Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science 292 (2013) 57–69

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

TAO+: Extending the Conceptual Framework TAO to Support Internal Agent Architectures in Normative Multi-Agent Systems

Emmanuel Sávio Silva Freire[a,1], Enyo José Tavares Gonçalves[b,2], Mariela Inés Cortés[a,3], Yrleyjânder Salmito Lopes[a,4] and Marcius Gomes Brandão[a,5]

Grupo de Engenharia de Software e Sistemas Inteligentes - GESSI/Brazil

[a] *Science Computer Department*
Universidade Estadual do Ceará - UECE
Fortaleza, Brazil

[b] *Science Computer Department*
Universidade Federal do Ceará - UFC
Quixadá, Brazil

Abstract

The existence of Normative Multi-Agent Systems, where the interaction between agents with different internal architectures is governed by norms, promotes the existence of an ontology suitable to define these related concepts. In this context, we highlight TAO, a conceptual framework for MAS, used as a foundation for the modeling language MAS-ML. However, TAO foresees limited support for representing the norm concepts and the different agent architectures. This paper describes the extension of TAO to allow the representation of these concepts.

Keywords: Normative Multi-Agent Systems; Conceptual framework; Template-Based Representation; Norms; Internal Architectures.

[1] Email: savio.essf@gmail.com

[2] Email: enyo@ufc.br

[3] Email: mariela@larces.uece.br

[4] Email: yrleyjander@gmail.br

[5] Email: marciusbrandao@gmail.br

1571-0661 © 2013 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2013.02.005

1 Introduction

An adequate definition of the conceptual framework is crucial to understand the business and elaborate a coherent computational solution in the context of the development project of complex systems [4]. Considering the diversity of concepts in Multi-Agent Systems (MAS), conceptual frameworks are critical to establish a base for modeling languages or agent oriented frameworks. The Taming Agents and Objects (TAO) [17] is a conceptual framework which provides the basis for software engineering methods based on agents and objects and is the basis of the modeling language MAS-ML [16].

The agents that compose a MAS can be designed according to different architectures [14] on the basis of proactivity and reactivity concepts. Each architecture is composed by specific structural and behavioural elements.

In order to cope with the heterogeneity, autonomy and diversity of interests among the different members, governance (or law enforcement) systems have been defined. The governance systems define a set of norms (or laws) that must be followed by the system entities [15]. In this context, norms are used to regulate the behavior of the agents in MAS by describing the actions that can be performed or states that can be achieved (permissions), actions that must be performed or states that must be achieved (obligations), and actions that cannot be performed or states that cannot be achieved (prohibitions). In addition, norms are used to cope with the autonomy, different interests and desires of the agents that cohabit the system [12]. The relation between norms and architectures are covered in recent works [2,13].

In TAO, the modeling of the norms concepts and agent architectures is limited thus, is required to adapt the concepts of TAO in order to support norm concepts along with other agent architectures defined in the literature [14]. This paper proposes the extension of TAO in order to represent the norm concepts in association with the internal agent architectures. The paper is structured as follows: the TAO, the norms for MAS and internal architectures are described in Section 2. The extension of the TAO is presented in Section 3. Section 4 presents a case study involving the proposed extension. Section 5 presents the related works. Finally, conclusions and future works are related in Section 6.

2 Background

2.1 *Taming Agents and Objects (TAO)*

The framework TAO (Taming Agents and Objects) provides an ontology that covers the fundamentals of Software Engineering based on agents and objects and supports the development of MAS in large-scale [17]. TAO presents the definition of each abstraction as a concept of its ontology, and establishes the relationships between them.

- Object: It is a passive or reactive element that has state and behavior and can be related to other elements.

- Agent: It is an autonomous, adaptive and interactive element that has a mental state. Its mental state has the following components: (i) beliefs, (ii) goals, (iii) plans and (iv) actions.
- Organization: It is an element that groups agents, which play roles and have common goals. It may restrict the behavior of their agents and their sub-organizations through the concept of axiom, which define the actions that must be performed.
- Object Role: It is an element that guides and restricts the behavior of an object in the organization. An object role can add information, behavior and relationships to the object instance that plays the role.
- Agent Role: It is an element that guides and restricts the behavior of an agent in the organization. An agent role defines (i) duties that define an action that must be performed by an agent, (ii) rights that define an action that can be performed by an agent and (iii) protocol that defines an interaction with the other elements.
- Environment: It is an element that represents the habitat for agents, objects and organizations. An environment can be heterogeneous, dynamic, open and distributed.

