must combine a sequence of actions associated with a primary set of concurrent actions, first individually and then collectively, according to a planning that meets individual and group objectives for each state of the environment.

A. The robot team Mecateam

The Mecateam robot soccer team was designed according the Concurrent Autonomous Agent Architecture [11] which comprises a cognitive, an instinctive and a reactive layer as presented in Fig. 1. This team has been participating in major competitions and has emerged as one of leading teams in simulated robot soccer in Brazil. The cognitive layer is responsible for the planning of the team, the reactive layer perceives and acts on the environment and the instinctive layer intermediates the communication between the cognitive and the reactive layer. The plans of the team are covered by cognitive and instinctive rules. The cognitive rules are responsible for defining the objectives of the agents and between each current objective and every future goals of the team, the cognitive rules promote a change in the state of the agent according to information about the environment from the instinctive rules. The instinctive rules receive this information from the reactive layer and determine the behaviors to be carried out by the reactive layer in the environment.

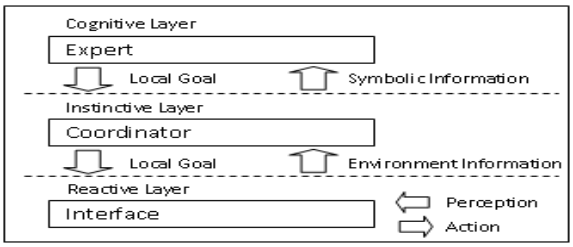

Fig. 1. Representation of the architecture of [10] (Adapted).

The individual plans of each player are composed of their cognitive and instinctive rules and their interactions. The performance of each player depends on its plans, changes in the environment and its internal state. In turn, the plans of the team are composed of the union of all individual plans and represent the possibilities of collective action in response to changes that occur in the environment.

The reactive layer has no decision-making behavior and for this reason it was not considered in the scope of this work.

B. Cognitive rules

The cognitive plan has some similar rules for the groups of agents (players): defenders (players 2, 3, 4 and 5) and midfielders and forwards (players 6, 7, 8, 9, 10 and 11). Table I presents two examples of cognitive rules by groups of players. In the first rule (rule 0) the *local_goal current* of the player is *none*. This corresponds to the beginning of the game, the goalkeeper and the defenders go to advance which means taking offensive actions in the game. The assertion *local_goal status active* activates the current local goal, now *advance*. The second rule (rule 3) changes the current local goal *side-attack* to the current local goal *mark* if the local goal *side_attack*fails. In this case the state *mark* is activated (*local_goal status active*).

TABLE I. Examples of cognitive rules

Goalkeeper and Defenders	Midfields and Forwards
(rule_0_start (if (logic(local_goal current none))) (then (logic(local_goal current advance)) (logic(local_goal status active))))	(rule_3_side_attack (if (logic(local_goal current side_attack)) (logic(local_goal status fail))) (then (logic(local_goal currentmark)) (logic(local_goal status active))))

In general the variable *local_goal current* has a set of possible values: none, mark, advance, side_attack and ending; and the variable *local_goal status* can assume the values: active, achieve and fail. Combinations among these values determine the individual player's goals. The set of particular values of *local_goal current* of a player depends on its role in the team, unlike the *local_goal status* of each player which has

the same set of values for all players. Table II shows the values of these variables per group of players.

TABLE II. SETS OF LOCAL GOAL CURRENT / STATUS BY GROUP OF PLAYERS.

		Players		
Variables		***1***	***2,3,4,5***	***6,7,8,9,10,11***
local_goal	***current***	mark, side_attack, none		
		advance		
		ending		
	status	fail, active, achieved		

C. Instinctive rules

The instinctive level of each agent contains a set of instinctive rules and a subset of it is presented in Fig. 2. These rules transmit the vision of the environment, from the reactive level to the cognitive level, and determine the behaviors of the reactive level on the environment. Between each current and each future goal of the team, the cognitive level should promote a change in the current state of the agent. This change depends on the results of the execution of a set of instinctive rules. Therefore, the instinctive rules intermediate the communications between the reactive and the cognitive level. The reactive level is responsible for: perceiving the situation of the environment, acting on the environment, activating reactive actions such as *intercept ball*, *drive_ball_forward*, *hold_ball*, etc.

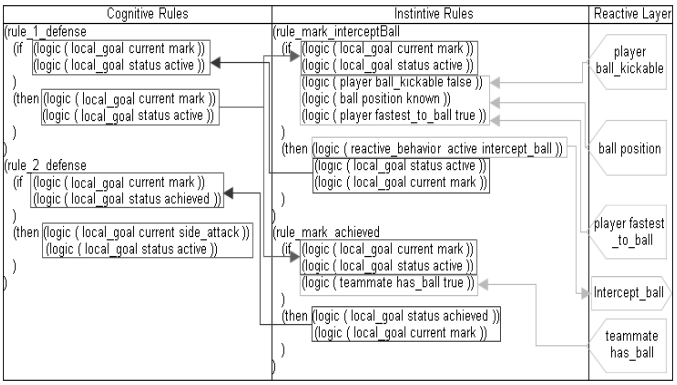

Fig. 2. Example of interactions between cognitive, instinctive and reactive rules.

IV. SPECIFICATION OF THE PLANS AS AUTOMATA

In this section we present a brief description of timed automata of UPPALL model checker used to model the rules and some examples of models.

A. UPPAAL automata

An automaton in UPPAAL is represented as a graph with a finite set of locations and transitions, represented as nodes and edges respectively. The initial location is represented by two concentric circles. Locations labeled "U" have priority over other locations. In the automaton of Fig. 6, the urgent location *SendingEnvironmentInfo* has priority over any other location because the environment state must be updated before any other action of the game is executed.

A transition can be controlled by guards and channels. The guards are logical expressions that determine the conditions for a transition to be triggered. For example the guard at the transition between the locations *None* and *Advance* (Fig. 3) determines that this transition will only be executed if the *LocalGoalCurrent == 4*. The channels synchronize the actions of two or more automata and can also be declared as urgent or broadcast to give priority to the corresponding transition and enable the communication with many automata at the same time, respectively.

B. Models of the rules

The left and right sides of the rules specify pre and post conditions which are modeled by the guards and changing in the values of variables in the automaton. Part of the three groups of players presented in Table II has the same set of rules and they are modeled as shown in Fig. 3 and Fig. 4.

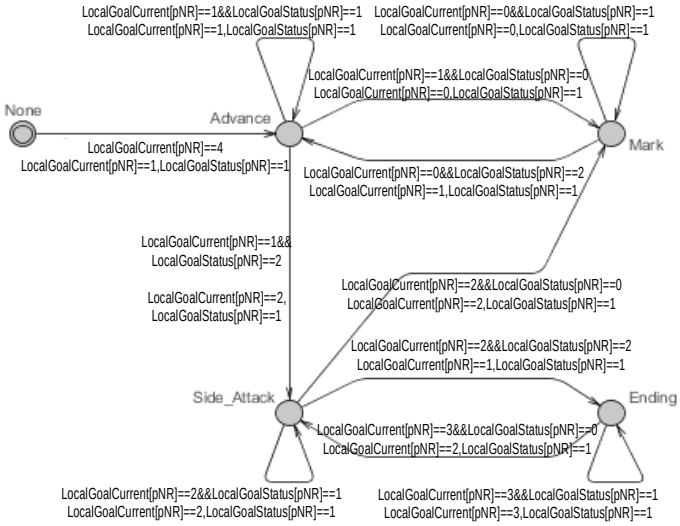

Fig. 3. Automaton of cognitive rules of the goalkeeper.

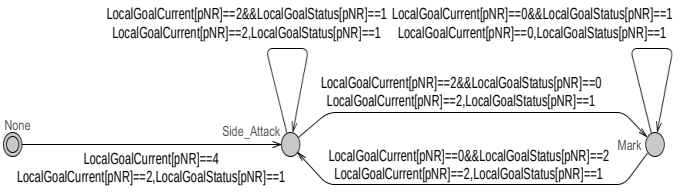

Fig. 4. Automaton of cognitive rules of the midfield and forward players.

The locations of the automata represent the current local goals of the players. InFig. 3, the locations corresponding to the possible goals of the cognitive rules of thegoalkeeper can be identified (see Table II): *None*, *Advance*, *Mark*, *Ending*, *Side_Attack*.In Fig. 4 the locations *None*, *Side_Attack* and *Mark* correspond to local current goalsof midfields and forwards players. For control purposes, the variables*LocalGoalCurrent* and *LocalGoalStauts* are declared in the automata. They correspondto the variables *local_goal current* and *local_goal status* of the rules, respectively.*LocalGoalCurrent* can assume the values: 0 to *mark*, 1 to *advance*, 2 to *side_attack*, 3 to*ending* and 4 to *none*.

LocalGoalStatus can assume the values: 0 to *fail*, 1 to *active* and2 to *achieved*. The combined values of both variables drive the change in location ofthe automaton. For example, *rule 2_defense* of Fig. 2 has state *mark* and status*achieved*. In the automaton of Fig. 4 that models this rule the corresponding location*Mark* has the guards *LocalGoalCurrent == 0* and *LocalGoalStatus == 2* thatcorrespond to state *Mark* and status *Achieved* respectively. So, a transition to thelocation *Side_Attack* occurs and the values of these variables are updated to*LocalGoalCurrent = 2* and *LocalGoalStatus == 1*.

Each instinctive rule was created as an elementary component to allow the removal or addition of rules for checking a particular player. Fig. 5 gives an idea of how the instinctive rules were modeled in automata.

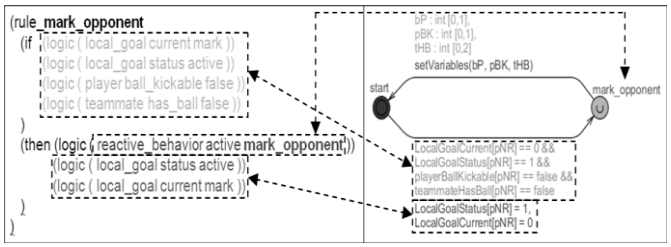

Fig. 5. Example of an instinctive rule and its respective automaton.

The automata presented up to here were used to verify individual players, without considering their interactions with the environment. For the verification of the entire team the model was enhanced with the communication channels *Startgame*, *ReStartGame* and *ActionInfo* between the environment and the players. Fig. 6a and Fig. 6b show examples of cognitive and instinctive rules resulting from the inclusion of these communications channels, respectively. A global boolean variable *EnvironmentInfo* was declared to control if the environment has processed the information resulting from the reactive actions in the previous round of the match and if the information about each player is available.

The channels *Startgame* and *ReStartGame* are declared broadcast to establish communication of the environment with all the players at the same time and they are declared urgent because the initial configuration of the game should be restored with priority over other actions. The channel *ActionInfo* represents the communication of each individual player with the environment and it is declared urgent because the environment should be restored to represent the actual situation of the game before any other action is executed.Fig. 7 shows the automaton of the environment. From its initial location a message (*Start Game!*) is sent to all players to inform that the environment is ready and that the players can start the game. In the location *WaitingForActions* the environment waits for all players to inform (through channel *ActionInfo*) the rules that were selected relative to a specific scenario of the match. After receiving all messages the transition to the location *SendingEnvironmentInfo* occurs and the function *setOpponentInformations()* is executed to update the current situation of the play considering the informations of the players and the actions of the adversary team which is simulated by the function *setOpponentInformations (oPHB,pTOA,pSOA)*. In case a goal is not done (*goalScored == false*) a new round is initiated by the function *enableNextRound()*, which is responsible to set the new state of the environment in order to the match continue. In case a goal is done (*goalScored == true*) the play is restarted. The function *setInitReinitValues()* carries out the configuration or reconfiguration of the environment in the beginning of the match or when a goal is done respectively. When a goal is done all players are informed by *ReStartGame!*channel.

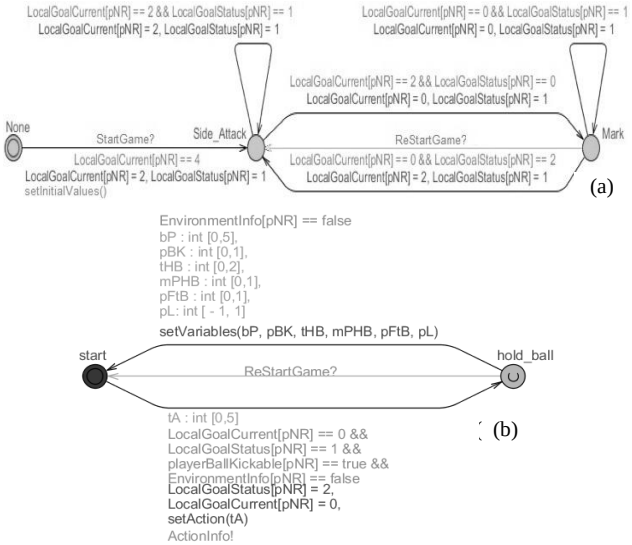

Fig. 6. Model of rules with communication channels: (a) cognitive rules of midfield and forward players, (b) an instinctive rule.

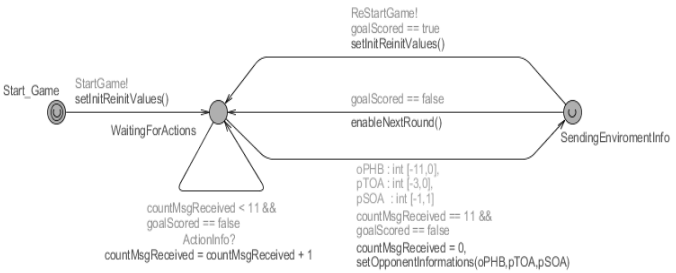

Fig. 7. Automaton of the environment.

The partial vision and the non-determinism of the environment were represented through a range of values previously defined by select label variable[1] of UPPAAL. For example, in Fig. 6 the select label variable (*oPHB: int[-11,0]*) receives a value between -11 and 0 each time a transition occurs from the location *WaitingForActions* to *SendingEnvironmentInfo* to indicate who player is with the ball when it is under control of the opponent team: {-1} – the ball is with the goalkeeper, {-2 to -5} – the ball is with the defensive players and {-5 to -11} – the ball is with the midfield or forward players. If the selected value is 0 the ball is in the field but it is not controlled by anyone.

V. MODEL CHECKING OF THE PLANS

Model checking consists of the specification of a finite model (an automaton or a variation of this type of representation) of a system and in the verification of desired

[1]Select label variables will take a non-deterministic value in the range of their respective types [9].

properties of the system through an exhaustive scan of the model state space, which is done automatically [11].

UPPAAL is a model checker designed for the verification of real-time systems. The UPPAAL tool set comprises a modeling environment, a simulator that is used to interactively monitor the execution of the model and a property checker that uses a query language which is a subset of the TCTL. The models in UPPAAL are designed as a network of timed automata as shown in the previous section of this paper [11]. The problem of state explosion is an intrinsic characteristic of model checking and solutions should be adopted to overcome this problem, such as abstractions in the models, compositional model checking [11], and other solutions.

A. Abstractions

To carry out model checking we did some abstractions in order to simply the model and reduce the state space: the soccer field was represented with one dimension, some match situations and time constraints were disregarded and the actions of the adversary team were simulated by a function in UPPAAL. In addition, we used an incremental verification process, presented in next section, which enabled the verification of individual agents initially, followed by the verification of groups of agents until the whole team in interaction with the environment was considered. Moreover, the multilayer architecture of Mecateam led to a modularization of the specification and verification considering the separation of cognitive and instinctive rules.

The representation of the field in one dimension (soccer simulation works with two dimensions) considered six regions, corresponding to the classic regions of a soccer field, which are: area, intermediate and midfield for both teams. This representation proved sufficient to explore relevant situations of the game such as the movements of the players and the partial view of the environment. It also allowed a significant simplification of the state space of the model and eliminated the problem of having to deal with calculations of movement and trajectory in the 2D environment, which should be resolved by a low level layer (reactive layer).

The following match situations were not represented because they are controlled by Soccerserver and are not part of the plans: corners, off sides, penalties, among others. In relation to time constraints, the cognitive and instinctive rules do not use this type of restriction. Although the Robocup environment considers a deadline for receiving agent messages, and our model has been devised to allow the inclusion of time constraints, we did not consider them in this version of the model. We emphasize that this did not cause any loss, considering the purpose of the cognitive and instinctive plan verification. The function that represents the adversary team simulates its behavior by choosing an offensive or defensive action, depending on the control of the ball is with the adversary or with it respectively. Random values are set through select label variables of UPPAAL to define which player will play, which actions will be executed and if the selected actions to be executed can succeed or not. So it was not necessary to create automata to represent the adversary team which has simplified the modeling.

We used pseudorandom values and data structures to confront the real situation of the match and the vision of each player with respect to it. This enabled a simple representation of the environment's uncertainty (caused by non-determinism) and the representation of the partial vision of the agents. An example is the position of the ball which is only partial determined by the agents and depends on their distances from it. The vision of each agent of the ball position is defined by the integer variable *bP: int[0,5]*, as seen in Fig. 7. This variable selects one value between 0 and 5 that corresponds to the soccer field areas, according to the position of the player in the field.

B. Verification of the plans

Due to the complexity of the system we defined a modularization of the verification process at several stages from simple plans to collective plans. This facilitated the analysis of model checking results and allowed the detection of errors in the subspaces of the whole model.

The process of verification of the cognitive and instinctive plans was done in five stages: 1. Verification of common properties related to common rules of all players or group of players (defenders or midfield players or offensive players). 2. Verification of the individual plans which consist of the particular rules of each agent in the team. 3. Modeling and verification of the environment. 4. Verification of each agent in communication with the environment. 5. Verification of the entire team in communication with the environment.

For the verification of the individual plans of each player we modularized the process, similarly as in composition model checking technique assume-guarantee [11]: first we assumed that the environment, the reactive layer and the instinctive layer were correct and applied model checking to the cognitive rules, and then we verified the instinctive rules assuming now that the cognitive rules were correct. This facilitated the analysis of the model checking and identification of errors.

In Table III are presented the properties used to verify the plans: safety (first and second line) and reachability properties (third line). These rules are specified as processes in UPPAAL which are instantiated according to the type of the rules as specified by the BNFs in the table. The safety properties are used to verify if the automata are free of deadlock (first line) or if the automata are consistent with the rules (second line), i.e., if the models had been constructed correctly. The reachability properties are used to verify if any automata location, corresponding to each cognitive or instinctive rule, is achieved from the initial location to determine if these rules are in fact used by the agents meaning that the reactive behaviors are achievable and the player will drive them.

A. Some results of model checking

As a result of the verification of the individual agents the occurrence of deadlock was observed in all players due to the bad construction of a rule (*rule_mark_hold_ball*) which led all plans to a cognitive state of the system in which there were no transitions possible. In addition, some locations were identified as not reachable in the instinctive rules, as for

example the state *advance* in the instinctive rule *Advanced_Pass_Ball_Forward* due to an error in this rule which was reflected in the model. After we had corrected the identified errors, the results of the model checking of the entire team showed that the model had preserved the reachability properties already verified before.

TABLE III. PROPERTIES.

Description	Formulae
1. Non-occurrence of deadlocks	A□ not deadlock
2. In all paths never occurs that the value of local goal current of the player is different from X and the location of the cognitive automaton is equivalent to X.	A□ not (LocalGoalCurrent[<pNR>] != X and P<pNR>_C.<CognitiveLocation>) Ex: A□ not (LocalGoalCurrent[1] != 0 and P1_C.Mark)
3. Exist a path for any location from the initial state.	E◊ P<pNR>_<Rule> Ex: E◊ P1_C.Side_Attack
Where the symbols declared in < > are defined as: <pNR> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 <Rule> ::= C.<CognitiveLocation> | <InstinctiveRule> <CognitiveLocation> ::= Mark | Advance | Side_Attack | Ending | None <InstinctiveRule> ::= <TypeOfCognitiveRuleAssociated> <AcronismAndInstinctiveRule> <TypeOfCognitiveRuleAssociated> ::= A | M | SA | E <AcronismAndInstinctiveRule> should be replaced by the name of an stinctive rule (which trigger a corresponding reactive action) such as: SB.seach_ball , HB.hold_ball , etc.	

Table IV shows some data about the verification. The specification consisted of 47 rules (10 cognitive and 37 instinctive). A total of 161 automata were instantiated: 11 automata from cognitive rules (one for each player), 147 automata from instinctive rules and one automaton of the environment. A total of 204 properties were checked. In the verification of each individual player all properties were checked but in relation to the complete team sate explosion occurred in the verification of 63 properties. Despite this state explosion situation, 141 properties (69.11%) were verified: 114 (80.85%) were satisfied and 27 (19.15%) were not satisfied. The maximum verification time of a property was 27.83 hours and the minimum was 0.01 seconds, approximated. It is worth mentioning that all the individual rules were completed checked but when the rules were processed collectively only the cognitive rules were completely checked and approximately 50% of the instinctive rules were not checked because of state explosion.

TABLE IV. TOTAL OF VERIFICATION.

Rules	Automata	Properties			
		Quantity Data		Time Data	
		Total 204		Minimum	Maximum
47 10 cog. 37 inst.	161 11 cog. 149 inst. 1 env.	Verified 141	Satisfied 114	~0.01s	8,5931s (~27.83h)
			Not satisfied 27		
		Not verified 63	-	-	-

VI. CONCLUSIONS

The verification of AAs and MASs plans is not straightforward, it is normally a laborious and complex activity.

This paper presented results of applying model checking on the verification of the agent plans of a robot soccer application with a three-tier architecture, where the environment is nondeterministic, dynamic and has partial vision. To minimize the state space explosion, some solutions were applied relative to abstractions of the model and for the process of verification. A modularization of the process of verification was carried out considering an evolution of the model from individual players to the whole team in order to identify errors as soon as possible, which otherwise could be difficult to interpret due to the size of the complete model.

The solutions proposed can be adapted for other similar applications and as a future work we intend to devise an interactive tool to support the development and verification of multi-agent systems with similar characteristics of this one.

REFERENCES

[1] A. El Fallah Seghrouchni et al, "Modelling, Control and Validation of Multi-Agent Plans in Dynamic Context", AAMAS, vol. 1, pp.44-51, Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1 (AAMAS'04), 2004.

[2] F. Marc, "Planification Multi-Agent sous Contraintes dans un Contexte Dynamique: Application Aux Simulations Aériennes", Thèse (Doctorat en Informatique), École Doctorale d'Informatique, Télécommunication et Électronique de Paris, Université Pierre et Marie Curie, 2005.

[3] T. A. Hezinger, "The Theory of Hybrid Automata", 28p, Electronics Research Laboratory, College of Engineering, University of California: Berkeley, 1996.

[4] L. Khatib et al, "Verification of plan models using UPPAAL", 1st International Workshop on Formal Approaches to Agent-Based Systems, Maryland, 2000.

[5] L. Khatib et al, "Mapping temporal planning constraints into timed automata", in: Proceeding of 8th International Syposium on Temporal Representation and Reasoning, 249p, IEEE Computer Society : Cividale Del Friuli, 2001.

[6] G. S. Costa, "Utilização da Verificação de Modelos para o Planejamento de Missões de Veículos Aéreos não-Tripulados", Dissertação (Mestrado em Engenharia Elétrica) - IME, Rio de Janeiro, 2008.

[7] C. B. Earle et al, "Verifying Robocup Teams", in: Proceeding of MoChArt 2008, 5th International Workshop on Model Checking and Artificial Intelligence, 189p, Patras, 2008.

[8] O. Santana Jr, C.F.G. Chavez, A. Loureiro da Costa, "MecaTeam Framework: An Infrastructure for the Development of Soccer Agents for Simulated Robots", IEEE Latin American Robotic Symposium, LARS, p 137-142, ISBN: 978-1-4244-3379-7, 2008.

[9] H. Kitano, "Robocup: The robot world cup initiative", in Proc. of The First International Conference on Autonomous Agent (Agents-97)) Marina del Ray, The ACM Press, 1997.

[10] G. Behrmann et al, "A Tutorial on Uppaal 4.0", Department of Computer Science, Aalborg University, 2006.

[11] A. Loureiro da Costa, G. Bittencourt, "From a concurrent architecture to a concurrent autonomous agents architecture", in: Third International Workshop in RoboCup, Springer Lecture Notes in Artificial Inteligence LNAI, 1856pp, pp.85-90, 1999.

[12] E. M. Clarke, O. Grumberg and D. A. Peled, "Model Checking". The MIT Press, 1999.

Parte IV

Short Papers - Resumos estendidos

Organizational Modelling of a Multiagent System basead in a Theater Play

Tatiane Dobrzanski e Gleifer Vaz Alves
Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná (UTFPR)
Câmpus Ponta Grossa Av. Monteiro Lobato, s/n, km 4
84.016-210 – Ponta Grossa – PR – Brasil
E-mail: tatianedki@gmail.com
gleifer@utfpr.edu.br

Antônio Carlos da Rocha Costa
Programa de Pós-Graduação em Modelagem Computacional
Centro de Ciências Computacionais
Universidade Federal do Rio Grande (FURG)
Av Itália, Km 8 – Câmpus Carreiros – 96.201-900
Rio Grande – RS – Brasil
E-mail: ac.rocha.costa@gmail.com

***Abstract*—Defining the social organization of a multiagent system (MAS) is a key point in agent modeling. This work presents the structural organizational of a MAS by means of MOISE+ model. The MOISE+ clearly defines three stages in agents organization: structural, functional and normative. In order to apply MOISE+ model we have chosen a (theater) play ("Auto da Compadecida", written by Ariano Suassuna), since one can clearly identify and modeling elements, such as, agents (characters), roles, links, missions, goals, social organization, as well as, the own plot of the play.**

I. Introdução

Sistemas orientados a agentes oferecem uma área promissora para o desenvolvimento de aplicações cuja interação e adaptabilidade a alterações no ambiente sejam essenciais [9]. Uma das características fundamentais destes sistemas é a capacidade de ação autônoma flexível de um agente em um ambiente.

Os agentes possuem a habilidade de se organizar, formar uma sociedade e dividir papéis e tarefas a fim de alcançar um objetivo comum. Para isso, eles relacionam aspectos sociais plausíveis ao comportamento humano, como a cooperação, coordenação e negociação [1], [9]. Sistemas multiagentes (*SMA*) são formados por um número de agentes em ambiente distribuído que levam em consideração essas características dos agentes [9].

Tendo as atribuições de um *SMA*, a sua modelagem é essencial. Assim como existem técnicas da engenharia de software aplicadas ao desenvolvimento de sistemas computacionais, metodologias e ferramentas de abstrações surgiram para sistemas orientados a agentes. Isso possibilitou a criação de uma área exclusiva, a engenharia de software orientada a agentes (do inglês, AOSE - *Agent Oriented Software Engineering*) [5]. Desde então, metodologias de desenvolvimento e modelos de organização de agentes têm sido propostas. Dentre as metodologias existentes para agentes é possível citar a GAIA, Prometheus, Tropos e Agent UML.

A organização de um *SMA* deve ser considerada como um aspecto central nestes sistemas. Ela especifica o conjunto de papéis que os agentes podem adotar, o conjunto de relacionamentos entre papéis, as normas, tarefas e compromissos dos agentes no sistema [1]. Com o estabelecimento de uma organização social é possível restringir o comportamento dos agentes, fazendo com que trabalhem em conjunto, cooperando, coordenando e negociando atividades para alcançar a finalidade do *SMA*. Como modelos de organização social de agentes pode-se citar: AGR, OperA, PopOrg, MOISE e MOISE+.

Em comparação com os modelos citados acima, o MOISE+ estabelece a *estrutura* (papéis e grupos), o *funcionamento* (esquemas sociais e missões) e as *normas* (obrigações e permissões) da organização social de agentes [4]. Estes itens compõem a *especificação organizacional*, que é instanciada pelos agentes alocados no *SMA*, costituindo, deste modo, a *entidade organizacional*.

Além disso, o MOISE+ possui a ferramenta da *AOSE*, a plataforma Moise[1], que possibilita simular a entidade organizacional. Esse modelo faz parte de um *framework* para *SMA* chamado JaCaMo (JACAMO,2012). O JaCaMo faz a integração do modelo de organização MOISE+ com a linguagem de programação para agentes, denominada Jason, através do compartilhamento distribuído de artefatos Cartago (JACAMO, 2012). Dessa forma, tem-se uma ferramenta integrada para o desenvolvimento de agentes inteligentes.

Assim, o objetivo principal deste trabalho é definir a especificação estrutural da modelagem organizacional de um *SMA* através do MOISE+. A especificação funcional e deôntica pode ser encontrada em [3]. O *SMA* escolhido foi extraído de uma peça teatral ("Auto da Compadecida", [7]), visto que em uma peça consegue-se identificar os agentes (personagens), papéis, ligações sociais, bem como o enredo da história. Características essas que vêm ao encontro, não somente da meta principal deste trabalho, mas também de metas futuras, como a aplicação de técnicas de Interactive Storytelling (ou em uma tradução livre: *história interativa*) [8], [2].

O restante do artigo está dividido da seguinte forma: a Seção II apresenta a organização de agentes através do modelo MOISE+, com a definição da estrutura. A Seção III apresenta a modelagem organizacional da peça "Auto da Compadecida" e por fim, a Seção IV apresenta a conclusão do artigo.

[1]Neste artigo, o termo Moise refere-se a plataforma para simulação de entidades organizacionais, sendo que o termo MOISE+ indica o modelo de organização de agentes apresentado por [4].

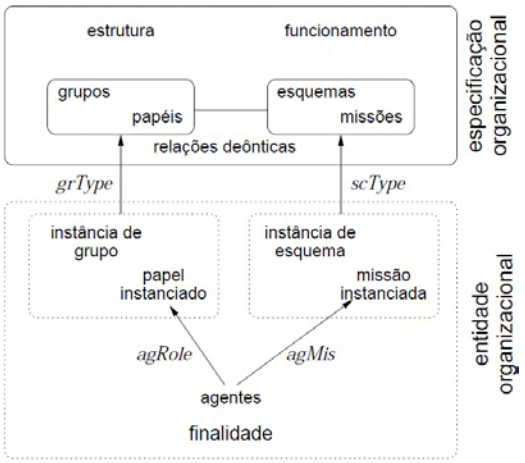

Fig. 1. Modelo MOISE+

II. MOISE+

O modelo MOISE+ proposto por [4] é centrado na organização de agentes e, com isso, duas ideias principais são definidas: a (*i*) especificação organizacional (*EO*) e a (*ii*) entidade organizacional (*EnO*) (Figura 1 extraída de [4]).

A especificação organizacional determina a estrutura, o funcionamento e as normas que regem a organização de agentes. A sua instanciação pelos agentes alocados no sistema estabelece a entidade organizacional.

Neste sentido, a especificação organizacional é formada pelo conjunto de especificações estruturais (EE), funcionais (EF) e deônticas (ED - também denominada normativa). A EE é composta pelos papéis e grupos, a EF pelos esquemas sociais e missões, e a ED pelas obrigações e permissões que os agentes possuem na organização. Como o escopo deste artigo reside apenas na especificação estrutural, a seguir, são apresentados apenas os elementos que constituem essa especificação, tais conceitos foram definidos em [4]. Contudo, a modelagem completa (especificação estrutural, funcional e dêontica) encontra-se em [3].

A. Especificação Estrutural

A especificação estrutural (EE) tem por objetivo determinar os papéis, as interações entre papéis e os grupos cujos agentes assumirão na organização.

O papel corresponde a função que o agente adota na organização. Ele é o elo de ligação entre o agente e a organização.

Os papéis podem apresentar herança, simplificando o processo da especificação de papéis. A relação de herança $\rho \sqsubset \rho'$ indica que o super-papel ρ é uma generalização do sub-papel ρ' e o sub-papel ρ' é uma especialização do super-papel ρ. Um tipo especial de papel, denominado papel abstrato, possui justamente o objetivo de simplificar a especificação de papéis em uma organização, sendo que não pode ser assumido por nenhum agente. Ademais, pode existir um super-papel comum a todos os outros papéis, o papel social (ρ_{soc}).

Ao assumir um papel, um agente pode interagir com outros através de uma ligação, denotada por $link(\rho_s, \rho_d, t)$, constituída por um papel de origem (ρ_s), um de destino (ρ_d) e um tipo de ligação (t).

Existem três tipos de ligações: (*i*) de conhecimento (*acq*); (*ii*) de comunicação (*com*) e de (*iii*) autoridade (*aut*). A ligação de conhecimento indica que o agente que assume o papel de origem têm permissão para conhecer e influenciar nas ações do que possue o papel de destino. A ligação de comunicação permite a comunicação entre agentes e a de autoridade determina que o papel de destino é subordinado ao de origem. Toda ligação de autoridade implica a existência de uma de comunicação, que, por sua vez, implica em uma de conhecimento.

Um papel também pode ser compatível com outro. De acordo com a relação $\rho_1 \bowtie \rho_2$, define-se que o agente com o papel ρ_1 pode assumir o papel ρ_2.

Apesar dos agentes adotarem um papel, eles somente o assumirão dentro de um grupo. Um grupo é composto de agentes com objetivos em comum. Uma especificação de grupo ($gt = (R, SG, L^{intra}, L^{inter}, C^{intra}, C^{inter}, np, ng)$) estabelece os papéis que podem ser assumidos no grupo (R), os sub-grupos (SG) pertencentes ao grupo, as ligações internas e externas ao grupo (L^{intra}, L^{inter}), as compatibilidades internas e externas (C^{intra}, C^{inter}) e a cardinalidade de papéis, com os respectivos valores máximo e mínimo, além da cardinalidade dos sub-grupos definidos na especificação de grupo.

Em uma ligação interna (L^{intra}), sendo $l \in L^{intra}$, todos os agentes que assumem o papel de origem da ligação l em um grupo gr_1 estão ligados por a todos os agentes que assumem o papel de destino da ligação l no mesmo grupo gr_1. Um ligação externa (L^{inter}) estabelece que todos os agentes que possuem um papel de origem estão ligados a todos os agentes que assumem o papel de destino, independentemente dos grupos em que os agentes assumem os papéis de origem e destino. Toda L^{inter} ocasiona em uma L^{intra}.

A compatibilidade interna (C^{intra}), pode-se dizer que um agente com um papel ρ_1 de um grupo gr_1 pode adotar um outro papel ρ_2 no mesmo grupo. Se a compatibilidade for externa (C^{inter}) um agente ρ_1 em um grupo gr_1 pode assumir um outro papel ρ_2 em outro grupo gr_2.

III. Modelagem Organizacional

A peça teatral "Auto da Compadecida" foi escrita por Ariano Suassuna em 1955 [7]. Baseada nos romances e histórias populares do sertão nordestino brasileiro, ela relata a história de dois amigos que juntos aprontam confusões que envolvem desde o padeiro da cidade e sua esposa, a Igreja, cangaceiros até Jesus Cristo e Nossa Senhora.

Essa Seção apresenta a especificação estrutural conforme o modelo MOISE+ de um *SMA* baseado nesta peça. Através do livro de [7], foram definidos papéis, a relação de herança, as ligações e compatibilidades que compõem uma organização de agentes.

A. Papéis e Grupos

A Figura 2 apresenta um diagrama da especificação estrutural da peça "Auto da Compadecida". Por meio dela, é possível

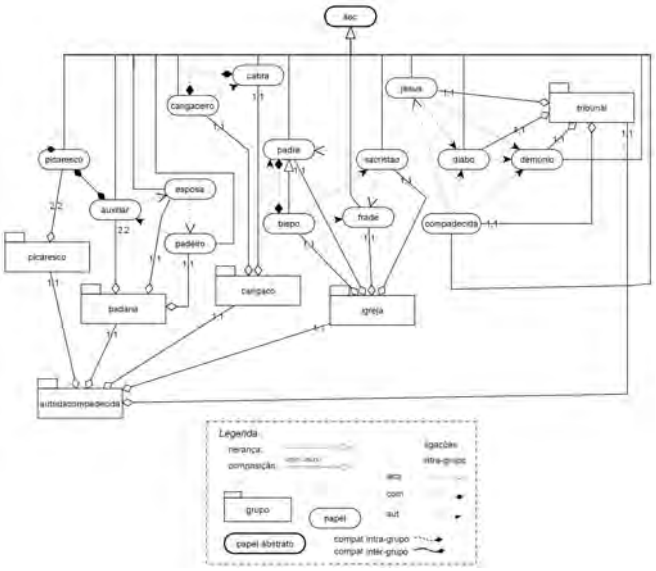

Fig. 2. Especificação estrutural da peça Auto da Compadecida

identificar o conjunto completo de papéis, grupos, compatibilidades internas e externas, além das ligações internas.

Analisando as funções que os personagens exercem na peça, foram definidos os seguintes papéis: Auxiliar de padeiro ($\rho_{auxiliar}$), Picaresco ($\rho_{picaresco}$), Padeiro ($\rho_{padeiro}$), Esposa (ρ_{esposa}), Jesus (ρ_{jesus}), Compadecida ($\rho_{compadecida}$), Bispo (ρ_{bispo}), Frade (ρ_{frade}), Padre (ρ_{padre}), Sacristão ($\rho_{sacristao}$), Cangaceiro ($\rho_{cangaceiro}$), Cabra (ρ_{cabra}) Demônio ($\rho_{demonio}$) e Diabo (ρ_{diabo}). Dentre estes papéis, destaca-se o de Auxiliar, Picaresco, Cangaceiro e Cabra.

Na peça, os personagens Chicó e João Grilo trabalham na padaria auxiliando o padeiro, deste modo, representam o papel de auxiliar ($\rho_{auxiliar}$) na organização. Além disso, como os dois aprontam trapaças e confusões para sobreviver a fome e a pobreza, eles representam típicos heróis picarescos da literatura nordestina [6], comportando o papel de picaresco ($\rho_{picaresco}$). Portanto, pode-se assumir que o papel de auxiliar é compatível com o de picaresco ($\rho_{auxiliar} \bowtie \rho_{picaresco}$) e vice-versa ($\rho_{picaresco} \bowtie \rho_{auxiliar}$).

O personagem Severino é um cangaceiro, assim como seu comparsa, o Cabra. No entanto, Severino tem autoridade sobre o Cabra, por isso os dois não podem assumir o mesmo papel (de cangaceiro). Para identificar essa ligação de autoridade, foi atribuído os papéis de cangaceiro e cabra aos personagens ($\rho_{cangaceiro}, \rho_{cabra}$), sendo que o papel de cabra é compatível com o de cangaceiro ($\rho_{cabra} \bowtie \rho_{cangaceiro}$).

Observando a interação dos personagens verifica-se a relação de herança. Um bispo possui todos os atributos de um padre, sendo portanto, uma especialização do papel de padre ($\rho_{padre} \sqsubset \rho_{bispo}$). Além disso, todos os papéis são especializações um papel abstrato social (ρ_{soc}).

Todos os papéis pertencem a um, dos cinco grupos: (*i*) Padaria ($gt_{padaria}$); (*ii*) Igreja (gt_{igreja}); (*iii*) Cangaceiros ($gt_{cangaco}$); (*iv*) Tribunal ($gt_{tribunal}$) e (*v*) Picarescos ($gt_{picaresco}$). O conjunto de especificações destes grupos ($gt_{autodacompadecida}$) e o conjunto de papéis ($R_{autodacompadecida}$) constituem a especificação estrutural ($ss_{autodacompadecida}$) do *SMA*, conforme apresentado abaixo:

$$ss_{autodacompadecida} = \langle \{gt_{autodacompadecida}\}, R_{autodacompadecida}, \sqsubset \rangle$$

A especificação do grupo auto da compadecida ($gt_{autodacompadecida}$) é bem formada se existir um, e somente um grupo (*i*) padaria, cangaço, tribunal, igreja e picaresco. O grupo padaria é considerado bem formado se existir apenas um padeiro, uma esposa e dois auxiliares. O grupo igreja é bem formado se comportar apenas um bispo, um frade, um padre e um sacristão. O grupo cangaço é bem formado se existir um cangaço e um cabra. O grupo tribunal é bem formado se houver um jesus, uma compadecida, um demônio e um diabo. O grupo picaresco é bem formado se existir dois e no máximo dois picarescos.

IV. Conclusão

Este trabalho apresenta a especificação estrutural da modelagem de um sistema multiagente baseado na peça "Auto da Compadecida" através do modelo de organização de agentes MOISE+.

Por meio deste modelo foi possível definir a estrutura, o funcionamento e as normas que regem o comportamento dos agentes, através da definição de papéis, das ligações entre papéis, relação de herança, das compatibilidades e dos grupos. Deste modo, foi possível analisar como os agentes se organizam no sistema. A peça "Auto da Compadecida" adaptou-se ao contexto de modelagem de agentes, visto que cada personagem da peça adotou um ou mais papéis, participou de grupos.

Por fim, é possível enumerar alguns trabalhos futuros: (*i*) Definição de esquemas sociais de forma que suportem variações de enredo; (*ii*) Aplicação de técnicas de *IS*; (*iii*) Implementação dos agentes através da linguagem Jason; (*iv*) Uso do *framework* JaCaMo.

References

[1] BARBOSA, R. D. M. (2011). *Especificação formal de organizações de sistemas multiagentes.* PhD Thesis, Universidade Federal do Rio Grande.

[2] CAVAZZA, M., CHARLES, F. and MEAD, S. J. (2010). Character-based interactive storytelling. *IEEE Intelligent Systems*, v. 17, p. 17-24.

[3] DOBRZANSKI, T. (2013). *Modelagem organizacional de um sistema multiagente através do modelo MOISE+*. Trabalho de Conclusão de Curso, Universidade Tecnológica Federal do Paraná, Câmpus Ponta Grossa.

[4] HÜBNER, J. F. (2003). *Organização de Sistemas Multiagentes.* PhD Thesis, Escola Politécnica, USP, São Paulo.

[5] JENNINGS N. R.; WOOLDRIDGE, M. (2000). Agent-oriented software engineering. *Handbook of Agent Technology. Anais AAAI/MIT Press.*

[6] PINHEIRO, S. R. (2002). O gótico e a picaresca se entrecruzam em cena cinematográfica do Auto da Compadecida de Ariano Suassuna. *Proceedings of the 2 - Congresso Brasileiro de Hispanistas*, v. 2.

[7] SUASSUNA, A. (2005). *Auto da Compadecida*. Agir, Rio de Janeiro.

[8] VUONO, V. (2008). Interactive storytelling via intelligent agents. *CSRS 2008 - 2nd Villanova University Undergraduate Computer Science Research Symposium*, v. 2.

[9] WOOLDRIDGE, M. (2009). *An introduction to multiagent systems*. ed. 2, Wiley Publishing.

Modeling Software Project Management with Norms and Reputation

Davy Baía
Pontifícia Universidade Católica (PUC-RIO)
Departamento de Informática
Rio de Janeiro, RJ
Email: davybaia@gmail.com

Elder Cirilo
Pontifícia Universidade Católica (PUC-RIO)
Departamento de Informática
Rio de Janeiro, RJ
Email: ecirilo@inf.puc-rio.br

Carlos Lucena
Pontifícia Universidade Católica (PUC-RIO)
Departamento de Informática
Rio de Janeiro, RJ
Email: lucena@inf.puc-rio.br

***Abstract*—The development of software projects and its results have received substantial attention from academia in the past years, mostly due to the fact that projects frequently do not achieve their expected results. Recent researches has highlighted how crucial the project manager efficiency is, in terms of project management since its activities have a straight impact in improving the success of such projects. Accomplishing such management activities in an efficient way, however, has been a challenge for most project managers. Through this paper, we propose the creation of a set of agents by applying norms and reputation concept in order to assist the project manager. We use norms to support the manager in the process of knowledge composition whereas reputation gives a better overview of each human resources in the project. Furthermore, to evaluate the proposed set of agents, we conducted exploratory case study, providing a detailed description of how the set of agents acted in a project simulation.**

I. Introdução

O desenvolvimento de software em grandes organizações é uma atividade que envolve equipes que trabalham de forma colaborativa na resolução de problemas que muitas vezes não são triviais. Os membros destas equipes seguem processos de desenvolvimento definidos pela organização na qual trabalham. Geralmente essas equipe possuem o papel de gerente de projetos, com o objetivo de conduzir o projeto e ou equipe, onde algumas de suas atividades são: alocar recursos, ajustar as prioridades, monitorar e controlar a evolução e manter a equipe do projeto concentrada na meta a ser alcançada[1]. Para realizar estas atividades com eficiência é necessário um conhecimento sobre o andamento do projeto e seus recursos. Por exemplo, uma das atividades do gerente de projeto é atribuir tarefa para cada membro da equipe. Para garantir a melhor alocação, tal atividade requer um conhecimento prévio dos recursos (pessoas). Caso contrário, isto é, atribuir uma tarefa para uma recurso que não está apto podem acarretar sérios problemas, como atrasos nas entregas ou entregas não bem sucedidas.

As atividades envolvem geralmente uma grande quantidade de informações, principalmente no contexto dos grandes projetos de software. Nesses casos, o registro e a análise manual dos dados tornam-se inviável, tanto pelo risco de ocorrência de erros decorrentes de falhas humanas, quanto pelo custo envolvido. Alguns estudos indicam que os projetos de TI continuam a ter uma taxa alta de insucesso [2] [3]. Existem evidências crescentes de que as habilidades do gerente de projeto pode ser fundamental para um desempenho eficiente e eficaz do projeto da equipe, melhorando seus resultados [4]. Para auxiliar as atividades do gerente de projeto, se faz necessário a existência de ferramentas automatizadas.

Existem tendências no uso de agentes para assumir determinadas tarefas. Assim, acreditamos que podemos utilizar agentes em ferramentas automatizadas para incorporar conhecimentos do gerente de projetos, auxiliando em suas atividades. Neste trabalho, modelamos as regras de avaliação do desempenho do recurso através de normas e o perfil do desenvolvedor através de sua reputação. Assim, podemos criar um sistema que auxilie o gerente de projeto a avaliar e equilibrar o ambiente de desenvolvimento em relação as atividades dos recursos. A utilização de sistema multiagente se torna essencial, pois oferece um abstração necessária para modelar o problema e automatizá-lo computacionalmente.

Para finalizar o processo e manter um ciclo de reconhecimento é utilizado o conceito de reputação. Por fim, para compreender o impacto da utilização de agentes com normas e reputação em ferramentas de auxílio ao gerente de projetos, foi realizado um estudo exploratório. O estudo foi executado através da simulação de um projeto. Resultados iniciais mostraram indicação positiva da utilização de agentes para auxiliar o gerente de projetos.

O documento está subdividido em Seções onde a Seção II é contextualizado Agente de Reputação baseado em Normas; a Seção III apresenta um cenário para aplicação das normas e sua descrição, assim como os resultados iniciais e perspectivas da aplicação do Agente de reputação; e por fim, na Seção IV o trabalho é finalizado com as considerações finais e trabalhos futuros.

II. Agente de Reputação baseado em Normas

Normas especificam padrões de comportamento representado por ações a serem executadas [5], [6]. Em outras ocasiões, padrões de comportamento são especificados através de objetivos que devem ser satisfeitos ou evitados. Nesse contexto, um agente pode assumir uma atividade do gerente de projeto, como por exemplo, acompanhar a evolução das atividades e seu estado por recurso (pessoas). Para isso, podemos utilizar normas como forma de modelar o que o gerente de projetos espera dos seus recursos (pessoas). Porém, para finalizar o

processo usamos a reputação para verificar se o recurso está de fato seguindo as normas.

De acordo com o dicionário Oxford, reputação é o que se diz ou se acredita sobre as qualidades de alguém ou alguma coisa. Sistemas de reputação apoiam a formação de reputação, pois o indivíduo começa a trabalhar de forma a aumentar sua reputação [7]. Os sistemas analisam o comportamento das pessoas para calcular pontos de reputação, que são publicados para a comunidade [8]. Por exemplo, no ebay o vendedor bem reputado obtém mais créditos para as suas ofertas. Como resultado, os vendedores investem em reputação e se comportar bem para sempre ter esse créditos, além de se um chancela de confiança. O uso de reputação pode ajudar o gerente de projeto em acompanhar a evolução de um recurso, ajustar a distribuição de novas atividades, recompensar os recursos e equilibrar o estado das atividades por recurso. Além disso, o próprio recurso trabalhará de forma a aumentar sua reputação, com isso mitigando a atividade do Gerente de projeto, pois o próprio recurso se autogerência . Na próxima seção iremos descrever um cenário de como foi a utilização de normas para modelar o que o gerente de projetos espera dos seus recursos.

III. Cenário

Nesta Sessão apresentaremos um cenário para utilização de agentes de reputação baseados em normas para auxiliar o gerente de projeto. A Figura 1 ilustra o sistema multi-agente, temos para cada papel um conjunto de normas e um agente. A reputação é gerada de acordo com o comportamento do recurso, se o recurso seguir as normas sua pontuação é elevada, caso ao contrario sua pontuação será baixada. Por exemplo, se o Recurso A tem atividades no estado KZW, e essas atividades correspondem uma porcentagem menor que X% do total de atividades, então o Recurso A é obrigado a aumentar para X+Y%. Caso Recurso A aumente para o desejado ele tem uma bonificação se não ele tem uma penalidade.

