Practical Reasoning in an Argumentation-based Decision BDI Agent: a Case Study for Participatory Management of Protected Areas

Pedro Elkind Velmovitsky[1], Jean Pierre Briot[2, 1], Marx Viana[1], Carlos Lucena[1]
[1] Laboratório de Engenharia de Software (LES), Pontifícia Universidade Católica (PUC-Rio), Rio de Janeiro, Brazil
[2] Sorbonne Universités, UPMC Univ Paris 06, CNRS, Laboratoire d'Informatique de Paris 6 (LIP6), Paris, France
{pvelmovitsky, mleles, lucena}@inf.puc-rio.br, jean-pierre.briot@lip6.fr, alessandro.sordoni@gmail.com

***Abstract*— This paper describes the implementation of an argumentation system used for participatory management of environmental protected areas, more precisely to model the decision of a park manager artificial agent. This implementation is based on a BDI agent architecture, namely the Jason/*AgentSpeak* framework/language. After introducing the principles of BDI architecture and of argumentation systems, we will detail how we model arguments within the BDI (Belief-Desire-Intention) architecture. Then, we present the argumentation-based model of deliberation and decision by the park manager agent as a case study. We show how our argument-based approach allows to model various cognitive profiles of park managers (more conservationist or more sensitive to social issues), through different knowledge bases. We show examples of decisions produced by the park manager agent and examples of traces of arguments used during deliberation, which could be a base for explaining decisions. Before concluding, we point out future directions, such as using argumentation as a basis for negotiation between various agents.**

Keywords. Agent architecture; BDI architecture; Argumentation; Decision; Participatory management.

I. INTRODUCTION

The general context of this work is an ongoing research project exploring computer support for *participatory management* of protected areas. We have designed a serious game (more precisely a role-playing game), inspired by real management of national parks in Brazil [2]. The objective is to train people about participatory management of protected areas, allowing players to explore *negotiation strategies* in order to address conflicting views: in these protected areas, the stakeholders (such as environmentalist, tourism operator, traditional population representative, etc.), discuss, negotiate and take decisions about *environment management*. In practice, these decisions are about the *type* (level) of *conservation* for each sub-part of the park. Example among predefined types are *Intangible* (full conservation) and *Extensive* (flexible indirect use of resources). A special role is the *park manager*, who acts as an arbitrator in the game, making a final decision about the types of conservation, justifying (or, at least, *explaining*) its decision to the stakeholders.

Our research project [2] explores the use of various advanced computer techniques as a support for *assistance* to the players. One of the approaches explored is the use of *argumentation* systems, as a support for *decision* and for *negotiation*, and last but not the least, for *explanation*. We use this approach to model the decision of the park manager, through an internal deliberation between arguments [5]. We may also model the decisions of other players (roles) and support automated negotiation between them (human or artificial, as for instance in [8]).

In this paper, we will describe the modeling and implementation of an artificial agent playing the role of the park manager, based on an argumentation system (based on [3]), and modeled and implemented in a *BDI agent architecture*. More precisely, we are using the *Jason* framework, an implementation of the *AgentSpeak* agent programming language, based on the BDI architecture [4]. Said another way, we are representing arguments and their management in *AgentSpeak*. The first version of the implementation has already been completed and tested. We will report on our experience as well as pointing out future directions. The reminder of this paper is organized as follows: Section II discusses related work, while section III focuses on the background for BDI and for the Jason platform. Section IV presents the use of argumentation theory in BDI, Section IV presents the modeling and implementation of a BDI agent using argumentation and Section VI results. Finally, Section VII presents our conclusions and future work.

II. RELATED WORK

JogoMan-ViP [8] is a distributed role-playing game similar in spirit to our SimParc project, as its domain is about participatory management of hydric resources. It has introduced artificial players, also implemented through *AgentSpeak*/BDI on top of the Jason framework. This pioneering work has been an important influence. That said, the model of decision of artificial players is relatively simplified and with a predefined and fixed protocol for negotiation (to facilitate the interface between artificial players and human players). Our objective in using argumentation as the basis for decision, explanation and negotiation is even more ambitious.