Additionally, Silva et al. [17] defined the following relationships in TAO: Inhabit, Ownership, Play, Specialization/Inheritance, Control, Dependency, Association and Aggregation/Composition.

2.2 Norms for Multi-Agent Systems

The norms are used to restrict the behavior of agents, organizations and sub-organizations during a period of time, and set sanctions to be applied if violated or fulfilled [15]. The main elements that compose the norm are [15]:

- Deontic concepts: deontic logic refers to the logical of requests, commands, rules, laws, moral principles and judgments. In MAS, such concepts are used to describe the constraints for the agent behavior (obligations, permissions and prohibitions).
- Involved Entities: norms are always defined to restrict the behavior of entities, thus the identification of the (group of) affected entities are essential.
- Actions: Specify the actions that are restricted by the norms. Such actions may be communicative actions, or non-communicative actions.
- Activation Constraints: Define the period in which the norms are active. Norms may be activated by a (set of) constraint, such as: the execution of actions, specifying time intervals, the reaching of system states or temporal aspects (such as dates) and also the activation/deactivation of other norm and fulfillment/violation of a norm.
- Sanctions: They are rewards or punishments that are applied when an entity fulfilled or violated a norm.
- Context: the norms are usually defined in a specific context that determines the application area, such as a particular environment or organization.

2.3 Internal Architectures

Internal architectures can be classified through the concepts of reactivity and proactivity. As reflex architectures, the simple reflex agents and model-based reflex agents can be highlighted, and like proactive architectures, goal-based agents and utility-based agents can be highlighted.

- Simple Reflex Agents [14]: condition-action rules are used to select actions based on the current perception. This architecture assumes that at any time the agent receives information from the environment through sensors. These perceptions consist of representations of the state through the aspects that are used by the agent to make the decision. A sub-system is responsible for processing the sequence of perceptions and selects the appropriate action from a set of possible actions for the agent. The agent performs the selected action through actuators.
- Model-based reflex agents: similar to simple reflex agents, model-based agents use condition-action rules to select their actions. Additionally, to deal with partially observable environments and to achieve a highest rational performance, this agent is able to store the current state of the environment in their internal state or model. A function called next function is introduced to map the perceptions and the current internal state to a new internal state, which is used to select the next action.
- Goal-based agents: they are based on models that are used to select a specific goal and the related actions that lead to the goal. This allows the agent to choose the state goal among the possibilities. The planning activity is dedicated to find out the sequence of actions that are capable of achieving the agent goals [14]. Therefore, the goal-based agent contains the next function and also includes the goal-formulation function, which receives the current state and returns the formulated goal and the problem-formulation function, which receives the state and the goal and returns the problem;
- Utility-based agents: considering the existence of multiple goal states, it is possible to set a measure of how a particular state is desired, in order to optimize the performance of the agent. Thus, the utility function is introduced in this architecture to measure the related utility according to the current goals [14]. The utility-based agents include the same elements of goal-based agents.

3 Extending the Framework TAO

The extension of TAO is focused on the representation of the internal architectures along with the static elements that compose a norm. The representation is done through templates used to define each abstraction in a schematic way. They list the set of properties and relationships of each element in the metamodel layer.

Internal Architecture	**Structural/ Mental Features**	**Behavioral Features**
Simple Reflex Agent	-	Perception and Action (oriented by Condition-Action Rules)
Model-based Reflex Agent	Goal and Belief	Perception, Next Fuction and Action (oriented by Condition-Action Rules)
BDI Agent	Goal and Belief	Plan and Action (oriented by the Plan chosen according to the Goal)
Goal-based Agent	Goal and Belief	Perception, Next Fuction, Goal-formulation Function, Problem-formulation function, Planning and Action
Utility-based Agent	Goal and Belief	Perception, Next Fuction, Goal-formulation Function, Problem-formulation function, Utility Function Planning, and Action

Table 1
New TAO Agent Features.

3.1 Representing the Internal Architectures

TAO describes an agent as an autonomous, adaptive and interactive element, where its structural features include goal and belief, and its behavioral features include action and plan [17]. However, depending on the agent internal architecture, some of these features cannot be modeled in the agent's behavior. Thus, the basic behavioral feature is defined by an action, structural, mental or behavioral aspects (Table 1).