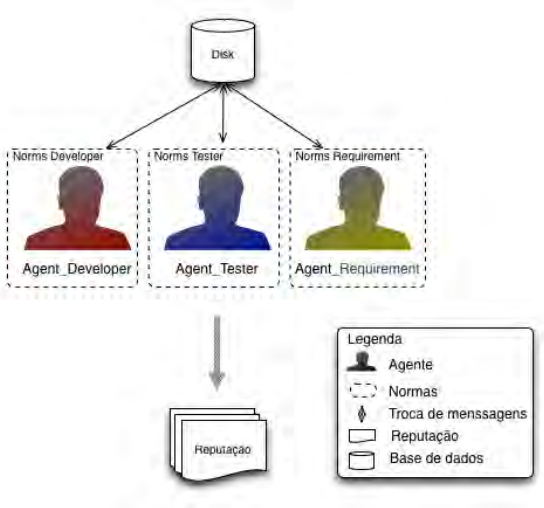

Fig. 1. Sistema Multi-Agentes

Na Seção seguinte descreveremos as normas como forma de recompensa e punição para os recursos envolvidos. O escopo desse cenário abrange o acompanhamento das atividades por recurso x atividade por estado.

A. Normas para estados da atividade e atividades por recurso

As atividades são tarefas a serem entregues ao qual é atribuída a um recurso. Cada atividade possui um estado, são esses: *Assigned, Closed, Resolved, Submitted, Opened e Activated* . Para cada projeto, o gerente precisa manter um equilíbrio sobre a porcentagem de atividades por recurso e seus estados, ajustar a distribuição das atividades e recompensar os recursos. Sendo estas, algumas das atividades que pode ser auxiliada por um agente. Para isso, utilizamos normas como forma de transcrever o que o gerente de projeto espera dos seus recursos. As normas possuem suas recompensas e punições, desta forma podemos gerar uma reputação para verificar se as normas estão sendo cumpridas. Assim, a reputação do recurso auxilia o gerente de projeto a verificar se de fato está sendo realizado o que é necessário para manter o equilbrio. Com isso, o gerente de projetos pode tomar decisões de acordo com a reputações geradas e os recursos podem seguir as normas para aumentar sua reputação. Desta forma, o agente assume o conhecimento e desejo do gerente de projeto, auxiliando em sua atividade. A fim de cumprir as metas do projeto, os estados supracitados possuem suas normas. Cada estado possui um limite em porcentagem ao qual o recurso precisa manter. O limite aplicado nesse trabalho foi a mediana. Porém, esses limites podem ser alterados de acordo com os objetivos do projeto. A escolha da mediana é justificada pelo fato de atender a nossa pesquisa.

Para o não cumprimento de cada norma existe uma punição e para o cumprimento existe uma recompensa. Por fim, assumiremos que cada recurso está associado a uma reputação, logo, o objetivo de cada recurso é aumentar ou manter sua reputação. Para isso, foi definido um conjunto de normas. Nesse artigo iremos tratar apenas a demonstração das normas para geração da reputação do recurso com o papel de desenvolvedor. As seguintes normas consideradas para o desenvolvedor foram:

(**Norma 1**): Se o Recurso X tem atividades no estado *Assigned*, e essas atividades possuem uma porcentagem menor que 26% do total de atividades, então o Recurso X é obrigado a aumentar para 26%.

- (Recompensas): Se estiver acima de 26%, a reputação do Recurso X aumenta um ponto.
- (Punições): Se estiver abaixo dos 26%, a reputação do Recurso X diminui um ponto.

(**Norma 2**): Se o Recurso X tem atividades no estado *Closed*, e essas atividades possuem uma porcentagem menor que 7% do total de atividades, então o Recurso X é obrigado a aumentar para 7%.

- (Recompensas): Se estiver acima de 7%, a reputação do Recurso X aumenta um ponto.
- (Punições): Se estiver abaixo dos 7%, a reputação do Recurso X diminui um ponto.

(**Norma 3**): Se o Recurso X tem atividades no estado *Resolved*, e essas atividades possuem uma porcentagem menor que 8% do total de atividades, então o Recurso X é obrigado a aumentar para 8%.

- (Recompensas): Se estiver acima de 8%, a reputação do Recurso X aumenta um ponto.
- (Punições): Se estiver abaixo dos 8%, a reputação do Recurso X diminui um ponto.

(**Norma 4**): Se o Recurso X tem atividades no estado *Submitted*, e essas atividades possuem uma porcentagem menor que 51% do total de atividades, então o Recurso X é obrigado a aumentar para 51%.

- (Recompensas): Se estiver acima de 51%, a reputação do Recurso X aumenta um ponto.
- (Punições): Se estiver abaixo dos 51%, a reputação do Recurso X diminui um ponto.

(**Norma 5**): Se o Recurso X tem atividades no estado *Activated*, e essas atividades possuem uma porcentagem menor que 1% do total de atividades, então o Recurso X é obrigado a aumentar para 1%.

- (Recompensas): Se estiver acima de 1%, a reputação do Recurso X aumenta um ponto.
- (Punições): Se estiver abaixo dos 1%, a reputação do Recurso X diminui um ponto.

Essas normas foram implementadas em Java com Jade e foram simuladas em uma base histórica de um projeto. Demostraremos seus resultados na sessão seguinte.

B. Resultados Iniciais e Perspectivas

As normas e o agente de reputação foram aplicados em uma base histórica, fornecida pela IBM. Foi criada uma simulação onde em uma determinada fatia de tempo o agente verificava as atividades e seu estado e pontuava de acordo com o cumprimento das normas. A Figura 2 demostra a evolução do recurso Developer 3.

Fig. 2. Evolução do Desenvolvedor 3

Podemos observar a reputação do Developer 3 em um certo ponto era negativa e depois ficou positiva. O gerente de projeto, com esses dados, poderia tomar decisões para atribuições de novas atividades para o desenvolvedor. Como por exemplo, depois de verificar a reputação negativa, o gerente de projeto poderia investigar o motivo. Supondo que o motivo foi atribuições de atividades com complexidade alta para tal desenvolvedor, o gerente poderia atribuir atividades mais simples para que a porcentagem das atividades no estado *Resolved* aumentasse. Com isso, a reputação do desenvolvedor poderá ser melhorada. Por outro lado, em posse da reputação, o desenvolvedor poderia agir de tal formal que sua reputação melhorasse, por exemplo, identificar quais normas não estão sendo cumpridas e verificar o que pode ser feito para pontuar. A continuação da analise segue o exemplo do recurso Developer 3, pois as considerações e criticas a esse exemplo servem como base para o analisarmos o restante dos recursos.

IV. Conclusão

Alguns fatores são determinantes para o sucesso ou fracasso de um projeto. Um desses fatores são as decisões tomadas pelo gerente de projetos para atingir as metas. Para auxiliar nessas decisões e nas tarefas do gerente de projeto, identificamos que um acompanhamento adequado com um conjunto de agentes contribui para alcançar essas metas. Para isso, sua criação deve seguir um método estruturado e definido. Neste contexto, foi utilizado normas para gerar a reputação. O conjunto de agente foi aplicado em um estudo de caso exploratório. Com isso, obtivemos indícios positivos que a utilização de agentes pode auxiliar o gerente de projeto em suas atividades. A realização do estudo de caso serviu como uma primeira avaliação da utilidade dos agentes para auxiliar o gerente de projeto. Pode-se concluir, então, que o estudo apresentado serviu como indicação positiva que aplicação de normas através de agentes por ser estudado mais profundamente em gerenciamento de projeto de software.

References

[1] PMBOK, *A Guide To The Project Management Body Of Knowledge - PMBOK Guides*, PMI, Ed. Project Management Institute, 2008.

[2] D. Rubinstein, "Standish group report: There's less development chaos today," *Software Development Times*, 2007. [Online]. Available: http://pdd.citsolutions.edu.au/Clients/DOGPM/documentation/Standish_Group_Chaos_Article_2006.pdf.pdf

[3] A. Budzier, "Why your it project may be riskier than you think," 2012. [Online]. Available: http://www.researchgate.net/publication/225070625_Why_your_IT_project_might_be__than_you_think/file/d912f4fc5f0ec7434f.pdf

[4] H. Taylor and J. P. Woelfer, "Leadership behaviors in information technology project management: An exploratory study," in *Proceedings of the 2011 44th Hawaii International Conference on System Sciences*, ser. HICSS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2011.280

[5] V. T. Silva, "From the specification to the implementation of norms: an automatic approach to generate rules from norms to govern the behavior of agents," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 1, pp. 113–155, Aug. 2008. [Online]. Available: http://dx.doi.org/10.1007/s10458-008-9039-8

[6] G. Boella and L. van der Torre, "Substantive and procedural norms in normative multiagent systems," *J. Applied Logic*, vol. 6, no. 2, pp. 152–171, 2008. [Online]. Available: http://icr.uni.lu/leonvandertorre/papers/jal08.pdf

[7] C. Prause and M. Eisenhauer, "First results from an investigation into the validity of developer reputation derived from wiki articles and source code," in *Cooperative and Human Aspects of Software Engineering (CHASE), 2012 5th International Workshop on*, 2012, pp. 126–128.

[8] A. John H. Baumertsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decis. Support Syst.*, vol. 43, no. 2, pp. 618–644, Mar. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.dss.2005.05.019

Integrating the Organizational Model $\mathcal{M}$oise$^+$ to a Cognitive Agent Architecture applied to Robocup Simulator 2D

Mateus Paiva Fogaça
Centro de Ciências Computacionais - C3
Universidade Federal do Rio Grande - FURG
Rio Grande – RS – Brazil
Email: mateus.p.fogaca@gmail.com

Eder Mateus Gonçalves
Centro de Ciências Computacionais - C3
Universidade Federal do Rio Grande - FURG
Rio Grande – RS – Brazil
Email: edergoncalves@furg.br

***Resumo*—This paper presents the initial results integrating the organizational model $\mathcal{M}$oise$^+$ to a cognitive agent architecture once instantiated the UvA Trilearn agent applied to Robocup Simulator 2D. At this moment, the main goal is to equate the team performance using an explicit organization compared to the same team considering the social aspects specification in an implicit form. From this integration, it is expected a more simple way to input a social organization to the team development.**

I. Introdução

Um sistema multiagente (SMA) é especificado e implementado a partir de três dimensões básicas: os agentes em si, os mecanismos de interação e comunicação entre os agentes e destes com o ambiente, e sua organização social. Destas dimensões, apenas os aspectos relacionados ao desenvolvimento interno dos agentes são naturalmente especificados de maneira explícita no projeto. No entanto, os aspectos relacionados a interação e comunicação dos agentes, bem como sua organização social, podem ser especificados de maneira implícita ao sistema ao qual farão parte.

Impor uma determinada organização a um grupo de agentes dar-se-á no sentido de estabelecer de forma explícita uma ou mais metas, sejam elas de contexto local aos agentes ou global ao sistema. Nesse caso, o papel da organização é permitir a um observador externo que se entenda para que propósito o sistema tende [1].

Em [2] é proposta uma arquitetura de agente que apresenta três níveis decisórios: nível reativo, nível instintivo e nível cognitivo. O nível reativo é composto por um conjunto de comportamentos de baixo nível que implementam as habilidades necessárias para que o agente atue no ambiente. O nível instintivo tem a função de identificar os estados do ambiente e definir qual o comportamento ativo no nível reativo. Além disso, estas informações de estado devem ser enviadas para o nível cognitivo que as utiliza, juntamente com a meta global do SMA, para determinar as metas individuais do agente. Estas metas individuais correspondem a estados desejados que devem ser alcançados pelos agentes. A escolha do melhor estado é feita a partir de um conjunto de funções preditivas que selecionam a melhor ação a ser tomada pelo agente. Esta seleção considera apenas o subconjunto de ações restringida pela modelo de organização.

A Robocup é uma entidade que visa estabelecer diretrizes para pesquisas nas diferentes áreas do conhecimento que viabilizam o desenvolvimento de artefatos robóticos em torno de problemas comuns. Um destes problemas padrão é a construção de equipes de futebol de robôs, reais e/ou virtuais. Entre as competições organizadas, tem-se aquelas voltadas para aspectos de software. No caso da simulação em 2D, tem-se uma excelente plataforma para desenvolvimentos envolvendo SMA's. A plataforma é denominada Soccerserver [3].

Este artigo apresenta a integração do modelo organizacional $\mathcal{M}$oise$^+$ à arquitetura de agente cognitivo proposto em [2]. Esta arquitetura é implementada tendo como base o agente UvA Trilearn [4], especialmente em seus componentes reativos, e que possui uma estrutura base adequada aquela proposta em [2]. O FURGBol-Sim é resultado da integração destas abordagens. Neste trabalho, apresenta-se os resultados iniciais desse processo de integração, onde o objetivo é igualar a performance do FURGBol-Sim com organização social com do código base UvA Trilearn. Isto é feito extraindo a organização social implícita ao UvA Trilearn e aplicando ao FURGBol-Sim utilizando o $\mathcal{M}$oise$^+$.

Este artigo é estruturado da seguinte forma. A próxima seção descreve a arquitetura interna do agente utilizado no FURGBol-Sim, e a seção III descreve como dá-se o aproveitamento do código fonte do UvA Trilearn para esta implementação. A seção IV descreve como dá-se a extração da organização social implícita ao código do UvA Trilearn e sua especificação e inserção no código segundo o modelo $\mathcal{M}$oise$^+$. A seção V descreve os resultados do processo de integração e a seção VI apresenta as conclusões deste trabalho e as perspectivas de futuros trabalhos.

II. Arquitetura de Agente Autônomo Concorrente

Nesta seção busca-se apresentar as características da arquitetura interna dos agentes que compõem o FURGBol-Sim, independente do modelo de organização SMA adotado.

Um ambiente complexo, como o Soccer Server, exige soluções que agreguem aspectos de baixo nível, formado por comportamentos reativos que atendam restrições de tempo

real, bem como aspectos deliberativos que permitam tratar com questões de planejamento e atendimento de metas, por exemplo. Não obstante, o agente deve estar inserido em um contexto social que permita compartilhar seus planos e metas. Nesse sentido, propõe-se um modelo de agente cognitivo, denominado agente autônomo concorrente [2], que apresenta três níveis decisórios: reativo, instintivo e cognitivo como descrito na figura 1.

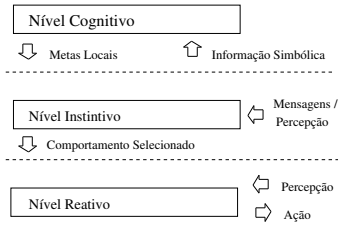

Figura 1. Arquitetura genérica de agente cognitivo

O *nível reativo* permite a interação do agente com o ambiente, tratando dos aspectos de percepção e ação. Deste modo, este nível é responsável pelo tratamento das restrições de tempo-real do agente. É formado por um conjunto de comportamentos reativos, que encapsulam as habilidades fundamentais do agente para atuar no ambiente. No caso do Soccer Server, citam-se como habilidades fundamentais, entre outras, passar, chutar, correr, marcar, interceptar a bola. Estes comportamentos podem ainda serem vistos como um nível de abstração hierárquico superior aos comandos básicos do ambiente.

O *nível instintivo* tem como funções básicas reconhecer o estado do jogo e executar a meta local do agente, que no âmbito do modelo $\mathcal{M}$oise$^+$ corresponde a especificação de uma missão. A execução da meta local, ou missão, é dada pela seleção de uma seqüência de comportamentos reativos que levam o estado atual do jogo a um estado desejado. A meta local permanece válida enquanto determinados parâmetros associados ao estado atual do jogo são verificados. Quando o estado do jogo muda, seja para um estado desejado ou não, uma nova meta local deve ser selecionada.

O *nível cognitivo* tem a função de integrar o agente no SMA, coordenando objetivos coletivos com seqüências de ações individuais. Deste modo, o nível cognitivo transforma as metas globais em metas locais, por intermédio de uma instanciação da especificação funcional (FS) do modelo $\mathcal{M}$oise$^+$, segundo os papéis que possui e o grupo ao qual o agente pertence. Comparado aos níveis inferiores, o nível cognitivo possui restrições de tempo real mais brandas, o que permite a execução de tarefas mais complexas. A escolha da meta local dá-se por intermédio de um conjunto de funções preditivas do ambiente, que escolhem dentro do subconjunto de ações do ambiente, previamente restrita pelo modelo de organização, aquela que leva ao melhor estado futuro do ambiente, segundo as metas globais do SMA.

III. UvA Trilearn

A arquitetura descrita na seção anterior é implementada a partir do código base do UvA Trilearn [4], que possui uma arquitetura de baixo nível, referente aos níveis hierárquicos mais próximos do ambiente, muito similar ao agente autônomo concorrente. A arquitetura do UvA Trilearn é descrita pela figura 2.

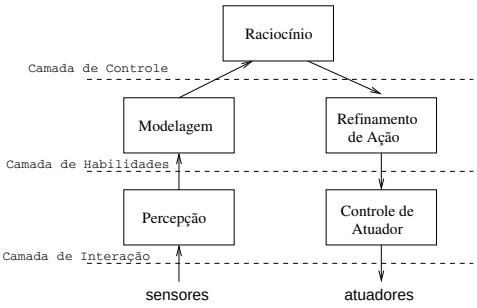

Figura 2. Arquitetura do Agente UvA Trilearn

A mais inferior é a *Camada de Interação*, que trata da interação do agente com o ambiente. A *Camada de Habilidades* usa a funcionalidade oferecida pela camada de interação para construir um modelo abstrato do mundo e para implementar os comportamentos dos agentes. A *Camada de Controle* contém os componentes deliberativos do agente. Sua função é escolher a melhor ação na camada de habilidades de acordo com o modelo de mundo e atual estratégia do agente.

O agente UvA Trilearn é implementado em C++ com suporte nativo em sistema operacional Linux, constituído por três threads: uma para percepção, uma para atuação, e outra para deliberação. A principal vantagem desta abordagem garantir um mínimo atraso em operações de entrada e saída com o servidor de simulação.

Para utilizar o UvA Trilearn como código fonte base, o objetivo é determinar uma estratégia de deliberação que garanta a escolha da melhor ação dado o modelo de mundo atual.

O mapeamento entre a arquitetura do agente autônomo concorrente e a implementação do UvA Trilearn é praticamente direta. A camada de interação mais o componente de refinamento de ação da camada de habilidades do UvA Trilearn correspondem ao nível reativo do agente autônomo concorrente. O componente de modelagem da camada de habilidades do UvA Trilearn implementa a função de identificação de estado do nível instintivo do agente autônomo concorrente. Finalmente, as funções restantes do nível instintivo mais o nível cognitivo do agente autônomo concorrente devem ser implementados na camada de controle do UvA Trilearn.

IV. Modelo de Organização Multiagente

O FURGBol-Sim é implementado segundo uma abordagem *top-down*, ou seja, da especificação social, que regula as relações entre os agentes que compõem a equipe, em direção as camadas mais baixas de implementação do agente, considerando que este possui uma arquitetura hierárquica que apresenta ambos aspectos, deliberativo e reativo. Deste modo, a descrição da equipe parte da especificação do SMA e de seu modelo de organização.

A idéia básica de um SMA é permitir que um grupo organizado de agentes cooperem na resolução de problemas que estão além das capacidades de resolução individual de cada

um deles. No entanto, esta definição contrapõem dois aspectos fundamentais da distribuição de problemas: buscar as metas do SMA e a autonomia dos agentes. A organização formal de um SMA permite identificar e ajustar o equilíbrio entre estes dois aspectos, por meio de um conjunto de restrições comportamentais adotada pelos agentes. Uma boa organização consiste em determinar um espaço de busca de ações menor que aquele determinado pelo ambiente, que corresponde a todos os mapeamentos entre percepções e ações, porém maior que aquele que leve a finalidade do SMA, de modo a respeitar a autonomia dos agentes [5].

Uma classificação para os modelos organizacionais divide-os em modelos baseados na dimensão *funcional*, *estrutural* e *deôntica*. Os modelos funcionais operam no sentido de alcançar as *metas globais* do sistema. Os modelos estruturais focam-se em conceitos como *papéis* e *grupos* para organizar os agentes. Já os modelos deônticos baseiam-se na definição de *normas* e *permissões* dentro do SMA.

Para o FURGBol-Sim, o modelo adotado é o $\mathcal{M}$oise$^+$. Este modelo organizacional foi selecionado uma vez que é o único que aborda as três dimensões, estrutural, funcional e deôntica. Segundo o modelo $\mathcal{M}$oise$^+$, as três dimensões organizacionais formam uma Especificação Organizacional (OS, do inglês *Organisational Specification*). Quando os agentes adotam uma determinada OS, eles formam um Entidade Organizacional (OE, do inglês *Organisational Entity*).

A Especificação Estrutural (SS, do inglês, *Structural Specification*) do modelo $\mathcal{M}$oise$^+$ é construída em três níveis: os comportamentos que um agente deve possuir quando ele é responsável por um papel, que corresponde ao seu *nível individual*; as relações de comunicação, autoridade e de conhecimento entre os papéis, que corresponde ao *nível social*; e a agregação dos papéis em grupos, que corresponde ao *nível coletivo*. A Especificação Funcional (FS, do inglês, *Functional Specification*) declara como o SMA alcança as *metas globais*, que são de caráter coletivo, decompondo-as em *planos* e distribuindo-os aos agentes por meio de *missões*. A Especificação Deôntica (DS, do inglês, *Deontic Specification*) descreve as obrigações e permissões dos papéis para as missões.

Para a implementação do SMA do FURGBol-Sim é utilizada a organização apresentada na Figura 3. Adotou-se a mesma organização de um jogador padrão do UvA Trilearn, onde todos o jogadores do time assumento o papel *deMeer*. Este papel possui duas atribuições: ficar bem posicionado, ou seja, se for o mais perto da bola deverá persegui-la, caso contrário ir para sua posição estratégica dentro do campo; e caso esteja com a posse da bola, chutá-la em direção ao gol.

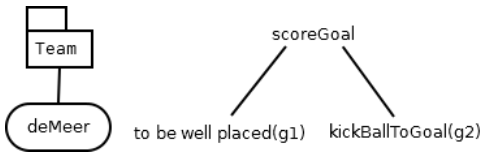

Figura 3. Organizão do FURGBol-Sim

V. Resultados

Para comparar as estratégias descrita de forma explícita e implicita, realizou-se 100 partidas das quais foram coletados os placares. A tabela I contém o somatório dos placares e a porcentagem de gols de cada equipe ao fim da simulação.

Placar	Estratégia Explícita	Implicita
Numérico	233	301
Porcentagem	43.6%	56.4%

Tabela I. Resultados da simulação

A diferença de desempenho entre as equipes é atribuida aos componentes aleatórios do Soccerserver. Este fator faz com que as ações dos jogadores não sejam precisas. Portanto, acredita-se que as duas equipes possuem o mesmo desempenho.

VI. Conclusões

Este artigo descreveu a arquitetura e a implementação da equipe FURGBol-Sim, projetada para o Soccer Server 2D. Esta equipe integra aspectos do modelo de organização $\mathcal{M}$oise$^+$ de SMA, uma arquitetura de agente cognitivo e o código fonte base do UvA Trilearn. Esta abordagem possui o mérito de restringir o espaço de busca de decisões do agente a um tamanho que garanta a realização dos objetivos coletivos do SMA, respeitando a autonomia dos agentes.

Nesta etapa do trabalho o objetivo é equipar o desempenho da equipe base original, com uma organização social implícita ao modelo, a uma equipe que integra os componentes de alto nível do agente cognitivo proposto em [2] junto com a explicitação da organização social original utilizando o modelo $\mathcal{M}$oise$^+$. Os resultados obtidos nos levam a acreditar que se obteve êxito no objetivo.

O resultado mais evidente deste processo é a modularização do processo de desenvolvimento em torno de conceitos sociais como meta globais, missões, papéis, etc. Deste modo, permite-se um desenvolvimento incremental da equipe de modo a atender um número cada vez maior de situações coletivas no ambiente. Não obstante, os próximos passos prevêem a extensão do número de situações atendidas pela equipe elevando a sua performance coletiva.

Referências

[1] V. Dignum and F. Dignum, "Modelling agent societies: Co-ordination frameworks and institutions," in *Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA'01)*. Berlin: Springer, 2001, pp. 191–204, lNAI 2258.

[2] A. L. da Costa and G. Bittencourt, "From a concurrent architecture to a concurrent autonomous agents architecture," in *International Joint Conference on Artificial Intelligence (IJCAI'99)*, 1999.

[3] M. Chen, E. Foroughi, F. Heintz, Z. X. Huang, S. Kapetanakis, K. Kostiadis, I. N. Johan Kummeneje, O. Obst, P. Riley, Y. W. Timo Steffens, and X. Yin, *RoboCup Soccer Server: for Soccer Server Version 7.07 and later*, May 2001, www.robocup.org.

[4] J. R. Kok, N. Vlassis, and F. Groen, "Uva trilearn 2003 team description," in *Proceedings CD RoboCup 2003*, D. Polani, B. Browning, A. Bonarini, and K. Yoshida, Eds., Padua, Italy, July 2003.

[5] J. F. Hübner and J. S. ao Sichman, "Aplicação de organização de sistemas multiagentes em futebol de robôs," in *XI Escola de Informática do SBC*, vol. 1. Lages-SC: Angelo Augusto Frozza, 2003, pp. 119–147.

Behavior Editor for Agents Based on Service Oriented Architecture

Saulo Popov Zambiasi
Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
Florianópolis, SC – Brasil
saulopz@gmail.com

Ricardo J. Rabelo
Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
Florianópolis, SC – Brasil
rabelo@das.ufsc.br

***Abstract*—Many efforts have been made to create agents more autonomous, flexible and dynamic. Concomitantly with the fact that agents are inserted in environments distributed, interconnected and with a lot of dynamics information, it agents need to communicates with others and other systems by the inter-operable way. The use of a standard, such as Service Oriented Architecture and web services, can provide an inter-operable way to agents communication and to executes some distributed operations. Thus, this paper presents a proposal of a behavior editor for agents based on Service Oriented Architecture. A prototype of a behavior editor and an executor of agents was implemented and it is also presented.**

Keywords—agents; behavior; service-oriented architecture.

I. INTRODUÇÃO

O cenário da computação distribuída tem aumentado significativamente a complexidade dos ambientes computacionais. Consequentemente, as pessoas veem-se inseridas em um ambiente pessoal cada vez mais distribuído e com uma grande quantidade de informações altamente dinâmicas. Dentro desse contexto, os agentes, como forma de auxiliar seus usuários, precisam interagir com este ambiente e manter a compatibilidade com os mais diversos tipos de conteúdos espalhados neste meio [2]. Para isso, técnicas para a criação de agentes e algoritmos para compor a forma como os agentes agem têm sido estudadas e aperfeiçoadas.

Uma maneira para resolver problemas de interoperabilidade nesse cenário distribuído é a utilização de estruturas e protocolos padronizados. A utilização de serviços distribuídos pelos agentes, via Arquitetura Orientada a Serviços (*Software Oriented Architecture* – SOA), traz consigo as vantagens dessa tecnologia. Nessa, as funções dos agentes se apresentam como serviços independentes, cada qual com interfaces de invocação bem definidas e que podem ser utilizadas sequencialmente ou paralelamente para compor processos de negócios [8].

Enquanto SOA é vista como uma forma bastante eficiente de executar programas remotamente via Internet, utilizando protocolos amplamente utilizados na atualidade, os agentes são uma tecnologia já firmada na computação como uma maneira para resolver problemas complexos [4]. Com a utilização de SOA em agentes, é possível que um agente possa se utilizar de chamadas à serviços web para efetuar algumas operações, ou mesmo processos de negócios. Porém, o usuário precisa poder personalizar as ações do seu agente, adicionar chamadas à serviços web, formas de interação, etc. de modo a este agente possa se adaptar às suas necessidades [9] e sem que o mesmo precise ser recompilado. Nesse contexto, esse artigo apresenta uma proposta de edição e execução de comportamentos de agentes integrados com SOA. Também é apresentado um protótipo implementado para os testes da proposta.

II. BREVE REVISÃO DA LITERATURA

Um agente computacional é visto como uma entidade que: (i) tem uma identidade persistente para realizar transações, manipular exceções, construir uma história de interações e definir a confiança para com outros agentes; (ii) percebe e responde ativamente às suas atividades no seu ambiente; (iii) se comunica com outros agentes ou pessoas [13]. Os agentes devem refletir a mesma dinâmica que um usuário apresenta na vida real, ou seja, assim como os interesses do usuário mudam com o tempo, os agentes também devem ajustar seus objetivos para corresponder com seus usuários [12].

Uma importante questão dos agentes está em construí-los de forma que possam ser facilmente personalizados para cada usuário. Muitos programas proveem simples parâmetros que permitem aos usuários personalizarem seus comportamentos explicitamente. Porém, em geral, essa interação é bastante limitada. Uma maneira sofisticada de se resolver esta limitação é por meio da dinamicidade e da escalabilidade. Com isso, muitos usuários podem ter disponível a possibilidade de programar suas preferências de forma explícita. Ainda assim, algumas tarefas, como personalizar um filtro de e-mails para selecionar mensagens urgentes, por exemplo, necessitam que a pessoa tenha certa noção detalhada e possa articular conceitos bastante sutis. Uma forma bastante interessante para resolução desse problema é deixar que especialistas desenvolvam os filtros e ao usuário fica apenas o papel de utilizá-lo e configurá-lo conforme suas necessidades. Entretanto, o problema de se lidar com a flexibilidade da forma como o agente se comporta vai além de simplesmente alterar a configuração de um agente para cada usuário [9].

Observa-se também a necessidade da utilização de padrões para comunicação entre agentes e outros softwares [2]. A melhor forma de se resolver esse problema é adotar algum tipo de tecnologia já difundida, tal como SOA, por exemplo.

SOA *"é um paradigma para organização e utilização de competências distribuídas que estão sob o controle de diferentes domínios proprietários"* [3]. As pessoas e organizações criam competências para resolver problemas específicos conforme suas necessidades, modeladas por um

conjunto de serviços de softwares [1]. Esses serviços possuem interfaces que podem ser publicadas e descobertas por consumidores de serviços e podem ser agrupados para a criação de diferentes aplicações e processos de negócios, utilizando-se de um modelo de comunicação baseado na troca de mensagens com baixo acoplamento [6], [10].

Os serviços web, por sua vez, são softwares que buscam a interoperabilidade através da interação entre um computador e uma rede. Estes serviços se comunicam entre eles e entre sistemas via mensagens baseadas no protocolo HTTP (*Hipertext Trasnfer Protocol*) [5].

Uma das vantagens de se trabalhar com serviços web em SOA está na estrutura flexível que permite a criação de novos serviços com a composição de outros serviços. Além disso, eles seguem os requisitos de SOA: (i) **Baixo acoplamento**, os serviços da arquitetura não devem ter uma dependência forte entre si; (ii) **Independência de implementação**, não se deve depender de características específicas de linguagens de programação ou ambientes de execução; (iii) **Configuração flexível**, os diferentes serviços devem poder ser ligados entre si de forma dinâmica e configurável; (iv) **Tempo de vida longo**, os serviços devem existir por tempo suficiente para que sejam descobertos e utilizados até se obter confiança em seu comportamento; (v) **Granularidade**, as funcionalidades de um sistema devem ser divididas em vários serviços; e (vi) **Distribuição**, os serviços devem ficar distribuídos, para aproveitar melhor os recursos computacionais [13].

Considerando que um serviço pode ser uma interface de um agente, podendo ser solicitado por um software ou agente, resultando na troca de mensagens, executando as atividades e interagindo via protocolo de troca de mensagens definida na descrição do serviço, pode-se então visualizar agentes em um conceito de SOA. Assim, são definidas entidades que atuam para a execução de determinada atividade ou para atingir algum objetivo. Por mais que se tratem de metodologias específicas, ambas trabalham com a noção de atividade, objetivo, tarefa e interação orientada a mensagens [11].

Para que os agentes possam trabalhar com SOA, deve haver três características: (i) um agente deve poder descobrir serviços e se adaptar a eles; (ii) devem existir serviços web com a descrição do protocolo para que um agente possa invocá-lo e; (iii) um agente deve ser capaz de realizar interações complexas com vários serviços ao mesmo tempo [7].

Assim, desenvolvedores podem encontrar e utilizar agentes pela invocação de serviços web. Tal integração traz benefícios imediatos em que torna um serviço web capaz de invocar um agente de serviço e vice-versa, permitindo assim, através dos conceitos e tecnologias de agentes, novas e avançadas funcionalidades na utilização de SOA [4].

III. GERENCIADOR DE COMPORTAMENTOS

Basicamente, a forma como as informações estão estruturadas/relacionadas são a base para a execução do algoritmo do agente. Dessa forma, tal estrutura é a primeira parte apresentada na proposta. A forma como o agente é executado está diretamente ligada com essa estrutura, permitindo que o usuário possa personalizar a ordem e forma da execução das operações em tempo de execução. Por fim, é mostrado brevemente o editor que permite que essa estrutura possa ser configurada para então ser executada pelo software servidor.

A. Estrutura das Informações

A Estrutura das Informações, visualizadas em um modelo entidade-relacionamento (ER) na Fig. 1., é utilizada para organizar as informações e a execução dos comportamentos.

A entidade *Service* é armazena os serviços web que podem compor os comportamentos do agente. Por meio desta, o sistema busca o WSDL do serviço web e apresenta para o usuário suas operações.

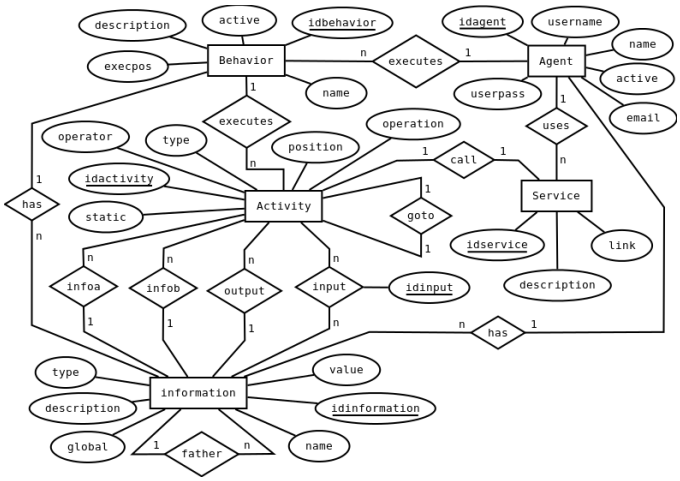

Fig. 1. Diagrama Entidade-Relacionamento do Sistema de Gerenciamento de comportamento dos agentes.

No protótipo, um agente possui uma estrutura de informações na forma de árvore, armazenadas na entidade *Information*. Uma informação pode ter várias informações filhas e cada filha pode ter outras diversas filhas. Para manter essa relação, em um atributo chamado *father* é definida a informação pai. Para a informação raiz, *father* é *null.* As informações podem ser de diferentes tipos, definidos no atributo *type*. Os tipos podem ser básicos como *string*, *integer*, *float*, *boolean*, *text*, ou um tipo complexo chamado *struct*. Se o tipo for *struct*, então a informação é caracterizada como pai, podendo conter *n* informações filhas. Para que uma informação possa ser visualizada e utilizada por outros comportamentos do agente, atribui-se *true* para o atributo *global* da informação. Caso contrário esta só pode ser visualizada no comportamento da qual a informação pertence, identificado em *idbehavior*.

Um agente pode possuir diversos comportamentos executando concorrentemente. Para que um comportamento entre em execução, deve ser atribuído *true* ao atributo *active* em *Behavior*. A estrutura de execução dos comportamentos proposta neste trabalho segue um formato de algoritmo. Cada linha de execução é uma atividade e está colocada em sua devida ordem. O atributo *execpos* se referencia à atividade em que a execução do comportamento se encontra. Se o usuário requisitar a interrupção de um comportamento, a posição de execução é salva para que o este possa voltar a se executar a partir da posição que parou.

As linhas de execução dos comportamentos são chamadas de atividades (*Activity*). São elas:

- ***assign call***: invoca um serviço web. Em *service* há um link para o serviço e em *operation* a operação. O *output* é ligado à uma *Information* e recebe o retorno da invocação do serviço. Os atributos *inputs* (*Information*) são os parâmetros passados à operação do serviço;
- ***assign call var***: funciona como uma *assign call*, mas utiliza o link para o serviço web e a operação na forma de variáveis (*Information*);
- ***assign static***: uma variável recebe um valor passado de forma estática e armazenado no atributo *static* da entidade *Activity*;
- ***assign var***: uma variável (*infoa*) recebe o valor de outra variável (*infob*);
- ***conditional static***: se uma condição (se-então) é satisfeita, o bloco interno é executado. Os atributos utilizados para efetuar a validação são *infoa*, *operator* (==, !=, <, >, <=, >=, contem) e *static* (valor estático);
- ***conditional var***: funciona tal como a atividade anterior, mas a segunda informação também é uma variável;
- ***conditional end***: aponta (*goto*) para o local onde o bloco condicional inicia. As atividades *conditional static* e *conditional var* precisam possuir uma ligação (*goto*) ao seu *conditional end*.
- ***loop static***: funciona como o *conditional static*, mas o bloco continua sendo executado enquanto a condição é satisfeita.
- ***loop var***: funciona como o *conditional var,* mas continua executando enquanto a condição é satisfeita.
- ***loop end***: Indica fim de bloco de um laço e deve ser ligado (*goto*) com seu início de bloco de laço também.

Essas informações são utilizadas pelo servidor para executar os comportamentos dos agentes, O usuário pode criar, alterar e excluir comportamentos por meio de uma interface web de configuração.

B. *Software Servidor*

O protótipo baseado na proposta desse artigo foi desenvolvido para suportar múltiplos usuários, cada qual com seu próprio agente. O servidor se mantém em um laço de execução, avaliando a cada período de tempo se o agente está ativo. Se um agente estava em execução e seu atributo *active* é alterado para *false*, significa que o agente foi desligado. O servidor envia uma mensagem para o agente avisando que o mesmo foi desligado. O agente salva suas informações e termina sua execução.

Cada agente é uma *thread* que possui uma lista de comportamentos e sua execução se dá na forma de um laço que efetua o gerenciamento desses comportamentos.

A utilização de *threads* se dá pois essas possuem suporte há múltiplas linhas de controle dentro de um processos. Elas ocupam o mesmo espaço de endereçamento, mas cada uma possui seu próprio ponteiro de controle de execução [14]. Um programa não precisa ficar esperando o retorno para continuar trabalhando em outras tarefas. Ou seja, podem haver vários agentes sendo executados simultaneamente no servidor. Os comportamentos (*Behavior*) dos agentes são compostos de várias linhas de execução (tal como um algoritmo) e precisam ser executados em paralelo. Nesse caso, também foram implementados como *threads*.

Os comportamentos se mantém em execução, executando cada uma de suas atividades. Há um ponteiro de execução (*execpos*) que indica qual atividade está em execução. Quando esta é executada, o ponteiro pula para a próxima atividade. O que define qual atividade é executada na sequência depende do retorno da execução da atividade corrente. Por exemplo, se uma atividade simples de atribuição estática *assign static* é chamada, ela retorna a posição dela acrescida de um, ou seja, a atividade seguinte. Se for um tipo condicional, a posição retornada é, ou a primeira atividade interna do bloco (se válida a condição), ou a próxima posição após seu final de bloco (se inválida). No caso do laço, retorna ou a primeira posição do bloco, ou a próxima posição do final de bloco e no final do bloco do laço, a posição é direcionada para o início do bloco.

Quando *execpos* possui um valor maior do que a quantidade de atividades que o comportamento possui, então o valor de *execpos* retorna para 1, e a execução do algoritmo recomeça.

C. *Interface Web de Configuração*

A interface web de gerenciamento do agente serve para criar um agente, adicionar informações, serviços e comportamentos. Cada comportamento pode ser ativar ou desativar.

Nessa interface o usuário pode cadastrar um novo serviço web para ser utilizado nos comportamentos do agente, alterar os existentes, ou visualizá-los. O usuário pode visualizar detalhes desses serviços, com suas operações e um link para o documento WSDL do serviço web.

A edição dos comportamentos (Fig. 2.) possui (à esquerda) informações específicas do comportamento (nome e descrição). Também há dois botões, o botão de ligar/desligar comportamento e o botão de recarregar informações na página.

Ao clicar no link “Detalhes”, aparece a lista de informações do comportamento e a lista das informações globais. O usuário pode acrescentar novas informações, editar ou excluir. Observa-se que informações que estão sendo referenciadas por alguma atividade não podem ser excluídas. Localizado na parte direita da Fig. 2. está o comportamento na forma de algoritmo.

Fig. 2. Editor de Comportamentos do agente.

Para adicionar uma nova atividade em um comportamento, o usuário deve clicar no ícone “+” nas linhas de atividades. O

ícone "-" serve para excluir uma atividade. Caso uma atividade seja um bloco, apenas o início do bloco e o final são excluídos, restando os elementos internos.

As setas servem para alterar a ordem de uma atividade. Ao clicar a seta para cima, uma atividade é trocada pela atividade anterior, se clicado na seta para baixo, a atividade é trocada pela posterior. Quando um início ou fim de bloco é movido, todo o bloco segue junto. Se ao mover um bloco para cima, for encontrado o final de outro bloco, então o bloco que foi movido é inserido dentro do bloco superior.

Quando o usuário passa com o mouse por cima de informações e outros itens da atividade, outras informações são apresentadas, como o valor de uma informação, o link de um serviço web, etc.

A atividade que se encontra em execução, no momento em que a página é carregada, é apresentada com uma cor azul claro. Para que o usuário possa visualizar o andamento da execução, é necessário clicar no botão recarregar, na parte esquerda da página. Neste protótipo, as informações não são carregadas automaticamente.

O ícone de edição, entre o "-" e a seta para cima, serve para editar um comportamento existente. Para cada tipo de atividade diferente, um formulário diferente é apresentado. No caso, é a edição de uma atividade de chamada de serviço web (Linha 2).

Um campo do formulário é reservado para a informação que deve receber o retorno da invocação do método, o segundo campo é o link do serviço e o terceiro a operação. Ao colocar ou alterar essas informações, deve ser selecionado o botão "ok" para salvar os dados, antes mesmo de selecionar os parâmetros. Em "Parâmetros:" há uma lista de parâmetros que já foram inseridos, na ordem que deve estar na operação do serviço web. Também à um botão de excluir, para retirar um parâmetro já inserido. Para adicionar um novo parâmetro, basta selecionar selecionar da lista de informações do comportamento e informações globais e clicar no botão "adicionar".

Quando uma atividade do tipo condicional ou laço é criada, automaticamente, e logo na sequencia, uma atividade de fechamento de bloco é criado (FIM SE ou FIM ENQUANTO).

IV. Considerações Finais

Este artigo apresentou uma proposta para edição do comportamento de agentes que permite que o usuário possa personalizar o agente para suas necessidades, em tempo de execução e sem que precise uma recompilação do agente.

Ainda, a proposta permite que o usuário possa configurar seu agente para efetuar chamadas a serviços web. Isso permite ao agente executar operações remotas, distribuindo e paralelizando o processamento e dando maior poder computacional. Essas operações também podem ser implementadas e fornecidas por terceiros. Além disso, por meio das chamadas a serviços web, o agente pode efetuar processos de negócios para os usuários dos agentes. Em tempo, a estrutura de algoritmos da proposta permite que possa ser efetuada uma orquestração de serviços web no agente.

Para avaliar a proposta em funcionamento, um protótipo foi desenvolvido. Este foi dividido em duas partes: (i) um editor de comportamentos que pode se utilizar de chamadas de serviços web para compor as atividades dos algoritmos dos comportamentos dos agentes, tal como chamadas de funções em linhas de execução em linguagem de programação e; (ii) um software servidor que mantem os agentes em execução, rodando paralelamente cada agente e cada comportamento desses agentes.

Os próximos passos podem seguir em diversos vieses. Não obstante, este artigo limita-se aqui a uma sugestão inicial, tal como a criação de uma interface de configuração dos comportamentos de forma gráfica, ou seja, a utilização de elementos de fluxogramas para compor os comportamentos ao invés de uma estrutura na forma de algoritmos. Tal recurso visa a facilitar a utilização do editor por usuários que não entendem de programação, mas que conseguem entender a lógica de um fluxograma de execução.

Referências

[1] Booth, D.; Haas, H.; McCabe, F. (2004). Newcomer, E.; et al. "Web Services Architecture". Disponível em: <http://www.w3.org/TR/ws-arch/> Acessado em Março/2013.

[2] Bush, J.; Irvine, J.; Dunlop, J. (2006). "Personal Assistant Agent and Content Manager for Ubiquitous Services". Wireless Communication Systems, 2006. ISWCS'06. 3rd International Symposium on. pg.169-173.

[3] Estefan, J. A.; Laskey, K.; McCabe, F.; Thorthon, D. (2008). "Reference Architecture for Services Oriented Architecture Version 1.0". OASIS. Disponível em: <http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf>. Acessado em: Março/2013.

[4] Greenwood, D.; Calisti, M.; at al. (2004). "Engineering Web Service - Agent Integration". IEEE International Conference on Systems, Man and Cybernetics. v2.

[5] Haas, H e Brown, A. (2004). "Web Services Glossary". Disponível em: <http://www.w3.org/TR/ws-gloss/>, Acessado em Março/2013.

[6] Huang, Y. e Chung, J.Y. (2003). "A web services-based framework for business integration solutions". Electronic Commerce Research and Applications, 2(1):15–26. Disponível em: <http://www.sciencedirect.com/science/article/B6X4K-48642HS-1/2/a751ffc1be1676f5b9955ea9050c160d>, acessado em Fevereiro/2013.

[7] Kuno, H. and Sahai, A.. (2002). "My agent wants to talk to your service: personalizing web services through agents". Proceedings of the First International Workshop on Challenges in Open Agent Systems. Pg 25-31.

[8] MacKenzie, C.; Laskey, K.; McCabe, F. at all. (2006). "Reference Model for Service Oriented Architecture 1.0". OASIS Standard. Disponível em: <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf>. Acessado em Fev/2013.

[9] Michaell, T., Caruana, R., Freitag, D., McDermott, J., Zabowski, D. (1994). "Experience with a learning personal assistant". Communications of the ACM, July.

[10] O'Brien, L.; Bass, L.; Merson, P. (2005). "Quality attributes and service-oriented architectures". Technical Note - Software Engineering Institute, Carnegie Mellon University.

[11] Ricci, A.; Buda, C.; Zaghini, N.. (2007). "An agent-oriented programming model for SOA & web services". Industrial Informatics, 5th IEEE International Conference on. v2.

[12] Sensoy, M. and Yolum, P.. (2008). "Evolving service semantics cooperatively: a consumer-driven approach". Springer Science+Business Media, LLC, Nov 9.

[13] Singh, M.P. and Huhns, M.N.. (2005). "Service-oriented computing: semantics, processes, agents". John Wiley & Sons, New York, NY, EUA.

[14] Tanenbaum, A.S.. (2010). "Sistemas Operacionais Modernos". Prentice Hall - Br. 3ed.

[15] Weiss, G.. (1999). "Multiagent systems: a modern approach to distributed artificial intelligence". MIT Press.

Model Oriented Approach to Code Generation for Normative Multi-Agent Systems

Robert M. R. Júnior, Emmanuel S. S. Freire e Mariela I. Cortés
Grupo de Engenharia e Sistemas Inteligentes (GESSI)
Departamento de Computação – Universidade Estadual do Ceará (UECE)
Fortaleza, Brasil
{robstermarinho, savio.essf}@gmail.com, mariela@larces.uece.br

***Abstract*—The increasing complexity of the normative multi-agent systems (MAS) development represents a challenge to software engineering. Model driven approach promotes a fast and consistent software development through the use of software models. In order to reduce the semantic gap that exists between modeling and implementation levels and surround the natural complexity associated to Normative MAS development, this work proposes the use of a model driven approach to develop Normative MAS. A template-based approach was used to automate the code generation process from NorMAS-ML models to the specific platform (JADE), it was named JAMDER 2.0 that contains all resources of JADE and adds new entities to adapt the concepts of each other.**

Keywords— Multi-Agent Systems; Norms; Model Driven Architecture; NorMAS-ML; JAMDER

I. INTRODUÇÃO

Sistemas Multi-Agente (SMA) normativos envolvem uma grande variedade de entidades, com isso, aumenta-se bastante a complexidade no processo de desenvolvimento [6]. O principal fator é a diferença semântica entre a fase de detalhamento do projeto ao longo de um conjunto de modelos, e a fase de implementação, que tem por objetivo a codificação do sistema. Nesse contexto, são necessárias linguagens de modelagem e programação que ajudem os desenvolvedores na construção de SMAs normativos e ferramentas que permitam a transição sistemática entre a modelagem e fases de implementação. Este artigo apresenta uma abordagem baseada na arquitetura orientada a modelos (MDA), a qual utiliza transformações e técnicas de geração de código a partir de artefatos de modelagem gerados pela linguagem de modelagem NorMAS-ML [6]. Diagramas da linguagem são gerados através do ambiente de modelagem MAS-ML Tool [7]. Como plataforma de implementação dessas entidades, destacamos o framework JAMDER 2.0 [10], definida como uma extensão do framework JAMDER (JADE to MAS- ML 2.0 *Development Resource*) [8]. Este trabalho está organizado da seguinte maneira: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 apresenta a abordagem MDA. A Seção 4 aborda o framework JAMDER 2.0 e os *templates Acceleo* para JAMDER 2.0. A seção 5 apresenta um estudo de caso e a seção 6 apresenta as conclusões e trabalhos futuros.