DOI reference number: 10.18293/SEKE2017-153

Simulación by [12] experimented with the use of artificial agents as assistants of human players, as an innovative and proactive type of interface. Assistant agents can make suggestions to human players, based on the model of a good strategy for the game combined with a learning mechanism. This work has also been an inspiration for our project. But we want to explore various models of the game, including *predictive* models (to estimate viability and resilience) and inner deliberation and justification models (by using *argumentation*).

Regarding argumentation systems, the theoretical framework by Rahwan and Amgoud [3] is interesting and a major source of inspiration, in that it is one of the first to use argumentation not only for *epistemic* reasoning (i.e., to create or modify knowledge) but also to *practical* reasoning (i.e., for reasoning about what to do and also how to do it). [13] is an implementation of an argumentation system based in Prolog. [14] is a preliminary integration of an argumentation system into a BDI (*AgentSpeak*/Jason architecture), with similar objective to ours, but focusing on the implementation of defeasible logic (logical rules that can be refuted).

III. BDI AND JASON

There are many ways to model the behavior of agents such as the BDI (Belief-Desire-Intention) model. To talk about this model, according to [4], we need to address the idea that we can talk about computer programs as if they had a "*mental state*". Thus, when we talk about a BDI system, we are talking about computer programs with computational analogues of beliefs, desires and intentions.

Beliefs are the information the agent has about the environment. This information, however, is something the agent believes in but it may not be necessarily true. As an example, an agent may perceive from its environment the fact that it is raining. However, the rain may stop before the next reasoning cycle of the agent – in this case, his belief is outdated and incorrect. *Desires* are the possible states of affairs that the agent might like to accomplish. That does not mean, however, that the agent will act upon it – it is a potential influencer of the agent's actions. *Intentions* are the state of affairs that the agent has decided to act upon. In other words, intentions can be considered as a selected option between the potential set of options/desires that the agent has decided to pursue.

Jason is a Java-based interpreter for the *AgentSpeak* language [4], providing a platform for the implementation and development of agents. This language is based on the BDI architecture [15] and allows programmers to customize the agent's knowledge base following logic sentences. The agents also have *goals* that express the wishes the agent wants to accomplish. For instance*, !buy(car)* means that the agent has the goal of buying a car. Furthermore, *AgentSpeak* provides a way to program and customize *plans* for the agents. These plans represent courses of action that the agents will take in order to achieve its goals. The overall syntax for a plan is *triggering event <- context: body*, where the *triggering event* represent changes which the agent will act upon; the *context* is used to check the current situation so as to determine whether a particular plan among various alternative ones is likely to succeed in handling the event; the *body* of a plan is the course of action the agent will take in order to handle the event that triggered the plan [4].

IV. ARGUMENTATION IN BDI

There have been several theories which look at formalizing the reasoning of autonomous agents based on mental attitudes, such as beliefs, desires and intentions (BDI). One of the main characteristics of this type of reasoning is the resolution of *conflicts*, since the goals and attitudes available to the agent may not always be compatible. In addition, the information that the agent has may not always be consistent, or it may be true at one moment but incorrect on the next one [3] and may be different for *another* agent. *Argumentation* is a promising approach to deal with such considerations: it is a mono-agent as well as a multi-agent process, in which an agent may decide alone, or adhere to the opinion of another agent, depending on the strength and validity of arguments. Furthermore, agents reserve the right to *revisit* their *opinions* in light of new information.