Besides the conceptual approach, Silva et al. [17] show agents through templates. Thus, the templates used to represent the new agent types are in following:

(Simple Reflex) Agent

Agent_Class Agent_Class_Name
  Perceives setOf{Perceives_Name}
  Actions setOf{Action_Name}
  Events generated: setOf{Event_Name}, perceived: setOf{Event_Name}
  Roles setOf{Role_Class_Name}, Relationships setOf{R_Name}
end Agent_Class

(Model-based Reflex) Agent

Agent_Class Agent_Class_Name
  Beliefs setOf{Belief_Name}
  Perceives setOf{Perceives_Name}
  NextFunction setOf{NF_Name}

```
    Actions setOf{Action_Name}
    Events generated: setOf{Event_Name}, perceived: setOf{Event_Name}
    Roles setOf{Role_Class_Name}, Relationships setOf{R_Name}
end Agent_Class
```

(Goal-based) Agent

```
Agent_Class Agent_Class_Name
    Goals setOf{Goal_Name}
    Beliefs setOf{Belief_Name}
    Perceives setOf{Perceives_Name}
    Actions setOf{Action_Name}
    Planning setof{Planning_Name}
    NextFunction setOf{Function_Name}
    FormulateGoalFunction setOf{GF_Name}
    FormulateProblemFunction setOf{PF_Name}
    Events generated: setOf{Event_Name}, perceived: setOf{Event_Name}
    Roles setOf{Role_Class_Name}
    Relationships setOf{R_Name}
end Agent_Class
```

(Utility-based) Agent

```
Agent_Class Agent_Class_Name
    Goals setOf{Goal_Name}
    Beliefs setOf{Belief_Name}
    Perceives setOf{Perceives_Name}
    Actions setOf{Action_Name}
    Planning setof{Planning_Name}
    NextFunction setOf{Function_Name}
    FormulateGoalFunction setOf{GF_Name}
    FormulateProblemFunction setOf{PF_Name}
    UtilityFunction setof{UF_Name}
    Events generated: setOf{Event_Name}, perceived: setOf{Event_Name}
    Roles setOf{Role_Class_Name}
    Relationships setOf{R_Name}
end Agent_Class
```

3.2 Inclusion of Norm Concepts in TAO

In order to include the static concepts of a norm in TAO, the creation of a new abstraction Norm and its relationships is necessary. Considering the definition and properties of norms established in Section 2.2, a new metaclass to represent this concept is required in TAO. Since that a norm is an element with state and behavior properties, and specific relationships this concept is modelled by the new metaclass Norm. Additionally, three new relationships between the elements of the conceptual framework with new abstraction Norm are introduced to indicate others properties of norms.

- Context: it defines that the environment, organization, or sub-organization are application context of the norm. The behavior of all the elements related to the environment, organization and sub-organization will be governed by the norms.
- Restrict: it defines which entity will have their behavior constrained by the norm. If the entity fulfils or violates the norms, a sanction will be applied.
- SanctionReward: it specifies the reward that can be received by the entity that has fulfilled the norm.
- SanctionPunishment: it specifies the punishment that can be received by the entity that has violated the norm.

The relationship template is used to define the links between the elements. For each relationship type, the template identifies the elements and its roles in the relationship.

Relationship

```
Relationship Relationship_Name
    CONTEXT : context, norm
    | RESTRICT : element, norm
    | SANCTION_REWARD : reward, norm
    | SANCTION_PUNISHMENT : punishment, norm
End Relationship
```

The template for a norm presents a Norm class which defines its state as the resource to be restricted, the behavior of its instances as a set of its properties and relationships that are common to all norm instances. According to Section 2.2, the behavior of a norm is defined based on its characteristics related to the deontic concepts and activation constraints.

Norm

```
Norm_Class Norm_Class_Name
    Restriction_Type Deontic_Concept_Name
    Resource <Element_Class_Name.property>
    Activation_Constraint setOf{Constraint_Type Constraint_Type_Name:
        (<Element_Class_Name_First> and/or <Element_Class_Name_
        Second>) or <date> or <Element_Class_Name.property:
        Operator = [(Element_Class_Name.property) or (value)]>}
    Relationships setOf {Relationship_Name}
```

End Norm_Class

3.2.1 Inclusion of Norm in TAO's Abstractions

Adaptations on the already existent abstractions in TAO are necessary in order to incorporate the norm concept. The concepts of right and duty in the agent role used to define the actions that can and must be executed by agents are substituted by the deontic concepts of permission and obligation defined by norms. Thus, the concepts of right and duty have been replaced by norm concepts in the Agent Role template for Goal-based Agent and Utility-based Agent.