II. TRABALHOS RELACIONADOS

Nesta seção, analisamos os principais trabalhos que abordam a geração de código para SMA normativos.

De Maria [4] propõe a geração de código baseado na arquitetura MDA para SMA utilizando a linguagem de modelagem MAS-ML e o *framework* de implementação ASF que contempla as entidades tipicamente encontradas em SMAs. Entretanto, possui suporte limitado aos conceitos normativos, pois a ferramenta é baseada em MAS-ML [11].

TAOM4E [13] é um ambiente de modelagem orientado a agente e suporta o desenvolvimento orientado a modelos. A ferramenta é um *plug-in* para a plataforma Eclipse, no entanto permite a geração de código apenas para agentes BDI (*belief-desire-intention*).

SMA *modeler* [12] é um metamodelo que possui independência com diferentes abordagens de desenvolvimento e plataformas de implementação e possibilita criar modelos de SMA. Porém este metamodelo não contempla todas as entidades de um SMA normativo e não possui uma ferramenta de modelagem.

Considerando a necessidade de uma abordagem de geração de código, a abordagem de De Maria [4] foi escolhida porque (i) sua geração de código é baseada na arquitetura MDA para SMA e (ii) utiliza a linguagem de modelagem MAS-ML que serviu como base para a definição da linguagem NorMAS-ML.

III. DESENVOLVIMENTO ORIENTADO A MODELOS

No desenvolvimento orientado a modelos (MDD), os diagramas da fase de modelagem possuem uma grande importância, pois qualquer modificação nos modelos conceituais reflete automaticamente no código gerado facilitando a manutenção e evolução dos sistemas além de trazer um foco maior na modelagem ao invés do código [5].

A arquitetura dirigida por modelos ou *Model Driven Architecture* (MDA) é uma iniciativa da OMG [9] com o intuito de formalizar os conceitos de MDD em um padrão para ser adotado pela comunidade e indústria de desenvolvimento de software. Com isso, a OMG propôs um processo de transformação, projetado para ser aplicado a diferentes linguagens de modelagem.

O código gerado a partir da transformação do modelo da aplicação é estabelecido pelo conceito ISM (*Implementation Specific Model*) dentro desta arquitetura. Neste contexto, podemos destacar o *plug-in* Acceleo [1] que permite a geração automática de código a partir de um metamodelo definido pelo

usuário que esteja de acordo com o EMF (*Eclipse Modeling Framework*). Como vantagens, o *plug-in* possui integração direta com o Eclipse e tem a possibilidade de extensão.

IV. Abordagem Baseada em Modelos para Sistemas Multi-Agente Normativos

No contexto da abordagem MDA, um suporte automatizado para a geração de código a partir de modelos de NorMAS-ML é proposto. A ferramenta MAS-ML Tool [7], desenvolvido como *plugin* para o Eclipse, permite a modelagem do diagrama de normas com base no metamodelo de NorMAS-ML. Esta ferramenta gera o arquivo masml. (formato XMI) que armazena a estrutura de dados das entidades e os aspectos estruturais e comportamentais definidos em NorMAS-ML. Estes arquivos representam a entrada para o processo de transformação a partir dos modelos de código.

Como o objetivo de fornecer a infraestrutura adequada voltada para a implementação de entidades de um SMA normativo de acordo com a linguagem NorMAS-ML, o framework JAMDER 2.0 é proposto.

A. *Jamder 2.0*

JAMDER 2.0 [10] é uma extensão do framework JAMDER [8] que incorpora os recursos oferecidos em nível de modelagem por NorMAS-ML, dentre eles, especificamente, as normas e suas propriedades. A extensão foi realizada por meio da inclusão de um conjunto de classes para representar os conceitos normativos. Essas classes são descritas a seguir: (i) para representar a entidade norma foi criada a classe *jamder.norms.Norm* que define as seguintes propriedades: o identificador, o tipo da norma, a entidade restringida, o contexto, a ação e a lista de restrições de ativação; (ii) a classe *jamder.norms.NormResource* representa qualquer tipo de recurso que será restringido; (iii) as ações vinculadas às normas foram representadas pela classe principal *jamder.norms.NormAction* e suas duas subclasses que definem operações do sistema: *AtomicAction* e *CompositeAction*; (iv) a classe *jamder.norms.NormConstraint* foi criada para a representação das restrições de ativação de uma norma, com suas respectivas subclasses que contemplam cada tipo de restrição: *jamder.norms.Before*, *jamder.norms.After*, *jamder.norms.Between* e *jamder.norms.IfConditional*. Ela está diretamente ligada à classe *jamder.norms.Date* responsável por definir uma data.

Outras classes existentes em JAMDER sofreram modificações para contemplar as propriedades definidas em NorMAS-ML. Essas modificações juntamente com as novas classes formam o framework JAMDER 2.0 que possibilita a modelagem das propriedades e relacionamentos das entidades que compõem um sistema multi-agente normativo. O código completo de JAMDER 2.0 encontra-se em: https://sites.google.com/site/uecegessi/estudo-de-caso/jamder-2-0

B. *Geração de código*

Para definir o processo de transformação para a geração de código utilizamos o plugin Acceleo [1]. O plugin permite desenvolvimento incremental em que o código pode ser gerado, modificado e reutilizado. Para formalizar a geração de código em Acceleo, é necessário estabelecer um template para cada entidade através da linguagem MTL (*Model Transformation Language*) [9]. Quando o template é executado no Eclipse é necessário especificar o modelo NorMAS-ML (arquivos .masml no formato XMI) e a pasta de saída para as classes (classes geradas em JAMDER 2.0). Dessa forma, a estrutura de cada entidade no diagrama de modelagem é verificada pelo template.

Uma norma é uma instância de uma classe que herda a classe *Norm* definida em JAMDER 2.0 e seu *template* possui uma regra de definição do construtor que contempla os vários casos de modelagem envolvendo a entidade definida como contexto e a entidade restringida segundo a especificação feita na modelagem. A Figura 1 apresenta parte da estrutura do *template* para a entidade *Norm*.

```
[comment encoding = UTF-8 /]
[module Norm('masml')]
[template public generateElement(c : NormClass)]
[comment @main/]
[file (c.name + '.java', false, 'UTF-8')]
import java.util.Hashtable;
import jamder.Organization;
import jamder.norms.*;
import jamder.roles.AgentRole;

public class [c.name/] extends Norm{
        [if ((c.normContext.organizationClass->size() >
        0) and
             (c.normRestrict.agentRoleClass->size() >
        0))]
        public [c.name/] (String name, NormType
normType, AgentRole       restrictAgentRoleClass,
Organization contextOrganizationClass, NormAction
        action, Hashtable<String, NormConstraint>
normConstraint){
                super(name, normType,
restrictAgentRoleClass, contextOrganizationClass,
        action, normConstraint);
}
        [else][if ((c.normContext.organizationClass-
>size() > 0) and (c.normRestrict.organizationClass-
>size() > 0)) ]
```

Fig. 1. Parte do template da classe Norm.

O template para gerar a entidade ambiente é responsável por criar uma instância de uma classe que herda a classe ambiente de JAMDER 2.0 e analisa, segundo a modelagem, todos os relacionamentos que o ambiente possui, a fim de instanciar em seu construtor as entidades relacionadas na seguinte ordem: organizações, agentes, papéis de agente e por fim, as normas e suas propriedades. Outro template de JAMDER modificado foi o da entidade agente. JAMDER considera cinco tipos de agentes de acordo com suas arquiteturas internas. Sabendo que todos os agentes têm como propriedades comuns estar em um ambiente e executar pelo menos um papel nas organizações de que fazem parte, a classe *jamder.agents.GenericAgent* é a classe que representa as propriedades e demais atributos comuns entre as três ramificações da hierarquia de agentes especificada em JAMDER 2.0. Dessa forma o template especifica os dois casos de herança: (i) quando o agente herda de outro agente; (ii) quando o agente não herda de outro agente, então nesse caso, ele herdará de *GenericAgent*.

V. Estudo de Caso

Com o objetivo de ilustrar a geração de código, este estudo de caso aborda a criação de um SMA normativo responsável pela

gestão de submissões de artigos. Esses sistemas são utilizados para a seleção dos artigos que serão publicados em um evento científico. Para isso, os autores devem submeter seus artigos até uma data determinada, a partir da qual, os avaliadores iniciam o processo de revisão. Após o término do período de revisão, os organizadores devem divulgar os resultados.

No ambiente *Conference Management*, é possível identificar a organização principal *Conference* e o tipo de agente: *user agent*, que pode exercer os papéis *author*, *speaker*, *organizer*, *conference chair*, *website manager* e *reviewer*. Estes papéis são definidos pela organização principal, juntamente com o papel de objeto *submitted*. As instâncias desse papel são exercidas pelas instâncias da classe *Paper*, que possui duas subclasses *ShortPaper* e *FullPaper*. Para o sistema de gestão de submissão de artigos foram definidas as seguintes normas: (i) N1: Os revisores estão proibidos de revisar seus próprios artigos; (ii) N2 (Punição para a violação da N1): Os revisores que violarem N1 devem ter seu papel cancelado.

A. *Geração de código*[1]

Cada entidade do sistema juntamente com as normas foram modeladas utilizando o diagrama de normas da MAS-ML Tool. A partir dessa modelagem segue a geração de código:

1) Normas: possuem uma estrutura simples de classe que contém apenas a chamada ao construtor. Os parâmetros do construtor serão todos instanciados dentro da classe Ambiente. A Figura 2 exibe o código gerado para as normas N1 e N2.

```
import java.util.Hashtable;
import jamder.Organization;
import jamder.norms.*;
import jamder.roles.AgentRole;
public class N1 extends Norm{
                  public N1 ( String name, NormType
normType, AgentRole restrictAgentRoleClass, Organization
contextOrganizationClass, NormAction action,
Hashtable<String, NormConstraint> normConstraint,
Hashtable<String, Norm> sactionPunishment){
                                    super(name, normType,
restrictAgentRoleClass, contextOrganizationClass, action,
normConstraint);
         }}
public class N2 extends Norm{
         public N2 (String name, NormType normType, AgentRole
restrictAgentRoleClass, Organization
contextOrganizationClass, NormAction action,Hashtable<String,
NormConstraint> normConstraint ){
                  super(name, normType,
restrictAgentRoleClass, contextOrganizationClass, action,
normConstraint);
         }}
```

Fig. 2. Código gerado para as normas N1 e N2.

2) Ambiente: É necessário identificar todas as propriedades que compõe cada uma das normas. Essas propriedades serão instanciadas dentro da classe do ambiente antes mesmo de instanciar a entidade Norma.

VI. Conclusão e Trabalhos Futuros

Neste trabalho, descrevemos uma abordagem dirigida a modelos para desenvolver SMAs Normativos através de geração automática de código a partir de modelos, com base em artefatos gerados pela MAS-ML Tool. O processo de geração automática de código é realizado usando *templates Acceleo* que são responsáveis por criar classes em JAMDER 2.0. Uma das vantagens da geração de código é encapsular o processo de transição da fase de concepção para a fase de aplicação, aumentando a produtividade e reduzindo a ocorrência de erros nas atividades de prototipagem. A combinação destas três ferramentas (JAMDER 2.0, NorMAS-ML e *Acceleo*) fornece um desenvolvimento mais fácil para SMAs normativos, aproveitando o uso da plataforma Eclipse.

Como trabalho futuro pode ser abordada a evolução da linguagem NorMAS-ML e do framework JAMDER 2.0 para suportar a modelagem e a geração de código das entidades considerando os elementos dinâmicos das normas.

[1]O estudo de caso está sendo parcialmente apresentado devido à limitação do número de páginas. Entretanto, a sua versão completa pode ser encontrada em: https://sites.google.com/site/uecegessi/estudo-de-caso/jamder-2-0.

References

[1] ACCELEO, "Acceleo OpenSource"; disponível em: http://www.acceleo.org/. Acessado em 15 de janeiro de 2013.

[2] Beydeda, S., Book M., E Gruhn, V. (2005) "Model-driven Software Development." Birkhäuser,

[3] Blois, M., Lucena, C. (2004) "Multi-Agent Systems And The Semantic Web – The SemanticCore Agent-Based Abstraction Layer." In: ICEIS - International Conference on Enterprise Information Systems, 2004, Porto. Proceedings of Sixth International Conference on Enterprise Information Systems ICEIS 2004. Porto: INSTICC, 2004. p. 263-270.

[4] De Maria, B. A. (2004) "Usando a abordagem MDA no desenvolvimento de sistemas multi-agentes. " Dissertação de Mestrado – Pontífica Universidade Católica do Rio de Janeiro.

[5] France, R.; Rumpe, B; (2007) "Model-Driven Development of Complex Software: A Research Roadmap" In: Future of Software Engineering (FOSE'07) co-located with ICSE'07, Minnesota, EUA.

[6] Freire, E. S. S. ; Cortés, M. I. ; Goncalves, E. J. T. ; Lopes, Y. S. (2012) "A Modeling Language for Normative Multi-Agent Systems". In: 13th International Workshop on Agent-Oriented Software Engineering (AOSE@AAMAS), 2012, Valencia (Spain). Proceedings of the 13th International Workshop on Agent-Oriented Software Engineering.

[7] Freire, E. S. S. ; Rocha Jr., R. M. ; Cortés, M. I. (2012) "Um Ambiente de Modelagem para Sistemas Multi-Agente Normativos". In: III Workshop on Autonomous Sotware Systems (Autosoft), 2012, Natal. Proceedings of III Workshop on Autonomous Sotware Systems.

[8] Lopes, Y. S. ; Goncalves, E. J. T. ; Cortés, M. I. ; Freire, E. S. S. (2012) "A MDA Approach Using MAS-ML 2.0 and JAMDER". In: 13th International Workshop on Agent-Oriented Software Engineering (AOSE@AAMAS), 2012, Valencia (Spain). Proceedings of the 13th International Workshop on Agent-Oriented Software Engineering.

[9] OMG. "Object Management Group." Disponível em: http://www.omg.org. Acessado em 15 de janeiro de 2013.

[10] 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), 2013.

[11] Silva, V. T.; Choren, R.; Lucena, C. J. P. (2007) "MAS-ML: A Multi-Agent System Modeling Language." Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA); In: Companion of the 18th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications; Anaheim, CA, USA, ACM Press, pp. 304-305.

[12] Santos, D. R. (2008) "Um Metamodelo para a Representação Interna de Agentes de Software. Dissertação de Mestrado." Porto Alegre: PUC.

[13] TAOM4E; "Tool for Agent Oriented Modeling." Disponível em: http://selab.fbk.eu/taom/

[14] Zambonelli, F.; Jennings, N. R.; Wooldridge, M. J. (2001) "Organisational Rules as an Abstraction for the Analysis and Design of Multi -Agent Systems." In: International Journal of Software Engineering and Knowledge Engineering, Volume 11, Number 3, p. 303-328.

Development of a communication mechanism between Pedagogical Agents in a Virtual Learning Environment

Desenvolvimento de um Mecanismo de Comunicação entre Agentes Pedagógicos em um Ambiente Virtual de Aprendizagem

A Geovane Griesang, Rejane Frozza, Rolf Fredi Molz, Gilberto Dessbesell Jr
Programa de Pós Graduação em Sistemas e Processos Industriais
Universidade de Santa Cruz do Sul (UNISC) – RS – Brasil
{geovanegriesang,frozza,rolf}@unisc.br,
gjunior@mx2.unisc.br

Rafael Peiter
Departamento de Informática
Universidade de Santa Cruz do Sul (UNISC) – RS – Brasil
rafapeiter@gmail.com

***Abstract*—The Intelligent Tutoring System (ITS) developed by a research group linked to the Department of Informatics UNISC (University of Santa Cruz do Sul) was no interaction between the tutor and companion pedagogical agents. Thus, the ITS was used as a basis in developing an interactive mechanism to coordinate communication of agents. For this, a protocol of interaction was modeled based on FIPA foundation. In addition, a facilitator and a variation of the method of roulette been developed to choose the agent that must interact with the student. Interaction scenarios were applied to homologate the engine.**

Keywords—teaching-learning process, intelligent tutoring systems, virtual learning systems, communication mechanism among pedagogical agents.

***Resumo*—O Sistema Tutor Inteligente (STI) desenvolvido por um grupo de pesquisa vinculado ao Departamento de Informática da UNISC (Universidade de Santa Cruz do Sul) não tratava a interação entre os agentes pedagógicos tutor e companheiro. Assim, esse STI foi usado como base no desenvolvimento de um mecanismo de interação para coordenar a comunicação dos agentes. Para isto, um protocolo de interação foi modelado baseado na fundamentação FIPA. Além disso, um agente facilitador e uma variação do método da roleta foram desenvolvidos para eleger o agente que deve interagir com o estudante. Cenários de interação foram aplicados para homologar o mecanismo.**

Palavras-chave— processo de ensino-aprendizagem, sistemas tutores inteligentes, sistemas virtuais de aprendizagem, mecanismo de comunicação entre agentes pedagógicos.

I. INTRODUÇÃO

A utilização de computadores em sala de aula tem contribuído para maior motivação dos estudantes em seu processo de aprendizagem, pois oferece mais diversidade na maneira com que o estudante desenvolve o conhecimento. O uso desses equipamentos na educação permite que novos programas educacionais sejam inseridos nos ambientes de ensino-aprendizagem, condizentes às expectativas dos estudantes atuais [1]. De modo geral, os chamados Sistemas Tutores Inteligentes (STI) podem ser definidos como sistemas educacionais que usam de técnicas de Inteligência Artificial (IA) para auxiliar os estudantes durante o seu processo de aprendizagem [2].

Entretanto, os agentes são considerados pedagógicos quando estão inseridos em sistemas que fazem uso do paradigma de agentes desenvolvidos com a finalidade de educar, possuindo como objetivo fundamental auxiliar os estudantes em seu processo de ensino-aprendizagem [3]. Portanto, os Sistemas Multiagentes (SMA) são *softwares* computacionais com vários agentes interagindo uns com os outros, como em uma sociedade de agentes. Entretanto, cada agente visa satisfazer suas próprias metas para que um objetivo maior e em comum possa ser atingido [4].

Desta forma, o objetivo deste trabalho é desenvolver um mecanismo de comunicação entre agentes pedagógicos de um STI, com a finalidade de permitir a interação entre eles. Para isto, o STI desenvolvido no Departamento de Informática da UNISC (projeto de estudantes e professores) foi utilizado para a validação da proposta. O mecanismo de comunicação procurou atender as necessidades do STI em questão, uma vez que esse sistema está em frequente estudo [5].

O artigo está organizado nas seguintes seções: a seção II aborda o STI usado como base para o desenvolvimento do mecanismo de comunicação; a seção III descreve alguns trabalhos relacionados; a seção IV apresenta as características do mecanismo desenvolvido; a seção V apresenta a conclusão.

II. STI BASE DESENVOLVIDO NA UNISC

Os agentes pedagógicos inseridos no software educacional são os agentes tutor (Dóris) e companheiro (Dimi), onde o agente tutor possui função parecida a de um professor, com capacidade de identificar características relativas à aprendizagem do estudante. Entretanto, o agente companheiro possui a função de atuar no ambiente como parceiro do estudante, ajudando-o nas tarefas propostas pelo STI.

Os agentes pedagógicos contidos no STI base possuem como característica a interação com o estudante, assim como, a comunicação com os demais módulos do Ambiente Virtual de Aprendizagem (AVA). Cada agente pedagógico atua de forma independe no sistema, não havendo interação entre eles. Essa arquitetura possui uma base de conhecimento e os módulos: perceptivo, cognitivo e reativo.

O módulo perceptivo se comunica com os demais, e com os agentes pedagógicos e a base de conhecimento. Ele é o responsável por extrair e armazenar dados referentes à interação do estudante com o STI e monitorar as ações do aluno. O módulo cognitivo executa as inferências sobre a base de conhecimento, sendo possível determinar as ações que devem ser realizadas pelo agente pedagógico, sempre com base nas suas percepções. O módulo reativo deve estabelecer a interface entre os agentes pedagógicos e o estudante, onde são exibidas as mensagens para os estudantes, além de executar as ações indicadas pelo módulo cognitivo [6].

III. TRABALHOS RELACIONADOS

Os trabalhos relacionados contribuíram para a definição das técnicas utilizadas neste trabalho. Moissa, em [7], usou o STI *Eletrotutor*, onde o agente Percepção foi inserido no sistema, assim como, um conjunto de funcionalidades para oferecerem suporte aos estados motivacionais identificados por este novo agente. Objetivo do autor foi fazer com que o agente Percepção monitorasse a comunicação entre a interface e o STI.

Portanto, esse agente Percepção tem finalidade semelhante ao agente Facilitador deste trabalho, pois também monitora os estímulos do ambiente. Além disso, o agente Facilitador participa efetivamente da comunicação entre os agentes pedagógicos do STI. É importante destacar que, o *Eletrotutor* usa linguagem de comunicação e protocolos baseado no padrão KQML, diferentemente deste trabalho. Assim, o *Eletrotutor* também trabalha com troca de mensagens.

O STI usado no presente trabalho pode ser usado para o ensino-aprendizagem de qualquer conteúdo, armazenado na base de conhecimento do sistema. Entretanto, diferentemente deste STI, os STIs usados em [8] e [9] possuem conteúdo específico. Segundo [8], o *STI MathTutor* é usado para auxiliar os estudantes sobre os fundamentos da estrutura da informação para os estudantes de Engenharia de Controle e Automação. O STI usado em [9] é destinado ao ensino de Lógica Matemática.

Os autores do artigo [8] não desenvolveram o STI conhecido como *MathTutor*, apenas descreveram a utilização dos agentes conectivos em um STI e, apresentaram as características do sistema estudado. Com foco no STI base, pode-se perceber que o *MathTutor* se assemelha por usar a arquitetura de troca de mensagens, mas se diferencia pelo fato do *STI MathTutor* usar as padrão KQML para a comunicação.

Segundo [9], o *HALYEN* foi desenvolvido com base na plataforma de desenvolvimento *JADE* que, segue as especificações FIPA [10]. Contudo, este trabalho também se baseia neste padrão, mas o STI base não foi desenvolvido com auxílio de *framework*, como o *JADE*. Uma semelhança entre os trabalhos está relacionada à figura de um agente centralizador, chamado de agente Facilitador no presente trabalho e, agente Coordenador no *STI HALYEN*. Ambos os agentes gerenciam as mensagens recebidas e enviadas pelos demais agentes.

IV. DESENVOLVIMENTO DO MECANISMO DE COMUNICAÇÃO

Inicialmente, foram levantadas as necessidades do STI base (problemas a serem solucionados). Em seguida, foi definida a linguagem de programação, o protocolo de comunicação e o formato das mensagens a serem usadas durante o desenvolvimento do mecanismo de comunicação. Na fase da heurística foi definido o algoritmo usado para determinar o agente que deve iniciar a interação com o estudante. Em seguida, foi realizada a integração de todas as etapas anteriores no STI base. Cenários de interação foram desenvolvidos para validar o mecanismo. Por fim, as decisões dos agentes puderam ser analisadas com auxílio de um questionário respondido por estudantes que usaram o STI.

A. Aspectos de implementação

Os agentes pedagógicos interagem com o estudante sempre que novos estímulos forem gerados no mesmo: pular uma página, voltar para outra página a partir da página de exercícios, permanecer muito tempo ou pouco tempo em uma página. Além disso, os agentes também podem gerar perguntas aleatórias sobre o assunto (aula) tratado no ambiente, caso o estudante fique algum tempo na página de exercícios.

Foram realizados diversos testes para levantar as necessidades do STI em questão, especialmente nos problemas relacionados à comunicação dos agentes pedagógicos. Os erros encontrados estavam diretamente relacionados à comunicação dos agentes com o estudante, pois muitas das mensagens eram apresentadas ao mesmo tempo para os estudantes. Portanto, isto poderia confundir o aluno, já nesses casos as mensagens poderiam ser repetitivas ou diferentes.

Após a análise das necessidades do STI, a linguagem de programação e as ferramentas para o desenvolvimento do mecanismo de comunicação foram escolhidas. Como o STI e os agentes pedagógicos foram implementados em Java, pois o STI foi desenvolvido nesta linguagem. Desta forma, também se definiu o PostgreSQL como o Sistema Gerenciador de Banco de Dados de Objeto Relacional do projeto (SGBDOR).

O protocolo que mais se aproximou das necessidades do STI base foi o FIPA *Contract Net Interaction Protocol* [10]. Assim, o protocolo desenvolvido foi baseado neste protocolo, onde algumas alterações foram realizadas, principalmente pelo fato dos agentes pedagógicos estarem implementados dentro do próprio, onde os objetos dos agentes pedagógicos apenas são instanciados. Ou seja, não foi necessário o uso de arquivos XML para a troca de informação entre os agentes do STI base.

Para a implementação do mecanismo de interação entre os agentes pedagógicos, foi necessário o desenvolvimento de um novo agente, chamado de agente facilitador. Portanto, os agentes pedagógicos não interagem diretamente entre si, essa comunicação sempre é gerenciada pelo agente facilitador. Quando o estudante interage com o STI, este novo agente trata os estímulos gerados no STI e aciona os agentes pedagógicos.

Contudo, esse agente facilitador consulta os agentes pedagógicos para decidir quem deve iniciar uma interação com

o estudante. Essa interação considera as habilidades de cada agente, como por exemplo, apenas a agente tutora Dóris possui a habilidade de fazer perguntas sobre o assunto estudado na disciplina. Assim, os agentes pedagógicos apenas conhecem suas próprias habilidades. Além disso, com o uso do agente facilitador pode-se adicionar novos agente pedagógicos no sistema sem que os agentes já existentes precisem ser alterados.

Tecnicamente, os agentes pedagógicos são instâncias de um objeto agente. Neste momento, apenas são definidas as habilidades desses agentes. Caso a comunicação fosse implementada nos próprios agentes, ambos teriam que conhecer a habilidade do outro agente para a tomada de decisão. A complexidade aumentaria com a inserção de novos agentes no ambiente. Portanto, apenas o agente facilitador precisa conhecer a habilidade dos demais agentes.

Além das habilidades de cada agente pedagógico, o agente facilitador também considera a quantidade de vezes (número de interações) que cada agente pedagógico interagiu com o estudante. Para isto, o agente facilitador usa uma heurística baseada no método da roleta para tomar sua decisão [11]. Então, o agente que interagiu menos vezes como estudante tem uma probabilidade maior de ser eleito para iniciar a iteração. Mas esse método apenas será executado se os agentes possuírem a habilidade de tratar o mesmo estímulo, como por exemplo, um pulo de página.

Por fim, cenários de interação foram elaborados, com o objetivo de validar cada etapa desenvolvida. Por exemplo, um dos cenários visado à execução de diversas tarefas (estímulos) no STI para verificar se apenas um dos agentes pedagógicos iria interagir iniciar a interação com o estudante.

Em seguida, foi adicionada à base de conhecimento do ambiente uma nova aula, referente ao Novo Acordo Ortográfico da Língua Portuguesa. Com isso, o ambiente foi usado por uma turma de Lógica para Computação da UNISC. Com base na análise de questionários respondidos por esses estudantes, pode-se comprovar que eles perceberam a maneira coordenada e não mais simultânea das mensagens enviadas pelos agentes.

V. CONCLUSÃO

A Este trabalho focou no estudo e no desenvolvimento de um mecanismo de comunicação entre agentes pedagógicos de um STI, para permitir a interação coordenada dos agentes pedagógicos com o estudante. Com isso, foi possível evitar a comunicação simultânea dos agentes pedagógicos. Portanto, foi adicionado ao mecanismo de comunicação um protocolo de interação entre agentes. Antes disto, alguns trabalhos relacionados também foram estudados para determinar as técnicas usadas no desenvolvimento do trabalho.

Optou-se pelo desenvolvimento de um protocolo baseado no FIPA [10]. Também foi desenvolvido o agente facilitador que, por sua vez, possui a função de facilitar/gerenciar os processos de comunicação. Assim, o protocolo de comunicação desenvolvido visa à troca de mensagens entre os agentes pedagógicos e o agente facilitador. Esse novo agente visa auxiliar a tomada de decisão, ajudando a determinar qual dos agentes deve interagir com o estudante.

Então, o agente facilitador usa uma heurística baseada no Método da roleta para a tomada de decisão. Esse método consiste em privilegiar o agente que interagiu menos vezes com o estudante, dando mais oportunidades para que esse seja o próximo agente escolhido para iniciar a interação. Entretanto, é importante destacar que o método apenas é executado quando mais de um agente possuir a habilidade de tratar o estímulo gerado e recebido pelo agente facilitador.

A linguagem Java foi usada no desenvolvimento, pois o STI também foi desenvolvido em Java. Assim, o PostgreSQL foi usado para o armazenamento e gerenciamento da base de conhecimento do sistema. Para a validação do mecanismo de comunicação desenvolvido, foram gerados cenários de interação. Por fim, o STI também foi avaliado por estudantes da disciplina de Lógica para Computação da UNISC. Portanto, pode-se concluir que o mecanismo de comunicação atendeu satisfatoriamente as necessidades de comunicação dos agentes pedagógicos com o estudante.

REFERENCES

[1] Cutmore, T. R. H., Hine, T. J., Maberly, K. J., Langford, N. M., Hawgood, G, "Cognitive ad gender factors influencing navigation in virtual environment". In International Journal of Human - Computer Studies, 2000, p. 223-249. .

[2] Guardia, R. B, "Asesores Inteligentes para apoyar el Proceso de Enseñanza de Lenguajes de Programación". Dissertação de mestrado em Ciências da Computação. ITESM (Instituto Tecnológico y de Estudios Superiores de Monterrey), 1997. México.

[3] Gürer, D, "The Use of Distributed Agents in Intelligent Tutoring". In: It's Workshop on Pedagogical Agents, 1998. San Antonio, Texas.

[4] Wooldridge, M, "An Introduction to Multiagent Systems". In Department of Computer Scienceat the University of Liverpool, 2009. Editora: John Wiley & Sons.

[5] Kühleis, R, "CHATTERDÓRIS – Um agente pedagógico com interação em linguagem natural". In Universidade de Santa Cruz do Sul (UNISC), monografia, 2011.

[6] Frozza, R., Silva, A. A. K. Da, Schreiber, J. N. C., Lux, B., Molz, K. W., Kipper, L. M., Borin, M. P., Carvalho, A. B. De, Baierle, J. L., Sampaio, L, "Agentes Pedagógicos Emocionais atuando em um Ambiente Virtual de Aprendizagem". In RENOTE - Revista Novas Tecnologias na Educação, UFRGS, 2011.

[7] Moissa H. E, "Arquitetura de um Agente Identificador de Fatores Motivacionais e Afetivos em um Ambiente de Ensino e Aprendizagem". In Universidade Federal do Rio Grande do Sul (UFRGS), 2001. Porto Alegre/RS.

[8] Frigo, L. B., Pozzebon, E., Bittencourt, G, "O Papel dos Agentes Inteligentes nos Sistemas Tutores Inteligentes". In Anais do WCETE - World Congress on Engineering and Technology Education, 2004. São Paulo/SP.

[9] González, S. M.; Tamariz, A. R.; Carneiro, E. C.; Almeida, J. S. de, "Agentes Inteligentes no Ambiente Virtual de Ensino de Lógica Halyen". In Conferência IADIS Ibero-Americana WWW/Internet 2007, Universidade Candido Mendes, Campos dos Goytacazes, RJ, Brasil.

[10] FIPA "Foundation for Intelligent Physical Agents". http://www.fipa.org, 2013.

[11] Oliveira, J. R. F, "O uso de algoritmos genéticos na decomposição morfológica de operadores invariantes em translação aplicados a imagens digitais". Tese Doutorado em Computação Aplicada - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 1998.

Collection Module Data To Support a Pedagogical Agent Affective

Marcus Rosa
Curso de Ciência da Computação - DINF
Universidade de Santa Cruz do Sul (UNISC)
Santa Cruz do Sul – RS – Brasil
marcus.mecks@gmail.com

Andrea Aparecida Konzen
Departamento de Informática - DINF
Universidade de Santa Cruz do Sul (UNISC)
Santa Cruz do Sul – RS - Brasil

Programa de Pós-Graduação em Informática na Educação
Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre - RS – Brasil
andrea@unisc.br

***Abstract*— This paper describes the development of a module to collect quantitative data in order to assist a computacional pedagogical and affective agent called 'MagaVitta'. This agent is part of the virtual game simulation of cities called 'Città'. The data collected by this module will assist the affective pedagogical agent to understand the current state of the simulation, allowing to infer emotions and display messages that will assist the student in the construction of the city in order to learn how to maintain the ecological balance of the city.**

Keywords—affective agent; affective pedagogical agent; emotions; student

I. INTRODUÇÃO

A Computação Afetiva, ramo da Inteligência Artificial, surgiu em meados da década de 1990 ao se perceber que mesmo um computador dito inteligente, capaz de resolver problemas de maneira lógica, não era, de fato, verdadeiramente inteligente. Desenvolver computadores emocionalmente inteligentes tem sido então um dos objetivos desse novo ramo da Inteligência Artificial uma vez que somos seres que se relacionam afetivamente o tempo todo. Somos seres sociais, inteligentes e também emotivos. Todas essas qualidades são indissociáveis e podem ser aplicadas na nossa relação com os computadores [7].

Dessa forma, a Computação Afetiva acabou, naturalmente, indo ao encontro da informática aplicada à educação, pois ela pode atribuir características afetivas aos agentes pedagógicos computacionais usados em sistemas tutores, jogos educativos e demais ambientes de ensino onde estudantes interagem com avatares virtuais.

Um projeto de criação de cidades virtuais com tecnologias de aprendizado e simulação de uma universidade federal em que o sistema educativo usa essa nova abordagem possui seu próprio agente pedagógico computacional afetivo. Dessa forma, neste trabalho foi desenvolvido um módulo para o agente pedagógico deste ambiente virtual que coleta dados quantitativos das construções inseridas na cidade virtual e das ações do usuário dentro do ambiente. Com esses dados, o agente, que está em constante contato com o usuário, infere emoções, tais como medo, quase sempre relacionado ao perigo de um desastre ecológico, ou alegria quando o usuário faz algo de bom para o meio ambiente.

Este artigo possui as seguintes sessões: (II) conceitos sobre computação afetiva, (III) o agente afetivo Maga Vitta, (IV) o módulo de coleta de dados quantitativos e, (V) conclusões finais.

II. COMPUTAÇÃO AFETIVA

Segundo Berch [2], a ideia de que razão e emoção são coisas distintas tem sido discutida desde Sócrates nos anos de 470 a 399 a.c e perdurou até o final do século XIX. Porém, no ultimo século, novos estudos de diversos pesquisadores (Piaget, Damásio e Le Doux, Vygotsky) acabaram por reescrever essa suposição. A afetividade passou a ser vista como parte do processo cognitivo.

Para Picard [7], a Computação Afetiva é a "computação que está relacionada com, que surge de ou deliberadamente influencia emoções". Assim, se quisermos computadores genuinamente inteligentes, adaptáveis às nossas necessidades e que interajam de forma natural conosco, então estes computadores precisarão de habilidades para reconhecer e expressar emoções, ter emoções ou até mesmo possuir inteligência emocional.

III. O AGENTE MAGA VITTA

Maga Vitta é o nome dado ao agente pedagógico que está inserido no jogo chamado Città [5] e que, por sua vez, faz parte de um projeto maior: de uma universidade federal que está relacionado com criação de cidades virtuais com tecnologias para aprendizado e simulação [4].

Conforme essa definição, o agente Maga Vitta pode ser considerado um companheiro virtual de aprendizado. Esse agente foi inicialmente proposto em [1]. Sua definição é de um agente autônomo ECA (Embodied Conversational Agent) dotado de capacidades afetivas e que auxilia usuários durante o

processo de construção de uma cidade virtual.

As emoções mostradas são baseadas em uma avaliação individual ou na avaliação de um significado cognitivo de um evento conforme descrito dentro do modelo OCC proposto por [6]. Neste modelo são analisadas as ações do usuário ao dispor os elementos da cidade propondo sugestões, questionamentos e informando sobre o processo de construção da cidade.

Em sua arquitetura, o agente Maga Vitta acaba mesclando noções de dois modelos cognitivos: o tradicional modelo OCC [6] que contém o appraisal das principais emoções inferidas pelo agente, e o modelo de [8], que considera a emoção surpresa que não é considerada pelo modelo OCC. Dessa forma, as emoções do agente são: Alegria, Tristeza, Preocupação, Surpresa, Raiva, Nojo e Medo.

Para que o agente Maga Vitta entenda o usuário, o agente constrói um Modelo do Usuário. Este modelo é obtido através da interpretação das ações do aluno durante a construção da cidade virtual. Esse modelo é guardado no perfil do usuário junto com outras características observadas durante a manipulação do ambiente virtual [4]. O Modelador ECA do agente possui a estratégia de sempre capturar o modelo do usuário para poder gerar uma coleção de recursos de fala e de movimentos faciais. O Organizador de Planos de Ação usa o modelo ECA gerado pelo Modelador ECA para organizar a coleção de ações do agente.

IV. Módulo de Coleta de Dados quantitativos

Para que o agente pedagógico possa demonstrar emoções e exibir as mensagens de aviso em resposta às ações do usuário, foi desenvolvido um módulo para coleta dos dados do ambiente virtual, o jogo Città.

Os dados são coletados das variáveis que representam os recursos do jogo: água, alimento, energia, atendimento médico, lixo e esgoto. O módulo a cada nova iteração do jogo, analisa todas as seis variáveis em busca de alterações nos valores desde a última análise. As mudanças de valores são salvas em uma base de dados para fins históricos e posterior consulta.

Dependendo do valor coletado o módulo responde ao agente informando a gravidade dessa variável e também indica qual a emoção apropriada. Com esses dados o agente pode buscar na base de dados a mensagem a ser exibida para o usuário no ambiente virtual.

Os valores das variáveis são atualizados sempre que uma construção é adicionada ou removida do jogo. As construções disponíveis no jogo adicionam valores diferentes para cada recurso, dependendo da quantidade de pessoas que cada uma acomoda. As construções são: casa (4 pessoas), prédio (40 pessoas), escola (40 pessoas), hospital (80 pessoas), igreja (5 pessoas), fábrica (80 pessoas), prefeitura (30 pessoas), mercado (20 pessoas), fazenda (10 pessoas), estação de tratamento de água (5 pessoas), estação de tratamento de esgoto (5 pessoas) e estação eólica (5 pessoas).

Também foi necessário determinar quanto uma pessoa consume de recursos (água, energia, alimento, atendimento médico) e produz detritos (lixo e esgoto). Depois de pesquisas em sites de concessionárias de água e eletricidade e de outros sites de ONGs chegou-se aos valores mostrados na Tabela 1.

TABELA 1. Valores de cada recurso para uma pessoa

Variável	Valor Real
Água	(Consome) 150 litros / dia / habitante
Alimento	(Consome) 2,0 kg / dia / habitante
Energia	(Consome) 3,2 kwa / dia / habitante
Esgoto	(Produz) 90 litros / dia / habitante
Espaço	(Consome) 1 m2
Lixo	(Produz) 20 kg. / dia / habitante

A. *Definição da gravidade de uma situação*

As variáveis do jogo são todas balizadas conforme uma escala de gravidade de 11 níveis, indo de -5 até +5. Essa escala é usada para calcular o quão grave é uma situação, isso é, o quanto um recurso está faltando (valores negativos) ou sobrando (valores positivos) bem como suas intensidades. As intensidades maiores (-5 e +5) representam valores de gravidade extrema para falta e excesso, respectivamente. A medida que o valor tende a zero a gravidade vai amenizando de intensidade até chegar no ponto de equilíbrio, zero. Com isso, quanto mais longe do ponto de equilíbrio, pior é uma situação, seja sobrando ou faltando um recurso do jogo [3].

A Figura 1 mostra a escala usada para inferir a gravidade de uma variável.

Fig. 1. Escala de gravidade da falta ou excesso de um recurso.

Com a escala também é possível inferir emoções, assim como os níveis mais distantes do ponto de equilíbrio representam situações mais graves, eles também representam emoções mais intensas, quase sempre medo, raiva ou tristeza, variando conforme o tipo de variável. Como o agente pedagógico foi desenvolvido para estar preocupado com o meio ambiente, a emoção de alegria é demonstrada no ponto de equilíbrio. A emoção de preocupação é demonstrada nos níveis 1 e 2 de cada lado da escala. Cada intensidade de cada variável diferente possui uma mensagem correspondente que possui o objetivo de alertar o usuário do ambiente virtual sobre a situação que a sua cidade está passando.

A variável água, por exemplo, no ponto de equilíbrio, faz o agente responder com a mensagem "Os moradores da sua cidade agora estão felizes. Não falta água nem para regar as plantas. Parabéns!" demonstrando, junto com a mensagem, a emoção de Alegria. Na gravidade -2, o agente alerta o usuário que há uma falta razoável de água exibindo a mensagem "Uma cidade não pode crescer sem água. Que tal construir algumas estações de água para sua cidade?". Nesse caso, a emoção demonstrada é de Preocupação. Quando a gravidade chega à -5, demonstrando falta grave de água na cidade (em proporção ao número de habitantes), o agente responde com a mensagem

"A situação é gravíssima. A sua cidade não irá crescer mais enquanto não houver água nela. Construa algumas estações coletoras de água o quanto antes!", junto com a emoção Medo.

B. Decisão sobre quando mostrar uma mensagem

As mensagens consideradas padrão exibidas pelo agente são mostradas conforme a resposta do algoritmo responsável por analisar os valores coletados do ambiente virtual.

Para qualquer uma das variáveis definidas no ambiente, o agente verificará essa variável comparando-a com o último valor coletado. Quando houver a mudança de uma variável x no ambiente, ela será comparada com o último valor correspondente gravado no módulo ou com x^1. Se o valor for diferente a variável do ambiente ou x^2, é comparada com zero para saber se o ambiente está equilibrado para o item que aquela variável representa. Se sim, a mensagem de incentivo parabenizando o aluno por ter conquistado o ponto de equilíbrio é exibida. Se a variável não for zero, então existe algum desequilíbrio acontecendo no ambiente. Se o valor x^2 for maior que o valor na base de dados (x^1), então houve uma piora no ambiente, caso contrário, houve uma melhora no ambiente. Nesse ponto, a mensagem correspondente, positiva ou negativa, será escolhida conforme o valor de x, assim o novo valor de gravidade para o recurso em questão é então gravado no módulo.

Este módulo foi finalizado, no entanto, para que o mesmo seja validado é necessário integrar com os outros módulos do jogo já existentes e módulos que ainda estão em processo de desenvolvimento. Assim, poderá ser considerado a base para próximas implementações tanto do jogo quanto do agente pedagógico.

Na medida em que o jogo fica mais complexo, incluindo novas maneiras de causar desequilíbrios ecológicos como fatores de poluição, desmatamentos ou mesmo desastres naturais ou não, o módulo acompanhará esse crescimento porque possui uma estrutura simples, porém robusta e flexível.

V. CONSIDERAÇÕES FINAIS

Conforme Picard [7] a Computação Afetiva une sistemas computacionais à capacidade de influenciar, expressar ou reconhecer emoções. A emoção é uma parte do processo racional, bem como, tem papel fundamental na capacidade de aprendizado de usuários, pois um aluno motivado aprende mais.

O módulo proposto e desenvolvido neste trabalho serve de auxílio ao agente pedagógico Maga Vitta. Sua principal função é de coletar os dados gerados pelo ambiente virtual e, através disso, ser capaz de determinar a melhora ou piora nas condições do ambiente observado. Conforme a evolução do ambiente, emoções são inferidas o que concede ao agente, características afetivas que servem para que ele tenha uma relação mais íntima com o usuário do jogo. Isso melhoraria a experiência do usuário dentro do ambiente e facilitaria a compreensão da preocupação com o meio ambiente.

Embora o desenvolvimento atual do jogo Città não contemple alguns itens importantes vistas, como os marcadores de poluição do ar, da água e da terra e penalidades na produção de recursos devido à poluição, o módulo foi planejado já prevendo esses dois aspectos. Qualquer outra necessidade do jogo, ou até mesmo do agente, quando este for desenvolvido, poderá ser contemplada pelo módulo. Além disso, o módulo também serve de base sólida para o desenvolvimento do agente que passa a contar com os dados coletados para finalmente ser desenvolvido, atuando no ambiente para o qual foi planejado.

Novas implementações podem ser sugeridas partindo do que foi construído nesse trabalho como, por exemplo: coleta de novos tipos de dados quantitativos, onde atualmente, o agente já recebe informações do módulo sobre os dados de água, alimento, energia, atendimento médico, lixo e esgoto. Esses dados também são salvos numa base de dados para posterior consulta. Seria natural, então, poder coletar outros tipos de dados como poluição de terra, água e ar e; novas inferências com base nos dados quantitativos, onde os dados quantitativos salvos na base de dados, hoje, servem apenas como histórico. Seria interessante então usá-los para que o agente aprenda mais sobre a evolução da cidade construída pelo aluno levando em consideração o tempo de vida da cidade.

AGRADECIMENTOS

Agradecemos ao Departamento de Informática da UNISC – Universidade de Santa Cruz do Sul e ao Programa de Pós-Graduação em Informática na Educação (PGIE) da UFRGS – Universidade Federal do Rio Grande do Sul pelo apoio no desenvolvimento da pesquisa e publicação deste artigo.

REFERÊNCIAS

[1] Axt, M. and Longhi, M. T. and Silveira, P. D. and Guimarães, L. N. (2008) "MagaVitta: Conversational Ecological Agent in a Interactive Collective Construction Environment for Basic Education", In: Agent-Based Tutoring Systems by Cognitive and Affective Modeling, Edited by Viccari, R. M. and Jaques, P. A. and Verdin, R. Porto Alegre.

[2] Bercht, M. (2001) "Direção a Agentes Pedagógicos com Dimensões Afetivas". Porto Alegre, Brasil. Academic Press.

[3] Konzen, A. ; Braitback, O.; Kist, L. ; Anjos, A. ; Moraes, L. ; Lima, C. ; Muller, D. ; Axt, M. (2011) "Maga Vitta: agente conversacional aplicado ao jogo educacional Città." In Simpósio Brasileira de Informática na Educação, Aracajú - 22 SBIE/17 WIE.

[4] Longui, M. T. and Nedel, L. P. and Viccari, R. M. and Axt, M. (2004) "Especificação e Interpretação de Gestos Faciais em um Agente Inteligente e Comunicativo". In: SBC Symposium on virtual reality. São Paulo, Brasil.

[5] Müller, D. N.; Oliveira, O. L. B. de; Remião, J. A. A.; Silveira, P. D.; Martins, M. A. R.; Axt, M. (2009) "Virtual Cities as a Collaborative Educational Environment." In Education and Technology for a Better World, pages 112-120, Springer.

[6] Ortony, A. and Gerald, L. C. and Allan C. (1988) "The Cognitive Structure of Emotions." Cambridge, USA.

[7] Picard, R. (1997) "Affective Computing". Cambridge, USA. Publishing Press.

[8] Roseman, I. J. and Spindel, M. S. and Jose, P. E. (1990) "Appraisals of emotion-eliciting events: Testing a theory of discrete emotions". In Journal of Personality and Social Psychology. Vol. 59.

Animated pedagogical agent as learning companion

Jun Hong Silva, Letícia Simioni Couto, Carla A. Barvinski and Valguima V. V. A. Odakura
Faculdade de Ciências Exatas e Tecnologia (FACET)
Universidade Federal da Grande Dourados (UFGD)
Email: {junx.03,leticiascouto}@gmail.com
{carlabarvinski,valguimaodakura}@ufgd.edu.br

***Resumo*—This work presents an Animated Pedagogical Agent (APA) developed to run in a Virtual Learning Environment (VLE) acting as companion student learning, trying to meet the emotional needs that environment. Was applied a survey with undergraduate students in a course in Computer whose results indicate the success of the APA.**

***Resumo*—Neste trabalho é apresentado um Agente Pedagógico Animado (APA) desenvolvido para executar em um Ambiente Virtual de Aprendizagem (AVA) atuando como companheiro de aprendizagem do aluno, tentando suprir as necessidades afetivas nesse ambiente. Foi aplicada uma pesquisa de opinião com alunos de graduação em um curso de Computação cujos resultados apontam o sucesso do APA.**

I. Introduction

Segundo Andrade e Vicari [1] os Ambientes Virtuais de Aprendizagem (AVA) estão longe de apresentar um modelo colaborativo como se deseja atingir. A ênfase, segundo os autores, quase sempre está no indivíduo. O ambiente de aprendizagem computacional também deve propiciar afetividade e motivação na interação com o aluno.