The classical logic proves inadequate to model such behaviors. For example, to verify the property of monotony: *If Φ, Δ and Γ denote sets of formulas in a formal reasoning system, then the property of monotony is stated as following: If Φ :- Γ and $\Phi \subset \Delta$ then Δ :- Γ.* The key here is that the addition of new formulas at Φ can never call into question the truth value of Γ; this is called a *closed world*. The interest in non-monotonic logic appears when we try to capture the notion of everyday reasoning, where definite conclusions are obtained from incomplete information which can be proven wrong or false; this is called an *open world* and *non-monotonous reasoning*, where the addition of new formulas at Φ can call into question the truth of Γ.

However, this logical approach is limited to *epistemic reasoning* and does not modulate *practical reasoning*, limiting its use in agent architectures. To deal with this, a new approach has been developed: the *argument*. As opposed to a proof, an argument may be invalidated, and by comparing arguments it is possible to manage inconsistencies in the agent's belief base. In order to formalize this notion, we refer to concepts used in [3] and [5]: *Let the classical deduction be denoted by :- and the logical equivalence $\equiv$. Then, an argument is a pair ⟨H, h⟩, such that 1) H is consistent; 2) H :- h; 3) H is minimal, that is, there is no subset of H which verifies 1 and 2.* From this definition, the authors define the attack relations *refute* and *block* between arguments: *Let ⟨H1, h1⟩ and ⟨H2, h2⟩ two arguments. ⟨H1, h1⟩* ***refute*** *⟨H2, h2⟩ iff $h1 \equiv \neg h2$; ⟨H1, h1⟩* block *⟨H2, h2⟩ iff $\exists h \in H2, h \equiv \neg h1$.*

V. MODELING AND IMPLEMENTATING MANAGER AGENT

To model the park manager, or other players in the game, along their respective roles, we used the formalism above. So, let us note *D* the set of desires, *B* the set of beliefs and *A* the set of actions. Let us suppose, for example, that: *B = {road, tourism_flow, beach}* and *A = {extensive_use, intangible_use}*. Each agent, then, will have the following rules and bases:

1) The *rules to generate desires* (later on, named *desires rules)* RD_i have the form: ϕ -: $\beta_1, ..., \beta_m, \phi_1, ..., \phi_n, \beta_i \in B$ *and* $\phi_i \in D$. If the agent has beliefs $\beta_1, ..., \beta_m$ and desires $\phi_1, ..., \phi_n$ then desire ϕ is satisfied. These rules belong to the base $BD = \{(RD_i, w_i)\}$, where RD_i represents a desire rule and w_i represents

the intensity (weight) of the conclusion of the desire ϕ. An example is: *+road : tourism_flow & beach <-+raise_tourism(3),* where *(3)* means that *Intensity* $_{(raise\_tourism)} = 3$;

2) The *decision rules RAj* have the form ϕ *-:* α, where $\alpha \in A$ and $\phi \in D$. If the agent takes the decision of performing action α, then desire ϕ is satisfied. These rules belong to the base $BA = \{(RAj, u_{jj}\}$, where *RAj* represents a rule of desire and u_j represents the utility of the action α according to the desire ϕ. An example is: *+extensive_use(0.75) -> +raise_tourism(0.75)*, with $Utility_{raise\_tourism}$ *(extensive_use) = 0.75*.

It is important to note that, for an agent to decide which action to choose, he must compare the gain of each action. These gains are defined as $gain_{\phi}(\alpha) = intensity(\phi) * u_{\phi}(\alpha)$. (see more details in [5]).

TABLE I. DESIRE RULES (RD BASE) AND DECISION RULES (RA BASE)

Name	Rule
RD_1	*+road : tourism_flow & beach <-+raise_tourism(3)*
RD_2	*+waterfall : true <- +~raise_tourism(2)*
RD_3	*+forest : true <- + protect_forest(3)*
RD_4	*+forest_fire: true <- +prevent_fire(3)*
RD_5	*+beach : ~protect_forest(A) <-+protect_forest_argument(C)*
RA_1	*+extensive_use(0.75) : true <- !raise_tourism_ extensive_ use (0.75)*
RA_2	*+intangible_use(1.0):true<-!protect_forest_intangible_use(1.0); !prevent_fire_intangible_use(1.0/2); !protect_forest_prevent_fire_intangible_use(1.0)*