Agent_Role (Goal-based and Utility-based Agent)

Agent_Role_Class Agent_Role_Class_Name
  Goals setOf{Goal_Name}
  Beliefs setOf{Belief_Name}
  Actions setOf{Action_Name}
  Protocols setOf{Interaction_Class_Name} U setOf{Rule_Name}
  Commitments setOf{Action_Name}
  Relationships setOf{Relationship_Name}
End Agent_Role_Class

Differently, according to [14], a simple reflex agent has not beliefs and goals. In the case of model-based reflex agent, beliefs exist only as a structural feature (Section 2.3). Thus, the following template represents the architecture for the corresponding agent role.

An organization defines a set of rules and laws to characterize the global constraints that agents and sub-organizations must obey [17]. Considering their semantic equivalence these concepts was substituted by norm in organization.

Finally, the norm concept in the environment is added in order to the definition of access restrictions related to services and resources.

Organization

Organization_Class Organization_Class_Name
  Norms setOf{Norm_Name}
  Actions setOf{Action_Name}
  Relationships setOf{Relationship_Name}
end Organization_Class

Environment

Environment_Class Environment_Class_Name

Norms setOf{Norm_Name}
Behavior setOf{Properties}
Relationships setOf{Relationship_Name}
Events generated: setOf{Event_Name}, perceived: setOf{Event_Name}
end Environment_Class

---

4 Case Study: TAC-SCM

TAC-SCM (Supply Chain Management) is a highly dynamic, stochastic and strategic environment [18] that allows the realization of simultaneous auctions. The game describes the scenario of a supply chain for the assembly of personal computers, consisting of a computer manufacturing, suppliers that provide components for the assembly of these machines and clients who demand built computers [1]. The environment simulates a sequence of days in which the agents need manage the supply chain. Agents have a bank account with null initial balance. Every simulated days, clients submit requests for budgets and select the budgets submitted by clients based on delivery date and offer price [1].

The complete modelling of the TAC-SCM is available on `https://sites.google.com/site/uecegessi/case-study-tac-scm-tao`. The templates used to define each abstraction that correspond to the metamodel layer are instantiated in order to exemplify the use of the templates to model the domain model layer in the context of TAC-SCM.

4.1 Modeling Agents

Moreover, in TAC-SCM five types of agents are identified: DeliveryAgent, SellerAgent, BuyerAgent, SupplierAgent and ManagerAgent. Thus, the internal architecture of each agent was chosen according to the respective role in the game.

DeliveryAgent class must achieve the goal of delivering products to consumers. To achieve this goal, a sequence of actions should be performed. The DeliveryAgent class can be represented by the agent template originally proposed by TAO.

DeliveryAgent

Agent_Class DeliveryAgent
Beliefs {OrdersToDelivery}
Goals {DeliveryProductsToConsumer}
Actions{CheckCurrentDeliveryDate, CheckAvailableProducts,
DeliveryProduct}
Plans {Delivery}
Events generated: {}, perceived: {}
Roles {Shipper}
Relationships{Inhabit_TACEnvironment, Play_Shipper}
end Agent_Class

The ManagerAgent class is responsible for managing all staff and allocates the resources. This agent tries to maximize profits and sales, note that these goals can be conflicting. Thus, the most appropriate architecture in this case is based on utility.

ManagerAgent

Agent_Class ManagerAgent
 Goals {MaximizeProfit, MaximizeSales}
 Beliefs {SalesInformation, ShoppingInformation, ProductiveInformation}
 Perceives {BuyerRequirements, SellerRequirements,
  ProducerRequirements, ShipperRequirements}
 Actions {RequestBuyParts, RequestPcProduction, RequestPcSelling,
  EvaluateBuyingPrice, EvaluateShoppingPrice}
 Planning {ManagerPlanning}, NextFunction {ManagerNextFunction}
 FormulateGoalFunction {ManagerGoalFormulation}
 FormulateProblemFunction{ManagerProblemFormulation}
 UtilityFunction {ManagerUtilityFunction}
 Events generated: {}, perceived: {}
 Roles {Manager}
 Relationships {Inhabit_TACEnvironment, Play_Manager}
end Agent_Class

The roles for reflex agents (BuyerAgent and SellerAgent) are illustrated below.