Uma maior percepção da afetividade pelos alunos, pode propiciar uma elevação da autoconfiança, iniciando um processo de motivação intrínseca que os estimula a interagir mais fortemente com o AVA [1]. Contudo, a afetividade tem um papel importante na aprendizagem, e este trabalho apresenta o desenvolvimento de um Agente Pedagógico Animado (APA) em um AVA com o objetivo de ser um companheiro de aprendizagem para o aluno.

O texto está organizado como se segue. Na seção II um APA é apresentado com ênfase na aparência do agente. Os resultados experimentais são detalhados nas seções II-A e II-B. Por fim, na seção III as considerações finais são apresentadas.

II. Agente Pedagógico Animado (APA)

Segundo Gulz e Haake [2], Agentes Pedagógicos Animados (APA) são visualmente representáveis, apresentam características físicas geradas computacionalmente e possuem papéis pedagógicos, adotando uma postura de instrutor virtual, mentor ou companheiro de aprendizagem. Os trabalhos de Gomes, Barbosa e Geyer [3], Baptista [4], Frozza et al [5] e Fontes et al [6] relatam o uso de um APA exercendo o papel de companheiro de aprendizagem. Um companheiro de aprendizagem pode ser caracterizado como alguém de idade próxima ou similar, que compreende o mundo de maneiras semelhantes [7]. Portanto, um APA, é um agente inteligente que tem um papel pedagógico, orientando e melhorando o aprendizado do aluno.

Figura 1. Agente pedagógico animado Carl.

O aspecto mais importante da concepção de um APA é sua aparência. Segundo Baylor [8] a aparência antropomórfica de um agente pode não ter influência no aprendizado mas tem grande impacto na motivação do estudante. Além disso, Baylor e Kim [9] comprovaram que agentes com aparência semelhante aos aprendizes auxiliam no processo motivacional, sendo influentes aspectos como gênero, etnia e idade. Baylor e Kim [9] comprovaram que uma escolha cuidadosa do APA pode influenciar tanto na aprendizagem dos alunos, como também estimular sua capacidade e competência para realizar as atividades.

Considerando que o APA proposto tem papel de companheiro na aprendizagem, e que o público alvo são estudantes universitários da área de Computação, definiu-se que a aparência ideal é a de um jovem e do sexo masculino. Partindo desta definição, foi escolhido o modelo oferecido pela Mixamo[1] representado na Figura 1. O agente expressa diferentes emoções em resposta ao desempenho do aluno em uma atividade proposta.

A definição das ações expressas pelo APA se baseiam nos trabalhos de Frozza et al. e Sansonet et al. [5], [10], os quais utilizaram o modelo de emoções OCC, desenvolvido por Ortony, Clore e Collins, cuja iniciais dos sobrenomes dão nome ao modelo. As ações implementadas no APA para expressar emoções são representadas por movimentos faciais, juntamente com movimentos corporais, conforme descrito na Tabela I.

Para a definição e a criação das ações do agente foi utilizado o aplicativo *Blender*[2], um *software* de código aberto que permite criação, modelagem, animação, texturização e renderização. Posteriormente as animações são carregadas e

[1]Disponível em http://www.mixamo.com/. Acesso em Abril de 2013.
[2]Disponível em http://www.blender.org/. Acesso em Março de 2013.

Expressão	Descrição
Parado	Movimentos leves das mãos, pescoço e tórax, dando a impressão de que o agente está parado.
Olá	Sorriso seguido de um aceno com a mão esquerda.
Aplaudir	Rosto alegre e mãos com movimentos de aplauso.
Adeus	Leve sorriso acompanhado de um aceno de uma das mãos da direita para a esquerda.
Positivo	Movimento do braço direito com polegar para cima, rosto com sorriso.
Decepção	Expressões de tristeza no rosto, com o pescoço ligeiramente inclinado e com as duas mãos sobre o rosto.
Tente novamente	Expressão de tristeza e braços curvados.
Vamos para a próxima	Braço direito com gesto de frente para trás e fechando a mão, com leve movimento da cabeça.

Tabela I. AÇÕES DO APA DESENVOLVIDO.

enviadas para o *Unity3D*[3]. Nesta etapa, é composto e definido o cenário para o agente e a criação de um *javascript* que contém estruturas para a chamada das ações do agente e o processo de comunicação com um AVA. O AVA escolhido para inserir o APA foi a plataforma Moodle[4], devido à ampla utilização neste segmento e por ser *OpenSource*.

O modelo do agente convertido em um *webplayer* através do *Unity3D* é instalado no mesmo servidor em que está o Moodle. O processo de interação entre o Moodle e APA só é realizado após o estabelecimento de comunicação entre *webplayer* e e consultas ao banco de dados através de uma página php criada e inserida no Moodle. São esses dois últimos componentes que acionam as animações, propiciando a interação entre as respostas dadas pelos alunos no questionário do Moodle e o APA. A Figura 2 ilustra a arquitetura de implementação do APA.

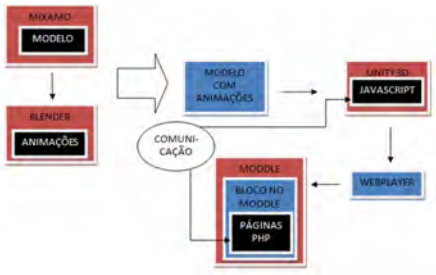

Figura 2. Arquitetura da implementação do APA.

O agente fica disponibilizado em um bloco no Moodle, no lado direito da tela, com dimensões que não causam distrações ao aluno, conforme a Figura 3. A ação do agente depende de cada resposta do aluno, variando de parabenização no caso de acerto até estímulo para estudar mais no caso de erro.

A Figura 3 ilustra o uso do APA em uma intervenção no AVA, em decorrência de resposta errada. Nesta situação, o APA

[3]Disponível em http://unity3d.com/. Acesso em Março de 2013.
[4]Disponível em https://moodle.org/. Acesso em Abril de 2013.

Figura 3. APA Carl no AVA dando *feedback* automático ao aluno.

expressa decepção conforme descrito na Tabela I e apresenta uma mensagem textual de estímulo ao aluno.

O APA apresenta as expressões definidas na Tabeça I. No caso de resposta correta, o APA pode utilizar as ações de aplauso ou positivo, juntamente com mensagem textual de parabenização.

Esse APA foi avaliado por um experimento aplicado em uma turma de alunos do curso de Bacharelado em Sistemas de Informação. O objetivo central da pesquisa de opinião foi avaliar se a presença do APA como companheiro de aprendizagem elevaria a motivação dos alunos para responder um questionário.

Optou-se por dividir a turma em três grupos para avaliar a eficácia dos estímulos, pois haviam, ao nosso ver, três modalidades desses. A primeira e mais rudimentar é o próprio *feedback* fornecido pelo Moodle, a segunda eram as mensagens a serem proferidas para os alunos e a terceira, as mesmas mensagens emitidas pelo APA. Desse modo, avaliamos na pesquisa de opinião esses três tipos de estímulos.

Cada grupo respondeu um mesmo questionário, contudo com diferentes formas de apresentação do *feedback* automático. Ao todo participaram 21 alunos, sendo 7 em cada grupo, divididos de forma aleatória. O experimento foi dividido em duas fases. A primeira fase consistiu na aplicação do questionário eletrônico de assunto relacionado a uma disciplina do curso. Enquanto que na segunda fase foi aplicada pesquisa de opinião visando mapear as percepções de cada grupo.

A. *Fase 1: Aplicação de questionário eletrônico avaliativo*

Cada um dos 3 grupos respondeu ao questionário eletrônico sobre o conteúdo da disciplina de gerência de projetos. O *feedback* do primeiro grupo foi o convencional do Moodle, as mensagens de *feedback* apresentadas ao segundo e terceiro grupo foram as mesmas, o diferencial foi a intervenção do APA como portador da mensagem de acerto ou erro para esse último. As mensagens buscavam estimular o aluno a prosseguir respondendo o questionário quando em situação de adversidade (erro), e a suprir as deficiências de conhecimento estudando mais. Nos demais caso, as mensagens eram de parabenização pelo acerto.

B. *Fase 2: Pesquisa de opinião*

Após a aplicação dos questionários eletrônicos os 3 grupos realizaram a segunda fase, respondendo a uma pesquisa de opinião sobre sua percepção de cada abordagem de *feedback*. Nessa pesquisa buscou-se saber o sexo e a faixa etária dos

participantes, seu conhecimento e utilização de avatares e motivação em responder o questionário. Os dados obtidos foram:

- 86,00% dos alunos está na faixa etária dos 20 a 30 anos, 9,00% tem mais do que 30 anos e 5,00% tem menos que 20 anos.
- O grupo era predominantemente masculino com 86,00% homens e apenas 14,00% mulheres.
- A grande maioria, 95,00% disse que já sabiam o que é um avatar, e 76,00% desses participantes já haviam utilizado anteriormente um avatar na Internet.

O grupo está na faixa etária de 20 a 30 anos, ou seja, são em maioria jovens e do sexo masculino. Além disso, a maioria já havia tido contato com um avatar.

Os alunos foram questionados se eles acharam divertido responder o questionário. Para 42,86% do grupo 1, 71,43% do grupo 2 e 71,43% do grupo 3 a resposta foi **sim**. O resultado aponta a importância do uso de recurso adicional para *feedback*, afinal os índices foram mais elevados para o APA e *feedback* textual do que o modo convencional do Moodle.

Indagou-se aos alunos se haviam se sentido estimulados a responder o questionário. Para: 57,14% do grupo 1, 100% do grupo 2 e 71,43% do grupo 3 a resposta foi **sim**. Confrontando os resultados, constatou-se que a presença do APA e do *feedback* textual estimularam mais do que o *feedback* experimentado pelo grupo 1.

Os grupos 2 e 3 avaliaram o teor das mensagens utilizadas no *feedback* do APA e do recurso textual. Os resultados foram:

- 57,14% dos participantes do grupo 2 e 71,43% dos participantes do grupo 3 consideraram as mensagens motivantes.
- A linguagem utilizada agradou 85,71% dos participantes do grupo 2 e 71,43% dos participantes do grupo 3.

Os dados demonstram que as mensagens utilizadas foram motivantes, contudo o *feedback* textual foi melhor recebido pelos alunos. Percebeu-se que a mesma mensagem quando proferida pelo AVA não teve a mesma eficiência, o que requer maiores estudos quanto à linguagem mais adequada para o agente.

O grupo 3 respondeu questões específicas avaliando características visuais do APA. Os resultados obtidos foram:

- Em relação à aparência do agente, 71,43% dos participantes preferem feições humanas e 28,57% preferem animações.
- Todos os participantes, 100,00% prefeririam interagir com um avatar feminino e não masculino. Desses, 85,71% eram do sexo masculino.
- A maioria dos participantes, 71,43%, preferem realizar um questionário com o acompanhamento de um avatar.
- 85,71% participantes consideraram o avatar simpático.

Os dados reforçam os estudos teóricos que apontam que a feição humana é a aparência ideal para um APA. No que se refere ao gênero, um avatar do sexo masculino não atendeu às expectativas do alunado. Resta saber se a preferência se mantém quando o público que interagir com o APA for do sexo feminino.

A maioria dos alunos já havia utilizado algum tipo de avatar e preferem a sua companhia. Não se pôde definir se essa preferência decorre da empatia que se criou entre aluno e avatar, uma vez que a maioria o considerou simpático, ou de experiências anteriores positivas.

Os dados demonstraram que o APA como companheiro de aprendizagem pode ser eficiente e suprir de alguma forma a lacuna de afetividade que existe em um AVA. Naturalmente, há a necessidade de aprofundamento e aperfeiçoamento da aparência e da abordagem, buscando melhores formas de comunicação.

III. Considerações finais

O objetivo da pesquisa foi alcançado com a criação de um APA de aparência atrativa e motivadora. A pesquisa de opinião demonstrou que o uso de agentes já está disseminado e sua aceitação é boa. A sua aplicação no âmbito pedagógico obteve sucesso, suas intervenções foram consideradas estimulantes, alegres e agradáveis. Os dados também demonstram que é necessária a realização de novas pesquisas que orientem quanto ao conteúdo das mensagens verbalizadas pelo APA.

Referências

[1] A. F. Andrade and R. M. Vicari, "Construindo um ambiente de aprendizagem a distância inspirado na concepção sociointeracionista de vygotsky," in *Educação on-line: teorias, práticas, legislação, formação corporativa.*, M. Silva, Ed. São Paulo: Loyola, 2003.

[2] A. Gulz and M. Haake, "Design of animated pedagogical agents-a look at their look," *Int. J. Hum.-Comput. Stud.*, vol. 64, no. 4, pp. 322–339, Apr. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.ijhcs.2005.08.006

[3] R. Gomes, D. N. Barbosa, and C. F. Geyer, "Lassalinho - um agente pedagógico animado em um ambiente multiagente para educação a distância," *RENOTE*, vol. 3, 2010.

[4] A. C. D. C. BAPTISTA, "Companheiros virtuais em dispositivos móveis: O caso do pequeno mozart." Master's thesis, Universidade do Aveiro, 2010.

[5] A. A. K. d. S. Rejane Frozza, J. N. C. Schreiber, B. Lux, K. W. Molz, L. M. Kipper, M. P. Borin, A. B. de Carvalho, J. L. Baierle, and L. Sampaio, "Agentes pedagógicos emocionais atuando em um ambiente virtual de aprendizagem," *Revista Renote*, vol. 9, 2011.

[6] L. M. de Oliveira Fontes, F. M. M. Neto, F. A. Diniz, D. G. Carlos, L. J. Júnior, and L. C. N. da Silva, "Um agente pedagógico animado de apoio à aprendizagem baseada em problema," *IEEE-RITA*, vol. 7, no. 4, pp. 182–188, Nov. 2012.

[7] K. Ryokai, C. Vaucelle, and J. Cassell, "Virtual peers as partners in storytelling and literacy learning," *J. Comp. Assisted Learning*, pp. 195–208, 2003.

[8] A. L. Baylor, "The design of motivational agents and avatars," *Educational Technology Research & Development*, 2011.

[9] A. Baylor and Y. Kim, "Simulating instructional roles through pedagogical agents," *International Journal of Artificial Intelligence in Education*, vol. 15, pp. 95–115, 2005.

[10] J. P. Sansonnet, D. W. Correa, P. Jaques, A. Braffort, and C. Verrecchia, "Developing web fully-integrated conversational assistant agents," in *Proceedings of the 2012 ACM Research in Applied Computation Symposium*, ser. RACS '12. New York, NY, USA: ACM, 2012, pp. 14–19. [Online]. Available: http://doi.acm.org/10.1145/2401603.2401607

Dynamic Modeling of Multi-Agent Systems Using MAS-ML Tool

Francisco R. O. de Lima, Állan R. Feijó, Robert M. Rocha Jr, Igor B. Nogueira, Enyo J. T. Gonçalves, Emmanuel S. S. Freire, Mariela I. Cortés
Grupo de Engenharia e Sistemas Inteligentes (GESSI)
Departamento de Computação – Universidade Estadual do Ceará (UECE)
Fortaleza, Brasil
{us.robson7, allanfeijo1987, robstermarinho, igor.bnog, savio.essf}@gmail.com, enyo@ufc.br, mariela@larces.uece.br

***Abstract* — Given the diversity of entities comprising multi-agent systems (MAS), the modeling of the dynamic aspects is complex and error prone. Thus, the existence of a tool capable of modeling of these systems and validating them automatically, can be crucial in order to increases the productivity. The goal of this work is present the evolution of MAS-ML tool to provide the support to the dynamic diagrams of sequence and activities defined on the MAS-ML 2.0 language.**

Keywords — multi-agent system; MAS-ML Tool; dynamic modeling;

I. INTRODUÇÃO

Em um cenário cada vez mais complexo, sistemas multi-gente (SMA) vem sendo cada vez mais utilizados para lidar com essa complexidade, tanto na indústria quanto na academia. O termo Sistema Multi-Agente (SMA) refere-se à subárea de Inteligência Artificial que investiga o comportamento de um conjunto de agentes autônomos, objetivando a solução de um problema que está além da capacidade de um único agente [1].

Neste contexto, várias linguagens de modelagem, frameworks de implementação e ferramentas foram criados para auxiliar o desenvolvimento de SMAs. Dentre eles, a linguagem MAS-ML (Multi-Agent System Modeling Language) [2] é capaz de modelar SMAs através dos diagramas estáticos de classe, papéis, organização, e dinâmicos de sequência e atividades [3]. A linguagem MAS-ML 2.0 [4] trata-se de uma extensão para permitir a modelagem de agentes com diversas arquiteturas internas [1], e seus papéis. Na sua versão atual, a ferramenta de suporte a modelagem MAS-ML tool [4] [5] não contempla nenhum diagrama dinâmico previsto na linguagem de modelagem MAS-ML 2.0, impossibilitando a modelagem do comportamento do sistema em tempo de execução. O presente artigo apresenta a evolução da ferramenta MAS-ML tool relacionada ao desenvolvimento dos diagramas dinâmicos de MAS-ML 2.0. O artigo é organizado como segue: Na Seção 2 é apresentado o referencial teórico. Na Seção 3, a evolução da ferramenta é apresentada. Na Seção 4, um estudo de caso é ilustrado. Na Seção 5, os trabalhos relacionados são comparados com as contribuições deste artigo. E por fim, na Seção 6, são apresentados as conclusões e os trabalhos futuros.

II. REFERENCIAL TEÓRICO

A. *MAS-ML 2.0*

MAS-ML [2] é uma linguagem de modelagem que estende a UML [6] e incorpora o conceito de agente definido no framework conceitual TAO (Taming Agents and Objects) [7] para a modelagem de SMAs. Originalmente, MAS-ML foi projetada para modelar apenas agentes pró-ativos orientados a objetivos e guiados por planos. Para possibilitar a modelagem das diversas arquiteturas de agente definidas por Russell e Norvig [1], Gonçalves [4] evoluiu a linguagem de maneira conservativa originando MAS-ML 2.0.

MAS-ML 2.0 contempla um conjunto de diagramas estáticos e dinâmicos. O diagrama de sequência de MAS-ML 2.0 consegue ilustrar diversas capacidades do SMA. Através de pathnames, podemos representar agentes mudando de um ambiente, organização, ou papel, e com uso dos estereótipos da linguagem podemos ilustrar a criação, a destruição e a interação entre as entidades. Por outro lado, o diagrama de atividades modela um fluxo de execução através de uma sequência de unidades subordinadas chamadas de ação. Na versão 2.0 de MAS-ML a representação de agentes reativos, baseados em objetivo com planejamento e baseados em utilidade foi proposta.

B. *MAS-ML tool*

A ferramenta MAS-ML tool [8] [5] é um ambiente de modelagem desenvolvido como um plug-in da plataforma Eclipse [9]. MAS-ML tool foi criada para dar suporte à modelagem dos diagramas contemplados na linguagem MAS-ML original, e na sua versão atual, a ferramenta fornece apoio para a construção dos diagramas de classe, organização, e papeis de acordo com MAS-ML 2.0. Visto que modelar SMAs sem o apoio de uma ferramenta torna o trabalho difícil de ser realizado e, podendo até certo ponto, ser considerado impraticável, é fundamental que ferramentas sejam propostas para o uso eficiente da linguagem.

MAS-ML tool foi gerada a partir dos plug-ins GMF (Graphical Modeling Framework) [10] e EuGENia [11]. O GMF é um framework para desenvolvimento de editores gráficos para modelos de domínio. Por outro lado, o EuGENia é capaz de automatizar os procedimentos necessários para o desenvolvimento de diagramas utilizando o GMF, cuja

abordagem é dirigida por modelos utilizando o próprio metamodelo da linguagem MAS-ML 2.0.

III. Evolução da Ferramenta

A estratégia adotada para implementar as extensões propostas segue a abordagem dirigida por modelos, utilizada originalmente para desenvolver a própria ferramenta. Neste caso é utilizado como modelo central o metamodelo da linguagem MAS-ML 2.0.

A. Criação do Diagrama de Sequência

O processo de criação do diagrama de sequência para MAS-ML tool se deu primeiramente, transcrevendo todos os elementos presentes no metamodelo da linguagem MAS-ML 2.0 para a linguagem Emfatic. Com a transcrição desses elementos, podemos identificar os elementos presentes no diagrama de sequência, e associar a eles suas respectivas representações gráficas.

A modelagem do diagrama de sequência em MAS-ML tool prevê adicionalmente a checagem do modelo gerado de forma a verificar a sua corretude em relação à definição no metamodelo da linguagem. Desta forma, foi definido na ferramenta um conjunto de regras de validação implementadas na linguagem OCL (Object Constraint Language) [11], descritas na Tabela 1.

TABELA 1 REGRAS DE VALIDAÇÃO DOS MODELOS (DIAGRAMA DE SEQUÊNCIA).

Regra	Propósito e Definição em OCL	
Regra 1	Todos os elementos do modelo devem ter um nome.	name.size() > 0
Regra 2	Se o agente possui plano, então ele possui ação.	self.ownedPlan->isEmpty() = false implies self.owendAction->isEmpty() = false and self.ownedPlan->isEmpty() = false
Regra 3	Se o agente possui plano, então não possui percepção.	self.ownedPlan->isEmpty() = false implies self.ownedPerception->isEmpty() = true and self.ownedPlan->isEmpty() = false
Regra 4	Se o agente possui planejamento, então ele possui percepção e ação.	self.ownedPlanning->isEmpty() = false implies (self.ownedPerception->isEmpty() = false and self.owendAction->isEmpty() = false) and self.ownedPlanning->isEmpty() = false
Regra 5	Se o agente possui plano, então ele não possui planejamento.	self.ownedPlan->isEmpty() = false implies self.ownedPlanning->isEmpty() = true and self.ownedPlan->isEmpty() = false
Regra 6	Caso o agente possua planejamento, então ele não terá plano.	self.ownedPlanning->isEmpty() = false implies self.ownedPlan->isEmpty() = true and self.ownedPlanning->isEmpty() = false

B. Ferramenta Gerada

O ambiente implementado trata-se de um plug-in da plataforma Eclipse, como mencionado anteriormente. Isso permite utilizar os recursos oferecidos pela plataforma de forma concomitante com a modelagem de SMAs. Dado que muitas plataformas de agentes são implementadas em Java, tais como JADE [13], Jadex [14], Jason [15]; o uso da plataforma Eclipse favorece uma possível geração de código dentro do mesmo ambiente de desenvolvimento. O projeto da ferramenta MAS-ML tool, juntamente com os plug-ins gerados encontram-se disponíveis em https://sites.google.com/site/uecegessi/masmltool.

IV. Estudo de Caso

O ambiente de aprendizagem Moodle [16] é usado por instituições de ensino como um ambiente de aprendizagem colaborativa, facilitando a comunicação entre professor e aluno, estimulando a troca de informações e compartilhando recursos via internet. Os diagramas de sequência e atividades foram criados para cada um dos seis agentes descritos, porém, devido à questão de espaço, vamos ilustrar neste trabalho apenas os diagramas para o AgenteBuscadorDeInformacoes. A Figura 1 ilustra o diagrama de sequência para o AgenteBuscadorDeInformacoes, que foi modelado como um agente baseado em objetivo e guiado por plano.

Este agente conta com dois planos pré-definidos: i) buscarInformacoesPessoas, que tem as ações de localizarPessoas, relacionarPessoas e exibirPessoasRelacionadas que são executadas em sequência. Este plano visa relacionar pessoas no contexto do Moodle para que as mesmas possam interagir entre si; e ii) buscarInformacoesDocumentos com as ações buscarDocumentos, relacionarDocumentos e exibirDocumentosRelacionados. Este plano visa encontrar documentos relacionados às pessoas e exibi-los.

Fig. 1. Diagrama de sequência para o AgenteBuscadorDeInformacoes

A Figura 2 ilustra o diagrama de atividades para o plano buscar informações pessoas do AgenteBuscadorDeInformacoes descrito no parágrafo anterior. Os demais agentes podem ser encontrados em https://sites.google.com/site/uecegessi/masmltool/modelagemdiagramasdinamicosmoodle.

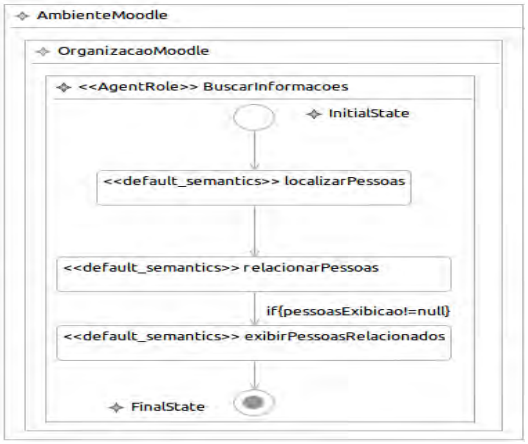

Fig. 2. Diagrama de Atividade para o plano buscar informações do AgenteBuscadorDeInformacoes

V. TRABALHOS RELACIONADOS

Ferramentas de modelagem são normalmente projetadas com foco no suporte a uma linguagem de modelagem específica, propagando suas vantagens e desvantagens para as ferramentas que as implementam. Tanto a linguagem AUML [17] quanto Anote [18] descrevem adequadamente papéis e suas propriedades. Com isso, as respectivas ferramentas de suporte não são capazes de modelar tais entidades.

MAS-ML [2] possui duas ferramentas de modelagem para SMAs. O VisualAgent [19] é baseado no metamodelo original da MAS-ML, e consequentemente, o suporte à modelagem de agentes com diferentes arquiteturas internas é limitado. Adicionalmente, apenas os diagramas estáticos propostos na linguagem foram contemplados. Por outro lado, MAS-ML tool [4] [5] é um ambiente de modelagem específico de domínio que atende à modelagem de sistemas multi-agente por meio da linguagem de modelagem MAS-ML 2.0 [4] e contempla os diagramas de classes e organização e papel de acordo com a versão 2.0 de MAS-ML. Duas vantagens de MAS-ML tool em relação à VisualAgent podem ser notadas: o fato de ter sido desenvolvida como um plug-in da plataforma Eclipse [9], e a capacidade de realizar verificação do modelo em relação ao metamodelo da MAS-ML. Adicionalmente, a nova versão de MAS-ML tool possibilita a modelagem de todos os diagramas estáticos e dinâmicos em conformidade com MAS-ML 2.0. Entretanto, as demais ferramentas citadas nessa seção não são capazes de modelar todos os aspectos estáticos e dinâmicos dos SMAs.

VI. CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho foi apresentada a evolução da ferramenta MAS-ML tool para o suporte à modelagem dos diagramas de sequência e atividades definidos em MAS-ML 2.0. Com isso, tanto os diagramas estáticos como os dinâmicos previstos na linguagem podem ser gerados através da ferramenta. Adicionalmente, a extensão proposta prevê a validação da boa formação dos diagramas gerados reduzindo falhas e tornando a modelagem mais coerente com as definições feitas em MAS-ML 2.0. Como trabalhos futuros relacionados com a evolução da ferramenta podem ser citados: (i) geração de código a partir dos diagramas construídos na ferramenta MAS-ML tool e (ii) realização de um estudo utilizando computação experimental com um público alvo formado por analistas/projetistas de sistemas multi-agente.

REFERÊNCIAS

[1] RUSSELL, S.; NORVIG, P. Inteligência artificial: uma abordagem moderna, 2ª Ed. Prentice-Hall: São Paulo, 2004.

[2] SILVA, V. T. Uma linguagem de modelagem para sistemas multi-agente baseada em um framework conceitual para agentes e objetos, Tese de doutorado. Rio de Janeiro: PUC, Departamento de Informática, 2004.

[3] SILVA, V. T.; CHOREN, R.; LUCENA, C. J. P. MAS-ML: A Multi-Agent System Modeling Language, In: Conference on Object-oriented programming, systems, languages, and applications, 18th annual ACM SIGPLAN; USA, ACM Press, 2007.

[4] GONÇALVES, E. J. T. (2009). Modelagem de Arquiteturas Internas de Agentes de Software Utilizando a Linguagem MAS-ML 2.0. Dissertação de Mestrado. Fortaleza: UECE, Centro de Ciência e Tecnologia.

[5] GONÇALVES, E. J. T.; OLIVEIRA, K. S. F.; CORTÉS, M. I.; FEIJÓ, A. R.; OLIVEIRA, F. R.; SILVA, V. T. MAS-ML TOOL: A Modeling Environment for Multi-Agent Systems. In: Proceedings of 13th International Conference on Enterprise Information Systems, Beijing 2011.

[6] UML, Unified Modeling Language Specification, versão 2.2, disponível em:<http://www.uml.org>, acessado em 2 de Junho de 2011.

[7] SILVA, V.; Garcia, A.; Brandao, A.; Chavez, C.; Lucena, C.; Alencar, P. (2003). Taming Agents and Objects in Software Engineering. In: Garcia, A.; Lucena, C.; Zamboneli, F.; Omicini, A; Castro, J. (Eds.), Software Engineering for Large-Scale Multi-Agent Systems, Springer-Verlag, LNCS 2603, pp. 1-26, 2003.

[8] FARIAS, K.; NUNES, I.; SILVA, V. T.; LUCENA, C. J. P. MAS-ML Tool: Um ambiente de modelagem de sistemas multi-agente, Fifth Workshop on Software Engineering for Agent-oriented Systems (SEAS@SBES 09), Brazil, 2009.

[9] ECLIPSE, Eclipse Platform, disponível em:<http:// www.eclipse.org>, acessado em 2 de Junho de 2011.

[10] GMF, disponível em:<http://www.eclipse.org/modeling/gmf/>, acessado em Junho de 2011.

[11] EuGENia disponível em:<http://www.eclipse.org/epsilon//>, acessado em Junho de 2011.

[12] OCL,disponível em :<http://www.eclipse.org/modeling/mdt/?project=ocl>, acessado em 20 de Junho de 2011.

[13] BELLIFEMINE, F. L.; CAIRE, G.; GREENWOOD, D. (2007). Developing Multi-Agent Systems with JADE. [S.l.]: Wiley (Wiley Series in Agent Technology).

[14] POKAHR, A.; BRAUBACH, L.; LAMERSDORF, W. (2003). Jadex: Implementing a BDI-Infrastructure for JADE Agents. EXP - In Search of Innovation (Special Issue on JADE), vol. 3, no. 3 , Telecom Italia Lab, Turin, Italy, S. 76-85.

[15] BORDINI, R. H.; WOOLDRIDGE, M.; HÜBNER, J. F. (2007). Programming Multi-Agent Systems in AgentSpeak using Jason, John Wiley & Sons.

[16] MOODLE, disponível em:<http://www.moodle.org.br>, acessado em Junho de 2011.

[17] ODELL, J.; PARUNAK, H. V. D.; BAUER, B. (2000). Extending UML for Agents. Proc. Of the Agent-Oriented. Information Systems Workshop (AOIS'00) at the 17th National Conference on Artificial Intelligence (AIII'00) (3-17).

[18] CHOREN, R., LUCENA, C. Agent-Oriented Modeling Using ANote, 3rd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems, 3rd; The Institution of Electrical Engineers, IEE, Stevenage, UK, 2004, pp. 74-80, 2004

[19] DE MARIA, B. A.; SILVA, V. T.; LUCENA, C. J. P.; CHOREN, R. VisualAgent: A software development environment for multi-agent systems, Proceedings of the 19º Simpósio Brasileiro de Engenharia de Software, Tool Track, Brazil, 2005.

Two Different Perspectives about How to Specify and Implement Multiagent Systems

André Mendes da Rosa, Alexander Gularte, Eder Mateus Nunes Gonçalves, Mateus Jung
Universidade Federal do Rio Grande - FURG
Rio Grande-RS, Brazil 96203-900
Email:seteeng@gmail.com, alexgularte@gmail.com, edergoncalves@furg.br, mateus_jung@furg.br

***Abstract*—This article review has as a goal to demonstrate that exists two different perspectives considering the actual literature about frameworks to specify and implement multi-agent systems in a formal way. On the one hand, there are those specific frameworks/methodologies for MAS where those obligatory requirements to guarantee the system correctness are encapsulated on the tool. On the other hand, there are those frameworks/methodologies that are based on those existing one and they are adapted to the multi-agent specificities, where those mechanisms to verify and validate the system are inherited from the original method/tool. On this paper are presented two methodologies based on the first perspective, considering three different dimensions on specifying MAS, and two adapted tool, Petri Nets and AUML, considering the second perspective.**

I. Introduction

Modern computational problems are inherently distributed. In these cases, a solution can be obtained through some kind of composition between parts dispersed in a real or virtual environment. A example like these is the recomposition of an electrical network after a blackout, and the control of a team of robots that play soccer. Problems like these share some pattern features [1]: they are physically and/or conceptually distributed, in the sense that their global state is composed by the aggregation of partially independent local states; and the tasks involved in solving these problems refer to different levels of abstraction, varying from global coordination protocols to local perception/action procedures, that use sensors to perceive the world state and effectors to act in the world. The relevance of these problems can be measured through the number of new methods/techniques or even new knowledge areas developed to treat them. It can be cited Pervasive and Ubiquitous Computing, Smart Grids and Multiagent Systems. All these issues has its fundamental knowledge based on the distributed systems theory.

One main reason for this situation is the absence of some pattern method/technique to develop this kind of solution. In this sense, it is necessary to establish some benchmarks about formal methods of specification to MAS. According to this study, it can be said about MAS:

- a MAS can be conceived from three dimensions: the agent itself, the communication/interaction aspects (environment and agents) and the organization model;
- The existing formal methods can be classified on two categories: those methods that was develop specifically as a multi-agent framework and has its own tools to verify and validate the system based on some kind of logic proof.

From this perspective, this paper has as a goal to define the minimal constraints about a formal method to specify, develop and implement MAS.

II. Requirements for a Multiagent Specification

A specification is formal if it is expressed in a language composed of the following three elements: rules for determining the correct formation of sentences (*syntax*); rules for interpreting sentences in a precise and meaningful regarding the considered domain (*semantics*), and rules to infer useful information from specifications (the *proof theory*). In a broad context, can be identified some metrics for evaluation of formalisms:

- *Expressiveness and required coding* - Expressiveness relates to the ability of the model to express formal aspects present in the real system;
- *Constructability, management and evolution* - The constructability is a ability to adapt to a system with modularized and incremental development processes;
- *Usability* - The usability concerns the ease with which the specification is performed;
- *Communicability* - Along the same idea of the previous criterion, the communicability allows well-trained people read and verify high quality specifications.

A multi-agent formalism should take into account three basic steps in their specification: requirements, design and implementation. In order to go from one step to another are necessary rules or propositions, i.e., a logical-mathematical formalism that enables the correlations the three stages.

III. Logic-based Formalism for MAS

In a MAS, there are three dimensions to consider:

- the *individual agents*, where the agent is able to sense changes in the environment, act according to its goals causing changes in the environment, communicate to coordinate your actions with other agents;
- the *communication and interaction* between agents, in other words, this dimension can be understood as protocols regulating the interactions between agents, enabling agents to use the functionality of others or allowing it access to external resources;
- the *social organization* carry out the agents representation such individuals inside a group organized

by concepts like roles, groups, norms and global and individual plans/missions.

Each of the two methods/formalisms/frameworks presented below represents some of these three dimensions.

A. Model of Organization for multI-agent SystEms+

The organizational specification of a MAS is useful to improve the efficiency of the system since the organization constrains the agents behaviors towards those that are socially intended: their global common purpose [2].

The MOISE+ has an explicit global plan and little dependency between the structure and functioning. The objective is an organization centered model where the first two dimensions can be specified almost independently of each other and after properly linked by the deontic dimension.

The organizational models that follow the organizational centered point of view usually are composed by two core notions: an Organizational Specification and an Organizational Entity. An Organization Entity is a population of agents functioning under an Organization Specification [3]. An Organization Entity is then created as the agents adopt the roles specified in the organization Specification.

A MOISE+ Organizational Specification is formed by a Structural Specification, a Functional Specification, and a Deontic Specification [3]. The three organizational dimensions of MOISE+ [4]:

- Structural Dimension (roles, groups, relations): A role is conceived as a set of behavioral constraints that an agent accepts since it joins a group in the organization;
- Functional Dimension (goals, global plans, missions): It defines a set of global plans for the MAS, which are structured in a social schema, as a goal decomposition tree, where each goal may be decomposed in sub-goals, and the responsibilities for the sub-goals are distributed in missions;
- Deontic Dimension (obligations, permissions): It specifies the relations between the structural specification and the functional specification, establishing which missions each role is obliged or has the permission to realize.

Through the levels shown, note that the MOISE+ may represent a real organization, showing a good degree of expressiveness. It has good evolution because the concepts of missions, set of plans and goals where all this concepts are assembled in a Social Scheme, thus, also has good usability, mainly by tree decomposition distributing the responsabilities in missions.

B. Social Commitments

Most agent communication languages are no longer defined in terms of the agents' mental attitudes, but in terms of social commitments [5]. However, commitments has not a clear and unequivocal character, and are not completely unrelated to the agent's reasoning, but this situation can be remedied through the combination of logic BDI with a logic of what is publicly grounded between agents.

By means of a reducionist logical characterization of social commitments, and due that individual mental attitudes are not enough to characterize social commitments, it should be combined a logic of individual mental attitudes with a logic accounting for the social and public feature of social commitments. Using the logic of grounding which extends a BDI-like logic by a modal operator of what is publicly established in a group of agents, as opposed to private mental attitudes.

Castelfranchi reduces social commitment of the debtor i to the creditor j w.r.t. the action α using mutual knowledge: "i and j mutually know that i intends to do α and this is j's goal, and that as for α j has specific rights on i (j is entitled by i to α)" [6] [5]. Replacing mutual knowledge with the notion of grounding, captures only the public feature of the i's intention, and also does not imply that this attitude holds.

Due to the combination of the BDI logic with a logic of what is public grounded between agents, the expressiveness is good, but is affected by the notion that commitment does not have a clear and unambiguous characterization. The constructability can be achieved through the theory of speech acts, formalizing commitments not only as effects of speech acts, but speech acts creating and managing commitments. The specification for presenting the logic grounding, modal operators and other special features, such as propositional social commitments still has low usability. In the communication part, it can be said that social commitments are more mature, however, must be well understood by well-trained people.

IV. Inherited Formalism for Multiagent Systems

This section describes two formalisms inherited from classic models of system specification: UML and Petri Nets.

A. Agent Unified Modeling Language (AUML)

MAS are often characterized as an extension of object-oriented systems, but unlike objects, agents are autonomous and interactive. Agents based on their internal states, its activities include goals and conditions that guide the execution of defined tasks. While objects require external control to execute its methods, agents know the conditions and the effects of their actions.

Participants of the FIPA Modeling Technical Committee and OMG-AUML Agent Work Group initially identified two areas for development of detailed specifications. These specifications are as follows: Class Diagrams - specify the internal behavior of an agent and relating it to the external behavior of an agent using and extending UML class diagrams; Interaction Diagrams - a generic term that applies to several types of diagrams that emphasize object interactions. These include collaboration diagrams, sequence diagrams, and the overview diagram of interaction.

According to FIPA Modeling Technical Committee, the areas of AUML Modeling are [7]: *Multiagent vs. single agent, Goal and soft goals, Social aspects, Environment, Workflow/Planning, Levels of abstraction, Temporal constraints, and Deployment and Mobility*.

We note that the AUML with its graphical notation, their extensions and adaptations is able to express and model the

various MAS, leaving the designer of such systems better able to lift requirements, design, build and implement, namely, has usability. And, through the various UML notations adapted one can have an overview of the system, i.e. the AUML shows a good degree of expressiveness, and has high constructability and it can be, in most situations, codified in a way more agile.

B. Petri Nets based Formalism for Multiagent Systems

The specification language is based on PN for structuring knowledge in various abstract levels and also provides generic mechanisms for use of several types of knowledge representation formalisms.

This model assumes that the agent, based on their mental model of the world, establishing priorities and setting goals for your performance environment, and to establish these goals, the agent has the job of identifying the best sequence and coordination of actions to achieve them.

The planning is directly linked to socialization, namely the role of the agent seeks to achieve the aims of agents society of which he is part. This model is modular, and from the planning module are defines the current goals of the agent that are passed on to the coordination module that selects the necessary actions to the module action can act on the environment.

The individual knowledge and the role that the agent has in society defines its personal strategy and together with the collective strategy of the society of agents, the agent performs their individual actions on behalf of social goal.

[1] proposed an approach to specify individual and social levels through the same formalism. This formalism is based on a specific model of High Level PN developed to interface between experienced professionals in the domain to be modeled and frameworks used to implement the system. Moreover, the proposed PN allows you to create and verify formally the mapping between individual and social levels through a hierarchical formalism that integrates the knowledge of the entire system.

The proposed model presents important aspects in the process of acquiring knowledge:

- The graphical representation allows minimizing communication problems between knowledge engineers and professionals in the area concerned. The model allows specifying concurrent tasks, as well as individual and social contexts;
- The mathematical model of PN can be used to check for problems such as inconsistencies, ambiguities and redundancies;
- It is possible to automatically transform the information in a knowledge base.

The use of PN is justified because it is a specification tool ideal for systems that require specifying competition and timing. In addition, knowledge based systems, as is the case with MAS, can be viewed as discrete event systems because changes state, or the occurrences of new events are driven by time. Aiming to represent a knowledge-based system using PN, it becomes necessary to extend the capability of representativeness tokens allowing manipulations represent the knowledge base when a rule is triggered. To meet this purpose, High Level PN are appropriate. This type of network allows associating preconditions and post-conditions that control the loading and firing of transitions. The transition firing entails a change in the knowledge base, which is updated by the manipulation of the chips. The distribution of tokens represents the state of the knowledge base.

V. Conclusion

In this paper is argued that the formalism to specify, implement and validate MAS can be classified on those that are specific for this kind of paradigm and those that are inherited from other methods. In this sense, four different methods were analysed considering some basic metrics for this kind of formalization.

The difference between AUML and the representation by PN seems to revolve around the adaptability of AUML, which can be shaped by the designer of the MAS in the way most suitable to represent the MAS in question, beyond AUML have a friendly graphical view system that makes coding more agile and simple, without forgetting, of course, that AUML can represent a wider variety of MAS.

Visually, PN and Moise are similar. Both transmit on its behalf, a great knowledge about the system as a whole. But Moise is very limited to such representation, once it only models the system, defining the rules of operation, structure and organization of MAS. PN go beyond, allowing the development of execution control, the system working at the individual level and global.

The main fact concluded from this work is the totally absence of frameworks/methodologies and even languages which encompass all dimensions of a MAS.

References

[1] E. M. N. Gonçalves, "Specifying knowledge in cognitive multiagent systems using a class of hierarchical petri nets," *Journal of Software*, vol. 7, no. 11, pp. 2405 – 2414, 2012, special Issue: Data and Knowledge Engineering in Open Social Network.

[2] L. Gasser, "Organizations in multi-agent systems," *Pre-Proceeding of the 10th European Worshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW'2001), Annecy*, 2001.

[3] J. F. Hübner, J. S. Sichman, and O. Boissier, "A model for the structural, functional, and deontic specification of organizations in multiagent systems," *Advances in Artificial Intelligence*, pp. 439–448, 2002.

[4] A. Hübner, G. Dimuro, A. Costa, and V. Mattos, "A dialogic dimension for the moise+ organizational model," in *Proceedings of the Workshop on LAnguages, methodologies and Development tools for multi-agent systemS (LADS 2010) at The Multi-Agent Logics, Languages, and Organisations Federated Workshops (MALLOW 2010), Lyon*, 2010.

[5] B. Gaudou, A. Herzig, D. Longin, and H. Noi, "Logical formalization of social commitments: Application to agent communication languages," in *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems. Budapest: International Foundation for Autonomous Agents and Multiagent Systems*. Citeseer, 2009, pp. 1293–1294.

[6] C. Castelfranchi, "Commitments: From individual intentions to groups and organizations," in *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, 1995, pp. 41–48.

[7] J. Odell, H. V. D. Parunak, and B. Bauer, "Extending uml for agents," in *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence*, 2000.

Multiagent Systems in Travel Planning

Diego Fialho Rodrigues*, Heber Amaral*, Alcione de Paiva Oliveira* and Simone Dornelas Costa†

*Departamento de Informática
Universidade Federal de Viçosa - UFV
Viçosa, MG - Brazil
diego.fialho@ufv.br, heberfa@gmail.com, alcione@dpi.ufv.br

†Departamento de Computação
Universidade Federal do Espírito Santo - UFES
Alegre, ES - Brazil
sidornellas@gmail.com

***Abstract*—Currently, as a result of globalization and other factors, the tourism market has remained high. Assisting the customer in order to satisfy their desires and constraints is a major challenge in this sector. This paper presents an agent-based system for developing a travel plan. In the system, according to the preferences informed by tourists, several agents cooperate in order to construct the travel plan that best suits the client needs. A tourism domain ontology was developed to enable communication between the agents. The system relies on the assistance of the greedy algorithm to trace the routes. A heuristic function was designed to ensure the fulfillment of the customer goals.**

***Keywords*—multiagent; ontology; Travel Planning**

I. Introduction

It's known that building a travel packet that satisfies the majority of desires and constraints of the clients is a tough task. There are several decisions to be taken and restrictions to be satisfied such as: displacements, schedules, time and cost constraints, accommodation restrictions, tourist attractions to be visited, etc. Therefore, a growing interest for automated systems that supports travel packets construction can be noticed nowadays. The agent-based systems approach can be helpful in this domain due to its characteristics: many autonomous agents (representatives from airlines, hotels, bus companies, tourist attractions) with their own goals interact to build a travel plan. To implement such a system is necessary to first establish a common vocabulary, using a lightweight ontology, so that agents can communicate. It is then necessary to design agents, the roles to be played and the interaction between agents in order to produce a travel plan that meets the users needs. This paper presents a Multiagent system (MAS) [1] for the travel planning problem. A tourism domain ontology was developed to enable communication between the agents. The system relies on the assistance of the greedy algorithm to trace the routes and the information required for the plan creation is located in a relational database. A heuristic function was designed to ensure the fulfillment of the customer goals. As related work, Lopez and Bustos [2] used a multiagent systems based on mobile devices to build a trip plan for a specific date. Schiafino and Amandi [3], presented an expert system, named *Traveller*, used for user assistance in solving travel problems.

II. Problem description

The problem to be addressed lies in the travel planning domain and the solution presented can be used to assist travel agencies. The problem consists in the construction of a travel plan that best satisfies the client needs, considering not only particular restrictions of the travel packet problem, but also clients constraints, like: time, money, preferences, and so on. The MAS approach is proposed in order to treat such a problem. The agents play roles in the system and interact in a collaborative way inside the environment. Each agent has its particular goals and the union of all agents' goals together reaches efficiently the general goal of building a travel packet. The Table I presents the roles in the system.

It is important to notice that each agent has an individual goal, which is related with the role played. The behavior of all agents and all their interactions, both with the environment and between the agents, shape the general behavior of the system.

The problem has some constraints that must be taken into account during the travel plan construction:

- The planning must fit in a time range (minimum and maximum date).
- There must be a minimum and a maximum number of cities to be visited.
- The travel time must be minimized.
- There must be a threshold cost.
- The travel must start in and return to the same place inside a period of time.

TABLE I. Roles occurring in the system

Role	Description
Planner Agent	It is the agent responsible for the development of the sequence of steps that make up the travel plan.
Tourist Agent	The agent that asks for the travel plan creation.
Travel Agent	The agent responsible for informing means of transport, accommodations, and interesting places to be visited.
Other Agents	Agents that represent airlines, bus companies, hostels, etc. These agents are responsible for informing prices, places available, and travel time.

- One city must not be visited more than once, except in the case it is used to return to another city.
- A person can visit the same kind of tourist attraction, in one or more cities, but not the same tourist attraction.

III. The System

To model the requirements of this project Tropos methodology [4] was used to help improve understanding of roles and goals of each agent. The main agents of the system are: *Planner*, *Tourist*, *Travel*, *Transport*, *Events Promoter* and *Lodging*.

The *Tourist* agent depends on the *Travel* agent to achieve the following objectives: (a) Perform Travel, (b) Cost Limited, (c) Limited Time, and (d) Minimum Transit Time. The objective (b) is related to financial constraints of the *Tourist* agent. The goal (c) is related to the time available to the *Tourist* agent to make the trip. The objective (d) is related to the *Tourist* agent desire to spend as little time as possible on transportation vehicles.

The *Tourist* agent also has a soft-goal, i.e., a goal that is important to be reached but it is not top priority, namely: *Enjoy the Journey*. This agent requires the asset: *travel plan*, which will be assembled by the *Planner* agent.