Suppose that the available actions are *extensive_use* and *intangible_use,* and the agent has the following beliefs about the area: *{"road","waterfall", "forest", "beach", "tourism_ flow", "forest_ fire"}*. Then, desire rules (*RD* base) and decision rules (RA base) are modeled as shown in Table I. Table II shows the *Plans* that are executed when the actions *extensive_use* and *intangible_use* become available. The *raise_tourism* desire related to *extensive_use* has utility *0.75*, the *protect_forest* and *prevent_fire* desires related to *intangible_us*e have utilities *1.0* and *0.5*, respectively.

It is important to note that the values of *intensity* and *utility* are modeled by the programmer, according to the manager's "personality". If this particular park manager is more *socioconservationist*, he may prefer a more *extensive use* of the area and therefore the utility from the desire *raise_tourism* may be bigger. The same applies if the park manager is more *preservationist* and therefore prefers a more *intangible use* of area.

With the rules and parameters mentioned above, there are two attack relations between explanatory arguments: refute attack between RD1 and RD2, since RD1's conclusion is *raise_tourism*, and RD2's conclusion is *~raise_tourism*; block attack between RD3 and RD5, since RD3's conclusion *protect_forest* is present in the body of RD5.

The park manager agent BDI, with its respective *RD* and *RA bases*, has been implemented utilizing the Jason plug-in for *Eclipse*. Then, an implementation of the mechanism to compare arguments in order to eliminate conflicts has been modeled in this architecture. For instance, in Table II, *!raise _tourism _extensive _ use (T)* is used to check if the *raise_tourism* desire was added from the *RD* in P_1 and P_2: since the agent has a rule that adds the *raise_tourism* desire, P_1 will be executed. If the agent's belief base did not include a *road*, for instance, then the *raise_tourism* desire would not be added from RD_1, and the agent would not believe that this desire is feasible – that is what the *not raise_tourism(A)* clause means. In this case, P_2 would be executed. The goals *!protect_forest_intangible_use(T)* and *!prevent_ fire_ intangible_ use(T)* are used in a similar fashion, and are not described here.

The *protect_forest_argument(C)* goal is used to check if a block attack relation exists in P_3 and P_4: in P_3, if the intensity of the negation of the *protect_forest desire*, denoted by *C*, is bigger than the intensity of the desire itself, denoted by *A*, then the *protect_forest* desire is removed. In P_4, if *C* is smaller than *A*, then the desire is not removed. In this implementation, *C* has been set to *4* and *A* to *3*, so P_3 applies and the desire is removed. The refute attack relation, on the other hand, is tested in P_5 and P_6. In P_5, if the intensity of the negation of the *raise_tourism* desire, denoted by *A*, is bigger than the intensity of the desire itself, denoted by *B*, then the *raise_tourism* desire is removed. In P_6, if A is smaller than *B*, then the desire is not removed. In this implementation, *A* has been set to *2* and *B* to *3*. So, P_6 applies and the desire is not removed. Lastly, the plan *!protect_forest_prevent_fire_intangible_use(T)* is used to consolidate all results. Since this goal is part of the body of the plan of the *intangible_use* action, its context needs to test if the desires *protect_forest, prevent_fire* and *raise_tourism* are available and if the action *extensive_use* is available. Then, it checks the gains of each action accordingly and selects an action to perform. For instance, P_7 and P_8 test if the desire *protect_forest* is present in the belief base and if the other desires are still feasible, if the *extensive_use* action is still available and if the gains of the *intangible_use* action are bigger than the gains of the *extensive_use* action. Similar clauses, omitted here, have been added for all possible combinations in the context, so that the agent will always know how to calculate the gains and choose an action. More details about the *AgentSpeak* language can be found in [4].