Buyer

Agent_Role_Class Buyer
 Beliefs {AuctionPrice, StockMissing}
 Actions setOf {Buy, Pay, RequestPrices}
 Protocols {FIPA_Protocol}, Commitments {Shop}
 Relationships {Association_BuyerAgent}
end Agent_Role_Class

Seller

Agent_Role_Class Seller
 Actions setOf {OfferProducts, ReceivePayment}
 Protocols {FIPA_Protocol}, Commitments {MakeSales}
 Relationships{Association_SellerAgent}
end Agent_Role_Class

4.2 Definition of Norms for TAC-SCM

For the TAC-SCM were established the following norms along with their modeling:

- N1: General Store organization's buyers are required to pay for items they bought.
- N2 (Punishment): Buyers that have violated the norm N1 are forbidden to buy items.

N1

```
Norm_Class N1
    Restriction_Type Obligation
    Resource Buyer.payGood
    Activation_Constraint{after: Buyer.buyGood}
    Relationships setOf{Context_GeneralStore_N1, Restrict_Buyer_N1,
        Sanction_N2_N1}
End Norm_Class
```

N2

```
Norm_Class N2
    Restriction_Type Prohibition
    Resource Buyer.buyGood
    Relationships setOf{Context_GeneralStore_N2, Restrict_Buyer_N2}
End Norm_Class
```

5 Related Work

In this section, we compared TAO+ with others conceptual frameworks and organization models. This comparison considers the able to representing the structural and dynamic properties of the elements that compose a MAS along with the representation of the normative elements and the internal agent architectures defined in [14].

The framework proposed by D'Inverno and Luck [6] defines a hierarchy composed by four layers having entities, objects, agents and autonomous agents. However, it presents the following limitations: (i) no dynamic aspect associated with the proposed entities is defined (ii) it does not provide activation constraint for norms, and (iii) it not possible the definition of permission norms.

KAoS is a conceptual framework that defines abstractions, such as entities, relationships and agents, as object extensions [3]. We can mention some weak points for this framework: (i) KAoS does not consider organizations, roles and environments; (ii) it does not explain in a satisfactory way, the distinction between an entity and an agent; (iii) it does not describe the object features or explains how it is extended by other abstractions; (iv) it does not define any dynamic aspect associated with the entities and (v) it describes only policies to restrict the access to properties of its abstractions.

The organization model Moise+ [11] is based in model Moise [10] that presents an organization-centred view considering three forms to represent the organizational restricts (roles, plans and norms). Moise+ allows the description of permission and prohibition norms for roles in context of an organization. This model presents the following weak points: (i) it does not define the agent properties; (ii) it does not allow the norm specification for agents and environments, (iii) it does not support the definition of sanctions and (iv) even allowing the modeling of the heterogeneous agents, it does not support the definition of all internal agent architectures defined in [14].

The organization model OperA [5] is a framework that allows the specification of MAS through of the distinction between the characteristics of the organization model and the behaviour of the agents. OperA allows the description of norms related to obligation, permission and prohibition for agents, agent roles and agent groups in context of organization. Additionally, it allows the definitions of restrictions for norm activation. However, this organizational model: (i) it does allow the modeling the structural aspects of agent organizations, (ii) it does not support the definition of reward, only punishment and (iii) it does not allow to restrict the agent behaviour in context of an environment.

TAO+ allows the representation of the structural and dynamic properties of the entities of MAS in association with the static elements of the norm (deontic concepts, entities involved, actions, activation constraints, sanction and context) that governs the behavior of these entities. In addition, TAO+ represents of the all internal agent architectures defined in [14]. Thus, the normative multi-agent systems with different agent architectures can be modelled in TAO+.

6 Conclusion and Future Works

This paper presents the extension of the TAO in order to enable the representation of main internal agent architectures and the concepts related to norms in association with the already existent entities.

The extension proposed involves the creation of the abstraction Norm and four new relationships. The properties of each abstraction and relationships are specified through templates. Additionally, adjustments in the templates of already existent abstractions are proposed.

The specifications of the four internal architectures of agents were added to the conceptual framework. Additionally, the agent role concept also has changed in consistency with the new agent definitions. Finally, the case study is centred on the modeling of TAC-SCM through the instantiation of the templates, aiming to illustrate the adequacy of the proposed extension.