The *Travel* Agent is the only agent that communicates with all other agents to get the following information from them: (a) *Tourist* agent preferences, (b) *Transportation*, (c) *Attractions*, and (d) *Lodging*. The *Travel* Agent needs this information to achieve its goal *Sell Travel Package*. This agent has as soft-goal to *Attend well Tourist* agent. It also has the task: *Assemble Travel Plan*.

The *Planner* agent needs several resources, which are available for this agent by communicating with the *Travel* agent, namely: (a) *Tourist* agent preferences, (b) *Lodging*, (c) *Transportation*, (d) Pre-defined Plan from *Tourism* agent, (e) *initial City*, and (f) *Attractions*.

The agents *Transport*, *Lodging* and *Event Promoter* are only responsible for providing the resources: *Transportation*, *Lodging* and *Attraction*, respectively, for the *Travel* agent. Fig. 1 shows through an use case diagram the actors and their functional dependencies.

In order to enable communication between the agents was necessary to develop an ontology for the travel domain. Although the ontology was created using OWL tools, it was stored as relational database due to its capabilities for fast retrieval of the data. This was an important feature as there were a great deal of instances (cities, attractions an hotels) to stores and manipulate. Fig. 2 shows the database schema used to store the ontology instances.

Most of the classes are self-explained. The class *Attribute* has a single property name, which stores the description of the attribute including it in a theme or a facet of the related classes. The attributes have many-to-many: with classes: *Accomodation*, *Attraction*, *Event*. For instance: (a) a rock concert, which is an event, can have the attribute "music" among others; (b) a resort have rooms that can have the attribute "whirlpool". The *Accomodation* class store information for

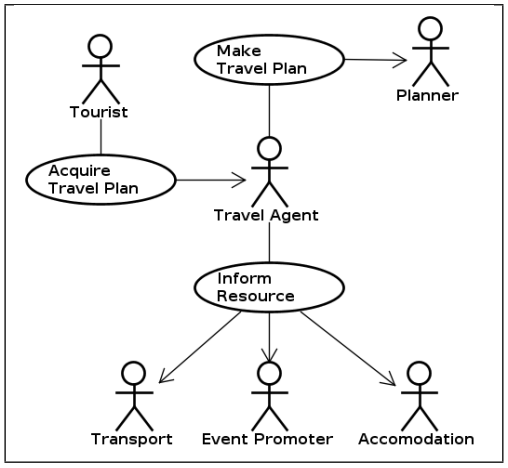

Fig. 1. System use case diagram.

inns, hotels and hostels. The property *kind* stores the type of accommodation. The *Transportation* Class represents the various means of transportation such as: bus, plane, ship, etc. The routes are specified from the *Link* class. Each route has the city of origin, destination, and a means of transportation. The associated class *LinkHasTransportation* specify the price and travel time between the cities for a particular mode of transport. These properties are important for assembling the travel plan.

A. The Planning Algorithm

In this work we adopted a greedy search for the assembly of the trip itinerary. In the greedy method decisions are made in an isolated way and in every step its decision is based on a heuristic function. The purpose of this function is to guide the greedy algorithm in the search for the most interesting attractions, i.e. those that meet the highest amount of desires and constraints of the tourists. Thus, we specified the

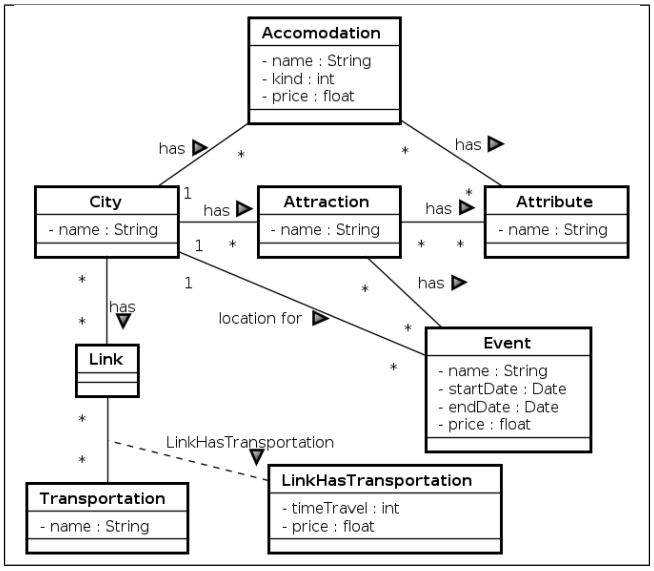

Fig. 2. The database schema.

criteria that influence, positively and negatively in the decision to participate in one attraction, namely: (a) the number of attributes that match the tourists preferences (b) the attraction cost, and this comprises the attraction cost plus lodging (it depends on the number of stay days), (c) the transportation cost, (d) and the transportation time to the city of the attraction. This is expressed the following function:

$$\begin{aligned} f(a) \quad = \quad & n(a) * k_1 \\ & -(c(a) + cl(a) * s(ca)) * k_2 \\ & -(tt(ca) * ct(ca) * k_3) \end{aligned} \tag{1}$$

Where: a denotes the attraction; ca stands for the city of the attraction; $f(a)$ is the heuristic value of the attraction; $n(a)$ is the number of attributes satisfied by the attraction; $c(a)$ is the cost of the attraction (ticket and other costs of the attraction); $cl(a)$ is the cost of accommodation in the city of the attraction; $s(ca)$ is the number of days staying in the city of the attraction; $tt(ca)$ is the time spent traveling to the city of the attraction; $ct(ca)$ is the cost of traveling to the city of the attraction; k_1, k_2 and k_3, are constant for adjustment of the equation. The great weight of the portion $(tt(ca) * ct(ca) * k_3)$ aims to make the algorithm select attractions in cities located close to each other. The algorithm also takes into account the financial cost of the return to hometown (co). The algorithm has addressed these constraints through the function $r(co)$ (cost of returning to co):

$$r(co) = (c(a) + cl(a) * s(ca)) - (cv(co, c)) \tag{2}$$

Where: co denotes the hometown; $cv(co, c)$ is the cost of returning hometown from the attractions' city. The value of $r(co)$ must always be more than 0.

IV. Results

Fig. 3 shows the output generated by the system for a simulation where the client requests a travel itinerary for a tourist region well known in Brazil, called "Serras Gauchas".

V. Conclusions

This paper presented a MAS in order to assemble tour packages according to the preferences and limitations of the tourists. In the MAS agents were used to search for information such as: accommodation, attractions, events, cities, transport, etc. A specific planner agent was used to assemble the travel plan from the information provided by other agents.

The greedy algorithm was used to assist in the assembly of the travel plan. The choice of cities was done in an iterative manner, being influenced by criteria: (a) positive: attributes of tourist interest, and (b) negative: cost and travel time to the next town. Because of this, it can be observed that the choice of cities have a behavior prone to formation of clusters, i.e. where it tends to group cities relatively close if they meet the interests of the visitor. The solution proved to be feasible in the tests with a small number of cities. Tests with larger number of cities need to be done.

```
Event  1
Name = Festa do Boi
city = caxias do sul
Accomodation = hotel de caxias
Number of Days = 2
Event Price = 8.0
Accomodation Price=250.0
Start Date = Sun Mar 04 00:00:00 BRT 2012
Event  2
Name = Festa da cultura gaucha farropilha
city = porto alegre
Accomodation = porto do sol
Number of Days = 6
Event Price = 12.0
Accomodation Price=120.0
Start Date = Tue Feb 21 00:00:00 BRST 2012
Event  3
Name = V Encontro do chimarrão
city = passo fundo
Accomodation = passo fundo grande hotel
Number of Days = 5
Event Price = 9.0
Accomodation Price=150.0
Start Date = Wed Mar 07 00:00:00 BRT 2012
Event  4
Name = Festival do churrasco
city = gramado
Accomodation = grama palace hotel
Number of Days = 4
Event Price = 11.0
Accomodation Price=200.0
```

Fig. 3. Itinerary generated for "Serras Gauchas".

As future work it would be interesting to use Internet to assemble tour packages. So it could seek the route information as distance, travel time between cities, etc., using Google Maps for instance. It would be interesting also to evaluate other search algorithms such as A*, which has the property to obtain optimal solutions for admissible heuristics. The system could also use a database of small pre-built routes to be dynamically joined.

Acknowledgment

This work is financed by funding agencies FAPEMIG, CNPq, FUNARBE and by the Gapso Company.

References

[1] M. Wooldridge, *An Introduction to MultiAgent Systems*. John Wiley & Sons Inc, 2002.

[2] J. S. Lopez and F. A. Bustos, "Multiagent tourism system: An agent application on the tourism industry," in *Proceedings of the International Joint Conference IBERAMIA/SBIA/SBR*, Brazil, 2006.

[3] S. Schiafino and A. Amandi, "Building an expert travel agent as a software agent," *Expert Systems with Applications*, vol. 36, pp. 1291–1299, 2009.

[4] P. Bresciani, P. Giordini, F. Giunchiglia, J. Mylopoulos, and A. Perini, "Tropos: An agent oriented software development methodology," *Journal of of Autonomous Agents and MultiAgent Systems*, vol. 8, no. 3, pp. 203–236, May 2004.

Towards a fault model for BDI agents: an initial study

Francisco J. P.Cunha, Elder Cirilo, Carlos Lucena
Laboratório de Engenharia de Software
Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Brasil
{fcunha,ecirilo,lucena}@inf.puc-rio.br

***Abstract*—Given the growing approaches based on the paradigm of multi-agent systems and hence the need to maintain the quality of the software produced, testing becomes an activity of a great importance. For the agent-oriented projects in the BDI architecture is not different. Thus, it is necessary for testing agents to have a suitable fault model that ensure the correctness of the proposed behavior. In this paper, we propose an initial analysis of a fault model. The structure of plans and goals of the agents is transformed into a sequence of actions in order to allow its verification and execution control.**

Keywords— fault model, tests in BDI agents, testability agents;

I. INTRODUÇÃO

Soluções de software baseadas em sistemas multi agentes são cada vez mais utilizadas. Agentes tem a capacidade de raciocinar usando a cooperação com outros agentes para alcançar metas estabelecidas [3]. Assim como nos paradigmas tradicionais de desenvolvimento, há a necessidade de garantir robustez e corretude, o que se dá através de testes [1]. Uma observação importante é que, testes em sistemas multi agentes não é uma tarefa trivial se comparada a sistemas tradicionais.

A proposta desse trabalho é realizar um estudo inicial sobre a testabilidade em sistemas de agentes BDI, identificando as características relevantes para elaboração de um modelo de falhas. Assim, analisamos quantitativamente a execução de um agente. No que segue, uma introdução da arquitetura BDI é apresentada na seção 2, na seção 3 é apresentado o mapeamento dos agentes para uma árvore de planos e metas e na seção 4 é analisada quantitativamente a execução. A conclusão é apresentada na seção 5.

II. BACKGROUND – ARQUITETURA BDI

Uma arquitetura de sucesso para desenvolver agentes deliberativos é o modelo BDI (Belief, Desire e Intention). [7] desenvolveu uma teoria de raciocínio prático humano que descreve o comportamento de maneira racional pelas noções de "Crenças", "Desejos" e "Intenções". A implementação do presente modelo substitui os dois últimos conceitos pelos conceitos de "metas" e "planos". Contudo, meta não é um conceito chave em sistemas BDI uma vez que são modeladas como eventos. Assim, estabelecer uma meta, corresponde a selecionar e executar um plano que possa manipular esse evento [5]. No restande deste trabalho consideramos os planos como manipulação de eventos [2]. Um plano é constituído por três partes: um evento padrão especificando o que é relevante, uma condição de contexto indicando em que situações o plano pode ser usado, e o corpo do plano [2].

Um ciclo de execução BDI é uma sequência como a descrita abaixo: (i) um evento ocorre; (ii) o agente determina um conjunto de instâncias de planos cujos eventos padrão correspondem ao evento gerador; (iii) o agente avalia as condições de contexto dos planos relevantes para gerar o conjunto de instâncias aplicáveis dos planos; (iv) uma das instâncias aplicáveis do plano é selecionada e executada; (v) se o corpo plano falhar, então um mecanismo de manipulação de falhas é acionado.

Este ciclo gera uma trilha de operações que compreende a sequência de ações com tentativas de execução e um indicador de sucesso ou fracasso. Existem algumas abordagens para lidar com o fracasso. Talvez a abordagem mais comum, e que é utilizada em diversas plataformas BDI existentes é a de selecionar um plano alternativo aplicável, e considerar apenas um evento como tendo falhado, quando não existem planos aplicáveis restantes. Ao determinar planos alternativos pode-se considerar o conjunto de planos aplicáveis existentes ou recalcular o conjunto de planos aplicáveis uma vez que a situação pode ter mudado desde que os planos foram determinados. Algumas plataformas BDI usam como tratamento de falhas, repetir os planos em caso de falha [2].

III. EXECUÇÃO BDI COMO ÁRVORE DE PLANOS E METAS

A execução de um agente BDI é um processo dinâmico que executa ações progressivamente. Na análise deste processo e, para torná-lo numa forma declarativa, aplicamos uma transformação de maneira que o comportamento do agente seja mapeado em uma árvore de planos e metas e que seja executado como uma sequência de ações. Os planos e metas podem ser visualizados como uma árvore onde cada meta tem como filhos os planos que lhes são aplicáveis e cada plano tem como filhos os seus subplanos [2]. Trata-se de um tipo de árvore "e/ou" onde cada meta é realizada por um dos seus planos ("ou") e cada plano precisa de todos os seus subplanos para ser alcançado ("e") [3]. A escolha do plano para cada meta na árvore não é um processo determinista. Além disso, ao considerar uma falha, precisa-se saber o que fazer para recuperação. Inicialmente, uma árvore de planos e metas é representada como um termo em Prolog seguindo uma gramática simples [3]. GPT é a abreviação de "Goal-Plan

Tree", AoGL é a abreviação de "Action or Goal List", e A é um símbolo.

```
⟨GPT⟩ ::= goal([]) | goal([⟨PlanList⟩])
⟨PlanList⟩ ::= ⟨Plan⟩ | ⟨Plan⟩, ⟨PlanList⟩
⟨Plan⟩ ::= plan([]) | plan([⟨AoGL⟩])
⟨AoGL⟩ ::= act(A) | ⟨GPT⟩ | act(A), ⟨AoGL⟩ | ⟨GPT⟩, ⟨AoGL⟩
```

Fig. 1 Uma gramática para transformação da árvore de planos e metas [2]

Como um exemplo, retirado do trabalho de Winikoff et al. [2], uma meta com dois planos aplicáveis, cada um contendo uma única ação, é representado pelo termo: goal ([plan ([act (a)]), plan ([act (b)])]). Um predicado não determinista exec é definido onde seu primeiro argumento (entrada) é a árvore de planos e metas e o segundo argumento (saída) é uma sequência de ações. Consideremos que a árvore de planos e metas contenha somente instâncias de planos aplicáveis. Assim, para transformar um "nó meta" em uma sequência de ações, selecionamos uma de suas instâncias aplicáveis do plano. O plano selecionado é transformado em uma sequência de ações (linha 2). Quaisquer uns dos planos aplicáveis podem ser escolhidos e não apenas o primeiro. Se o plano selecionado é executado com sucesso então o resultado do trace é uma sequência de ações para a execução de uma meta (linha 3). Caso contrário, é executada a recuperação de falhas (linha 10), que é feita tomando os planos restantes, ou seja, excluindo o plano que já foi tentado. A sequência de ações resultantes é anexada à sequência de ações do plano de falha para obter uma sequência completa de ação para o objetivo. Especificamente, um plano aplicável é selecionado e executado, e se for bem sucedido, então a execução para. Se não for bem sucedido, então um plano alternativo é selecionado e a execução continua, ou seja, as sequências de ações são anexadas.

```
1  exec(goal([]),[]).
2  exec(goal(Plans), Trace) :- remove(Plans,Plan,Rest), exec(Plan,Trace1),
3      (failed(Trace1) -> recover(Rest,Trace1,Trace) ; Trace=Trace1).
4  exec(plan([]), []).
5  exec(plan([Step|Steps]), Trace) :- exec(Step,Trace1),
6      (failed(Trace1) -> Trace=Trace1 ; continue(Steps,Trace1,Trace)).
7  exec(act(Action), [Action]).
8  exec(act(Action), [Action,fail]).
9  failed(Trace) :- append(X,[fail],Trace).
10 recover(Plans,Trace1,Traces) :-
11     exec(goal(Plans),Trace2), append(Trace1,Trace2,Traces).
12 continue(Steps,Trace1,Trace) :- exec(plan(Steps),Trace2),
13     append(Trace1,Trace2,Trace).
14 % remove(A,B,C) iff removing element B from list A leaves list C
15 remove([X|Xs],X,Xs).
16 remove([X|Xs],Y,[X|Z]) :- remove(Xs,Y,Z).
```

Fig. 2 Mapeamento da árvore de planos e metas em uma sequência de ações [2]

IV. Comportamento dos Agentes BDI

É importante considerar o número de possibilidades de comportamentos existentes para um agente BDI que está tentando atingir uma meta. Usando o mapeamento anterior, vemos a execução de um agente BDI como a transformação de uma árvore de planos e metas para uma sequência de ações. Assim, a respeito do número de possibilidades para o comportamento de agentes BDI, derivam fórmulas que permitem calcular o número de comportamentos, de sucesso e fracasso para uma determinada árvore de planos e metas [3]. Assumindo que, todas as subárvores de um nó plano ou meta têm a mesma estrutura podemos definir a profundidade de uma árvore de planos e metas como o número de níveis de "nós-meta" que ele contém. Uma árvore de profundidade igual a 0 é um plano sem submetas, enquanto que uma árvore com profundidade d > 0 ou é um nó plano com filhos que são nós-meta de profundidade d, ou um nó meta com filhos que são planos em profundidade d - 1. Assumimos que todos os planos de profundidade d > 0 têm k submetas e que todas as metas têm j instâncias aplicáveis do plano. Este pode ser o caso de cada meta ter j planos relevantes, cada qual resultando em exatamente um plano aplicável, mas também pode ser o caso de outras formas, por exemplo, uma meta pode ter 2 planos relevantes, metade dos quais são aplicáveis na situação atual.

A seguinte terminologia foi utilizada nas seções abaixo: (i) pressupomos que a estrutura da subárvore com raiz em um nó plano ou meta é determinada unicamente por sua profundidade e, portanto, podemos denotar uma meta ou plano em profundidade d como g_d ou p_d, respectivamente; (ii) usamos $n^{\checkmark}(x_d)$ para identificar o número de caminhos de execução bem sucedida de uma árvore de profundidade d e com raiz em x; (iii) analogamente, usamos $n^{\times}(x_d)$ para denotar o número de caminhos de execução mal sucedidas de uma árvore de profundidade d com raiz x; (iv) estendemos a notação para as sequências $x_1;...;x_n$ onde cada x_i é uma meta ou ação e ";" denota uma composição sequencial. Abreviamos a sequência de n ocorrências de x por x^n.

A. *Caso Base: Comportamento com execuções de sucesso*

Iniciamos o processo calculando o número de caminhos bem sucedidos na árvore de planos e metas na ausência de falhas [6]. O número de caminhos possíveis para a alcançar uma meta é a soma do número de caminhos em que seus filhos podem ser alcançados. Por outro lado, o número de caminhos para se alcançar um plano é o produto do número de maneiras em que os seus filhos podem ser alcançados [2]. Sendo uma árvore com raiz g, assumimos que cada um dos seus j filhos podem ser alcançados de n diferentes maneiras, então, ao selecionar um dos filhos o número de maneiras na qual g pode ser alcançado é jn. Da mesma forma, para uma árvore com raiz p ("plan"), assumimos que cada um dos seus filhos k filhos podem ser alcançados de n maneiras diferentes, então, ao executar todos os seus filhos, o número de maneiras na qual p pode ser executado é n^k. Um plano sem filhos pode ser executado exatamente de uma maneira. Representamos o cenário descrito acima respectivamente pelas equações: $n^{\checkmark}(g_d) = jn^{\checkmark}(p_{d-1})$; $n^{\checkmark}(p_0) = 1$; $n^{\checkmark}(p_d) = n^{\checkmark}(g_d)^k$.

B. *Adicionando Falha*

Estendendo a análise para incluir falhas, determinamos o número de execuções mal sucedidas. Nenhum mecanismo de tratamento de falhas será considerado. Para determinar o número de execuções mal sucedidas, precisamos saber onde cada falha pode ocorrer. Em sistemas BDI há dois lugares onde as falhas ocorrem: quando uma meta não possui instâncias de planos aplicáveis e, quando uma ação dentro do corpo de um plano falha [2]. Neste trabalho consideramos somente metas com instâncias aplicáveis. Logo, um plano pode falhar devido à falha de quaisquer umas das ações e submetas presentes no corpo do plano. Mais especificamente, um plano falha se a tentativa de executar sequencialmente seu corpo falha.

Generalizando, podemos assumir que há um número de ações antes, depois e entre as submetas de um plano. Um plano

sem submetas é considerado consistente e de uma única ação. Assim, o número de caminhos mal sucedidos de uma meta é definido por $n\times(gd) = j$, de um plano que é uma folha na árvore $(d=0)$ por $n\times(p0) = \varphi$ e um plano $(d > 0)$ por $\varphi + (n\times(gd) + \varphi n\checkmark(gd)) * (n\checkmark(gd) - 1/ n\checkmark(gd) – 1)$ [2].

C. *Adicionando tratamento de falha*

Uma forma comum de lidar com as falhas é responder a falha de um plano, tentando aplicar um plano alternativo ao evento gerador. Como resultado, é mais difícil ocorrer uma falha. A única maneira é se todos os planos falharem. O efeito da adição do tratamento de falha é converter possíveis falhas para sucessos, ou seja, uma execução que de outra forma seria mal sucedida, é estendida para uma longa execução que pode ser bem sucedida. O número de execuções de uma meta com j instâncias de planos aplicáveis e φ ações no corpo do plano, pode ser representado por: $n\times(g_d) = j!n\times(p_{d-1})^j$; $n\times(p_0) = \varphi$; $n\times(p_d) = \varphi + (n\times(g_d) + \varphi n\checkmark(g_d))(n\checkmark(g_d)^k -1/ n\checkmark(g_d) -1)$.

D. *A probabilidade da execução falhar*

A introdução do tratamento de falhas torna a possibilidade de falha na execução muito menor. Quando olhamos unicamente para o número de possibilidades de falhas, verificamos que o número de possibilidade de ocorrer uma falha é grande. Acontece que, o mecanismo de tratamento de falhas reduz drasticamente esse número devido a probabilidade de uma falha, de fato, ocorrer.

E. *Análise dos números e equações*

Analisando as equações apresentadas verificamos que percorrer exaustivamente cada caminho possível na árvore de planos e metas é, de fato, inviável. A medida que a profundidade e largura da árvore aumentam, o número de possibilidades de caminhos cresce exponencialmente. Para dimensões de uma árvore pequena, talvez seja possível fazer algum tipo de verificação, mas sem dúvida, trata-se de uma solução não-escalável.

V. CONCLUSÃO

Como contribuição, o presente trabalho revisitou a literatura relacionada a testabilidade em SMA de agentes BDI. Essa revisão inicial revelou que, o número de comportamentos possíveis na execução de agentes BDI de fato cresce à medida que a profundidade e a largura das árvores de planos e metas aumentam. Uma observação interessante é que, a introdução de tratamento de falha faz uma diferença significativa no número de comportamentos. Ao considerar o teste do sistema como um todo, concluímos sobre a testabilidade de sistemas de agentes BDI que, de acordo com as equações apresentadas, tentar obter a garantia de correção do sistema por meio de testes do sistema como um todo, não é viável. Na verdade, a situação é ainda pior quando consideramos não apenas o número de possíveis execuções, mas também a probabilidade de falha. Então, o teste do sistema de agentes BDI parece ser impraticável, nessas condições. Sobre testes unitários e testes de integração, apesar dos testes de unidade e integração serem relevantes, nem sempre é clara a forma de aplicá-los utilmente para sistemas de agentes. Para os agentes BDI, ao testar uma submeta, pode ser difícil de assegurar que o teste abrange todas as situações em que podem ser tentados. Igualmente, ao testar um agente sem o resto do sistema (incluindo outros agentes) pode ser difícil garantir a cobertura das diferentes possibilidades.

No entanto, a conclusão parece ser a de que SMA BDI são verificáveis através de testes quando restringimos certas caracteristica da representação do comportamento dos agentes. Neste sentido, constatamos que existem diveras lacunas em aberto e, assim, a existência de uma área promissora de pesquisa. Podemos considerar como tópicos de pesquisa: (i) a investigação de mecanismos para geração de planos de testes automatizados; (ii) rastreamento e mapeamento das condições de contexto e trocas de mensagens para a sequência de ações geradas; (iii) análise de um modelo de falhas; e (iv) estudo de viabilidade de extensão do framework JAT para utilização do modelo de falhas proposto.

REFERENCES

[1] Sommerville, I., Software Engineering (Sixth edition), Addison Wesley (2000).

[2] Winikoff, M., Cranefield, S., On the testability of BDI agent systems. Information Science Discussion Paper 2008/03, University of Otago, Dunedin, New Zealand, 2008.Disponível em http://www.business.otago.ac.nz/infosci/pubs/papers/dpsall.htm.

[3] Sudeikat, J., Validation of BDI Agents. In: Bordini, R.H., Dastani, M.M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2006. LNCS (LNAI), vol. 4411, pp. 185–200. Springer, Heidelberg (2007).

[4] Low, C.K., Chen, T.Y., Rönnquist, R.: Automated test case generation for BDI agents. In: Autonomous Agents and Multi-Agent Systems, vol. 2, pp. 311–332 (1999).

[5] Nunes, I., Lucena, C.J.P., Luck, M. (2011), BDI4JADE: a BDI layer on top of JADE, in Louise A. Dennis, Olivier Boissier and Rafael H. Bordini, ed., Ninth International Workshop on Programming Multi-Agent Systems (ProMAS 2011), Taipei, Taiwan, pp. 88-103.

[6] Padgham, L., Winikoff, M., Developing Intelligent Agent Systems: A Practical Guide. John, Wiley and Sons (2004).

[7] Bratman, M.: Intentions, Plans, and Practical Reason. Harvard Univ. Press (1987).

Simulating Consumers Energy Profiles through Multiagent Systems

Fernanda P. Mota[1], Vagner S. da Rosa[2], Graçaliz P. Dimuro[3], Silvia S. da C. Botelho[4]

Computational Science Center- Federal University of Rio Grande
Av. Itália km 8 – Campus Carreiros – 96.201-900– Rio Grande – RS – Brasil
{[1]nandap.mota, [2]vsrosa, [3]gracaliz, [4]silviacb.botelho}@gmail.com

***Abstract*— Simulation of home use of electric energy is a very powerful tool for the purpose of studying, planning and managing at electric energy distribution companies. This paper presents a NetLogo-based multi-agent system for energy consumption simulation in residential areas. Several possible consumers profiles and household appliances with different powers are modeled and simulated using computational agents. Simulation results are presented and discussed.**

Keywords*— Multiagent Systems, NetLogo, Electricity Consumption

I. INTRODUÇÃO

O consumo de energia é um dos principais indicadores do desenvolvimento econômico e do nível de qualidade de vida de qualquer sociedade. Ele reflete tanto o ritmo de atividade dos setores industrial, comercial e de serviços, quanto à capacidade da população para adquirir bens e serviços tecnologicamente mais avançados, como automóveis, eletrodomésticos e eletroeletrônicos.

Segundo a ANEEL (Agência Nacional de Energia Elétrica), em maio de 2012 o consumo de energia elétrica cresceu 3,8% no Brasil em relação ao mesmo período do ano de 2011, atingindo 36.900 Gigawatts-hora (GWh) [1]. O setor residencial foi outro que teve crescimento do consumo acima da média: 4,3%. O destaque desse segmento também foi a Região Nordeste, que concentrou 36% do aumento.

No entanto, a expansão acentuada do consumo de energia elétrica, embora possa refletir o aquecimento econômico e a melhoria da qualidade de vida, possui aspectos negativos tais como: a possibilidade do esgotamento dos recursos utilizados para a produção de energia, o impacto ao meio ambiente produzido por essa atividade e os elevados investimentos exigidos na pesquisa de novas fontes e construção de novas usinas [2].

Neste sentido, o trabalho em questão envolve esforços direcionados em prover dados que auxiliam na análise deste tipo de situação, por meio das técnicas de simulação baseada em agentes e mais especificamente com suporte da ferramenta NetLogo. Deste modo, as seções a seguir, as quais demonstram mais detalhes desta proposta estão organizadas da seguinte maneira: seção 2 descreve resumidamente os aspectos conceituais sobre sistemas multiagentes e a ferramenta NetLogo; a seção 3 demonstra o modelo inicial de simulação desenvolvido; a seção 4 relata os resultados que foram gerados pela execução do modelo e por fim, na seção 5 têm-se considerações finais e os trabalhos futuros.

II. SISTEMAS MULTIAGENS E NETLOGO

No contexto de Inteligência Artificial, podem-se definir agentes como entidades computacionais que, inseridos em um ambiente, são capazes de perceber e atuar sobre o mesmo. Um agente computacional possui atributos como operar sob controle autônomo, perceber seu ambiente, persistir por um período de tempo, adaptar-se a mudanças e ser capaz de assumir metas [3].

Os Sistemas Multiagentes têm como principal característica a coletividade e não um indivíduo único, e desta forma, passa-se o foco para a forma de interação entre as entidades que formam o sistema e para a sua organização [4].

Existem diversos ambientes de programação que foram projetados para trabalhar-se com a modelagem baseada em agentes, contudo com diferentes vantagens conforme tabela 1.

TABELA I. COMPARAÇÃO ENTRE ALGUNS AMBIENTES DE MODELAGEM BASEADA EM AGENTES [5].

Métrica\Plataforma	Ascap	Mason	Repast	NetLogo	SWARM
Quantidade de Usuários	Baixa	Crescente	Grande	Grande	Baixa
Linguagens	Java	Java	Java Python	NetLogo	Java Objective C
Velocidade de Execução e Programação	Média	Mais Rápida	Rápida	Média	Média
Facilidade de Aprendizagem	Média	Média	Média	Média	Média
Documentação	Boa	Pouca	Pouca	Muita	Boa

Neste trabalho optou-se pela ferramenta NetLogo [6], especialmente por oferecer facilidade de programação, portabilidade, documentação abundante, acesso e uso gratuitos. Nele podem ser dadas instruções a centenas ou milhares de agentes, os quais trabalham paralelamente [7].

III. MODELO DE SIMULAÇÃO NO NETLOGO

O modelo de simulação de consumo de energia baseado no paradigma de agentes e implementado na ferramenta NetLogo (Fig. 1), possui as seguintes características:

- Os perfis dos consumidores de energia elétrica, que são baseados na renda familiar de acordo com os dados do IBGE [8] foram representados com uma cor para cada tipo de renda (Fig1), assim:
  - Azul: possuem renda até R$800,00;
  - Amarelo: possuem renda de R$ 800,00 a R$ 1.245,00 ;
  - Vermelho: possuem renda de R$6.225,00 a R$ 10.375,00;
  - Vermelho: possuem renda maior que R$ 10.375,00.
- Os consumidores e os eletrodomésticos foram modelados como agentes computacionais racionais, ou seja, se um equipamento já estiver ligado eles não irão ligar novamente até que o mesmo seja desligado e não esquecerão o chuveiro ligado quando saírem de casa. Especificamente no que se refere ao NetLogo, podem-se implementar perfis diferentes de agentes por meio da criação de distintas *breeds*. Estas representam a ideia de espécies de agentes, onde cada uma incorpora um conjunto de diferentes instruções a serem executados, como por exemplo, na simulação de uma colmeia os agentes da espécie operária possuem comportamentos distintos da espécie rainha.
- Os agentes que representam os usuários podem ficar um período de tempo dormindo, este pode variar de 0 até 8 horas de sono. No entanto, o usuário não irá consumir enquanto estiver dormindo.
- Os agentes que representam os usuários podem ficar um período de 8 horas fora de casa. No entanto, o usuário não irá consumir enquanto estiver fora de casa.
- O consumo de energia de cada equipamento foi calculado de acordo com os dados fornecidos pelas distribuidoras de energia *light* [9] e CEEE [10].
- Os equipamentos possuem a mesma probabilidade de serem escolhidos, com exceção:
  - **Renda1:** chuveiro, geladeira e televisão que possuem uma probabilidade maior de serem escolhidos.
  - **Renda2:** chuveiro, *freezer*, geladeira e televisão que possuem uma probabilidade maior de serem escolhidos.
  - **Renda3 e Renda 4:** chuveiro, *freezer*, lavadora de roupas ,geladeira e televisão que possuem uma probabilidade maior de serem escolhidos.
- A geladeira e o *freezer* são os únicos equipamentos que permanecem ligados por um período de 24 horas.
- O modelo está simulando a estação verão, demonstrando os equipamentos que ele utiliza neste período e tempo que cada um deles é utilizado.

Fig. 1: Interface do modelo que simula o consumo de energia elétrica dos quatro tipos de consumidores.

IV. RESULTADOS PRELIMINARES

Foram modeladas quatro tipos de rendas, onde cada renda é composta por famílias de 1 a 3 pessoas, as simulações duraram um período de 24 horas e pelas seguintes características:

A. *Renda 1 (renda familiar até R$830,00):*

Nesta renda cada família pode conter uma lista de 3 a 10 equipamentos, essa lista varia de acordo com um valor randômico no início da simulação. No entanto, essa renda terá no mínimo: uma geladeira, uma televisão e um chuveiro elétrico. Conforme pode ser observado na Fig. 2, as residências tiveram um consumo médio de 64,48 kWh e um desvio padrão de 5,28 kWh.

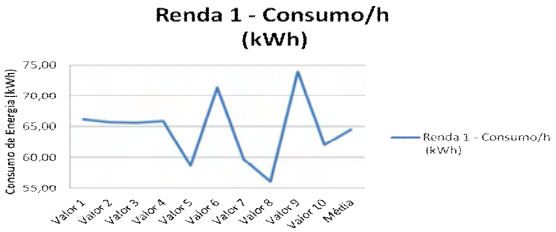

Fig. 2: Simulação do consumo de energia elétrica da renda 1, onde o eixo horizontal representa os dias de consumo e o eixo vertical representa o consumo de energia em kwh para cada dia simulado.

B. *Renda 2 (renda familiar de R$830,00 a R$1.245.00):*

- Nesta renda cada família pode conter uma lista de 4 a 15 equipamentos, essa lista varia de acordo com um valor randômico no início da simulação. No entanto, essa renda terá no mínimo: uma geladeira, uma televisão, lâmpada de 60 *watts* e um chuveiro elétrico. Conforme pode ser observado na Fig. 3, as residências

tiveram um consumo médio de 112,14 kWh e um desvio padrão de 4,90 kWh.

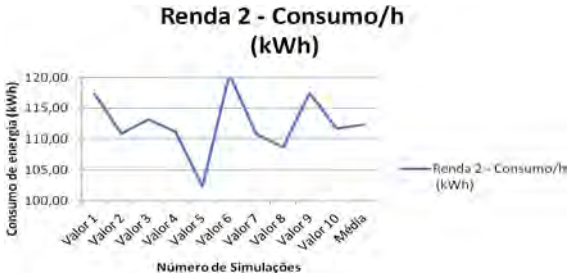

Fig. 3: Simulação do consumo de energia elétrica da renda 2, onde o eixo horizontal representa os dias de consumo e o eixo vertical representa o consumo de energia em kwh para cada dia simulado.

C. Renda 3 (renda familiar de R$6.225,00 a R$ 10.375,00):

Nesta renda cada família pode conter uma lista de 4 a 20 equipamentos, essa lista varia de acordo com um valor randômico no início da simulação. No entanto, essa renda terá no mínimo: uma geladeira, um *freezer*, uma televisão e um chuveiro elétrico. Conforme pode ser observado na Fig. 4, as residências tiveram um consumo médio de 395,75 kWh e um desvio padrão de 24,65 kWh.

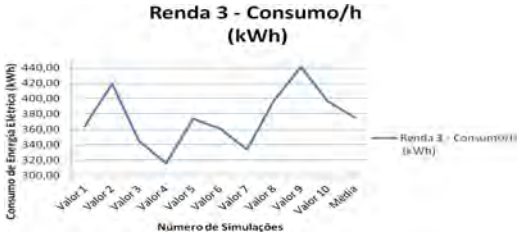

Figura 4: Simulação do consumo de energia elétrica da renda 3, onde o eixo horizontal representa os dias de consumo e o eixo vertical representa o consumo de energia em kwh para cada dia simulado.

D. Renda 4 (renda familiar maior que R$ 10.375,00):

Nesta renda cada família pode conter uma lista de 4 até a 20 equipamentos, essa lista varia de acordo com um valor randômico no início da simulação. No entanto, essa renda terá no mínimo: uma geladeira, freezer, lavadora de roupas, uma televisão e um chuveiro elétrico. Conforme pode ser observado na Fig. 5, as residências tiveram um consumo médio de 374,69 kWh e um desvio padrão de 36,99 kWh.

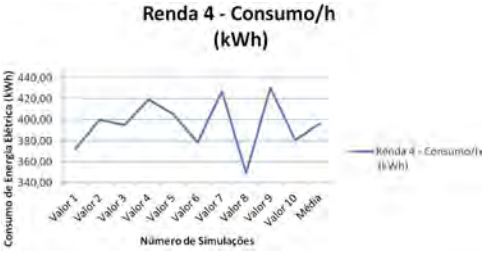

Fig. 5: Simulação do consumo de energia elétrica da renda 4, onde o eixo horizontal representa os dias de consumo e o eixo vertical representa o consumo de energia em kwh para cada dia simulado.

V. Conclusões

Como demonstrado nos resultados pode-se dizer que a utilização do paradigma de agentes bem como da ferramenta NetLogo é uma alternativa interessante para simulações de cenários de perfis de usuários de energia elétrica, pois os diversos comportamentos inerentes a sociedade utilitária deste serviço podem ser mapeados respectivamente em deferentes tipos de agentes os quais são inseridos em uma ambiente virtual.

Enfatizamos que estes resultados são reduzidos a escopo de um teste inicial e estamos buscando dados reais para futuramente compararmos os resultados atingidos na simulação com os dados de consumo de energia reais da cidade do Rio Grande. No entanto, esta avaliação inicial aponta para futuros trabalhos com mais elementos e maior número de perfis a serem analisados, sendo este a motivação para os próximos passos do trabalho.

Como trabalhos futuros serão criados perfis que não são econômicos e serão simuladas outras estações tais como: inverno, primavera e outono. Outra perspectiva é o teste considerando punições que afetem o comportamento dos agentes, como por exemplo, as multas por excesso de consumo de energia.

Referências

[1] Multiner , “ Consumo de energia elétrica cresce no Brasil e Belo Monte é garantia para essa demanda,” disponível em: http://www.multiner. com.br/ multiner/ Default. aspx?Tab Id=117, acessado em: dezembro de 2012.

[2] ANEEL. “Atlas de Energia Elétrica do Brasil, ” disponível em: www.aneel.gov.br/ arquivos/ pdf/livro_atlas.pdf, acessado em dezembro de 2012.

[3] S. Russel , P. Norvig, “Artificial Intelligence: A modern approach”, 2nd edition, Pearson Education, 2003.

[4] G. P. Dimuro, A. C. R. Costa, L. A. Palazzo, “Systems of exchange values as tools for multi-agent organizations,” journal of the Brazilian Computer Society, 11(1):3150, 2005.

[5] P. Sapkota, “Modeling Diffusion Using an Agent-Based Approach, ” PhD thesis, University of Toledo, 2010.

[6] S.Tisue, U. Wilensky, ” Netlogo: A simple environment for modeling complexity, ” in International Conference on Complex Systems, Boston, 2004.

[7] U. Wilensky, “NetLogo, ” disponível em: http://ccl.northwestern.edu/ netlogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999.

[8] IBGE. “Pesquisas de Orçamentos Familiares 2008- 2009, despesas, rendimentos e condições de vida,” 2009.

[9] Light, “Simulador de consumo de energia elétrica, ” disponível em: http://www.aessul.com.br/areacliente/servicos/simula.asp, acessado em janeiro de 2013.

[10] CEEE, “Simulador de consumo de energia elétrica, ”disponível em: www.ceee.com.br/pportal/ceee/component/controller.aspx?cc=1221, acessado em janeiro de 2013.

Multiagent Systems Simulation of Dengue in Minas Gerais (Brazil)

Katia Cristina A. Damaceno Borges, Willian Magno Pereira Reis, Alcione de Paiva Oliveira
DPI – Departamento de Informática
UFV- Universidade Federal de Viçosa
Viçosa-MG -Brazil
{katia.borges, willian.reis} @ufv.br, alcione@dpi.ufv.br

***Abstract*— Due to high rates of spread of Dengue fever, as well as high impact on global health and its high cost to public coffers, it is necessary a study on appropriate measures to be taken prior to installation of the epidemic. However, due to the high complexity of the variables involved, it is difficult to describe or make predictions about the advance of the epidemic. Thus, this paper proposes the modeling and simulation of a subsequent multi-agent system applied to the scenario of Dengue in Brazil. The modeling was performed using the software Netlogo and the chosen endemic region was Uberaba in Minas Gerais-Brazil. The results showed that the simulator behaved in a manner consistent with the field data.**

Keywords—Disease spread, agent-based model, NetLogo®

I. INTRODUÇÃO

Dengue é uma doença infectocontagiosa causada por quatro sorotipos de vírus DENV-1, DENV-2, DENV-3 e DENV-4 [6]. Segundo Kumar et al. [4] a primeira evidência documentada sobre a dengue vem de uma antiga enciclopédia médica chinesa de 992. No entanto, nos séculos 18 e 19, o mosquito vetor da doença (*Aedes aegypti*) e o vírus da dengue começaram a espalhar para novas regiões geográficas devido ao aumento do volume de comércio e transporte entre os diferentes continentes. A dengue ocorre preferencialmente em regiões tropicais e subtropicais. O vetor responsável por sua transmissão se reproduz em água parada. A principal complicação da dengue é a dengue hemorrágica que tem alto potencial de mortalidade [5]. De acordo com Barreto et al. [1] a dengue se distribui ao longo da faixa do equador à 35 º acima e abaixo desde, conforme Figura 1**.**

Figura 1. Distribuição da dengue em 2007 fonte: [12]

Até o momento não existe vacina eficaz ou um remédio especial contra a dengue. Assim, o controle da doença, atualmente, é realizado combatendo os mosquitos adultos e eliminando os criadouros, de acordo com as diretrizes da OMS (Organização Mundial de Saúde) [2]. A eliminação de criadouros impede o desenvolvimento do inseto nas fases pré-adultas o que resulta na eliminação do transmissor.

É de suma importância que os gestores municipais e estaduais tracem as metas de controle da doença, impedindo que esta se transforme em uma epidemia [2]. A utilização de simulação da epidemia da Dengue pode ser de suma importância para que a tomada de decisão seja o mais eficaz e eficiente possível.

Diante deste contexto, é proposto um simulador baseado em agentes que pode auxiliar no entendimento da proliferação da doença. O modelo busca simular um ambiente real criando situações onde as características relativas à estação climática (temperatura), eliminação de focos e evolução da doença devido a diferentes contágios podem ser parametrizados. Este artigo está organizado da seguinte forma: a seção II apresenta o quadro da dengue no Brasil; na seção III é descrita a proposta do simulador para a propagação da dengue; na seção IV são apresentados os resultados obtidos na simulação computacional e finalmente a seção V traz as considerações finais para este trabalho e propostas para trabalhos futuros.

II. DENGUE NO BRASIL

De acordo com estimativas realizadas pela organização mundial de saúde, anualmente ocorrem 50 milhões de infecções, sendo que destas 500.000 casos de Febre Hemorrágica da Dengue (FHD) e destes vão a óbito cerca de 21.000, principalmente em crianças [7]. Ruas et al.[9] define e explica os estágios que o vetor Dengue pode assumir. São eles: Ovo, Larva, Pupa e Mosquito.

De acordo com Westaway [12], a transmissão da dengue ocorre quando um mosquito fêmea não contaminado pica uma pessoa contaminada, mantém o vírus na saliva e depois do período de incubação de 8 a 12 dias, retransmite a doença. O mosquito costuma picar nas primeiras horas da manhã ou no final da tarde, evitando as altas temperaturas. Existem suspeitas do ataque de alguns durante a noite. Após a ingestão do sangue contaminado, o vírus permanece no mosquito por toda a vida, sendo transmitido a seus descendentes.

Após o contagio, o ser humano apresenta um período de incubação da doença é de 3 a 15 dias. E então aparecem os

primeiros sintomas, que costumam durar de 5 a 6 dias. A única forma de transmissão é através da picada do mosquito contaminado.

A dengue é uma doença que desperta preocupação mundial, e mobiliza a comunidade cientifica para estudar formas de combate e erradicação da doença.

2.1 Cenário da Dengue em Uberaba

Uberaba é um município situado na região do Triângulo Mineiro, no Estado de Minas Gerais. Ela possui uma população de 295.988 habitantes, dos quais 289.376 residem em área urbana [3].

O Levantamento Rápido do Índice de Infestação por *Aedes aegypti* (LIRAa) é uma metodologia utilizada pelos municípios que faz um levantamento dos índices larvários do *Aedes aegypti*. Foi desenvolvido pelo Ministério da Saúde em 2002 para contribuir na disponibilização mais rápida de informações entomológicas geradas pelos gestores e profissionais que trabalham no controle da dengue [8].

De acordo com dados do LIRAa, entre Janeiro - Fevereiro de 2013, Uberaba foi classificada com índice 5,3%, o que a enquadra no grupo de risco. O Informe Epidemiológico divulgado pelo Ministério da Saúde em 22/02/2013 [7], Uberaba é o segundo município do estado de Minas Gerais em número de casos confirmados de Dengue com 3.348 casos. Só fica atrás de Ipatinga com 4.784 casos.

No Gráfico 1, pode-se ver a distribuição trimestral do número de casos da dengue ocorridos em 2010 no município de Uberaba, segundo o resumo informativo de 23/02/2012 divulgado pelo Secretária da Saúde [8] sobre a situação atual da dengue em Minas Gerais. Estes dados foram utilizados como comparativos para os experimentos feitos no simulador.

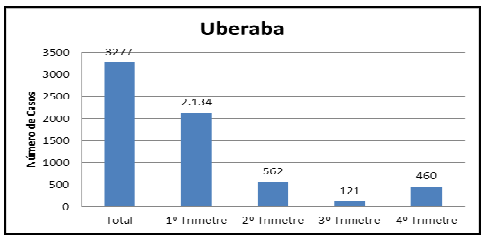

Gráfico 1: Casos de Dengue em Uberaba [11]

III. Modelagem Proposta

Para este trabalho foi modelado um SMA (Sistema Multi-agente) reativo no framework para construção de agentes NetLogo. Originalmente, o Netlogo foi desenvolvido para solucionar problemas relacionados a fenômenos sociais e naturais com alta complexidade e no decorrer do tempo este passou a ser também utilizado para fins educacionais e de pesquisa. [10].

Na modelagem de epidemias faz-se necessário o entendimento do relacionamento das variáveis epidemiológicas da doença em questão. Para tanto, no modelo proposto são criados três agentes distintos: o mosquito, o exterminador e a pessoa. Algumas características são desconsideradas para efeito de simplificação e restrição do escopo. A seguir são descritas as características e justificativas de sua utilização.

Os agentes mosquitos foram modelados apenas como fêmeas já fecundadas e podem assumir 2 estados: infectado, que possui o vírus da dengue, e o não-infectado. Estes agentes transitam pelo ambiente aleatoriamente e buscam as pessoas para se alimentarem. Sempre que encontram uma pessoa, eles podem picá-la dada uma probabilidade informada pelo usuário. Caso o mosquito esteja infectado e pique uma pessoa não infectada, ele contamina esta pessoa. Caso contrário, se ele não está infectado e pica uma pessoa infectada, ele passa para o estado infectado. No ambiente existem pontos que contem água. Casos os mosquitos encontrem esse pontos, e já tenham picado alguma pessoa, eles se reproduzem em proporção a estação do ano corrente, ou seja, a variação da temperatura interfere a reprodução. Os novos mosquitos herdam as características do pai, ou seja, caso este esteja infectado, os gerados também estarão. Estes agentes tem período de vida de 33 dias, ou até serem mortos pelas armadilhas do exterminador.

Os agentes pessoas, também como os mosquitos, assumem estados infectados ou não. Caso eles sejam infectados por algum mosquito, ele passa para o estado infectado que tem duração de 15 dias. Após este período voltam ao estado não contaminado. Existe uma probabilidade deste agente morrer ao se contaminar, se resultar em dengue hemorrágica. Essa probabilidade varia de acordo com o número de vezes que o agente contraiu a doença. Na primeira vez a primeira probabilidade é de 1%. Na segunda vez, de 10%. Na terceira de 15%. E a partir da quarta 25% [1].

O agente exterminador anda pelo ambiente, a procura dos focos de mosquito. Quando ele encontra um foco, ou seja, um mosquito, deixa uma armadilha no local, que mata os mosquitos que passarem por aquele local.