VI. RESULTS

The results of the implementation in Jason, as logged by the agent, are: 1) *extensive_use* action added from desire *raise_tourism*, with utility *0.75* and total gain *2.25*; 2) *raise_tourism* negation added with intensity *2* and desire *raise_tourism* with intensity *3* has bigger force; 3) *protect_forest* negation added with intensity *4* and desire *protect_forest* with intensity *3* removed; 4) *intangible_use* action added from desire *prevent_fire* with utility *0.5* and total gain *1.5*; 5) *extensive_use* action executed with total gain *2.25*, because *2.25* is bigger than *1.5*. Thus, the resulting decision by the park manager is *extensive_use.*

Suppose that we want to change the profile of the park manager from socioconservationist to preservationist. Then, the intensity of *~protect_forest* will be slightly adjusted from 4 to 2, causing the desire *protect_forest* to not be removed in the attack relation. The total utility of the intangible_use action will be 3*0.5 (*prevent_fire* desire) + 3*1.0 (*protect_forest* desire) = 4.5, surpassing the utility of the *extensive_use* action. The result, as traced by the agent, would be as following: 1) *extensive_use* action added from desire *raise_ tourism*, with utility *0.75* and total gain *2.25*; 2) *raise_tourism* negation

added with intensity *2* and desire *raise_tourism* with intensity *3* has bigger force; 3*) protect_forest* negation added with intensity *2* and desire *protect_forest* has bigger force *3*; 4) *intangible_use* action added from desire *protect_forest* with utility *1* and total gain *3*; 5) *intangible_use* action added from desire *prevent_fire* with utility *0.5* and total gain *1.5*; 6) *intangible_use* action executed with total gain *4.5*, because 2.25 is bigger than 2.25. Thus, the resulting decision is *intangible_use*.

This example shows the flexibility of the BDI model that makes use of the argumentation: it allows for the agent to deliberate and decide while eliminating conflicts between the bases. In this case, the conflict between the *protect_forest* desire and its negation is resolved by comparing their respective forces.

TABLE II. PLANS

Name	Plan
P_1	*+!raise_tourism_extensive_use(T): raise_tourism(A) <- .print (" extensive_use action added from desire raise_tourism, with utility ",D, " and total gain ", D*T)*
P_2	*+!raise_tourism_extensive_use(T) : not raise_tourism(A) <- .print ("extensive_use action added from desire raise_tourism, with utility ",D, " and total gain ", 0)*
P_3	*+protect_forest_argument(C) : protect_forest(A) & C>A <- .print ("protect_forest negation added with intensity ",C, " and desire protect_forest with intensity ", A, " removed");- protect_forest(A)*
P_4	*+protect_forest_argument(C) : protect_forest(A) & C<A <- .print ("protect_forest negation added with intensity ",C, " and desire protect_forest has bigger force ", A)*
P_5	*+~raise_tourism(A): raise_tourism(B) & A>B <- .print("raise_tourism negation added with intensity ", A, " and desire raise_tourism with intensity ",B, " removed");- raise_tourism(B)*
P_6	*+~raise_tourism(A): raise_tourism(B) & A<B <- .print("raise_tourism negation added with intensity ", A, " and desire raise_tourism with intensity ",B, " has bigger force")*
P_7	*+!protect_forest_prevent_fire_intangible_use(A) : not protect_ forest(E) & prevent_fire(D) & raise_tourism(C) & extensive_use (B) & C*B<=(D*A/2) <- .print("intangible_use action executed with total gain: ",(D*A/2),", because ", (D*A/2), " is bigger than ", C*B)*
P_8	*+!protect_forest_prevent_fire_intangible_use(A) : not protect_ forest(E) & prevent_fire(D) & raise_tourism(C) & extensive_use (B) & C*B>(D*A/2) <- .print("extensive_use action executed with total gain: ",C*B,", because ", C*B , " is bigger than ", (D*A/2))*