As future works, we consider the formalization of the proposed templates through the approach presented in [17]. This work can help to join of extensions of MAS-ML [16], related to agent norms [7,8] and internal agent architectures [9].

References

[1] Arunachalam, R., *The 2003 supply chain management trading agent competition*, Third International Joint Conference on Autonomous Agents & Multi Agent Systems. July 2004. [S.l.: s.n.]. p. 113-120.

[2] Campos, G. A. L., E. S. S. Freire and M. I. Cortés, *Norm-Based Behavior Modification in Reflex Agents*, 14th International Conference on Artificial Intelligence (ICAI@WORLDCOMP). Proceedings of the 14th International Conference on Artificial Intelligence, 2012, Las Vegas, USA.

[3] Dardenne, A., A. Lamsweerde and S. Fickas, *Goal-directed Requirements Acquisition.* Science of Computer Programming. v.20, p.3-50, 1993.

[4] Dieste, O., N. Juristo, A. Moreno and J. Pazos, *Conceptual Modeling in Software Engineering and Knowledge Engineering: Concepts, Techniques and Trends*, Chang, S.K. (eds.): Handbook of Software Engineering and Knowledge Engineering Fundamentals. World Scientific Publishing Co., 2001, Vol.**1**.

[5] Dignum, V., "A model for organizational interaction: based on agents, founded in logic, " PhD dissertation, Universiteit Utrecht, SIKS dissertation series 2004-1, 2004.

[6] D'Inverno, M. and M. Luck. "Understanding Agent Systems, " New York: Springer, 2004.

[7] Freire, E. S. S., M. I. Cortés, E. J. T. Goncalves and Y. S. Lopes, *A Modeling Language for Normative Multi-Agent Systems.* 13th International Workshop on Agent-Oriented Software Engineering (AOSE@AAMAS). Proceedings of the 13th International Workshop on Agent-Oriented Software Engineering, Valencia (Spain), 2012.

[8] Freire, E. S. S., M. I. Cortés, E. J. T. Goncalves and Y. S. Lopes, *NorMAS-ML: A Modeling Language to Model Normative Multi-Agent Systems.* 14th International Conference on Enterprise Information Systems (ICEIS). Proceedings of the 14th International Conference on Enterprise Information Systems, Wroclaw (Poland), 2012.

[9] Gonçalves, E. J. T., M. I. Cortés, G. A. L. Campos, G. F. Gomes and V. T. Silva, *Towards the Modeling Reactive and Proactive Agents by Using MAS-ML*, Special Track on Agent-Oriented Software Engineering Methodologies Infrastructures, and Processes (AOMIP) in 25th Annual ACM Symposium on Applied Computing (SAC 2010). Proceedings of the 25th Annual ACM Symposium on Applied Computing, Sierre, Suíça, 2010.

[10] Hannoun, M., "MOISE: un modèle organisationnel pour les systèmes multi-agents." Thèse(Doctorat), École Nacionale Supérieure des Mines de Saint-Etienne, 2002.

[11] Hübner, J. F., J. S. Sichman and B. Olivier, *A model for the structural, functional and deontic specification of organizations in multiagent systems*, Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence, 2002, Springer-Verlag London, UK.

[12] López y López, F., "Social Power and Norms: Impact on agent behavior," PhD thesis, University of Southampton, Faculty of Engineering and Applied Science, Department of Elec-tronics and Computer Science, 2003.

[13] Meneguzzi, F. and M. Luck, *Norm-based behaviour modification in BDI agents*, 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10-15, 2009, Budapest, Hungary.

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

[15] Silva, V. T., C. Braga and K. Figueiredo, *A Modeling Language to Model Norms*, The International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2010), 9th. Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, 2010, Toronto, Canadá.

[16] Silva, V., R. Choren and C. Lucena, *MAS-ML: A Multi-Agent System Modelling Language*, International Journal of Agent-Oriented Software Engineering, Interscience Publishers, 2008, vol.2, no.4.

[17] Silva, V., A. Garcia, A. Brandao, C. Chavez, C. Lucena and P. Alencar, *Taming Agents and Objects in Software Engineering*, 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, ISBN 978-3-540-08772-4.

[18] Wellman, M. P., P. Stone, A. Greenwald and P. R. Wurman, *The 2001 Trading Agent Competition*, IEEE Internet Computing, v. 13, p. 935-941, 2002.