O simulador permite que o usuário forneça entradas para o ambiente de simulação. Essas variáveis são o número de mosquitos infectados, o número de mosquitos não infectados, o número de pessoas infectadas, o número de pessoas sãs, o número de exterminadores, estação (primavera, verão, outono, inverno) corrente, a probabilidade de picada do mosquito e o número de pontos de água parada (criadouro) A duração da simulação, é um período de 90 dias, que foi considerado como o tamanho da estação.

IV. Experimentos e Resultados

No intuito de aferir o simulador, foram utilizados dados do município de Uberaba mostrados no Gráfico 2. Como os dados estavam agrupados em trimestres, considerou-se que cada trimestre seria uma estação para a entrada (Verão, Outono, Inverno e Primavera).

Os experimentos foram divididos em quatro partes, de acordo com as estações. Para cada etapa foi utilizado como entrada 150 agentes pessoas não infectados e 1 agente exterminador. A probabilidade de um agente mosquito picar um agente pessoa, dado que estes estejam num mesmo quadro, é de 80%. As demais entradas tiveram alterações acompanhando as características da realidade.

Como a taxa de pluviosidade varia de acordo com a

estação, sendo maior no verão e menor no inverno, foi considerado que a quantidade de água parada no ambiente seguiria esta proporção. Para o verão o número de quadros de água foram 15, no outono 8, 4 no inverno e 10 na primavera.

A quantidade de mosquito varia de acordo com a temperatura e por isso foi variada nas entradas. No verão foram utilizados como entrada 70 mosquitos não contaminados e 7 contaminados. No outono 50 e 5, 40 e 4 no inverno e 60 e 6 na primavera. Conforme dito na seção 3, a temperatura influencia na reprodução dos vetores, então no verão a taxa de reprodução é a maior e decresce progressivamente até o inverno.

Na Figura 2 observa-se que o número de mosquitos é maior onde se encontra água limpa. Estes mosquitos geralmente estão no mesmo estado, dado que herdam todas as características do pai. Onde há uma concentração de mosquitos infectados, as pessoas ao redor tendem a estarem infectadas. No centro do ambiente não ocorre infecção, pois como se vê, não há nem pessoas e nem mosquitos infectados.

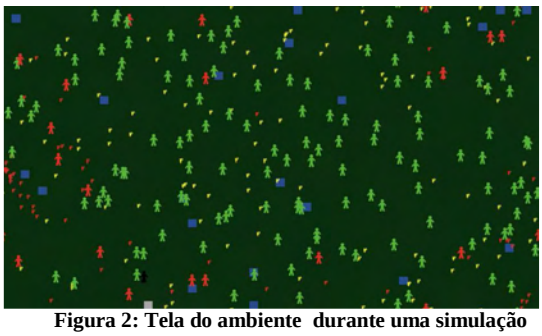

Figura 2: Tela do ambiente durante uma simulação

Para validar o simulador foi realizada uma série de experimentos com os dados acima para cada estação e a média do número de casos de dengue foi utilizado para a geração do Gráfico 2 e comparado com os dados reais ilustrados no Gráfico 3.

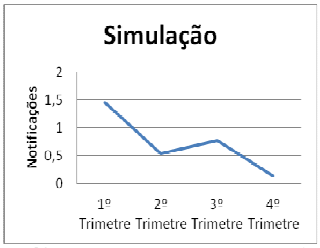

Gráfico 2: Resultado da Simulação

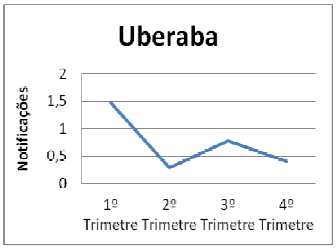

Gráfico 3: Resultados Reais

Como pode observar os resultados da simulação com as entradas acima citadas fornecidas, tiveram um comportamento muito parecido com os dados do número de casos de Uberaba. Isto demonstra que o simulador é capaz de simular a proliferação da dengue.

V. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Simulações baseadas em SMA proporcionam a simulação de situações complexas vivenciadas no mundo real, entretanto, com o beneficio de estar dentro de um ambiente controlado. O que torna as simulações úteis para análises aplicáveis a situações reais com uma boa relação custo benefício.

A dengue é uma doença de risco elevado e altamente complexa. Gerando gastos ao poder público e ocasionando eventuais mortes a população. Diante disto, faz-se necessário a compreensão do comportamento das diversas variáveis envolvidas no processo de disseminação da doença.

Neste cenário, a utilização de um simulador baseado em SMA é útil para o entendimento do processo endêmico da doença. Para isso, o simulador proposto neste trabalho teve por objetivo auxiliar a compreensão dos profissionais envolvidos no controle, prevenção e gestão da dengue.

Os resultados obtidos mostram uma alta similaridade com os dados reais, conforme demonstrado nos Gráficos 5 e 6. Embora, o modelo tenha um escopo reduzido, produziu resultados satisfatórios. Outra vantagem é que o simulador possui uma interface simples e de fácil manejo, podendo ser utilizado pelos gestores públicos no controle da dengue.

Pretende-se estender este modelo, abrangendo dados ainda mais realistas em trabalhos futuros, incluindo variáveis que simulem a variação pluviométrica e geração de água parada. Outra melhoria poderia ser realizada nos agentes exterminadores dando a estes um comportamento mais próximo ao real.

Agradecimentos

Ao CNPQ (Conselho Nacional de Desenvolvimento Científico e Tecnológico).

Referencias

[1] Barreto, Maurício L., and Maria Glória Teixeira. "Dengue no Brasil: situação epidemiológica e contribuições para uma agenda de pesquisa." *estudos avançados* 22.64 (2008): 53-72.

[2] Dengue (2013) Disponível em: http://www.dengue.org.br acessado em 04/03/2013.

[3] IBGE (2013) Disponivel em: http://www.ibge.gov.br/cidadesat/topwindow.htm acessado 04/03/2013.

[4] KUMAR, K., SINGH, P. K., TOMAR J., BAIJAL S. Dengue: epidemiology, prevention and pressing need for vaccine development. Asian Pacific Journal of Tropical Medicine, Volume 3, Issue 12, December 2010, Pages 997-1000.

[5] Maciel, I. J., Siqueira Júnior, J.B., and Martelli, C.M.T. "Epidemiologia e desafios no controle do dengue." *Revista de Patologia Tropical* 37.2 (2008).

[6] Mcbride W. J.H , bielefeldt-ohmann H. Dengue viral infections; pathogenesisand epidemiology. Microbes and Infection, Volume 2, Issue 9, July 2000, Pages 1041-1050.

[7] Ministerio da Saúde (2013) Informe epidemiologico dengue 22_02_13.

[8] Secretária da Saúde (2013) disponível em : http://www.saude.mg.gov.br Acessado em 03/03/2013.

[9] Ruas et. al. An Agent-Based Model for the Spread of the Dengue Fever: A Swarm Platform Simulation Approach. In: Agent-Directed Simulation Symposium of the 2010 Spring Simulation Multiconference. Orlando; 2010.

[10] Tissue S, Wilensky U. (2004) NetLogo: A Simple Environment for Modeling Complexity. In: International Conference on Complex Systems. Boston.

[11] Westaway EG, Brinton MA, Gaidamovich SY, Horzinek MC, Igarashi A, Kaariainen L, Lvov DK, Porterfield JE, Russell PK, Trent DW 1985. Flaviviridae.Intervirology 24: 183-192.

[12] http://gamapserver.who.int/mapLibrary/Files/Maps/World_DengueTransmission_Extension_2007.png acessado em 13/03/2013

Use of High Performance Computing in Agent-Based Social Simulation: A Case Study on Trust-Based Coalition Formation

Luciano M. Rosset, Luis G. Nardin and Jaime S. Sichman
Laboratório de Técnicas Inteligentes – EP/USP
Av. Prof. Luciano Gualberto, 158 – trav. 3
05508-970 – São Paulo – SP – Brasil
{luciano.rosset,luis.nardin}@usp.br, jaime.sichman@poli.usp.br

***Abstract*—Computer models based on agents have shown to be very useful in the field of social simulation, especially for its versatility and ease to model complex systems. In the context of agent simulations, Nardin and Sichman developed a model for the study of coalition formation among agents, based on trust. Later, the need of more efficient ways for simulating the model was acknowledged in order to explore large-scale scenarios. The solution found was the adoption of a high performance agent-based computing platform. This article intends to explore the use of such platform on the Nardin and Sichman's model by migrating its code to the Repast HPC tool, which is executed on the Blue Gene/P supercomputer.**

I. Introduction

In the last decades, computer simulation has proved to be a viable approach for science, in addition to the traditional deductive and inductive approaches [**?**]. According to Banks [**?**], computer simulation consists on the reproduction of real systems through computer models, making the study of these systems' dynamics possible without interfering on them. The application of this approach to the study of social science problems enables the reproduction of social systems behaviours through the use of computational methods, which is called social simulation.

Adequate modelling of social systems is essential for obtaining a useful simulation. Among the available modelling paradigms, agent-based modelling has many adequate abstractions for representing such systems: the main one is the agent abstraction. A comprehensive definition proposed by Ferber [**?**] states that an agent is a physical or virtual entity who (a) is capable of acting in an environment; (b) is capable of communicating with the others; (c) is driven by a set of tendencies (individual goals to achieve or a satisfaction utility to optimize); (d) has its own resources; (e) is capable of perceiving the environment (limited perception); (f) has (eventually) a partial representation of this environment; (g) has competences and offers services; (h) may eventually reproduce itself; and (i) tends to behave in order to satisfy its goals using the available resources and competences and taking into consideration its internal perceptions, representations and the received communication.

According to Davidsson [**?**], the intersection between social simulation and agent-based computing brings forth a new area denoted Agent-Based Social Simulation (ABSS), whose main purpose is to provide models and tools for simulating social phenomena.

In the context of ABSS, Nardin and Sichman proposed an agent-based simulation model, named *Trust and Coalition* (or simply T&C), that integrates the notions of coalition formation and trust in order to enable the analysis of the impacts of trust on the formation of partnerships among autonomous agents [**?**]. This model was implemented using NetLogo [**?**], which is an educational agent modelling and simulating environment. Based on this implementation, several experiments were carried out identifying that the use of trust is relevant for the formation of partnership in fully heterogeneous scenarios in which the agents have high levels of trust intolerance and volatility [**?**].

Although these simulations were successfully performed and provided data for analyses concerning the correlation between trust and coalition formation, there is an uncertainty about the influences that different environment features, such as scale and topology, may cause on such correlation. Thus, in order to analyse such influences, the execution of simulations considering larger populations, such as millions of individuals, is required. However, since the T&C model execution is highly computational demanding, simulations comprising a larger agent population would require an unmanageable amount of time for execution using conventional Agent-Based Modelling and Simulation (ABMS) tools, e.g. NetLogo. For instance, it takes about 1 hour to run a simulation considering 2,500 agents in an Intel i5 2.5 GHz and 4 GB RAM.

High Performance Computing (HPC) is a good path to follow in order to fulfil the needs of large-scale simulations [**?**]. Therefore, in a first step towards an analyses of the T&C model in a large-scale environment, we decided to migrate its previous NetLogo implementation to Repast for High Performance Computing (Repast HPC) [**?**], and this article aims to present some details of such implementation.

The remainder of the article is organized as follows. In section II, we briefly present Repast HPC and its main features, as well as the motivation for selecting this tool. An overview of the *Trust and Coalition* simulation model is presented in Section III and its implementation using Repast HPC is described in Section IV. Since this is a ongoing work, we describe in Section V some of our intended future work.

II. REPAST HPC

Some ABSS models demand great computational capabilities and consume an unmanageable amount of time and memory when simulated monolithically, i.e., the model's simulation is performed using a single process. The single process has not only strict processing restrictions, but may also be overwhelmed by memory needs. In order to overcome the monolithic approach limitations and decrease the simulation execution time, some approaches were proposed. Among these different approaches, the most common ones are (i) *parallel computing* in which one computer composed of several processors execute the simulation in parallel, and (ii) *distributed computing* in which several different computers connected via a network process the simulation in parallel [**?**]. The former is known to be the fastest option as it has a reduced communication overhead, but no software tool capable of executing agent models based on this approach is available. Based on the *distributed* approach, several software tools were proposed lately, such as SWAGES [**?**], FLAME [**?**] and Repast HPC [**?**].

Among these tools, we chose to use Repast HPC in this work because it is the one that presents the greatest advantages concerning flexibility and general use, due to its easy structure based on contexts and projections [**?**]. Moreover, it handles transparently all required inter-process communication and synchronizes agents status in different processes, when needed, in order to optimize cross-process information sharing. Additionally, Repast HPC is the only available, tested and supported platform that runs in a Blue Gene/P supercomputer, which is the target machine for performing our large-scale experiments.

Repast HPC is a cross-platform C++ based environment for large-scale ABMS. It was developed to run large-scale agent models which complexity or number would overwhelm a single process. Its focus is on enabling distributed runs over multiple processes that communicate and share agents using Message Passing Interface (MPI). Each individual process is responsible for executing the behaviours of a subset of all agents in the simulation. Therefore, each process has a scheduler, a context and the projections associated with the context. Repast HPC core components are:

- *AgentId* – A number that uniquely identifies an agent in the simulation. It is composed of the agent's *identity number*, *process rank* and *type*. The *identity number* is a number unique in a specific *process rank*. The *process rank* is the process number the agent is associated to, being unique for each process. The *type* is a number that specifies a class of agents, with the same behaviour.
- *Context* – It is a simple container that groups agents of the same type together; the developer accesses the agents properties through the context.
- *Projection* – It imposes a structure in which the agents are organized. The structure defines relationships among the agents using the semantics of a projection. Three projections are provided: *network*, which consists of a set of nodes and links between them, defining a graph; *grid*, where topological information is included, and agents may occupy a one, two or three-dimensional matrix, that eventually may be wrapped (a 2D wrapped grid works as a torus); and *continuous space*, which basically is a grid which coordinates are floating-points.
- *Scheduler* – It allows agents to schedule events and avoids processes beginning new tasks before other processes end their current ones, which guarantees a consistent simulation execution.
- *Data Collection* – It gathers agents information and write them into files/structures. It allows to produce output files, composed of aggregated or non-aggregated data from all simulation processes.

All simulations in Repast HPC are managed by a set of controllers, one per process, which manages all components such as *Contexts*, *Projections*, *Data Collections* and *Scheduler*. In the beginning of the simulation, each controller creates agents, assigns to each one an *AgentId* and associates them with a *Context*. *Projections* are then created and associated to the *Context* for further positioning of the agents. After this initial setup, the controllers create several events, which trigger actions execution. The sequence of the organization and execution of the events is performed by the *Scheduler*. These events generation and actions execution are performed in steps named *ticks* and the simulation runs until a predefined condition is met or a specified number of ticks is reached. Besides executing the agents behaviours, Repast HPC may also gather useful information during the simulation, through its *Data Collection* feature, writing them into output files.

III. TRUST AND COALITION MODEL

The simulation model used in this work is the *Trust and Coalition* (T&C) proposed by Nardin and Sichman [**?**]. The simulation model proposes a spatial Prisoner Dilemma (PD) game that integrates the notions of coalition formation and trust, which purpose is to enable the analyses of the impacts of trust on the formation of partnerships through coalitions.

The simulation model is composed of an environment represented by a grid and a group of agents, each of them located at one position of the grid. It runs iteratively for a specified number of cycles. At each cycle, each agent interacts with its neighbours (von Neumann or Moore neighbourhood) playing a 2-players PD game, with the following payoff matrix values: *T*emptation=5, *R*eward=3, *P*unishment=1, and *S*ucker=0.

At each cycle, each agent is in either one of the possible states: *independent*, *coalition leader*, or *coalition member*. When *independent*, the agent is not associated to any coalition and chooses between cooperating or not with its neighbours based on its own strategy. A strategy is randomly assigned to the agent at the beginning of the simulation, and there are 3 different possible strategies: Tit-For-Tat (TFT), Probabilistic Tit-For-Tat (pTFT), or Random. When the agent is a *coalition leader* or a *coalition member*, it always cooperates with the neighbours of the same coalition and does not with other neighbours, representing its commitment to the group and its trust on its leader.

Based on the agent's state, its payoff is calculated differently. The *independent* agent payoff is the sum of all the payoffs obtained by playing the PD game against its neighbours.

The *coalition leader* agent payoff is a tax calculated over the sum of all its *coalition members* payoffs. The *coalition member* payoff is defined as follows: (i) the sum of all the *coalition members* payoff is calculated, (ii) the value paid as tax to the *coalition leader* is subtracted from this value, and (iii) each *coalition member* receives the even division of this remaining value by the number of *coalition members*.

After the payoff is calculated, each agent can change its state. The *independent* agent decides to associate to a coalition if its payoff is the smallest amongst all its neighbours. In this case, the agent associates with the coalition with greatest payoff. If both agents are *independent*, then a new coalition is formed and the agent with the greatest payoff becomes the *coalition leader* and the other the *coalition member*. On the other hand, the *coalition member* remains or leaves a coalition based on a *trust* value in its *coalition leader*. If its payoff is not the greatest amongst its neighbours, it decreases the trust in the *coalition leader*, otherwise it increases it. When the *trust* value decreases to a value below a *trust threshold*, then the agent leaves the coalition and becomes *independent*, otherwise it remains in the coalition. The *coalition leader* becomes independent only when its coalition disppears.

IV. Implementation

As the T&C model has only one *Type* of agent, our Repast HPC model uses a single *Context* to hold $n_x \times n_y$ agents. A two-dimensional grid *Projection* is created, and each agent is located in one cell of grid. The grid is then divided into m_x by m_y processes (n_x and n_y must be respectively multiple of m_x and m_y). Since the agents positioned in the border of the grid require to interact with agents running on other processes, there is a component handled by Repast HPC, called *buffer*, that synchronizes this exchange of data. Therefore, each controller accounts for $n_x/m_x \times n_y/m_y$ agents (grid cells, in fact) plus $2(n_x/m_x + n_y/m_y) + 4 \times 2^{(size(buffer)-1)}$ agents taking into account the buffered ones.

The sequence of methods execution divides the events into smaller ones, corresponding for agent's decisions, payoff calculation and coalition management, each of them scheduled to run simultaneously by all processes.

The only limitation we found in Repast HPC was the data collection feature, which does not allow gathering data independently from individual agents, but only by process. Therefore, we will use HDF5[1], since it allows data collection of individual agents similarly to what was done in [**?**].

V. Future Work

Since this is an ongoing project, we are still migrating the model to Repast HPC and no results are currently available. At the moment, we have performed some preliminary tests using Repast HPC and migrated part of the simulation model. However, as soon the migration is completed, we shall perform some further analyses considering:

- *larger populations* – This study has a social approach and one very important point to tackle is the effect of different population sizes. HPC allows us to study the behaviour of populations with the size of entire cities;
- *different topologies* – We want to analyse the effect of narrower rectangular grids, wrapped grids (torus) and other neighbourhoods;
- *more detailed parameter sweep* – Earlier NetLogo simulations were made with a very large variation of the parameters. One example is the *tax* paid by *coalition members* that ranged from 0% to 100% varying by 25%. HPC will allow swifter simulations, therefore making it possible to perform a larger number of simulations.

[1] http://www.hdfgroup.org/HDF5/

Acknowledgements

Luciano M. Rosset and Jaime S. Sichman are partially supported by CNPq/Brazil. We would like to acknowledge the computing time provided on the Blue Gene/P supercomputer supported by the Research Computing Support Group (Rice University) and Laboratório de Computação Científica Avançada (LCCA-CCE, Universidade de São Paulo).

References

[1] R. Axelrod, "Advancing the art of simulation in the social sciences," *Complexity*, vol. 3, no. 2, pp. 16–22, 1997.

[2] J. Banks, Ed., *Handbook of Simulation : Principles, Methodology, Advances, Applications, and Practice*. New York: John Wiley & Sons, 1998.

[3] J. Ferber, *Les Systèmes Multi-Agents: Vers une Intelligence Collective*, ser. Informatique, Intelligence Artificielle. Paris: InterEditions, 1995.

[4] P. Davidsson, "Agent based social simulation: A computer science view," *Journal of Artificial Societies & Social Simulation*, vol. 5, no. 1, p. 7, 2002. [Online]. Available: http://jasss.soc.surrey.ac.uk/5/1/7.html

[5] L. G. Nardin and J. S. Sichman, "Simulating the impact of trust in coalition formation: A preliminary analysis," *Advances in Social Simulation, Post-Proceedings of the Brazilian Workshop on Social Simulation*, pp. 33–40, 2011.

[6] U. Wilensky, *NetLogo*, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, 1999. [Online]. Available: http://ccl.northwestern.edu/netlogo/

[7] L. G. Nardin and J. S. Sichman, "Trust-based coalition formation: A multiagent-based simulation," in *Proceedings of the 4th World Congress on Social Simulation*, Taipei, TW, 2012.

[8] J. T. Murphy, "Computational social science and high performance computing: A case study of a simple model at large scales," in *Proceedings of the 2011 Computational Social Science Society of America Annual Conference*, Santa Fe, 2011. [Online]. Available: http://computationalsocialscience.org/conferences/17-2/csssa-2011-papers

[9] N. Collier and M. North, "Parallel agent-based simulation with repast for high performance computing," *SIMULATION:Transactions of the Society for Modeling and Simulation International*, pp. 1–21, 2012.

[10] R. M. Fujimoto, *Parallel and Distributed Simulation Systems*, 1st ed., ser. Wiley series on parallel and distributed computing. New York: John Wiley & Sons, 2000.

[11] M. Scheutz, P. Schermerhorn, R. Connaughaton, and A. Dingler, "SWAGES: an extendable distributed experimentation system for large-scale agent-based ALife simulations," in *Proceedings of the 10th International Conference on the Simulation and Synthesis of Living Systems*, 2006.

[12] S. Coakley, M. Gheorghe, M. Holcombe, S. Chin, D. Worth, and C. Greenough, "Exploitation of high performance computing in the FLAME agent-based simulation framework," in *High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*, 2012, pp. 538–545.

Using Interest Management to Improve Load Balancing in Distributed Simulations

Felipe C. Bacelar, Carlos J. P. de Lucena
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
Rio de Janeiro, Brasil
{fbacelar, lucena}@inf.puc-rio.br

Pierre Bommel
CIRAD
Unidade GREEN
Montpellier, France
bommel@cirad.fr

***Abstract*— This paper presents an approach to distribute an agent-based simulation over a network of computers. The developed work aims at improving the load balancing of the simulation distribution. In order to reach such objective, we propose to use an Interest Management technique presented by Brian Logan and Georgios Theodoropoulos who proposed to distribute a simulation by dynamically partitioning the environment according to the Interest of the agents. In order to assess its efficiency, we have re-implemented this model using the distribution mechanisms provided by some of the main multi-agent system platforms.**

Keywords—Load Balancing; Interest Management; Spheres of Influence; Distributed Simulation; Multi-agent Systems

I. INTRODUÇÃO

Agentes podem ser entendidos como entidades autônomas localizadas em um ambiente. Eles são capazes de se comunicar entre si e interagir com seu ambiente [13]. Sistemas multiagentes podem ser compostos por conjuntos de agentes que interagem com o propósito de resolver um problema ou alcançar um objetivo. Em certos casos, os agentes colaboram entre si para atingir uma meta em comum. Em outros, os objetivos dos agentes podem ser opostos [11].

O comportamento autônomo dos agentes e sua capacidade de tomada de decisão são fatores que tornam a utilização de sistemas multiagentes muito atrativa para o desenvolvimento de simulações.

Em nosso contexto é possível definir simulação como uma representação do comportamento de um sistema, real ou imaginário, através do tempo [5]. O emprego de simulações tem sido uma forma muito eficaz de estudar e analisar problemas reais. A possibilidade de modelar diferentes cenários para o mesmo problema e reproduzi-los diversas vezes permite a identificação de situações incomuns e comportamentos inesperados.

Como os agentes costumam ser unidades complexas de software, manter um número elevado deles em uma simulação pode ser muito custoso. Simulações de larga escala, com centenas ou milhares de agentes, tendem a ter seu desempenho comprometido.

Uma solução para este problema é distribuir a simulação entre diversos computadores. Contudo, simplesmente dividir a simulação em partes e executar cada uma delas em uma máquina diferente pode não ser uma boa solução. Entre outros problemas, é preciso considerar o balanceamento de carga. Balanceamento de carga é uma técnica que visa dividir coerentemente a carga de trabalho entre computadores de uma rede, maximizando o desempenho e evitando sobrecarga.

O presente trabalho, referente a uma pesquisa em andamento, propõe uma abordagem para melhorar o balanceamento de carga de uma simulação distribuída explorando o conceito de esferas de influência como estratégia de gerenciamento de interesse dinâmico.

II. GERENCIAMENTO DE INTERESSE

Em simulações distribuídas de larga escala a comunicação do tipo *broadcast* deve ser evitada para reduzir o custo de comunicação entre as máquinas da rede. Uma proposta para resolver este problema é conhecida como Gerenciamento de Interesse [14]. O princípio deste modelo é baseado na idéia de que as entidades raramente usam todas as informações disponíveis, mas elas podem manifestar interesse em apenas um subconjunto de informações que são relevante para elas. Um agente deve ser provido de acesso apenas ao conjunto de elementos com os quais poderá interagir (ler/atualizar). Contudo, esse conjunto pode mudar com o tempo. Ao se mover pelo ambiente, um agente pode desviar de um obstáculo e passar a ter um campo de visão maior, aumentando o número de variáveis que poderá ler, por exemplo. O mecanismo de Gerenciamento de Interesse deve ser capaz de se adaptar a estas mudanças [6], [7], [9], [12].

A. *Esferas de Influência*

Com o intuito de solucionar o problema do gerenciamento de interesse dinâmico, Logan e Theodoropoulos criaram o conceito de Esferas de Influência [6], [7], [9], [12]. O modelo da simulação é divido em conjuntos de Processos Lógicos (LP na sigla em inglês) concorrentes, cada um mantendo uma parte disjunta do espaço do sistema. Os LPs podem ser processos lógicos de agentes ou processos lógicos de ambiente [6], [7], [9]. Todos os LPs de uma simulação geram um número limitado de tipos de eventos. Diferentes tipos de eventos geram diferentes efeitos sobre o estado da simulação.

A esfera de influência de um evento pode ser entendida como “o conjunto de variáveis lidas ou atualizadas em decorrência do evento”. A esfera de influência de um LP em

um dado intervalo de tempo é a união das esferas de influência dos eventos gerados por esse LP no intervalo. A interseção das esferas de influência dos LPs gera uma ordenação parcial das variáveis de estado em que os primeiros elementos são os acessados pela maior quantidade de LPs e os últimos são acessados pela menor quantidade [9]. Esta ordenação parcial pode ser utilizada para decompor o estado da simulação de forma a manter um determinado agente no mesmo nó da plataforma distribuída em que se encontram os elementos com os quais ele mais interage na simulação. Isto garante uma redução no custo de comunicação entre as máquinas.

Em [6], [7], [9] e [12] o estado da simulação pode ser decomposto usando um novo conjunto de processos lógicos chamados *Communication Logical Processes* (CLP). Cada CLP armazenará um subconjunto do estado da simulação. Inicialmente, toda a simulação é de responsabilidade de um único CLP. Com o progresso da simulação, o CLP realiza o cálculo aproximado das esferas de influência dos agentes. Se o CLP ficar sobrecarregado (atingir um determinado limite de tráfego de rede, por exemplo), um novo CLP é criado e um subconjunto disjunto do estado da simulação é atribuído a ele, normalmente os últimos elementos da ordenação parcial gerada pela interseção das esferas de influência dos LPs. O processo é então repetido para o novo CLP, monitorando a carga e criando novos CLPs caso necessário.

Segundo Logan e Theodoropoulos, o custo computacional de calcular as esferas de influência é muito alto. Qualquer implementação eficiente pode apenas aproximar o resultado ideal [9]. A abordagem apresentada pelos autores foi implementada em um framework chamado PDES-MAS [10].

III. Abordagem Proposta

No framework PDES-MAS os CLPs são organizados naturalmente em uma árvore e cada CLP possui informação de roteamento que indica quais tipos de eventos são relevantes para seu CLP pai e para seus filhos (LPs e outros CLPs) [9], [10].

Para simplificar a implementação e adaptação a qualquer plataforma distribuída de sistemas multiagentes, o trabalho sugere evitar o uso de CLPs.

Muitas plataformas de sistemas multiagentes podem ser distribuídas entre máquinas de uma rede local ou até mesmo pela Internet. Em geral, a plataforma mantém informações sobre todos os nós conectados e é capaz de gerenciar a comunicação entre eles.

O presente trabalho propõe utilizar o conceito de esferas de influência para decompor o estado de uma simulação e distribuí-la entre os nós de uma plataforma que forneça recursos de distribuição. Cada nó assumirá o papel de um CLP passando a conter uma parcela do estado da simulação. No entanto, o nó não é responsável por guardar informações de roteamento, deixando para a plataforma o gerenciamento da comunicação.

Com a finalidade de evitar gargalos na simulação, são estabelecidos limites que, uma vez atingidos, classificam um nó da plataforma como sobrecarregado. Alguns critérios úteis são: uso de CPU, uso de memória e tráfego de rede.

A simulação é iniciada no hospedeiro principal da plataforma e quando um dos limites estabelecidos é alcançado, o processo de distribuição é disparado. Utilizando a ordenação obtida através das esferas de influência dos LPs, uma parte do estado da simulação é migrada do hospedeiro principal para um nó disponível na plataforma. Se o nó em questão ficar sobrecarregado, a migração continua para o próximo nó disponível. O processo é interrompido quando não há mais nós sobrecarregados, incluindo o hospedeiro principal, ou quando não há mais máquinas disponíveis na plataforma.

Durante a execução da simulação, se qualquer nó da plataforma ficar sobrecarregado, o processo de migração é disparado novamente.

IV. O Modelo da Simulação

Para ilustrar o método proposto, está em desenvolvimento uma simulação simples representando um cenário de derramamento de petróleo. Este tipo de desastre é comum e pode causar diversos tipos de impactos ao meio-ambiente. Entre os mais frenquentes é possível listar: envenenamento de peixes, bloqueio da luz solar impedindo a fotossíntese das algas e morte por afogamento de pássaros que tiveram suas penas cobertas de petróleo (ficando incapazes de voar).

O modelo da simulação é composto por um pequeno conjunto de classes de agentes. Há uma classe para representar os barcos recolhedores, uma classe para manchas de petróleo e uma classe representando células de mar. O ambiente é apresentado em um *grid* composto por células de mar.

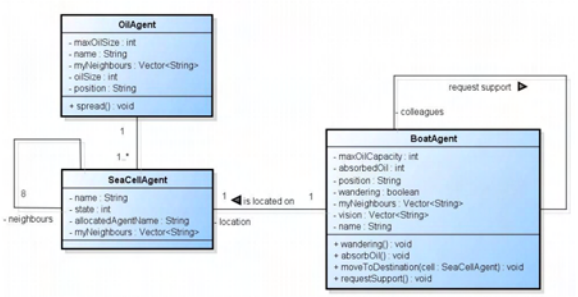

Fig. 1. Diagrama de classes do modelo de derramamento de petróleo

Como pode ser visto no diagrama apresentado na figura 1, o agente de mancha possui apenas o comportamento de se espalhar. O agente barco tem o comportamento de vaguear pelo ambiente até que uma mancha de petróleo entre em seu campo de visão. Ao avistar uma mancha, o agente barco percorre o caminho até ela e a absorve assim que a alcança. Se a capacidade do barco não for suficiente para absorver todo o petróleo, outro barco recolhedor é solicitado.

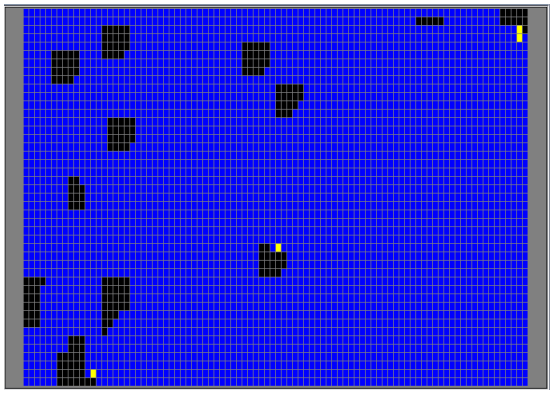

Fig. 2. Execução da simulação em desenvolvimento

A figura 2 apresenta a simulação em desenvolvimento. As células azuis do *grid* representam as células do mar. As células em preto, por sua vez, são representações das manchas de petróleo. Por último, as células em amarelo são os barcos recolhedores.

Para exibir graficamente a interação entre os agentes foi utilizado o *framework* "Agent. GUI". O *framework* provê uma interface gráfica pré-definida que pode ser adaptada às necessidades do desenvolvedor. É completamente desenvolvido em Java e baseado no *framework* JADE [4].

JADE é uma sigla para Java Agent DEvelopment Framework. Trata-se de um *framework* desenvolvido em Java que fornece recursos para implementação de sistemas multiagentes [1]. Sua plataforma de agentes é usada no projeto desenvolvido no presente trabalho, pois possui todos os recursos de distribuição necessários [1], [2], [3].

V. Considerações Finais

O presente trabalho é referente a um projeto de pesquisa em andamento que visa melhorar o balanceamento de carga ao distribuir simulações. Para isto, propõe a utilização de uma técnica de gerenciamento de interesse que busca manter cada agente no mesmo nó da rede que se encontram as partes da simulação com as quais ele mais interage.

Uma implementação de uma simulação em uma plataforma distribuída utilizando o método proposto está em desenvolvimento. Outras técnicas de gerenciamento de interesse estão sendo analisadas para eventualmente atuarem em conjunto com o conceito de esferas de influência na solução final do projeto. Além disto, uma estratégia para selecionar o melhor nó disponível da plataforma (máquina mais potente ou com melhor tempo de resposta) deve ser discutida.

References

[1] Bellifemine, F.; Bergenti, F.; Caire, G.; Poggi, A., JADE - A Java Agent Development Framework. In Proceedings of Multi-Agent Programming, p. 125-147, 2005.

[2] Bellifemine, F.; Caire, G.; Trucco, T.; Rimassa, G. JADE Programmer's Guide. Available at: http://jade.tilab.com/doc/programmersguide.pdf Access: 10 dec. 2012

[3] Caire, G.; Rimassa, G.; Bellifemine, F. JADE: a versatile run-time for distributed applications on mobile terminals and networks. In Proceedings of SMC (2), p. 1882-1888, 2004.

[4] Derksen, C.; Branki, C.; Unland, R. Agent.GUI: A Multi-agent Based Simulation Framework. In Proceedings of FedCSIS, p. 623-630, 2011.

[5] Fujimoto, Richard M. Parallel and Distributed Simulation Systems. New York: Wiley Interscience, 2000. 300 p.

[6] Lees, M.; Logan, B.; Minson, R.; Oguara, T.; Theodoropoulos, G. Distributed Simulation of MAS. In Proceedings of the Joint Workshop on Multi-Agent and Multi-Agent-Based Simulation (MAMABS'04), p. 21-30, 2004.

[7] Lees, M.; Logan, B.; Minson, R.; Oguara, T.; Theodoropoulos, G. Modelling Environments for Distributed Simulation. In 1st International Workshop on Environments for Multi-Agent Systems (E4MAS), in conjunction with the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS04), p. 150-167, 2004.

[8] Lees, M.; Logan, B.; Theodoropoulos, G. 2007. Distributed Simulation of Agent-based Systems in HLA, ACM Transactions on Modelling and Computer Simulation, Vol. 17, 3, Available at: http://doi.acm.org/10.1145/1243991.1243992) ISSN: 1049-3301.

[9] Logan, B.; Theodoropoulos, G. The distributed simulation of multi-agent systems. In Proceedings of the IEEE 89(2), p. 174-186, 2001.

[10] Oguara, T.; Chen, D.; Theodoropoulos, G.; Logan, B., and Less, M. An Adaptive Load Management Mechanism for Distributed Simula-tion of Multi-agent Systems. Proceedings of the Ninth IEEE International Workshop on Distributed Simulation and Real-Time Applica-tions, pp. 179–186. October 2005.

[11] Parunak, H. V. D.; Brueckner, S.; Fleischer, M. et Odell, J., 2003. A Design Taxonomy of Multi-Agent Interactions. Pages 123-137 of : Paolo Giorgini, Jörg P. Müller, James Odell (ed), Agent-Oriented Software Engineering IV : 4th International Workshop, AOSE 2003, Melbourne, Australia, July 15, 2003, Revised Papers. Lecture notes in computer science LNCS, vol. 2935. Springer.

[12] Theodoropoulos, G; Logan, B. An Approach to Interest Management and Dynamic Load Balancing in Distributed Simulation. In Proceedings of the 2001 European Simulation Interoperability Workshop, p. 565-571, 2001.

[13] Wooldridge, M.; Jennings, N.R., Intelligent Agents: Theory and Practice. Knowledge Engineering Review, 10(2), p. 115-152, 199

[14] Morse, K. L.; Interest Management in Large-Scale Distributed Simulation; Volume 96, Issue 27 of Technical report (University of California, Irvine. Dept. of Information and Computer Science), 1996

Simulation and Analysis of Malaria Using Multiagent Systems

Laurence N. da S. Costa, Diana Francisca Adamatti

***Abstract*—Malaria is a disease that affects hundreds of millions of people globally and causes about 1.2 million deaths every year. Just in Brazil, there is about three hundred thousand cases of malaria per year. It's a very serious problem of public health in the countries which the disease is endemic. This paper proposes the creation of a computational model of malaria, based on Multiagent Systems (MAS), which covers aspects such as infection, mortality, length of incubation and prevention. The choice to use MAS and Netlogo allowed the creation of a system which has simpler implementation, but highly configurable and, with the public health experts help and successive refinements, it can deliver results more and more reliable.**

***Index Terms*—Malaria, Multiagent Systems, Simulation, NetLogo**

I. Introdução

A malária é uma das doenças mais letais do mundo e causa grandes prejuízos econômicos e sociais nas regiões de risco [1]. No Brasil, a grande maioria dos casos concentra-se na região Norte, mais precisamente na região da Amazônia Legal, a qual engloba nove estados brasileiros: Acre, Amapá, Amazonas, Mato Grosso, Pará, Rondônia, Roraima e Tocantins. A malária é uma doença causada por protozoários do gênero plasmódio, transmitidos por um mosquito do gênero anofelino [2][3][4]. Quando a fêmea do mosquito pica uma pessoa a fim de obter sangue, inocula uma saliva anticoagulante. É através dessa saliva que os protozoários invadem o corpo do hospedeiro humano. Não existe vacina, sendo a prevenção a melhor maneira de combater a doença.

Do ponto de vista computacional, um agente é uma entidade dotada de capacidade de autonomia, podendo tomar decisões e escolher a melhor maneira de atingir seus objetivos. Os agentes são capazes de analisar uma situação, gerar alternativas e escolher a que melhor atenda seus objetivos, e serem capazes de interagir com outros agentes computacionais para obtenção de suas metas [5].

Sistemas Multiagentes (SMA) são compostos por agentes que, inseridos em um ambiente, interagem uns com os outros, a fim de satisfazer um objetivo ou conjunto de objetivos. Os agentes inseridos nesse ambiente possuem características distintas de capacidade e percepção do ambiente. Em um SMA, os agentes podem trabalhar em conjunto para atingir objetivos gerais, ou então terem objetivos individuais, mas que precisam da interação de outros agentes para completá-los.

Embora haja estudos sobre epidemias com sistemas multiagentes [6], a quantidade de material sobre simulação da Malária em português é escassa [7][8], e não utiliza a mesma ferramenta utilizada por esse trabalho, o Netlogo. Existem trabalhos que fazem modelagem estatística [9], contudo, utilizando-se a abordagem de sistemas multiagentes, é possível criar uma simulação bastante flexível, pois torna-se possível modificar vários parâmetros do cenário, tais como taxa de contágio e índice de mortalidade.

Este artigo está organizado em 3 seções: a seção II mostra o modelo computacional da malária; a seção III apresenta os resultados dos testes, e a seção IV conclui o artigo e apresenta propostas para trabalhos futuros.

II. Modelo Computacional

No modelo apresentado, cada retângulo representa um tipo de agente, e as setas representam as interações que ocorrem entre diferentes tipos. Por exemplo, a interação entre um agente *mosquito sadio* com um *Homem doente P. Vivax* resulta em um terceiro tipo de agente, o *Mosquito transmissor P. Vivax*. Essa relação simula um mosquito sadio que, ao picar um indivíduo com malária, é infectado com o plasmódio da mesma espécie que infectou a pessoa picada pelo mosquito.

Na Figura 1 é mostrado o modelo completo, que representa as possibilidades viáveis. Em seguida são mostrados os diferentes ciclos que o mosquito, o plasmódio e o ser humano podem estar inseridos.

Todos os ciclos partem do princípio que o mosquito está inicialmente sadio, e contrai as formas infectantes do plasmódio através de indivíduos doentes.

A. Ciclos do Modelo

O modelo desenvolvido é composto de vários ciclos.

Ciclo 1

Passos: 1 → 2 → 3 → 4a

O mosquito sadio pica um indivíduo infectado (1) e torna-se vetor da malária (2). Os vetores picam indivíduos sadios (3) e estes contraem a doença (4a).

Ciclo 2

Passos: 1 → 2 → 3 → 4a → 4b

Semelhante ao ciclo 1. O mosquito sadio pica um indivíduo infectado (1), torna-se vetor da malária (2), estes picam indivíduos sadios (3), os quais ficam doentes (4a). Entretanto, conseguem se curar (4b).

Ciclo 3

Passos: 1 → 2 → 5a → 6

O mosquito sadio pica um indivíduo infectado (1), torna-se vetor da malária (2) e estes picam indivíduos curados da malária (5a). Adoecem novamente (6).

Ciclo 4

Passos: 1 → 2 → 3 → 4a → 4b → 5b

Semelhante ao ciclo 2. O mosquito sadio pica um indivíduo infectado (1) e torna-se vetor da malária (2). Estes picam indivíduos sadios (3), os quais

ficam doentes (4a), mas conseguem se curar (4b). Entretanto, plasmódios latentes nas células do fígado ficam ativos, e o indivíduo contrai novamente a malária (5b). [1]

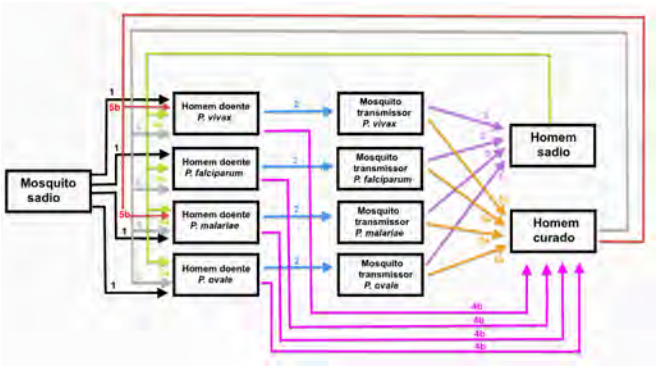

Figura 1. Modelo computacional completo

III. Resultados

Foi elaborado um cenário com muitas pessoas doentes, poucas sadias e muitos mosquitos, chamado de "caso extremo". Os testes foram executados sem medida de prevenção e com medida de prevenção.

Foram executados 10 iterações para os dois tipos de teste, todos com os mesmos dados iniciais. A tabela II mostra as taxas de contágio de cada plasmódio, a Tabela I exibe as taxas de mortalidade e a Tabela III mostra a configuração inicial dos agentes.

Tabela I
Taxas de Mortalidade

Plasmódio	taxa de mortalidade
P. vivax	10%
P. falciparum	80%
P. malariae	1%
P. ovale	1%

Tabela II
Taxas de Contágio

Plasmódio	taxa de contágio
P. vivax	90%
P. falciparum	13%
P. malariae	4%
P. ovale	3%

Tabela III
Dados Iniciais do Caso Extremo

inicial pessoas	300
inicial mosquitos	80
inicial vivax	70
inicial falciparum	70
inicial ovale	70
inicial malariae	70

[1] A recaída ocorre apenas para as espécies *P. Vivax* e *P. malariae*.

A. Resultados sem medida de prevenção

Após a execução de 10 simulações, a Tabela IV exibe a média dos resultados.

Tabela IV
Resultados do Caso Extremo

doentes vivax	139,7
doentes falciparum	0
doentes ovale	0
doentes malariae	0
mortes	79,8
curados	7.409,6
recaídas vivax	0
recaídas malariae	0

A quantidade de pessoas contaminadas com o plasmódio *P. vivax* é de 139,7, sendo que muitas dessas pessoas podem já ter contraído a doença várias vezes. Os contaminados pelas outras espécies de plasmódio são zero ou muito próximo disso; a possível causa é que os mosquitos transmissores desses plasmódios morreram antes de conseguirem transmitir o protozoário. No caso do *P. falciparum*, as pessoas contaminadas com esse plasmódio devem ter morrido antes do mosquito ter uma chance de picá-las.

A média de cura é de 7.409,6 pessoas, sua imensa maioria eram doentes de *P. vivax*. Esse número mostra que, embora a taxa de contágio seja muito alta, as chances de cura também são grandes, embora o risco de morte não seja zero. Na simulação, as pessoas recuperam-se totalmente (passam de doentes para sadios), mas ela não mostra as sequelas que esse plasmódio pode vir a deixar na pessoa a qual ele infectou.

B. Resultados com medida de prevenção

A medida de prevenção trata-se de um agente que se move aleatoriamente pelo ambiente e mata todo mosquito que estiver ao redor dele, seja sadio ou transmissor. A ideia desse agente de prevenção é simular um dedetizador que pulveriza inseticida nas áreas percorridas por ele.

Tanto os dados iniciais quanto as taxas de contágio dos testes com medida de prevenção são os mesmos dos sem a medida de prevenção, apenas foi acrescentado o agente que vai diminuir a população de mosquitos no ambiente. Também foram executadas 10 simulações para este caso. A média dos resultados está ilustrado na Tabela V.

Tabela V
Resultados do Caso Extremo com Prevenção

doentes vivax	1,4
doentes falciparum	0
doentes ovale	0
doentes malariae	0
mortes	81
curados	1.140,9
recaídas vivax	59,7
recaídas malariae	0

O acréscimo de um agente de prevenção trouxe mudanças significativas na quantidade de doentes e na de curas. Mesmo

com um pequeno aumento de óbitos, o número de infectados com o plasmódio *P. vivax* diminuiu drasticamente. A quantidade de pessoas curadas pode ser considerada elevada à primeira vista, mas é bem menor que os apresentados pelos testes sem prevenção.

IV. Conclusões e Trabalhos Futuros

O imenso número de casos registrados todos os anos, principalmente em regiões com Índice de Desenvolvimento Humano (IDH) baixo, torna a Malária um dos principais problemas de saúde pública no mundo.

Este trabalho apresentou um modelo computacional para a doença da Malária, apresentando os principais ciclos existentes. Na literatura pesquisada, um modelo assim não foi encontrado. Desta forma, acredita-se que este modelo é uma contribuição do trabalho realizado.

Os sistemas multiagentes e o Netlogo proporcionaram o desenvolvimento de um sistema muito versátil, sendo possível realizar simulações com os mais variados dados de entrada, forneceram recursos para construir interfaces limpas e permitem que refinamentos posteriores possam ser feitos com relativa facilidade. O sistema desenvolvido fornece resultados iniciais, mas já mostra como a dinâmica da doença funciona e como medidas de prevenção podem ser eficazes, além de contribuir com um modelo para a ferramenta Netlogo.

Como trabalhos futuros, vislumbra-se a implementação de outras medidas de prevenção, bem como permitir ao usuário maior controle dessas medidas através da interface. Também acredita-se que a avaliação mais rigorosa do modelo por especialistas em saúde pública seja uma próxima etapa, tornando o modelo desenvolvido mais fidedigno a realidade.

Referências

[1] "Malaria," http://portal.saude.gov.br/portal/saude/profissional/area.fm?id_area=1526, 2012, acesso em: maio de 2012.

[2] D. Varella, "Malaria," http://drauziovarella.com.br/doencas-e-sintomas/malaria/, 2012, acesso em: maio de 2012.

[3] "Malária: Sintomas, tratamento e prevenção," http://www.brasilescola.com/doencas/malaria.htm, 2012, acesso em: maio de 2012.

[4] C. F. D. Control and Prevention, "Anopheles mosquitoes," http://www.cdc.gov/malaria/about/biology/mosquitoes/, 2012, acesso em: maio de 2012.

[5] S. O. Rezende, *Sistemas Inteligentes - Fundamentos e Aplicações*. Manole, 2003.

[6] C. N. da Fonseca, "Um modelo baseado em agentes para simulação experimental de mecanismos de controle da disseminação da dengue," Mestrado em Modelagem Computacional, Universidade Federal do Rio Grande, 2012.

[7] A. P. B. da Silva, W. P. Tadei, and J. M. M. dos Santos, "Variabilidade genética em populações de anopheles darlingi (diptera: Culicidae) e relação ao comportamento da atividade de picar, analisada por rapd," *Acta Amazônia*, vol. 40, no. 3, pp. 585–590, 2010.

[8] W. P. Tadei, J. M. M. dos Santos, W. L. de Souza Costa, and V. M. Scarpassa, "Biologia de anofelinos amazônicos," *Revista do Instituto de Medicina Tropical de São Paulo*, vol. 30, no. 3, pp. 221–251, Maio-Junho 1988.

[9] F. T. M. Costa, "On the pathogenesis of plasmodium vivax malaria: Perspective from the brazilian field," *International Journal of Parasitology*, vol. 1, no. 1, 2012.

[10] A. P. Gomes, R. R. Vitorino, A. de Pina Costa, E. G. de Mendonça, M. G. de Almeida Oliveira, and R. Siqueira-Batista, "Malária grave por plasmodium falciparum," *Revista Brasileira de Terapia Intensiva*, vol. 23, no. 3, pp. 358–369, 2011.

[11] WHO, "World malaria report 2011," World Health Organization, Tech. Rep., 2011.

[12] SVS, "Sivep - malária," http://portal.saude.gov.br/portal/arquivos/pdf/boletim_malaria_2010_2011.pdf, 2011, acesso em: maio de 2012.

[13] ——, "Manual de diagnóstico laboratorial da malária," Ministério da Saúde, Tech. Rep., 2005.

[14] L. P. Reis, "Coordenação em sistemas multi-agente: Aplicações na gestão universitária e futebol robótico," Ph.D. dissertation, Faculdade de Engenharia da Universidade do Porto, Julho 2003.

[15] J. S. Garcia, A. C. B.; Sichman, *Sistemas Inteligentes - Fundamentos e Aplicações*. Manole, 2003, ch. Agentes e Sistemas Multiagentes, pp. 269–306.

[16] U. Wilensky, "Netlogo," http://ccl.northwestern.edu/netlogo/, 1999.

Agent-Based Simulation to a Decision Support System to Pollutant Dispersion

Narúsci dos S. Bastos[1], Bianca P. Marques[1], Diana F. Adamatti[1]

[1] Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Caixa Postal 474 – 96.201-900 – Rio Grande – RS – Brasil

Abstract .- **This paper presents a decision support system (DSS) prototype that uses Agent-Based Simulation adjusted to a simulated pollutant dispersion. The goal is to assist in choosing a suitable location for the construction of new industries through computational simulation tools that make predicting risks that may occur in the chosen region. The DSS allows the good choice of a location for industry to avoid the pollution that directly affects the population.**

Keywords—agent-based simulation; decision support system; pollutant dispersion

I. INTRODUÇÃO

Este artigo apresenta o modelo de um sistema de apoio à decisão (SAD) que utiliza Simulação baseada em Agentes ajustado a um simulador de dispersão de poluentes. O objetivo é auxiliar na escolha de um local apropriado para a construção de novas indústrias, através de ferramentas de simulação computacionais que fazem a previsão de riscos, que podem ocorrer na região escolhida.

Com a evolução industrial e o aumento populacional essas indústrias acabam atingindo a população com a remessa de poluentes lançados a atmosfera, e o SAD permite que ajudemos a indústria, escolhendo um lugar que tem o melhor escore do calculo feito pela formula, para que evite esta poluição ligada diretamente com a população

II. SISTEMA DE APOIO À DECISÃO

Um Sistema de Apoio à Decisão é um modelo genérico de tomada de decisão que analisa um grande número de variáveis para que seja possível o posicionamento a uma determinada questão. Segundo Turban et. al. [2], um Sistema de Apoio a Decisão (SAD) é um sistema de informação baseado em computador que combina modelos e dados em uma tentativa de resolver problemas semiestruturados e alguns não estruturados com intenso envolvimento do usuário.

A ideia desse trabalho é desenvolver um SAD baseado em Sistemas Multiagentes, para que a tomada de decisão seja da forma ambientalmente mais correta.

Com o crescimento populacional e industrial tecnológico, necessita-se também do crescimento da preservação ambiental, pois com o crescimento industrial, a poluição também aumenta causando danos ambientais e para a saúde da população. A simulação computacional é utilizada para prever essas consequências futuras auxiliando na tomada de decisão, avaliando os riscos que serão causados, sem perturbar esta região que está sendo avaliada. Este simulador escolhe o local que tem o melhor escore do cálculo feito através da formula, para a inserção de uma nova indústria, tendo em vista que a poluição não atinja diretamente a população, que é representada pelos multiagentes na simulação. O simulador foi subdividido em quatro módulos: manipulação de mapas, inserção de novos objetos, propagação da poluição e SAD.

A. Funcionamento do SAD

A Figura 2 ilustra o funcionamento do SAD. O sistema segue os seguintes passos:

1. Escolha dos possíveis locais para inserção da indústria;
2. Definição dos parâmetros da indústria que são passados ao simulador;
3. Início do processamento do simulador;
4. Armazenamento dos dados que serão acessados pelos agentes;
5. O coordenador acessa os dados armazenados;
6. O coordenador repassa as informações para os demais agentes participantes;
7. Cada agente processa sua avaliação individual;
8. O agente decide o local a ser inserida a indústria de acordo com seu perfil;
9. Cada agente informa sua decisão ao coordenador;
10. O coordenador contabiliza o número de votos;
11. O coordenador repassa a decisão final para o simulador;
12. O simulador gera um mapa final com a localização da nova indústria;
13. O mapa é apresentado ao usuário.

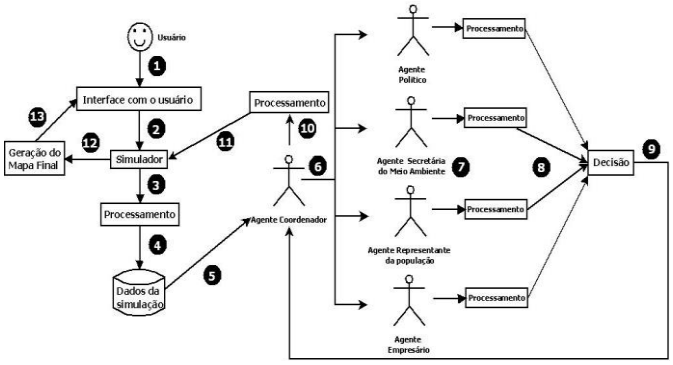

Figura1. Fluxograma do SAD

A ideia desse SAD foi apresentada inicialmente por Nunes et. al. [4].

B. Agentes e seus critérios de avaliação

Para a tomada de decisão do local a ser inserida a nova indústria, os agentes baseiam-se em um cálculo de avaliação representado pela Figura 2, proposto por Nunes et. al. [4]. Nesta fórmula, é feito um somatório ponderado do critério de avaliação multiplicado pela taxa atribuída ao mesmo. Após este cálculo, a localização da indústria que possui a maior avaliação é escolhida pelo agente que possui o melhor escore do calculo feito pela formula.

```
AVALIAÇÃO = ( (Custo X Tx_Custo) + (Tempo de Contrução X Tx_TempoConst) +
              (Impacto ambiental X Tx_ImpAmbieltal) +
              (Opinião Pública X Tx_opiniãoPública) +
              (Geraçaõ de empregos X TX_geraçãoEmpregos) +
              (Lucros X Tx_lucros) +
              (Impostos X Tx_impostos)
            )
```

Figura2.Cálculo de avaliação

Foram criados os seguintes critérios de avaliação com os respectivos pesos:

- **Político:** custo: 0.1, tempo de construção: 0.8 , impacto ambiental: 0.6 , opinião publica: 1, geração de empregos: 1, lucros: 0.5 , impostos: 1.
- **Empresário:** custo: 1, tempo de construção: 1, impacto ambiental: 0.3, opinião publica: 0.2, geração de empregos: 0.5 , lucros: 0.6, impostos: 0.7.
- **Secretaria do meio ambiente:** custo 0.7, tempo de construção 1, impacto ambiental 1, opinião publica 0.5, geração de empregos 0.5, lucros 0.6, impostos 0.7
- **Representante da população:** custo: 0.5 , tempo de construção: 1, impacto ambiental: 0.7, opinião publica: 1, geração de empregos: 1, lucros: 0.1, impostos: 0.7

Para fazer o cálculo, nesta pesquisa, usa-se para os valores das taxas: Tx_custo, Tx_TempoConst, Tx_ImpAmbiental, Tx_opiniãoPública, Tx_geraçãoEmpregos, Tx_lucros e Tx_impostos, os valores correspondentes de cada taxa são randômicos e variam de 0 a 100.

C. Ferramenta NetLogo

O Netlogo é uma ferramenta que possui uma linguagem de programação simples, sendo este voltado para modelagem e simulação de fenômenos naturais e/ou sociais [4][5].

Essa ferramenta é especialmente adequada para modelar sistemas complexos, que evoluem ao longo do tempo. Os desenvolvedores podem dar instruções a dezenas, centenas ou milhares de agentes, que funcionam de forma independente, interagindo entre si e com o ambiente. Tornando-se assim possível de explorar a ligação entre o comportamento dos indivíduos locais e padrões macroscópico que surgem através de suas interações. Suas simulações são particularmente endereçadas a áreas de conteúdos como ciências naturais e sociais, incluindo a biologia, medicina, física, química, matemática, ciência da computação, ciência econômica e psicologia social.

D. Implementação com a ferramenta NetLogo

Os agentes a serem implementados no SAD são: os políticos, os secretários do meio ambiente, a população, os empresários e as indústrias, criou-se as variáveis intrínsecas a cada um dos agentes, variáveis estas que são os critérios de avaliação, mencionados na seção *B*, Após serem definidas as características dos *agentes* implementou-se o cálculo de avaliação, dentro da variável *eat*, como pode-se visualizar na linha1, em que permite realizar as funções especificadas a cada "*tick*", iteração. Entre as linhas 2 e 8 atribui-se o random para determinar os valores dos pesos, aleatoriamente. Seguindo o algoritmo descrito na linha 9 logo abaixo, a cada iteração, a variável *set P_total* recebe o valor do somatório ponderado do critério multiplicado pelo peso atribuído ao mesmo.

```
 1.  to politician-eat
 2. set custo random 100
 3. set tempo random 100
 4. set impacto random 100
 5. set op random 100
 6. set emprego random 100
 7. set lucro random 100
 8. set impostos random 100
 9. set P_total (P_cust * custo)+ (P_temp * tempo)+
(P_impact * impacto)+(P_op * op)+
 10.     (P_emp * emprego)+ (P_luc * lucro)+ (P_imp
* impostos)
 11. set P_avaliacao (P_total)
 12. end
```

Para a apresentação do resultado final, inseriu-se dentro do bloco *go*, que recebe o comando de execução das iterações enviadas pelo *Button Go* ou *Go once*, os agentes criados com a função *eat*, logo uma estrutura de controle *IF* em que cada um dos resultados de cada agente é comparado com o outro, para

identificar o que apresenta o maior valor, sendo encontrado é impresso na tela através do comando *user-message* a melhor hipótese, a cada iteração realizada. Como se pode observar o algoritmo a seguir:

```
to go
  ask industries [ industry-eat ]
  ask politicians [ politician-eat ]
  ask secretaries [ secretary-eat ]
  ask populations [ population-eat ]
  ask entrepreneurs [ businessman-eat]
  tick
if ( P_avaliacao >  S_avaliacao and  P_avaliacao >
Pop_avaliacao and P_avaliacao >  E_avaliacao)  [
user-message (word "O agente político apresentou a
melhor avaliação:" P_avaliacao " ") ]
 if (S_avaliacao > P_avaliacao and S_avaliacao >
Pop_avaliacao and S_avaliacao > E_avaliacao)
  [ user-message (word "O secretário do Meio
Ambiente apresentou a melhor avaliação:" S_avaliacao
" ") ]
  if (Pop_avaliacao > S_avaliacao and Pop_avaliacao
> P_avaliacao and Pop_avaliacao > E_avaliacao)
  [ user-message (word "Representante da População
apresentou a melhor avaliçao:" Pop_avaliacao " ")]
  if (E_avaliacao > S_avaliacao and E_avaliacao >
P_avaliacao and E_avaliacao > Pop_avaliacao)
  [ user-message (word "O empresário apresentou a
melhor avaliação:" E_avaliacao " ") ]
 Tick
End
```

III. RESULTADOS PARCIAIS

O simulador gerou os resultados dos cálculos de avaliação para cada agente, com os valores definidos para cada critério de avaliação (custo, tempo construção, Impacto ambiental, opinião publica, geração empregos, lucros e impostos) e usando valores randômicos para os pesos (Tx_custo, Tx_tempo construção, Tx_Impacto ambiental, Tx_opinião publica, Tx_geração empregos, Tx_lucros e impostos), sendo realizadas diversas iterações, sempre começando do zero novamente (ou seja, a iteração anterior não influencia na próxima) apenas para fazer uma média das escolhas realizadas pelo SAD. Executou-se duas simulações, cada uma com 206 iterações, logo exportou-se para Excel os valores de cada iteração recebida para cada agente, obtendo-se os resultados da media para cada um deles, como podemos ver na Tabela 1, a comparação dos resultados:

TABELA 1: RESULTADOS PARCIAIS OBTIDOS

Média		
Agente	Iteração 1	Iteração 2
Político	15.117.500.000.000.000	21.820.500.000.000.000
Empresário	22.587.961.783.439.500	16.021.500.000.000.000
Secretário do meio ambiente	15.166.800.000.000.000	18.286.600.000.000.000
Representante da população	14.615.941.176.470.600	14.030.400.000.000.000

Na primeira iteração, de acordo com os critérios de avaliação fica claro que com os valores hipotéticos, o Empresário apresenta o melhor resultado, tendo a maior média, de 22.587.961.783.439.500, sendo escolhido como o melhor local para instalar a industria. Já na segunda iteração o melhor resultado, para instalar a industria é apresentado pelo agente político, com uma média de 21.820.500.000.000.000.

IV. CONCLUSÃO E TRABALHOS FUTUROS

O simulador proposto visa escolher o melhor local a ser inserida a nova indústria utilizando sistemas multiagentes, por assemelharem-se ao comportamento humano. Como estes agentes alimentam o sistema de apoio à decisão, tem-se comportamentos e decisões similares as que serão tomadas no mundo real.

A proposta apresentada neste artigo ainda é inicial, em relação a sua implementação, sendo assim os resultados apresentados são iniciais e precisam de maiores análises. Contudo, percebe-se que a ferramenta NetLogo permite a implementação de todos os aspectos desejados para o SAD de forma ampla e facilitada.

Como trabalhos futuros, temos a inserção de modelos mais reais de Propagação de Poluição que é um coeficiente que faz parte do simulador de desenvolvimento, bem como os ajustes das regras e cálculos do agente coordenador.

Pretende-se ainda estudar a viabilização do uso de outras técnicas como estratégias de decisão coletiva. Outra meta seria estender o trabalho desenvolvido não apenas para o cálculo de impactos gerados pela instalação de indústrias como também para outros tipos de construções. Assim mesmo que não tome uma decisão autônoma, o simulador auxiliará os respectivos órgãos responsáveis a fazer a melhor escolha.

REFERENCES

[1] Camila D. Thomasi, Gerson L. Nunes, Priscila S. Teixeira, Márcio M. Jugueiro, Diana F. Adamatti e Carlos R. A. Tagliani (2011). "**Um sistema para previsão de impactos gerados pela instalação de indústrias e sua influência sobre ecossistemas costeiros no extremo sul do Brasil**." em: WCAMA – Workshop de Computação Aplicada ao Meio Ambiente e aos Recursos Naturais – CSBC.

[2] Efraim Turban, Dorothy Leidner, Ephraim Mclean e James Wetherbe (2010). "**Tecnologia da Informação para Gestão"**. Parte V, Sistemas Gerenciais e Sistemas de Suporte à Decisão. Ed. Bookman, 6ª Edição.

[3] Gerson L. Nunes, Camila D. Thomasi, Márcio M. Jugueiro e Diana F, Adamatti (2011). "**Um Sistema de Apoio a Decisão baseado em agentes para simulação de impactos gerados pela instalação de indústrias**". Em: WESAAC 2011 – Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações. Curitiba.

[4] NetLogo 5.0 (2012), User Manual, Fevereiro. Acesso em 26.05.2012.

[5] Prisma, "Breve introdução a ferramenta NetLogo" http://cftc.cii.fc.ul.pt/PRISMA/capitulos/netlogo/topico3.php, acesso em 26 de maio.2012.

[6] Solange Oliveira Rezende. **Sistemas Inteligentes - Fundamentos e Aplicações.** 1. ed. Manole: São Paulo, 2002. pg. 270-303.

[7] L .O. Alavares; J. S. **Sichman Introdução aos sistemas multiagentes**. em: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. Jornada de Atualização em Informática. Brasília - UnB, 1997. p. 1-37.

A Brownian Agent approach for modeling and simulating the population dynamics of the schistosomiasis contagion

Renato L. Cagnin, Ivan R. Guilherme, Filipe Marcel F. Gonçalves, and Alexandro Baldassin
Departamento de Estatística, Matemática Aplicada e Computação
Universidade Júlio de Mesquita Filho (UNESP)
Rio Claro, Brazil
rlcagnin@rc.unesp.br, ivan@rc.unesp.br, filipemfg@gmail.com, alex@rc.unesp.br

***Abstract*—Multiagent Simulations have been successfully employed in the studies of population behavior. We present a brownian agent approach for simulating the Schistosomiasis infection. Early results suggest that the proposed model describes relevant aspects of the infection conditions.**

Keywords—Simulation; schistosomiasis; brownian agents; contagion.

I. Introdução

Sistemas Multiagentes têm sido adotados com êxito na simulação de diversos problemas [1]. Neste tipo de simulações, pressupõe-se que os agentes individuais que formam o sistema são dotados de certa autonomia e inteligência. Estes agentes interagem para que um propósito, ou um comportamento global seja atingido. Estas interações entre os diversos agentes influenciam a dinâmica do sistema como um todo.

As simulações Multiagentes (MABS) vêm sendo empregadas para o estudo de sistemas complexos como a modelagem de sistemas ecológicos, modelagem do comportamento de populações para a previsão de criminalidade, dinâmica de populações biológicas, dispersão de informações, formação de grupos, entre outros [2]. Uma outra importante utilização de MABS tem sido feita no contexto do contágio e disseminação de doenças, como por exemplo a malária [3].

Há diversas abordagens para a modelagem de MABS. Neste trabalho a modelagem utilizada é dos agentes brownianos. Um agente browniano é caracterizado por ser capaz de gerar um campo de informação local que também pode influenciá-lo ou a outros agentes. A vantagem no uso de agentes brownianos se deve à facilidade de modelagem de problemas que apresentam difusão de informação ao longo do tempo e espaço [4].

Neste trabalho é apresentado a utilização do modelo de agentes brownianos para a simulação do contágio da esquistossomose [5].

II. A Esquistossomose

A esquistossomose é uma verminose debilitante que pode ser letal e é causada no ser humano pelo parasita platelminto adulto, do gênero *Schistossoma*. O ciclo da doença apresenta dois hospedeiros: o caramujo do gênero *Biomphalaria* – hospedeiro intermediário, e o homem – hospedeiro definitivo. A esquistossomose ocorre em regiões onde existem córregos, açudes e baixo ou inexistente saneamento básico. No Brasil, é considerada ainda um grave problema de saúde pública, pois acomete milhões de pessoas, provocando anualmente um número expressivo de formas graves e até mesmo óbitos [6].

O ciclo da doença inicia-se quando um homem contaminado elimina ovos do verme ao excretar em uma região hídrica favorável ao desenvolvimento do parasita. Destes ovos eclodem larvas ciliadas – Miracídios, que penetram na epiderme do caramujo e o contamina. Após um período de maturação no interior do molusco, estas larvas sofrem metamorfoses e são liberadas pela pele do caramujo, agora como larvas com cauda, denominadas Cercárias – a forma infectante da doença. As Cercárias buscam o hospedeiro definitivo e penetram na pele do homem. Agora sem a cauda, são carregados pela corrente sanguínea e, quando adultos, se alojam nas veias do intestino ou da bexiga do hospedeiro, causando assim a esquistossomose. Os vermes adultos se reproduzem, gerando novos ovos que serão expelidos pelo homem – fechando assim o ciclo da doença [6].

O combate à esquistossomose pode ser feito combinando estratégias de combate ao molusco *Biomphalaria* junto a medidas de melhoria de saneamento, saúde e educação, já que o ciclo de vida do parasita depende tanto das interações entre as larvas e os hospedeiros, quanto das condições para o desenvolvimento das larvas [6].

III. Agentes Brownianos

A modelagem do problema ocorrerá a partir da técnica de agentes brownianos. Um Agente Browniano é caracterizado por um conjunto de variáveis de estado u_i^k, onde i=1,...,N se refere a um agente i individual e k indica suas diferentes variáveis de estado [4]. Na modelagem dos agentes utilizados na simulação serão adotadas duas observáveis: espaço (Eq. 1) e velocidade (Eq. 2), presentes em todos os agentes pois assume-se que todo agente apresenta mobilidade no espaço, e duas internas: a informação e energia. A variável de informação interna (Eq. 3) permite caracterizar a natureza dos agentes vetores (homem e caramujos), no caso do problema em questão, caracterizar o fato do agente estar ou não infectado. A

energia interna do agente (Eq. 4), permite definir quão ativo um agente estará num dado estágio da simulação. No caso do problema proposto, isso é o equivalente à reserva metabólica dos indivíduos das diferentes populações de parasitas. Os parasitas (Miracídios e Cercárias) sobrevivem temporariamente consumindo sua reserva de nutrientes.

$$u_i^1 = \vec{r}_i, \quad (1) \qquad u_i^2 = d\frac{\vec{r}_i}{dt} = \vec{v}_i, \quad (2)$$

$$u_i^3 = \theta_i, \quad (3) \qquad u_i^4 = e_i. \quad (4)$$

O comportamento individual de cada agente apresenta um comportamento determinístico, baseado em fatores que influenciam o meio sobre o qual está inserido. No caso das populações biológicas, este se apresenta na forma das restrições associadas ao recurso hídrico (lagoas, rios e açudes); já no caso de populações humanas, refere-se às necessidades biológicas que levam o indivíduo humano à um ambiente de possível contágio. Nos agentes são especificados uma componente estocástica para permitir representar a livre busca pelo ambiente geográfico, ou seja, cada agente poderá se deslocar aleatoriamente em torno de uma posição média, simulando o andar errante em busca de recursos, como por exemplo, alimentos. Dessa forma, a dinâmica das variáveis poderá ser influenciada por fatores estocásticos e determinísticos.

O modelo do ambiente para o problema corresponde a um ambiente geográfico, onde os recursos hídricos, essencial na dinâmica do contágio, são modelados na forma de potenciais atrativos determinísticos para os agentes miracídios, cercárias e caramujos. Dessa forma, o recurso hídrico pode ser modelado a partir de uma função matemática que descreve uma área de influência para os agentes citados. Para caracterizar os ambientes lacustres, esses recursos são representados por geometria circular ou elíptica.

De acordo com o modelo do agente e do ambiente, é possível determinar a variação da velocidade individual do agente (Eq. 5) e também o consumo de energia (Eq. 6) de cada um dos agentes.

$$d\frac{u_i^k}{dt} = f_i^k + F_i^{stoch}, \quad (5) \qquad d\frac{u_i^4}{dt} = -m_i(t), \quad (6)$$

onde f_i^k, , representam as influências determinísticas (recursos hídricos e campo de informação, discutido a seguir) sobre o agente, F_i^{Stoch}, as influências estocásticas e m_i, a taxa metabólica do agente.

O consumo de energia será aplicado somente em agentes da população de parasitas (Miracídios e Cercárias), pois assume-se que o tempo de vida dos agentes da população de parasitas é muito menor que o tempo de vida da população de vetores. Dessa forma, o número de agentes do tipo vetores permanece constante ao longo de cada simulação.

A informação interna do agente está presente somente nos agentes caramujos e humanos, e varia somente de acordo com as condições propícias ao contágio, esta variável contribui para a construção de áreas locais de influência em determinadas regiões do ambiente, denominadas de campo de informação.

No modelo, a interação entre os agentes das diferentes populações será modelada por um campo de informação. O campo de informação é construído a partir da contribuição individual de cada agente segundo o valor de sua informação interna atual. Dessa forma, quanto maior o número de agentes concentrados numa dada região do espaço, maior será o efeito deste campo sobre os demais agentes que compõem o sistema. Admite-se que somente o vetores não contaminados poderão se contaminar e produzirão regiões em que terão maiores chances de contágios para as populações de parasitas. Ou seja, um agente vetor já contaminado não influencia mais o campo de informações acessível aos agentes do tipo parasita. Para simular a necessidade da busca de um hospedeiro, os agentes parasitas sentem a influência do campo, caracterizada como uma força determinística que os atrai até o(s) agente(s) vetores. Por outro lado, os agentes vetores são os geradores do campo de informações contribuindo com o valor do seu estado num dado ponto do espaço.

IV. O Processo de Simulação

Um simulador com uma interface Web foi desenvolvida. Na interface são informados os parâmetros de entrada agrupados em três categorias: dados de um cenário geográfico (dimensões da região geográfica, escala, números e localizações de recursos hídricos), dados demográficos das populações (tamanho da população de humanos e de caramujos e sua distribuição no espaço geográfico), dados sobre o tempo (duração da simulação).

Durante a simulação uma lista de todos os agentes é mantida em memória e será modificada segundo os passos a seguir:

1. Atualização dos campos de informação: a lista de agentes é percorrida para cada agente da classes de vetores (Humanos ou caramujos), o campo associado é atualizado, ou seja, o valor associado à novas posições dos agentes vetores é alterado de acordo com os eventos que ocorreram;

2. Proliferação e contaminação: a contaminação ocorre da seguinte forma: se o agente for do tipo miracídio ou cercária, e houver um humano ou caramujo respectivamente, em um determinado raio, esses agentes são contaminados. Uma vez contaminados, tais agentes não são curados. Já a proliferação se dá a seguir: todo agente do tipo humano, que se encontra contaminado, gera um novo agente miracídio, que será adicionado ao final da lista de agentes, o mesmo ocorre com os agentes caramujos que estão contaminados, contudo o novo agente criado é um agente cercária.

3. Atualização das posições dos agentes: a posição, velocidade e energia de todos os agentes no sistema são atualizados. Caso a energia de um determinado agente se esgote, o respectivo agente é removido da simulação.

A simulação consiste na execução das fases 1, 2 e 3, nessa ordem, para um número pré-determinado de passos de simulação, o algoritmo para o processo de simulação é apresentado na Figura 1.

```
Definição do ambiente e das populações  (vetores e parasitas)
Enquanto a simulação não terminar
        Remover possíveis agentes mortos
        Atualizar a energia dos agentes
        Modificar a posição dos vetores
        Atualizar o campo de informações
        Modificar a posição dos parasitas
        Verificar possíveis contágios
        Criar novos parasitas
FimEnquanto.
```

Figura 1: Algoritmo para o processo de simulação.

Por serem agentes Brownianos, os agentes do tipo Parasitas modificam sua dinâmica ao serem influenciados pelo campo de informações gerado pela população de agentes da classe Vetores. Se estes agentes conseguirem atingir agentes da classe Vetores, o contágio ocorrerá, mudando o atributo "estado" do agente Vetor, o que desencadeará a criação de novos agentes do tipo Parasitas.

V. Resultados

Uma série de simulações foram executadas tendo como objetivo verificar a dinâmica do contágio das populações. Para isso, serão analisadas a influencia das localizações dos recursos hídricos e a influência da população de humanos e de caramujos, pois como já discutido, o ciclo de contágio da doença é complexo e envolve a interação de diferentes populações.

Dessa forma, um conjunto inicial de simulações apresentará mudanças na localização e dimensão dos recursos hídricos do ambiente mantendo inalteradas populações de vetores, que nos demais conjuntos serão modificadas, mantendo o ambiente inalterado.

Todas as simulações foram executadas no tempo máximo de 500 passos. A velocidade médida da propagação do contágio era calculada ao término de cada simulação. Ao todo, 30 simulações foram realizadas para cada conjunto de parâmetros testado. As simulações são iniciadas com as diversas populações de agentes dispostas ao longo do espaço geográfico. Os agentes do tipo humanos estarão dispostos aleatoriamente, simulando uma ocupação desordenada, geralmente característica em regiões de baixo saneamento. As demais populações que dependem do recurso hídrico, são alocadas ao longo das dimensões desses recursos.

Os fatores antes citados (distribuição e tamanho de recursos hídricos e densidade populacional) influenciam diretamente na propagação dessa informação ao longo do tempo e do espaço. Na Figura 2, é demonstrado que o tamanho das populações de vetores influencia na velocidade média de propagação do contágio. Assim como a distribuição e o tamanho dos recursos hídricos, influenciam diretamente na distribuição dos agentes caramujos e alteram o número de humanos localizados em suas proximidades. Isso significa que um maior número de humanos se tornam suscetíveis ao contágio.

Uma elevada ocupação de humanos permite que os caramujos sejam rapidamente contaminados (Fig. 2). O mesmo comportamento é observado com o número de caramujos. Em ambos os casos verifica-se que o aumento significativo de indivíduos da população de parasitas aumenta a possibilidade de contágio a partir dos indivíduos vetores. O que aumenta a propagação da doença ao longo do espaço. Isso se deve pois os vetores contaminados, criam novos parasitas fechando o ciclo da doença. Este comportamento era esperado segundo as características do ciclo da doença como já discutido anteriormente.

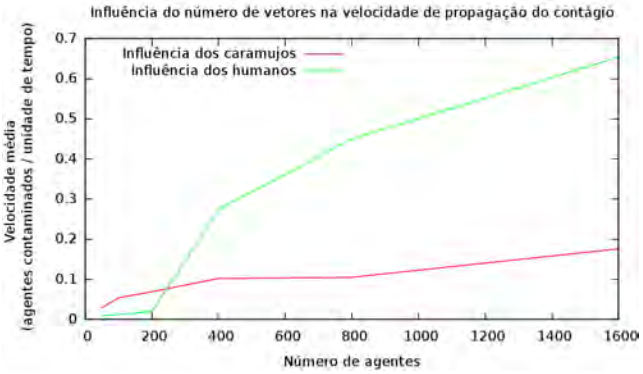

Figura 2: Influência da população de caramujos e humanos

VI. Conclusões

Uma abordagem multiagente foi utilizada para a modelagem e simulação da dinâmica das populações no contágio da esquistossomose. Em geral, os resultados obtidos a partir do simulador, demonstraram que o fator localidade é determinante para o contágio, e este fator é influenciado pela densidade populacional de vetores encontrados ao longo do espaço geográfico. O simulador foi submetido a uma série de testes verificando diversos parâmetros, e de forma geral, o simulador permitiu representar os aspectos relevantes para o demostrar a dinâmica das populações no contágio da esquistossomose.

A validação do modelo com dados reais serão realizados em trabalhos futuros. Em decorrência da complexidade dos cenários reais, a simulação vai requerer a conexão com modelos de dados geográficos reais, e em decorrência do elevado número de agentes, implementar uma versão do simulador que utiliza o processamento paralelo.

References

[1] F. Bousquet and C. Le Page, *Multi-agent simulations and ecosystem management: a review*. Ecol. Model . 176. 2004. pp. 313–332.

[2] V. Furtado, *Modelagem e Simulação Multiagente da Criminalidade*, *Comp.Scien. 0002*. Dec. 7, 2008.

[3] C. Linard, N. Poncon, D. Fontenille and E. F. Lambin. *A multi-agent simulation to assess the risk of malaria re-emergence in southern France , Ecological Modelling 220* (2009) pp. 160–174.

[4] F. Schweitzer. *Brownian Agents and Active Particles : Collective Dynamics in the Natural and Social Sciences*, Ed. Springer-Verlag, 2003 Berlin.

[5] H. Hu, P. Gong, B. Xu. *Spatially explicit agent-based modelling for schistosomiasis transmission: Human–environment interaction simulation and control strategy assessment.* Epidemics 2 (2010) 49–65

[6] V. Schall, C. L. Massara, M. J. Enk, H. S. Barros. *Os Caminhos da Esquistossomose. Parte I Dentro do nosso corpo*. Centro de Pesquisas René Rachou/Fiocruz. 2007. (Série Esquistossomose, 8).

[7] J. Shin, Y. Park. *Brownian agent-based technology forecasting*. Technological Forecasting & Social Change 76 (2009) 1078–1091.

Self-Regulation of Social Exchange Processes in MAS: a cultural and evolutionary BDI agent society model

Andressa von Laer*, Graçaliz P. Dimuro* and Marilton S. Aguiar†
* Prog. de Pós-Graduação em Computação
Universidade Federal do Rio Grande
Rio Grande, Brazil
Email: {andressavonlaer, gracaliz}@gmail.com
† Prog. de Pós-Graduação em Computação
Universidade Federal de Pelotas
Pelotas, Brazil
Email: marilton@inf.ufpel.edu.br

***Abstract*—Agent interactions are often understood as service exchange processes between pairs of agents, followed by the evaluation of these services by the agents involved, generating social exchange values. For the social equilibrium of the agent society there should be an adequate balance of those values, which can be obtained by the regulation of the social exchange processes by the agents themselves. A hybrid evolutionary model of self-regulation of processes of social exchanges in MAS was proposed for Netlogo. However, certain characteristics involved in social exchanges are more appropriately dealt with BDI agents. This paper proposes to develop a model of agent society composed by cultural evolutionary BDI-like agents, for the self-regulation of processes of social exchanges, using the JaCaMo framework.**

I. Introdução

Na Teoria das Trocas Sociais de Piaget [Piaget e Smith 1995] as interações são entendidas como processos de trocas de serviços entre pares de agentes, seguidos da avaliação destes serviços por parte dos agentes envolvidos, gerando assim valores das trocas sociais. Para que haja o controle das trocas sociais entre agentes, o balanço dos valores envolvidos nas trocas deve ser continuamente mantido, tanto quanto possível, perto do equilíbrio, havendo assim a regulação de trocas sociais.

Vários trabalhos já foram realizados pelo grupo de pesquisa no sentido da regulação de trocas sociais em Sistemas Multiagentes (SMA) [Dimuro et al. 2011], [Pereira et al. 2008a], [Pereira et al. 2008b], [Macedo et al. 2012]. Em particular, em [Macedo 2013] foi proposto um modelo híbrido evolucionário de autorregulação de processos de trocas sociais entre agentes em um SMA, baseado em Teoria dos Jogos [Fiani 2006] e Algoritmos Genéticos (AG) [Linden 2008], procurando tornar os agentes independentes e reguladores dos processos de trocas com seus parceiros. A implementação foi realizada no NetLogo[1].

Entretanto, na literatura observa-se que certas características envolvidas em trocas sociais são mais adequadamente tratadas com agentes cognitivos, como Agentes BDI (*Belief, Desire, Intention*) [Rao e Georgeff 1991], [Rao e Georgeff 1992].

Os Algoritmos Genéticos (AGs) e Culturais (ACs) situam-se dentro de um paradigma na Inteligência Artificial (IA) que acredita na possibilidade de reproduzir características humanas em uma máquina para que esta possa resolver problemas. Os AGs são a base dos ACs, porém estes dispõem de um componente chamado Espaço de Crenças. Os ACs baseiam-se na ideia de que a cultura também evolui, e sua evolução é mais rápida que a genética, possibilitando uma melhor adaptação do agente ao ambiente [Reynolds e Zanoni 1992].

A proposta apresentada neste trabalho é de mapear a ideia do Jogo de Autorregulação de Processos Trocas Socais (JAPTS), introduzido no trabalho de [Macedo 2013] para um sistema de agentes BDI, baseados em algoritmos genéticos e culturais.

Neste trabalho, adota-se a plataforma JaCaMo[2], que é um framework para programação de sistemas multiagentes que combina três ferramentas/tecnologias separadas: Jason, CArtAgo e MOISE+, para modelagem da organização do sistema multiagente.

Este artigo está organizado da seguinte forma: a Seção II aborda os conceitos de sistemas multiagentes relacionados ao modelo de arquitetura BDI e também são descritos os processos de um agente BDI no ambiente Jason; a Seção III aborda conceitos, definições e estrutura dos Algoritmos Genéticos e, também, apresenta conceitos e funcionamento dos Algoritmos Culturais; a Seção IV apresenta a ideia dos agentes BDI Evolucionários Culturais através da combinação entre as três tecnologias abordadas neste artigo; e, por fim, a Seção V apresenta as considerações finais deste trabalho.

II. Agentes BDI

Entre as diversas abordagens existentes na área de Inteligência Artificial (IA) em que a arquitetura dos agentes é baseada no comportamento humano, a que mais se destaca é o mo-

[1]Disponível em: `http://ccl.northwestern.edu/netlogo/`

[2]Disponível em: `http://jacamo.sourceforge.net`

delo BDI, baseado na teoria de raciocínio prático humano desenvolvido pelo filósofo Michael Bratman em [Bratman 1987]. A estrutura deste modelo é descrita em [Bratman et al. 1988].

O modelo BDI explica o comportamento humano através das seguintes atitudes: crenças, desejos e intenções, e supõe que as ações são derivadas a partir do processo chamado raciocínio prático constituído de passos. No primeiro passo os objetivos do agente são determinados através de um conjunto de desejos que devem ser alcançados. No segundo passo determina-se quais ações tomar (planos), através do uso dos meios disponíveis, para que estes objetivos sejam alcançados.

A arquitetura BDI tem se destacado em sistemas multiagentes como um importante método de modelagem e desenvolvimento de agentes racionais. As três atitudes mentais que compõem a arquitetura BDI são:

- Crenças (*Beliefs*): representam a visão que um agente possui do seu ambiente. Podem ser vistas como o provável estado do ambiente, isto é, como um componente informativo do estado do sistema. Um agente pode ter crenças sobre o mundo, sobre outros agentes, sobre interações com outros agentes e crenças sobre suas próprias crenças.
- Desejos (*Desires*): representam estados desejáveis que o sistema poderia apresentar e/ou que o agente gostaria de alcançar. Eles influenciam o agente a agir de forma a realizar metas, e as ações tomadas são realizadas através das intenções, causadas pelos desejos.
- Intenções (*Intentions*): são estados que os agentes pretendem alcançar, e estão interligadas com os desejos dos agentes da seguinte maneira: se um agente decide seguir uma meta específica, então esta meta torna-se uma intenção. Uma intenção quando adotada, acarretará em um direcionamento no raciocínio prático futuro, ou seja, enquanto se tem uma intenção particular haverá consideração por ações que são consistentes para a realização desta intenção.

Um agente BDI permanece comprometido com algum plano até que ele seja totalmente executado, a intenção (pela qual o plano foi desenvolvido) seja atingida, seja inatingível, ou não seja mais útil. Se por algum motivo o agente perceber que não será capaz de atingir seus objetivos, ele deve rever seus conceitos afim de fazer novas escolhas de planos e intenções. Este comprometimento do agente com seus planos e intenções é um fator importante no modelo computacional de um agente BDI.

Jason[3] [Bordini et al. 2007] é uma plataforma de desenvolvimento de sistemas multiagentes baseada em um interpretador para uma versão estendida da linguagem de programação *AgentSpeak(L)* que oferece uma série de extensões que são necessárias para o desenvolvimento de sistemas multiagentes. *AgentSpeak* é uma linguagem de programação orientada a agentes e baseia-se em eventos e ações. É baseada em implementações de sistemas BDI já existentes na época, tais como *Procedural Reasoning System* (PRS) [Georgeff e Lansky 1986] e o *Distributed MultiAgent Reasoning System* (dMARS) [Luck e Wooldridge 2004].

Em Jason os desejos dos agentes são representados pelos eventos que ocorrem no interpretador, ativando os planos. O Jason possui uma estrutura chamada "base de crenças" (*belief base*) para armazenar as crenças do agente, esta estrutura consiste em um conjunto de predicados sobre um estado do ambiente. As crenças representam o que o agente "acredita" ser verdade no ambiente, não significa que seja mesmo uma verdade. Os objetivos de um agente indica o que ele fará, ou o estado do ambiente que ele deseja atingir. Por exemplo, dado um determinado objetivo g, o agente se compromete em alterar o estado do ambiente até que acredite que o determinado objetivo é verdadeiro. Os planos de um agente são armazenados em uma espécie de *biblioteca de planos*, onde inicialmente armazenam os planos que o próprio programador escreveu. Para que um objetivo seja alcançado, a cada ciclo de raciocínio um plano é executado, podendo alterar o ambiente.

III. Algoritmos Genéticos e Culturais

Os Algoritmos Genéticos (AGs) foram criados nos anos 60 por John Holland, e desenvolvidos em meados dos anos 70 pelos seus alunos da Universidade de Michigan. Holland tinha como objetivo estudar o fenômeno "evolução" por reprodução (Darwinismo) [Darwin 1859] e de alguma maneira trazer isto para a computação [Goldberg 1989].

A evolução por seleção natural ocorre quando os indivíduos mais adaptados ao meio ambiente tem mais chances de sobreviver. Neste sistema, explorado pelos Algoritmos Genéticos, os cromossomos podem ser explorados e combinados, no intuito de formarem melhores candidatos à solução e, consequentemente, encontrar soluções aproximadas para problemas de grande complexidade computacional [Aguiar e Toscani 1997].

Nos cromossomos dos indivíduos está codificado o conhecimento que cada um possui. Há mecanismos de reprodução que modificam esta formação, os mais utilizados são: as mutações, responsáveis por fazer certas alterações, que ocasionalmente são benéficas aos cromossomos; inversões, responsáveis por uma inversão no código do cromossomo; e o cruzamento (*crossover*), que faz uma troca com o material genético dos cromossomos geradores, e é o mecanismo que influencia na eficiência dos AGs.

Por ser uma técnica robusta e efetiva em uma grande área de problemas, os AG's são utilizados em um grande número de problemas e modelos científicos e da engenharia. Eles nem sempre garantem a solução ótima, mas geralmente encontram uma solução aceitável e de maneira rápida.

Por outro lado, as interações e o comportamento social dos humanos também são uma forte influência na otimização da raça humana, além da genética e da evolução, pois as interações sociais ajudam em uma melhor adaptação do ser ao ambiente. Baseado nesta ideia, Robert Reynolds propôs os Algoritmos Culturais (AC's) [Reynolds e Zanoni 1992], como complemento às outras técnicas de computação evolutiva.

Segundo Reynolds, os AC's modelam a evolução cultural de um sistema computacional ao longo do tempo e possuem um mecanismo explícito de aquisição, armazenamento e

[3]Sigla do inglês *A Java-based AgentSpeak Interpreter Used with Saci for Multiagent Distribution Over the Net*.

integração da experiência e do comportamento dos indivíduos e de grupos.

A utilização dos ACs é indicada para diversos tipos de problemas, como: a) quando se tem grande espaço de busca a ser explorado; b) quando se tem problemas de otimização restrita; c) onde a adaptação pode ocorrer em vários níveis e em várias taxas dentro do espaço populacional e de crença; d) quando se trabalha com ambientes dinâmicos (i.e. *World Wide Web*, etc); entre outros.

IV. Agentes BDI Evolucionários Culturais

A definição de Agentes BDI Evolucionários Culturais surge através de um mapeamento entre os elementos da arquitetura BDI e os Algoritmos Genéticos na plataforma Jason. Este mapeamento consiste em:

- os "cromossomos" dos AGs passam a ser representados na forma das crenças (*beliefs*) dos agentes;
- o processo de reprodução (*crossover*, mutação, etc.) é guiado através dos objetivos dos agentes BDI, ou seja, a reprodução se torna um objetivo com o qual o agente se comprometerá em alcançar executando uma sequência de planos existentes no Jason (passos necessários para o alcance do objetivo final);
- no processo de evolução de suas crenças, os agentes podem se valer da "cultura do SMA", que é alimentada pelos próprios agentes. Esta cultura pode ser tratada como um artefato do CArtAgO da plataforma JaCaMo, onde nela estão informações comuns e disponíveis a todos os agentes BDI.

V. Conclusão

Na área de Inteligência Artificial um dos desafios encontrados é a representação de conhecimento, por ser uma propriedade humana difícil de ser representada a fim de ser interpretada por máquinas. Por isso se faz necessário a criação de ferramentas que facilitem o desenvolvimento de agentes artificiais autônomos.

O presente trabalho apresentou uma proposta para estender o trabalho apresentado em [Macedo 2013], interligando os temas ali trabalhados com a arquitetura cognitiva BDI, pois estas facilitam o tratamento das características subjetivas envolvidas no modelo. Até o momento foi feita a implementação de uma aplicação baseada no problema do Caixeiro Viajante – que tenta determinar uma menor rota para percorrer uma série de cidades (visitando cada uma pelo menos uma vez) retornando a cidade de origem – para demonstrar a possibilidade do mapeamento das características de uma arquitetura BDI utilizando a plataforma Jason.

Agradecimentos

Este trabalho teve apoio financeiro do CNPq (Proc. 560118/10-4, 305131/2010-9, 476234/2011-5), FAPERGS (Proc. 11/0872-3) e do Projeto RS-SOC (FAPERGS Proc. 10/0049-7).

Referências

[Aguiar e Toscani 1997] Aguiar, M. S. e Toscani, L. V. (1997). Algoritmos genéticos. In *I Workshop sobre métodos formais e qualidade de software*, pages 78–87, Porto Alegre/RS.

[Bordini et al. 2007] Bordini, R. H., Hübner, J. F., e Wooldridge, M. (2007). *Programming Multiagent Systems in AgentSpeak using Jason*. University of Liverpoll: Wiley.

[Bratman 1987] Bratman, M. (1987). *Intention, plans, and practical reason*. Harvard University Press.

[Bratman et al. 1988] Bratman, M. E., Israel, D. J., e Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. Technical Report 425, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025. Revised version.

[Darwin 1859] Darwin, C. (1859). *On the Origin of Species*. John Murry, London.

[Dimuro et al. 2011] Dimuro, G. P., da Rocha Costa, A. C., Gonçalves, L. V., e Pereira, D. R. (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(3):143–161.

[Fiani 2006] Fiani, R. (2006). *Teoria Dos Jogos*. CAMPUS.

[Georgeff e Lansky 1986] Georgeff, M. P. e Lansky, A. L. (1986). Procedural knowledge. In *Proceedings of the IEEE Special Issue on Knowledge Representation*, volume 74, pages 1383–1398.

[Goldberg 1989] Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*. Artificial Intelligence. Addison-Wesley.

[Linden 2008] Linden, R. (2008). *Algoritmos Genéticos*. Brasport, 2a edition.

[Luck e Wooldridge 2004] Luck, M. e Wooldridge, M. (2004). The dmars architecture: A specification of the distributed multiagent reasoning system. In *Autonomous Agents and Multiagent Systems*, pages 1–2.

[Macedo 2013] Macedo, L. F. K. (2013). Uma abordagem evolucionária e espacial para o jogo da autorregulação de processos de trocas sociais em sistemas multiagentes. Dissertação de mestrado, Universidade Federal do Rio Grande.

[Macedo et al. 2012] Macedo, L. F. K., Dimuro, G. P., Aguiar, M. S., da Rocha Costa, A. C., Mattos, V. L. D., e Coelho, H. (2012). Analyzing the evolution of social exchange strategies in social preference-based mas through an evolutionary spatial approach of the ultimatum game. In *Social Simulation (BWSS), 2012 Third Brazilian Workshop on*, pages 83–90.

[Pereira et al. 2008a] Pereira, D. R., Gonçalves, L. V., Dimuro, G. P., e Costa, A. C. (2008a). Constructing bdi plans from optimal pomdp policies, with an application to agentspeak programming. pages 240–249, Santa Fe, Argentina.

[Pereira et al. 2008b] Pereira, D. R., Gonçalves, L. V., Dimuro, G. P., e Costa, A. C. (2008b). Towards the self-regulation of personality-based social exchange processes in multiagent systems. In Zaverucha, G. e Costa, A. L., editors, *Advances in Artificial Intelligence - SBIA 2008*, volume 5249 of *Lecture Notes in Computer Science*, pages 113–123. Springer Berlin Heidelberg.

[Piaget e Smith 1995] Piaget, J. e Smith, L. (1995). *Sociological Studies*. Taylor & Francis.

[Rao e Georgeff 1991] Rao, A. S. e Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. In Fikes, R. e Sandewall, E., editors, *Proc. 2nd Intl. Conf. on Principles of Knowledge Representation and Reasoning)*, pages 473–484, San Mateo. Morgan Kaufmann.

[Rao e Georgeff 1992] Rao, A. S. e Georgeff, M. P. (1992). An abstract architecture for rational agents. In *Proc. of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 439–449. Morgan Kaufmann.

[Reynolds e Zanoni 1992] Reynolds, R. e Zanoni, E. (1992). Why cultural evolution can proceed faster than biological evolution. In *Proceedings of International Symposium on Simulating Societies*, pages 81–93. sn.

In-silico Simulation of Indoor Panic Situations using Reactive Agents

Giorgio P. F. G. Torres, Willian C. Farago, Alcione de Paiva Oliveira
Departamento de Informática
Universidade Federal de Viçosa
Viçosa, Minas Gerais, Brazil
{giorgio.torres; willian.farago}@ufv.br, alcione@dpi.ufv.br

***Abstract* – Panic situations sadly happens frequently. Even in cases where there are people trained to deal with emergency situations, most people panic, resulting in possible tragedies. In this article, it is investigated how primitive agents, with low capacity for deliberation and limited perception of the environment, behave in a panic scene. We conducted a case study, taking as example the tragedy in Santa Maria, Rio Grande do Sul - Brazil, where a fire in a nightclub resulted in the deaths of 241 persons.**

Keywords – simulation; simulação; multi-agent systems; sistemas multiagentes; MAS; SMA; Repast Simphony simulation

I. INTRODUÇÃO

O comportamento de multidões possui muitas variáveis, algumas previsíveis e outras que surgem aleatoriamente, dependendo de circunstâncias do momento. Esta complexidade torna imprevisível o comportamento global. Para evitar tragédias como a ocorrida na boate Kiss na cidade Santa Maria, Rio Grande do Sul, Brasil [3] onde um incêndio resultou na perda de 241 vidas, faz-se necessário a investigação minuciosa dessas variáveis e de seus impactos em ambientes específicos como o da boate. Segundo [6], uma linha de investigação com foco em prevenção muito difundida e abordada atualmente, para problemas com as características anteriores, é a simulação baseada em Sistemas Multiagentes. A vantagem de uma simulação por meio desta técnica é o fato de não ser necessário modelar o comportamento global, bastando a modelagem do comportamento de cada agente e, a partir da execução do sistema e pela interação entre os agentes, observar o surgimento do comportamento global. Em [4] é apresentado um SMA reativo para analisar o espalhamento da Influenza por meio de agentes reativos, mostrando a adequação deste tipo de modelagem para processos estocásticos. A simulação por meio de SMA do comportamento de aglomerados de pessoas em situações de pânico não é inédito. [1] desenvolveram um modelo SMA com esse objetivo, porém as situações modeladas eram muito genéricas e não levavam em conta as peculiaridades de uma casa noturna. As mesmas observações são aplicáveis aos trabalhos de [6] e de [2]. Este artigo descreve uma simulação por meio de sistemas multiagentes para uma situação de pânico para ambientes fechados e com muitos obstáculos, similar ao ambiente onde ocorreu a tragédia mencionada. É investigado como agentes primitivos, com baixa capacidade de deliberação e percepção limitada do ambiente, se comportam em um cenário de pânico.

A próxima seção aborda a simulação do comportamento de pessoas por meio de sistemas multiagentes. A seção 3 descreve como foi realizada a modelagem. Na seção 4 são apresentadas as simulações e resultados e na seção 5 são apresentadas as conclusões.

II. SIMULANDO COMPORTAMENTO DE PESSOAS COM SISTEMAS MULTIAGENTES

Os Sistemas Multiagentes (SMA) são compostos por unidades computacionais, com diferentes graus de capacidade de deliberação, dotadas de autonomia (são capazes de decidir sem intervenção de agentes externos) denominadas agentes, que podem interagir uns com os outros e executar alguma ação no estado do ambiente no qual foram inseridos. O uso de SMA vem crescendo nas ultimas décadas, segundo [5] por dois motivos principais. Primeiramente, em função da crescente complexidade da vida moderna, criando uma demanda por sistemas computacionais cada vez mais complexos, dinâmicos e com execução distribuída. Em segundo lugar, [5] afirma que os SMA têm a capacidade de desempenhar um importante papel no desenvolvimento e análise de modelos e teorias. É justamente esta segunda característica que torna os SMA interessantes para os estudos dos problemas do comportamento de multidão. Os SMA são adequados para modelar sistemas com muitos elementos autônomos, onde cada elemento executa ações segundo os estímulos percebidos localmente. Problemas como esse são modelados segundo uma abordagem *bottom-up* onde são definidos os aspectos individuais dos agentes, de forma a permitir que ocorra a emergência dos aspectos coletivos pela interação entre os agentes. Sendo assim, é possível verificar como ocorre o deslocamento, em massa, de pessoas em uma situação de pânico, como no caso de incêndios em locais fechados.

Um problema com a modelagem de deslocamento de multidão baseada em agentes é verificado nos casos onde é necessário simular uma quantidade elevada de pessoas (na casa das centenas de milhares). Nestes casos a demanda por poder computacional extrapola a capacidade das máquinas atuais. Uma solução é tratar a multidão como um todo, aproximando seu comportamento ao de um fluxo contínuo de um fluido, eliminando as características individuais. Desta forma é

Este trabalho conta com o apoio financeiro das instituições de fomento FAPEMIG, FUNARBE, CNPq e da empresa Gapso.

possível estimar o fluxo de circulação/evacuação para multidões grandes e densas, mas com alguma perda de fidelidade [6]. Como o objetivo desta pesquisa é focar em ambientes menores, com capacidade para alguns milhares de pessoas, optou-se por utilizar agentes específicos que dessem uma resposta mais fiel da realidade.

O sistema foi modelado e desenvolvido sobre o *framework* Repast Simphony[1], em sua versão 2.0. O Repast Simphony é um *framework* para desenvolvimento de sistemas multiagentes e é de código aberto. As definições de espaço, tempo, representação das pessoas e dispersão da fumaça, foram feitas considerando os recursos e limitações do *framework*.

III. Modelagem dos Agentes

Os agentes foram modelados como agentes reativos simples para testar a hipótese de que a ausência de capacidade cognitiva pode prejudicar a evacuação em casos de pânico. Foi feita uma classe chamada *Customer* para representar uma pessoa. O Repast utiliza essa classe para instanciar todos os agentes criados no sistema, totalizando 1000 instâncias dessa classe para representar o público presente na boate. Cada *Customer* contém um atributo *energy*, que representa a resistência do agente às toxinas presentes no ambiente. Esse atributo *energy* é decrementado a cada rodada do simulador em função da quantidade de toxina existente no espaço em que o agente se encontra.

A classe *Customer* contém os seguintes métodos executados pelo simulador:

1) public void wander(); este método é o que contém o algoritmo de fuga da boate. Ele está agendado para começar no tick 1 (unidade de tempo do Repast) e é chamado a cada 1 tick de simulação. Mais adiante será detalhado o processo decisório dos agentes, a dinâmica da emissão de fumaça tóxica e o decrescimento da energia do agente.

2) public void die(); este método diminui a energia do agente à medida em que este está exposto às toxinas. Quando a energia zera, o agente para de se movimentar pelo ambiente. Para representar esse estado foi criada uma classe DeadCustomer que não possui atributos, nem métodos, e serve apenas para evitar sobrecarga do simulador.

O princípio básico da simulação é de que o agente tome decisões bem simples e locais. Logo, um *Customer* decide apenas se vai correr contra a fumaça, ou seja, para a célula do espaço que contém menor concentração de gases, ou se vai andar aleatoriamente. Essa decisão é tomada de modo aleatório e o agente tem 50% de chance para cada uma. Após ter escolhido o modo de andar, o agente vai dar dez passos nesse modo. Essa decisão de modelagem foi feita para que o agente pudesse sair de situações de mínimo local, ou seja, para que ele não ficasse preso a um lugar de baixa concentração de fumaça. Em cada um dos modos, o agente sabe as células livres para as quais ele pode efetuar um deslocamento. Não é permitido que dois agentes ocupem um mesmo espaço no *grid*. Porém quando um agente "morre", outros agentes podem ocupar o espaço, ou seja, podem passar por cima. No modo de movimento aleatório, a cada *tick* o agente escolhe uma outra posição aleatoriamente para se deslocar dentre as células vazias (que não contém outro *Customer*, ou parede ou mobília).

O agente tem 50% de chance para cada um dos modos de movimento, pois no movimento aleatório ele pode decidir ir para uma célula de alta concentração de fumaça, para sair de salas sem saída. Se este modo tivesse maior probabilidade de acontecer, o agente perderia sua energia mais rápido e poderia levá-lo de encontro com o foco da fumaça. Já no movimento de ir contra a concentração de fumaça, o agente é guiado para lugares com maiores chances de ter saída. No entanto, se este modo tivesse maior chance, o agente poderia ficar preso em lugares de mínimos locais, como banheiros e salas fechadas. Por esses motivos apresentados que o agente tem metade das chances de escolha para cada um dos modos de movimento.

IV. Simulações e Análise/Discussão de Resultados

Foram efetuadas ao todo 20 simulações. Em cada simulação os seguintes parâmetros foram modificados: quantidade de energia do agente; número máximo de passos em cada modo de movimento; número de agentes presentes; e a quantidade de fumaça difundida por *tick*. Os testes revelaram que:

1) Os únicos fatores que contribuíam para uma maior taxa de sobrevivência dos agentes foram a quantidade de energia inicial que é dada a eles, a quantidade de passos que poderiam dar em cada modo de movimento, e a quantidade de fumaça expelida pela fonte;

2) Mesmo com uma quantidade de energia muito grande e/ou uma quantidade de fumaça pequena difundida a partir do foco e um grande número na quantidade de passos para sair de mínimos locais, os agentes não eram capazes de sair do estabelecimento, ficando em torno de 450 a 500, o número de agentes cuja energia se tornou zero, como mostrado na Fig. 1, para cada 1000 agentes simulados.

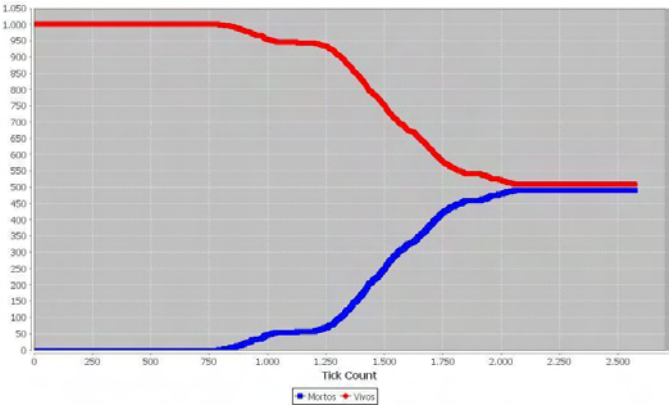

Figura 1. Evolução das taxas de agentes com energia menor que zero (azul) e com energia maior que zero (vermelho).

A. Definição de instâncias para simulação

O mapa utilizado para execução das simulações foi criado baseando-se nos dados da reportagem feita pelo site [3] e está demonstrado na Fig. 2.

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

Foi feita uma regra de equivalência entre as dimensões da boate 23,18 metros de fachada por 26,45 metros de comprimento. Tomou-se como valor máximo de pessoas por metro quadrado o valor 6. Cada pessoa ocupa uma posição no grid. Sendo assim, segundo os cálculos, a área mais fiel seria de 57 unidades de grid na fachada e 65 de comprimento totalizando 3705 espaços no grid.

Para definir os espaços livres e ocupados também foi feita uma relação de equivalência. Segundo os bombeiros entrevistados pelos meios de comunicação, a capacidade máxima de pessoas na boate seria de 691. Este número é obtido a partir da divisão da área ocupável (toda a área da boate descontando paredes e mobília) por 0,4 metros (espaço mínimo para uma pessoa de 70 kg). Sendo assim chegou-se a uma área ocupável de aproximadamente 276,4 metros quadrados. Fazendo a relação com o espaço em grid chegou-se ao valor de 1670 posições ocupáveis por agentes no grid e 2035 não ocupáveis (parede e mobília). Depois de definidos os parâmetros descritos e cientes das restrições do simulador, foi feita uma pesquisa por várias imagens da boate para tentar estabelecer, com o máximo de exatidão possível a posição e espessura das paredes e da mobília, chegando ao resultado visível na Fig. 2. As paredes são representadas com um tom mais escuro e a mobília em um tom mais claro.

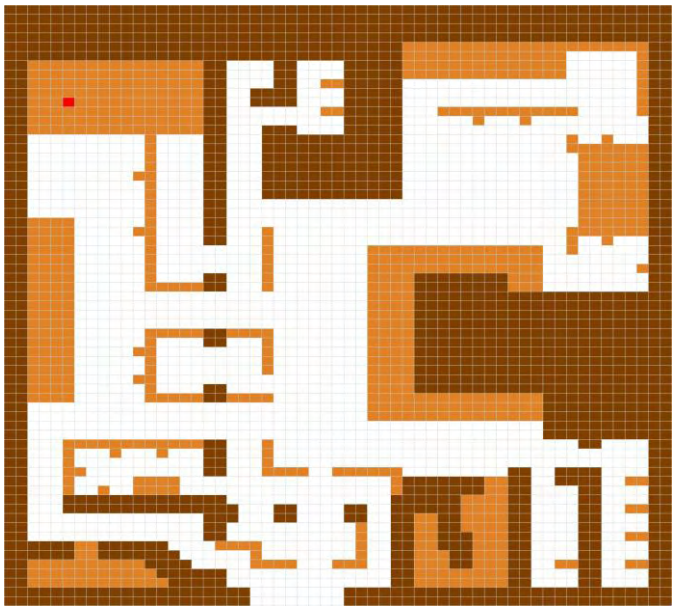

Figura 2. Marcações feitas no espaço para representar o mapa da boate Kiss.

V. Conclusão

A partir dos resultados pôde-se perceber que, em casos de situação de pânico, a capacidade deliberativa de um agente é um fator que proporciona maiores chances de sobrevivência ao mesmo. Visto que a quantidade de agentes que tiveram sua energia zerada foi, em torno de, duas vezes maior que o observado no caso real da boate Kiss.

Como trabalhos futuros devem ser projetados agentes com maior capacidade de deliberação e dotados de base de conhecimento de senso comum, que tenham capacidades de tomar decisões que uma pessoa comum tomaria em situações de risco para as quais eles não estão previamente preparados.

Como trabalhos futuros pretende-se utilizar diferentes mapas de ambiente para podermos analisar diferentes arquiteturas de ambientes e sua evasabilidade, ou seja, sua capacidade de evacuação de pessoas, mesmo que estas sejam restritas em sentidos e inteligência, como foi o foco deste trabalho.

Referências

[1] J. E. Almeida, R. Rosseti and A. L. Coelho. “Crowd simulation modeling applied to emergency and evacuation simulations using multi-agent systems,” DSIE'11 - 6th Doctoral Symposium on Informatics Engineering, FEUP - Engineering Faculty of Porto University, Porto. 2013.

[2] V. Bansal, R. Kota, and K. Karlapalem. “System issues in multi-agent simulation of large crowds,” In, 8th International Workshop on Multi-Agent-Based Simulation (MABS), Honolulu, USA, Springer Berlin / Heidelberg, 8-19. 2007.

[3] Globo Comunicação e Participações S.A. Como foi a tragédia em Santa Maria, http://g1.globo.com/rs/rio-grande-do-sul/tragedia-incendio-boate-santa-maria-entenda/platb/ ultimo acesso 15 de março de 2012.

[4] M. P. Nicoletti, C. B. Rizzi and R. L. Rizzi. “Simulação do espalhamento da influenza na cidade de Cascavel-PR utilizando agentes computacionais,” Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCações (WESAAC). Florianópolis, SC. 2012.

[5] G. Weiss, (Ed.). “Multiagent systems: a modern approach to distributed artificial intelligence,” [S.l.]: The MIT Press, 1999. ISBN 0-262-23203-0.

[6] S. Zhou, D. Chen, W. Cai, L. Luo, M. Y. H. Low, F. Tian, V. S. H. Tay, D. W. S. Ong and B. D. Hamilton. 2010. “Crowd modeling and simulation technologies,” ACM Trans. Model. Comput. Simul. 20, 4, Article 20 (October 2010), 35 pages. DOI = 10.1145/1842722.1842725 http://doi.acm.org/10.1145/1842722.1842725.

Using Agent Coordination Techniques to Support Rescue Operations in Urban Disaster Environments

Alan D. Barroso, Felipe de C. Santana, Victor Lassance, Luis G. Nardin, Anarosa A. F. Brandão, Jaime S. Sichman*
Laboratório de Técnicas Inteligentes (LTI) – Escola Politécnica (EP) – Universidade de São Paulo (USP)
Av. Prof. Luciano Gualberto, 158 – trav. 3 – 05508-970 – São Paulo – SP – Brasil
{alan.barroso, fesantana, victor.lassance.silva, luis.nardin}@usp.br, {anarosa.brandao, jaime.sichman}@poli.usp.br

***Abstract*—This extended abstract describes a task allocation and coordination policy that aims to maximize the efficiency of teams when they are rescuing the victims and protecting the city's heritage in the case of an urban disaster. Our approach considers the existence of local and global information that could help this coordination. Supposing that communication is limited and unreliable, we also present a comparison of those coordination techniques, aiming to improve the task allocation process. This abstract comprises our motivation, the main goal and the key conceptual aspects of the research, as well as the development steps and specification of the whole project.**

I. Introdução

A expansão urbana observada no Brasil nas últimas décadas causou o crescimento acelerado e desordenado das cidades, potencializando assim a ocorrência de desastres, principalmente em áreas mais densamente povoadas. Além disso, o Brasil irá sediar eventos de dimensão internacional, como a Copa do Mundo de 2014 e os Jogos Olímpicos de 2016, aumentando as consequências de um possível desastre urbano.

No caso específico de desastres naturais, pesquisas e dados científicos recentes indicam que eles vêm se tornando mais frequentes, intensos, dinâmicos e complexos [1]. No Brasil, alguns dos desastres que têm ocorrido com maior frequência e que poderiam ter um trabalho de prevenção esbarram em custos altos, o que faz com que os responsáveis posterguem a implementação da solução do problema. Dessa forma, como ocorre com o caso de enchentes e deslizamento de terras, o gerenciamento de desastres deve ser efetivo para minimizar os danos causados, através de ações eficientes de coordenação e alocação dos reduzidos recursos existentes em um ambiente parcialmente observável.

Nesse trabalho será proposta uma abordagem de alocação de tarefas que considera a existência de informações locais e globais com o objetivo de coordenar as entidades de salvamento para que maximizem a eficiência no resgate a vítimas e na proteção do patrimônio da cidade em caso de desastre. Supondo uma comunicação limitada e incerta, esse trabalho também apresentará uma comparação de técnicas de coordenação de forma a melhorar a alocação de tarefas.

Na seção II são apresentados os principais tópicos conceituais utilizados nesse trabalho, que são Sistemas Multiagentes e Técnicas de Coordenação como o *Partial Global Planning*. Já a seção III descreve a estrutura de fases de implementação deste trabalho, enquanto na seção IV detalha-se sua especificação por meio da descrição dos quatro algoritmos de coordenação que serão implementados. A seção V traz uma breve conclusão, descrevendo nossos próximos passos.

II. Aspectos Conceituais

Para melhor entendimento da abordagem proposta nesse trabalho, nesta seção são apresentados os aspectos conceituais relacionados às escolhas quanto às estratégias de coordenação de tarefas. O primeiro aspecto é referente ao modo como o projeto pode ser visto como um sistema multiagente. O segundo é referente ao tipo de técnica de coordenação escolhida.

A. A competição Robocup Rescue e seu simulador

A adoção do paradigma multiagente na abordagem apresentada para alocação de tarefas é fortemente dependente do domínio do problema em que estamos trabalhando, em particular do simulador da competição *Robocup Rescue*[1]. O simulador possui três blocos principais: *kernel*, simuladores e agentes. O *kernel* serve como bloco central e junta todas as informações pertinentes aos problema. Os simuladores têm como função principal simular os diversos fatores que influenciam o andamento de uma rodada da competição. O terceiro bloco é composto por agentes programáveis, que são responsáveis por reduzir as perdas dentro do ambiente de desastre.

Os agentes programáveis possuem um certo grau de autonomia: a decisão de realizar ou não uma ação depende somente da vontade do próprio agente. Esses agentes autônomos devem então se organizar e se coordenar para poder resolver o problema proposto na competição. De acordo com [2], a descrição do problema acima se aproxima bastante à um sistema multiagente. Esse fator possibilita ao projeto o uso de técnicas de coordenação e comunicação já estudadas em sistemas desse tipo, como as descritas a seguir.

B. Técnicas de Coordenação

De acordo com [2], existem três abordagens principais em técnicas de coordenação para sistemas multiagentes: planejamento centralizado para planos distribuídos, planejamento distribuído para plano centralizado e planejamento distribuído para planos distribuídos. A primeira abordagem diz respeito a uma figura central que faz todo planejamento das tarefas e

(*) Jaime S. Sichman is partially supported by CNPq and FAPESP, Brazil.

[1] http://www.robocup2013.org/?lang=en.

comunica aos respectivos agentes suas tarefas. A segunda leva em conta que todos os agentes entram em cooperação para elaborar um plano comum a todos. Na terceira abordagem, a mais complexa, cada agente elabora individualmente um plano próprio, tendo um objetivo comum de cooperar com os outros agentes.

Escolhemos a terceira abordagem por duas razões principais, relacionadas ao modo como a comunicação entre agentes do simulador é feita. Para um agente se comunicar com os outros, ele leva um ciclo de simulação e, para receber a resposta, um segundo ciclo, caso a mensagem chegue corretamente ao destinatário. Isso dificulta o uso da primeira técnica de forma competitiva, pois o tempo envolvido para distribuir os planos entre os agentes seria muito grande. Al;ém disto, a comunicação entre agentes chega até mesmo a ser inexistente em alguns cenários, fazendo com que a segunda abordagem também não possa ser selecionada.

Por essas razões, foi escolhida a técnica de planejamento distribuído para planos distribuídos, mais especificamente através do uso da técnica denominada *Partial Global Planning*. A técnica, descrita em [3], baseia-se na existência de dois níveis de planejamento simultâneos, um local e outro global. No primeiro, o agente cria o seu próprios plano; já no segundo, os agentes trocam informações para que os planejamentos locais levem em conta os conhecimentos dos outros agentes.

III. Fases do Projeto

Esse trabalho visa analisar os métodos de coordenação atualmente utilizados pelas entidades de proteção, para poder sugerir a integração destas diferentes entidades, através de um mecanismo de comunicação criado com forte embasamento teórico e prático. Para atingir esse objetivo, o trabalho deve ser dividido em três fases, cada uma essencial para o resultado final.

A. Fase 1 - Competição Robocup Rescue

Essa fase do projeto está voltada para a participação do grupo na competição *Robocup Rescue*, que ocorrerá em Eindhoven, Holanda, em junho de 2013. A *Robocup Rescue* é uma competição internacional que tem como objetivo colocar em prática técnicas de coordenação de agentes para diminuir os danos e as perdas humanas causadas em um ambiente de desastre. No ambiente de simulação, são utilizados agentes com diferentes papéis, como do Corpo de Bombeiros, Agentes de Polícia e Ambulâncias, cada um deles com funcionalidades específicas. Um dos membros do grupo, Luis Gustavo Nardin, participou desta competição nos anos 2011 e 2012, como detalhado em *LTI Agent Rescue Team Description* [4] [5].

O desenvolvimento das técnicas de coordenação envolve o estudo dos aspectos teóricos e práticos que envolvem a competição como um todo, desde a teoria sobre sistemas multiagentes até as regras da competição e o funcionamento do simulador. Portanto, a participação na competição será essencial para o andamento do nosso trabalho, pois com ela poderemos adquirir os conhecimentos teóricos sobre sistemas multiagentes e técnicas de coordenação, bem como conhecer o funcionamento do simulador da *Robocup Rescue*, que utilizaremos em fases posteriores do projeto.

B. Fase 2 - Modelagem e análise de estratégias atuais de coordenação

Está será uma das fases essenciais do projeto, pois a partir dela o trabalho tomará um viés mais prático. Tentaremos aplicar os conhecimentos adquiridos anteriormente para poder modelar e analisar quantitativamente os métodos de coordenação atualmente utilizados pelas entidades de proteção em cenários de desastre.

Para analisar os métodos de salvamento atuais, serão pesquisadas as técnicas de coordenação das equipes de resgate (Bombeiros, SAMU e Defesa Civil) da cidade de São Paulo, através de visitas aos seus centros de operações e contatos com seus responsáveis. Quando essas informações tiverem sido coletadas, as entidades de resgate serão modelados no ambiente do simulador e, posteriormente simuladas. A partir destes resultados, uma análise do funcionamento desses agentes será feita para determinar a melhor forma de integrá-los.

C. Fase 3 - Aplicação das técnicas de coordenação na integração das entidades de salvamento modeladas e análise dos resultados

Essa será a última e conclusiva fase do projeto, na qual serão integradas as estratégias de coordenação e comunicação criadas na fase 1 com o comportamento modelado na fase 2. Após realizada a junção das estratégias criadas e dos métodos de coordenação aplicados, o desempenho da solução final será avaliado quantitativamente através do simulador, analisando então quais foram os ganhos trazidos pela integração entre as diferentes equipes, e pelo uso de algumas técnicas adicionais de coordenação. Para realizar essa análise de desempenho, serão definidos parâmetros quantitativos referentes à simulação, como por exemplo o número de civis resgatados e o número de prédios que foram destruídos pelo fogo. Dessa forma, o resultado poderá ser expresso em porcentagem de melhora ou piora de cada um dos parâmetros selecionados.

Esta terceira fase é fundamental para o projeto, demonstrando seus resultados conclusivos: mostrar como o uso de algumas técnicas específicas de coordenação e a integração entre as diferentes entidades pode maximizar a eficiência no resgate a vítimas e na proteção do patrimônio da cidade em caso de desastre.

IV. Especificação

Esse trabalho utiliza como base o código do time implementado pela equipe do LTI, participante da competição de 2012. Adicionalmente, apresentamos quatro principais algoritmos de coordenação para melhorar o projeto existente: (i) o particionamento baseado em refúgios para os policiais, (ii) o desbloqueio preventivo, (iii) a inclusão de ações nos agentes de resgate ao escutar, na simulação, o pedido de ajuda dos civis, e (iv) a determinação da eficiência de um resgate de acordo com parâmetros pré-definidos. Outras técnicas serão estudadas para uma implementação futura; no entanto, devido ao curto tempo restante até a competição, estas não serão implementadas por completo para a competição desse ano.

É interessante ressaltar que como esses algoritmos são voltados para a competição *Robocup Rescue 2013*, estes nem sempre irão refletir de maneira idêntica a forma como os

agentes de resgate se comportariam em um ambiente real. Os algoritmos descritos abaixo visam exclusivamente aumentar o *score* final do time na competição.

A. Particionamento do mapa baseado em refúgios

Uma das tarefas mais difíceis na competição é bem distribuir os agentes de cada equipe, para que estes estejam sempre próximos aos focos de incêndio e às vítimas de soterramento, podendo chegar assim rapidamente a esses locais. Uma das técnicas utilizadas para a distribuição dos agentes pelo mapa é o particionamento do mapa, de modo que cada agente ou grupo de agentes fique restrito a um setor designado no mapa.

A ideia dessa técnica consiste em reservar áreas próximas aos refúgios para que um policial fique em constante ronda em torno de um refúgio. Os policiais alocados nessa tarefa terão como objetivo retirar os bloqueios encontrados nessa região, na tentativa de limpar a entrada e os principais caminhos próximos ao refúgio. Essa técnica será utilizada somente em mapas que contém mais do que um número mínimo definido de policiais disponíveis, pois deverá haver um número mínimo de policiais para efetuar um desbloqueio preventivo.

B. Desbloqueio preventivo

Dentro do ambiente do simulador, quando uma ambulância decide salvar um civil preso nos escombros, ela deve se locomover até o local e tirá-lo dos destroços. Para finalizar o processo de salvamento, a ambulância deve ainda transportar o civil machucado do local do acidente até um refugio para que ele possa ser atendido pelos médicos. Um dos grandes problemas encontrados durante esse processo é a presença de bloqueios entre a ambulância e o civil, e entre o civil e o refúgio. A presença desses bloqueios impede temporariamente a ambulância de alcançar o civil e/ou levá-lo até um refúgio, causando a morte do mesmo e consequentemente a diminuição do *score*.

Uma solução para o problema, utilizada atualmente, é chamar um agente de polícia no momento em que a ambulância encontra um bloqueio. No entanto, o tempo levado para o agente de polícia atingir o bloqueio e retirá-lo pode aumentar o tempo de chegada no local do acidente.

Dessa forma, é proposta a utilização de uma técnica de desbloqueio preventivo: uma ambulância, ao determinar que irá salvar um civil, envia uma mensagem de *broadcast* para todos os agentes de polícia. O agente de polícia mais próximo do civil começa então um processo de busca e remoção de bloqueios entre o civil e a ambulânci,a e entre o civil e o refúgio. Esse processo tem por objetivo principal aumentar as chances de sobrevivência do civil e, consequentemente, do nosso *score*.

C. Inclusão de ações ao escutar pedido de ajuda dos civis

No simulador, os agentes civis, quando soterrados, podem enviar uma mensagem pedindo ajuda para serem socorridos. Os agentes de salvamento próximos ao civil podem escutar essa mensagem e descobrir que existe uma pessoa em perigo naquele local. Essa informação pode então ser repassada aos outros agentes para que alguém vá em direção ao civil para resgatá-lo. Na solução implementada atualmente, os agentes não estão tomando ações baseados nessa mensagem, e precisam entrar nas casas para encontrar as vítimas, o que aumenta o tempo de salvamento. A implementação desse mecanismo pode ajudar a aumentar as chances de resgate e, consequentemente, melhorar o *score* do nosso time.

D. Determinação da eficiência de um resgate

Nem sempre é possível chegar no local do acidente a tempo de salvar um civil. Em certas situações, devido à presença de um incêndio, por exemplo, os civis morrem antes mesmo que a ambulância chegue ao local. Para diminuirmos o desperdício de recursos ao deslocar uma ambulância até o local do incêndio e não ser capaz de salvar o civil, uma solução é de ponderar a ida de uma ambulância a um local ao invés de outro pela eficiência calculada do resgate. Essa eficiência pode levar em conta, por exemplo, o tempo de deslocamento até o local da vítima, a presença de bloqueios conhecidos no caminho e a presença de incêndios nas proximidades do local onde o civil se encontra.

Como dito anteriormente, essa técnica visa somente aumentar o *score* da competição e não reflete necessariamente o comportamento real de uma ambulância. Em uma situação real, esse cálculo poderia levar em conta outros fatores que não estão presentes no simulador e, portanto, não podem ser incluídos na competição.

V. Conclusões

Neste trabalho, foi brevemente apresentado nosso projeto de um time de agentes para resgate de civis em situações de desastre. Descrevemos as suas etapas e as técnicas que ainda serão implementadas. Atualmente, o projeto encontra-se na primeira fase, onde foram realizados os estudos teóricos necessários e as principais estratégias a serem implementadas para a competição já foram definidas. Os próximos passos incluem a implementação e testes das estratégias acima citadas no simulador da competição, além da participação efetiva na competição em junho de 2013. Posteriormente, ao iniciar a segunda fase do projeto, serão realizadas visitas aos centros de coordenação e estudos sobre os protocolos de atuação das entidades de salvamento do município de São Paulo, com o intuito de modelar e analisar esses protocolos no simulador.

References

[1] D. Guha-Sapir, F. Vos, R. Below, and S. Ponserre, "Annual disaster statistical review 2010: The numbers and trends," Centre for Research on Epidemiology of Disasters, Tech. Rep., 2011. [Online]. Available: http://www.cred.be/sites/default/files/ADSR_2010.pdf

[2] M. Wooldridge, *An Introduction to Multiagent Systems*, 2nd ed. Chichester, UK: John Wiley & Sons Ltd., 2009.

[3] E. Durfee and V. Lesser, "Partial global planning: a coordination framework for distributed hypothesis formation," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 21, no. 5, pp. 1167–1183, 1991.

[4] A. H. Pereira, L. G. Nardin, and J. S. Sichman, "LTI agent rescue: A partial global approach for task allocation in the robocup rescue," *Revista de Informática Teórica e Aplicada*, vol. 19, no. 1, pp. 71–92, 2012.

[5] A. B. M. da Silva, L. G. Nardin, and J. S. Sichman, "Um método baseado em particionamento para exploração de ambientes de desastre," in *Anais do 9o. Encontro Nacional de Inteligência Artificial*. Curitiba, BR: Sociedade Brasileira de Computação, 2012.

Using DCOP to Model Resource Allocation: A Review of Algorithms

Alexander R. Gularte, Diana Francisca Adamatti
Programa de Pós Graduação em Computação
Centro de Ciências Computacionais
Universidade Federal do Rio Grande (FURG)
Rio Grande – RS – Brasil

***Abstract*—Distributed Constraint Optimization Problem (DCOP) is a formalism that is widely used for coordination in multiagent systems. The advantage of applying these algorithms for multiagent coordination is due to the fact that them are distributed, robust and scalable. The aim of this work is to present a revision of the complete and incomplete algorithms, generally found in the literature and how these approaches can benefit the resource allocation in Multiagent Systems.**

I. Introdução

Diversos formalismos podem ser utilizados para coordenação de Sistemas Multiagente (SMA), contudo os Problemas de Otimização de Restrição Distribuída (DCOP - *Distributed Constraint Optimization Problems*) vêm ganhando destaque na literatura por ser uma abordagem robusta e escalável. Os DCOPs estão associados a outros três formalismos: Problemas de Satisfação de Restrição (CSP - *Constraint Satisfaction Problem*)[Apt 2003], Problemas de Otimização de Restrição (COP - *Constraint Optimization Problems*)[Schiex et al. 1995] e Problemas de Satisfação de Restrição Distribuída (DCSP - *Distributed Constraint Satisfaction Problems*)[Yokoo et al. 1992].

Um CSP consiste em associar valores a variáveis, de tal forma que um conjunto de restrições é satisfeito. Cada uma das *n* variáveis do problema (X_1,X_2,...,X_n) assume um valor que pertence a um determinado domínio discreto. Um conjunto de domínios (D_1,D_2,...,D_n) especifica o domínio de cada uma das variáveis. Uma restrição é definida por um predicado. Uma restrição P_k = (xk_1,...xk_j) é um predicado definido sobre o produto cartesiano $Dk_1 \times \ldots \times Dk_j$. O predicado será verdadeiro se os valores associados às variáveis satisfazem a restrição. A solução para um CSP é equivalente a encontrar uma associação de valores para cada uma das variáveis envolvidas no problema de tal forma que todas as restrições sejam satisfeitas.

A extensão dos CSPs para coordenar a resolução cooperativa de problemas distribuídos em SMA foi proposta por [Yokoo et al. 1992]. Para tornar isso possível, as variáveis do CSP foram distribuídas entre os agentes. Essa abordagem pôde ser aplicada, por exemplo, na alocação de tarefas em SMA, considerando cada tarefa como uma variável e os domínios das variáveis como o conjunto de agentes capazes realizá-las.

O trabalho de [Schiex et al. 1995] propôs generalizar as funções booleanas dos CSPs em funções valoradas. Esses valores denotam o impacto causado pela violação da restrição. Além disso, permitem representar níveis de preferência em relação às possíveis associações de valores. O processo de resolução de um COP tende a ser mais complexo que um CSP, pois, não basta encontrar uma associação que satisfaz todas as restrições, é necessário encontrar a associação que otimiza o valor das funções de custo. Isso implica em uma maior exploração do espaço de estados.

Baseando-se nos formalismos mostrados anteriormente, diversos pesquisadores propuseram os DCOPs, que podem ser vistos tanto como uma distribuição dos COPs, como uma generalização dos DCSPs. Um DCOP é formalmente definido como a tupla (X , D , C , A, α), onde $X = \{x_1,x_2,\ldots,x_n\}$ é um conjunto de n variáveis, $D = \{D(x_1),D(x_2),...,D(x_n)\}$ um conjunto de domínios discretos no qual cada elemento corresponde ao domínio de uma variável, C um conjunto de funções de custo, A o conjunto de agentes e α o mapeamento de agentes e variáveis. Encontrar uma solução de custo mínimo para um DCOP é um problema NP-Hard [Modi et al. 2005].

Em [Yeoh 2010] foi descrita uma possível taxonomia para abordagens de solução para DCOPs presentes na literatura. Neste artigo é apresentada uma extensão da taxonomia de Yeoh onde os algoritmos incompletos são expandidos nos que tem garantia de qualidade e nos que não tem. A taxonomia estendida é mostrada na Figura 1.

Fig. 1. Taxonomia dos algoritmos para solucionar DCOPs.

Inicialmente, os algoritmos podem ser divididos em completos e incompletos. Os completos fornecem soluções ótimas ao custo de muita computação e trocas de mensagens. Por outro lado, os incompletos encontram soluções sem garantia de qualidade mais rapidamente.

Na classe dos algoritmos completos, os parcialmente centralizados compreendem os algoritmos nos quais alguns agentes transferem as suas informações de restrição (funções

de custo) para que outros agentes centralizadores realizem o processamento. O algoritmo mais conhecido dessa classe é o OptAPO (*Optimal Asynchronous Partial Overlay*) [Mailler e Lesser 2004]. O OptAPO apresenta uma abordagem baseada na cooperação de mediadores, onde alguns agentes atuam como mediadores resolvendo centralmente subproblemas que se sobrepõem.

Diferentemente dos parcialmente centralizados, nos descentralizados cada agente tem acesso apenas as suas restrições. Essa classe se divide em uma abordagem baseada em busca e outra em inferência. Os de inferência geralmente utilizam programação dinâmica para propagar as funções de custos de um agente para outro. O algoritmo mais referenciado na literatura que utiliza essa abordagem é o DPOP (*Distributed Pseudo-tree Optimization*) [Petcu e Faltings 2005]. Já os algoritmos baseados em busca empregam estratégias de busca distribuída para explorar o espaço de soluções até encontrar aquela que possui o custo mínimo.

A classe dos algoritmos incompletos tende a ter maior aplicação em problemas reais em virtude do reduzido custo computacional. Apesar de não possuírem garantia de encontrar a solução ótima, alguns fornecem limites de distância máxima entre a solução ótima e a aquela que pode ser encontrada. Os algoritmos sem garantia tendem a ser os mais rápidos para solucionar DCOPs, contudo não apresentam qualquer garantia em termos de convergência e qualidade da solução.

II. Algoritmos Completos

A. ADOPT - Asynchronous Distributed Constraint Optimization

O algoritmo ADOPT (*Asynchronous Distributed Optimization*), proposto por [Modi et al. 2005], se destaca como o primeiro método completo, distribuído, assíncrono e com comunicação local. Essa última característica demonstra a escalabilidade do método, pois um agente se comunica apenas com seus vizinhos. Dois agentes são considerados vizinhos sempre que existir ao menos uma função de custo entre eles. No ADOPT, a assincronia é alcançada à medida que os agentes podem mudar o valor de suas variáveis sempre que percebam a existência de uma solução melhor que a atual. Além disso, essas decisões são realizadas baseando-se em informações locais. A busca assíncrona no ADOPT é uma variante da busca em profundidade *Branch-and-Bound* (BB).

B. DPOP - Distributed Pseudo-tree Optimization

Diferentemente do ADOPT, o algoritmo DPOP (*Dynamic Programming Optimization Protocol for* DCOP) proposto em [Petcu e Faltings 2005] utiliza uma estratégia de inferência baseada em programação dinâmica. A principal vantagem do DPOP é utilizar um número linear de mensagens, o que reduz o *overhead* dos algoritmos de busca distribuída. A complexidade do DPOP depende do tamanho de suas mensagens de utilidade, que são limitadas exponencialmente.

O algoritmo DPOP é composto por três fases: criação da árvore DFS (*Depth First Search*), propagação das mensagens de utilidade e propagação das mensagens de valor. A primeira fase consiste em gerar a árvore DFS para o grafo de restrições do DCOP. Em seguida, ocorre um processo de propagação de utilidades que começa nas folhas e vai até a raiz. A ideia dessa fase é ir coletando informações de utilidade a fim de fornecer subsídios para os nós superiores escolherem os valores que proporcionam maior ganho de utilidade. Terminada essa fase, inicia-se um processo de propagação de valores, onde, desde a raiz, os nós começam a associar os valores ótimos a suas variáveis.

C. OptAPO - Optimal Asynchronous Partial Overlay

A maioria dos algoritmos para solução distribuída de DCOPs mantem uma separação total entre os conhecimentos dos agentes ao longo do processo de resolução. Contudo, em muitos casos, se os agentes pudessem compartilhar as suas funções de custo, soluções estáveis poderiam ser encontradas mais rapidamente. Motivado por esse inconveniente, [Mailler e Lesser 2004] apresenta um novo algoritmo baseado em uma técnica parcialmente centralizada denominada mediação cooperativa. Assim, o OptAPO fornece uma resolução rápida, distribuída e assíncrona sem gerar o *overhead* decorrente das comunicações excessivas. O OptAPO se propõe a explorar a velocidade das técnicas centralizadas, ao mesmo tempo que de forma distribuída consegue identificar as subestruturas do problema.

III. Algoritmos Incompletos

A. Algoritmo Soma-Máxima

Uma das formas de representar um DCOP é através de grafos bipartidos denominados grafos-fatores. Um grafo bipartido é composto por arestas não direcionais e dois conjuntos de nós. Nesses grafos, cada aresta conecta nós de conjuntos diferentes. No caso dos grafos-fatores, um conjunto de nós representa as variáveis das funções (nós de variável), enquanto os outros nós representam as funções (nós de funções). As arestas conectam as variáveis às funções sempre que uma variável for argumento para uma função.

[Farinelli et al. 2008] propôs representar um DCOP através de grafos-fatores e então aplicar o algoritmo soma-máxima, extensão do soma-produto [Kschischang et al. 2001], para encontrar soluções aproximadas através de troca de mensagens locais. Quando aplicado em um grafo-fator acíclico, o soma-máxima tem garantia de encontrar a solução ótima, por outro lado em grafos cíclicos essa garantia não existe. O soma-máxima tem a vantagem de apresentar ótima escalabilidade, uma vez que a complexidade no cálculo das mensagens depende apenas do número de vizinhos e não do número total de agentes.

O algoritmo soma-máxima opera através da troca de mensagens enviadas de nós de função para nós de variável e de nós de variável para nós de função. Essas mensagens tratam-se de funções de custo que ao longo da execução do algoritmo são somadas e margilizadas. Essas funções resumem todo custo existente na porção do grafo da qual elas provêm. O algoritmo termina quando todos os nós de variável tenham recebido todas as mensagens de seus vizinhos. Nesse momento, cada nó de variável terá subsidios para decidir qual melhor valor para assumir, uma vez que irá conhecer o custo existente em todas as direções ao seu redor.

B. Algoritmo Soma-Máxima Branch-and-Bound

O trabalho de [Stranders et al. 2009] identificou um gargalo em potencial nas mensagens que são enviadas das funções para as variáveis no soma-máxima. Para gerar essas mensagens é necessário produzir todas as possíveis combinações de valores para as variáveis. Contudo, o espaço de estados em questão cresce exponencialmente em relação ao número de variáveis e o número de possíveis valores/estados de cada variável. Com base nesse inconveniente, [Stranders et al. 2009] propõe dois algoritmos de poda. O primeiro deles é executado em uma etapa de pré-processamento, antes de executar o soma-máxima propriamente dito, e baseia-se na poda de estados dominados. Esses estados consistem em valores atribuídos às variáveis que comprovadamente não levam a uma solução ótima global. O segundo algoritmo emprega a técnica de busca *Branch-and-Bound* para otimizar a exploração de estados necessária ao gerar as mensagens de funções para variáveis. Estudos posteriores referenciam esses dois algoritmos como uma evolução do soma-máxima, que pode ser chamado de Soma-Máxima *Branch-and-Bound* [Macarthur 2011].

C. Algoritmo Soma-Máxima Limitado

O trabalho de [Rogers et al. 2011] propõe uma abordagem que fornece um limite entre a solução encontrada pelo soma-máxima e a ótima. Essa evolução foi chamada soma-máxima limitado (*bounded max-sum*). A ideia básica do soma-máxima limitado é remover os ciclos do grafo-fator para obter resultados ótimos com o algoritmo soma-máxima. Entretanto, para remover ciclos é necessário ignorar algumas dependências entre funções e variáveis (arestas do grafo-fator), o que acaba gerando um segundo problema, diferente do inicial representado pelo grafo cíclico. Não existe garantia de que a solução ótima para o grafo livre de ciclos será também uma solução ótima para o grafo cíclico. Contudo é possível estabelecer uma distância limite entre a solução ótima do grafo cíclico e aquela encontrada através do grafo acíclico. Uma etapa chave dessa abordagem é escolher as dependências que serão removidas, considerando o impacto que elas terão na qualidade da solução.

IV. Modelagem de Alocação de Tarefas com DCOP

Alocação de tarefas envolve associar um conjunto de tarefas a um conjunto de agentes, onde ambos variam com o tempo, ou seja, o ambiente é dinâmico. Cada agente recebe uma recompensa baseada em uma função de utilidade que determina o ganho para cada associação agente-tarefa. Assim, uma solução ótima será a associação cujo ganho total para todos os agentes seja maximizado.

Dentre as possíveis soluções para o problema de alocação de tarefas, os DCOPs se destacam por serem uma representação flexível o suficiente para representar as mudanças rápidas no ambiente sem a necessidade de criar uma representação completa do mesmo. [Macarthur 2011] utiliza essa abordagem através de grafos-fatores, onde os agentes são representados por nós de variável e as tarefas por nós de função. As principais contribuições do trabalho de Macarthur são três algoritmos para alocação dinâmica de tarefas: *fast-max-sum*, *branch-and-bound fast-max-sum* e *bounded fast-max-sum*, os quais foram desenvolvidos com base no algoritmo soma-máxima. Além desse trabalho, outros algoritmos para alocação de tarefas foram desenvolvidos também com base em DCOPs, por exemplo, o LA-DCOP [Scerri et al. 2005] e Swarm-GAP [Ferreira Jr et al. 2008].

V. Trabalhos Futuros

Pretende-se investigar domínios de problemas reais no qual a alocação de tarefas/recursos seja carente de abordagens distribuídas e eficientes. A partir disso, um determinado domínio será escolhido e modelado sob a perspectiva de sistemas multiagentes, adotando DCOP para realização da coordenação. Além disso, diferentes algoritmos serão avaliados a fim de encontrar o mais adequado para o domínio em questão.

References

[Apt 2003] Apt, K. (2003). *Principles of constraint programming*. Cambridge University Press.

[Farinelli et al. 2008] Farinelli, A., Rogers, A., Petcu, A., e Jennings, N. R. (2008). Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 2*, AAMAS '08, pages 639–646, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[Ferreira Jr et al. 2008] Ferreira Jr, P. R., Boffo, F. S., e Bazzan, A. L. (2008). Using swarm-gap for distributed task allocation in complex scenarios. In *Massively Multi-Agent Technology*, pages 107–121. Springer.

[Kschischang et al. 2001] Kschischang, F., Member, S., Frey, B. J., e andrea Loeliger, H. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519.

[Macarthur 2011] Macarthur, K. S. (2011). *Multi-Agent Coordination for Dynamic Decentralised Task Allocation*. PhD thesis, University of Southampton.

[Mailler e Lesser 2004] Mailler, R. e Lesser, V. (2004). Solving distributed constraint optimization problems using cooperative mediation. In *International Conference on Autonomous Agents: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-*, volume 1, pages 438–445.

[Modi et al. 2005] Modi, P. J., Shen, W.-M., Tambe, M., e Yokoo, M. (2005). Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1):149–180.

[Petcu e Faltings 2005] Petcu, A. e Faltings, B. (2005). A scalable method for multiagent constraint optimization. In *International Joint Conference on Artificial Intelligence*, volume 19, page 266. LAWRENCE ERLBAUM ASSOCIATES LTD.

[Rogers et al. 2011] Rogers, A., Farinelli, A., Stranders, R., e Jennings, N. R. (2011). Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 175(2):730–759.

[Scerri et al. 2005] Scerri, P., Farinelli, A., Okamoto, S., e Tambe, M. (2005). Allocating tasks in extreme teams. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 727–734. ACM.

[Schiex et al. 1995] Schiex, T., Fargier, H., Verfaillie, G., et al. (1995). Valued constraint satisfaction problems: Hard and easy problems. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 631–639. Citeseer.

[Stranders et al. 2009] Stranders, R., Farinelli, A., Rogers, A., e Jennings, N. R. (2009). Decentralised coordination of mobile sensors using the max-sum algorithm. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 299–304. Morgan Kaufmann Publishers Inc.

[Yeoh 2010] Yeoh, W. (2010). *SPEEDING UP DISTRIBUTED CONSTRAINT OPTIMIZATION SEARCH ALGORITHMS*. PhD thesis, UNIVERSITY OF SOUTHERN CALIFORNIA.

[Yokoo et al. 1992] Yokoo, M., Ishida, T., Durfee, E. H., e Kuwabara, K. (1992). Distributed constraint satisfaction for formalizing distributed problem solving. In *Distributed Computing Systems, 1992., Proceedings of the 12th International Conference on*, pages 614–621. IEEE.

Authors Index - Índice de Autores