VII. CONCLUSION AND FUTURE WORK

In this paper, we have presented a prototype architecture of an artificial agent able to make decisions by comparing arguments. The architecture is based on a BDI architecture (namely, the Jason/*AgentSpeak* framework/language). We have modeled an argumentation system through different knowledge base layers and the relation of attacks between arguments, as a basis to select best viable arguments. We have tested the architecture by modeling a park manager in a serious game for participatory management of protected areas. The park manager artificial agent makes decisions about conservation types by examining and reasoning (comparing) facts and arguments about the protected area situation and concerns. As a future work, already under progress, we plan to completely automate the management of attacks between arguments, in order to increase the agent's autonomy and reasoning capacity. In addition, being able to track the agent's reasoning to choose an argument over another enables the agent to provide the user feedback about what is happening in the system. Last, argumentation could also be used, not only for internal deliberation within a single agent, but also for negotiation between agents by exchanging and evaluating arguments. This leads to the prospects of artificial negotiating players.

REFERENCES

[1] Hocine, N.; Gouaich, A.. A survey of agent programming and adaptive serious games. RR-11013, 2011, pp. 8. http://hal-lirmm.ccsd.cnrs.fr/lirmm-00577722

[2] Briot, J.-P., Irving, M. D. A., Vasconcelos Filho, J. E., de Melo, G. M., Alvarez, I., Sordoni, A., de Lucena, C. J. P. Participatory Management of Protected Areas for Biodiversity Conservation and Social Inclusion. In D. Adamatti (Ed), Multi-Agent Based Simulations Applied to Biological and Environmental Systems, Advances in Computational Intelligence and Robotics (ACIR) Book Series, IGI-Global, pp. 295–332.

[3] Rahwan, I.; Amgoud, L. An argumentation based approach for practical reasoning. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems. ACM, 2006. p. 347-354.

[4] Bordini, R. H.; Hübner, J. F.; Wooldridge, M. Programming multi-agent systems in AgentSpeak using Jason. John Wiley & Sons, 2007.

[5] Sordoni, A., Briot, J.-P., Alvarez, I., Vasconcelos, J., Irving, M., Melo, G. Design of a participatory decision making agent architecture based on argumentation and influence function – Application to a serious game about biodiversity conservation, RAIRO – An International Journal on Operations Research, 44(4):269–284. 2010.

[6] Barreteau, O. *et al*. Our companion modelling approach. Journal of Artificial Societies and Social Simulation, v. 6, n. 1, 2003.

[7] Le Page, C. *et al.* Participatory agent-based simulation for renewable resource management: the role of the Cormas simulation platform to nurture a community of practice. Journal of Artificial Societies and Social Simulation, v. 15, n. 1, p. 10, 2012.

[8] Adamatti, D. F.; Sichman, J. S.; Coelho, H. An analysis of the insertion of virtual players in GMABS methodology using the Vip-Jogoman prototype. Journal of Artificial Societies and Social Simulation, v. 12, n. 3, p. 7, 2009.

[9] Barreteau, O. *et al*. Role-playing games for opening the black box of multi-agent systems: method and lessons of its application to Senegal River Valley irrigated systems. 2001.

[10] Dignum, F. *et al*. Games and agents: Designing intelligent gameplay. International Journal of Computer Games Technology, v. 2009, 2009.

[11] Wei, W.; Alvarez, I.; Martin, S. Sustainability Analysis: Viability Concepts to Consider Transient and Asymptotical Dynamics in Socio-Ecological Tourism-based Systems. Ecological Modelling, V. 251, P. 103-113, 2013.

[12] Guyot, P.; Honiden, S. Agent-Based Participatory Simulations: Merging Multi-Agent Systems and Role-Playing Games. Journal of Artificial Societies and Social Simulation, V. 9, N. 4, 2006.

[13] Kakas, A.; Moraitis, P. Argumentation based decision making for autonomous agents. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems. ACM, 2003. p. 883-890.

[14] Panisson, A. R. *et al.* An Approach for Argumentation-based Reasoning Using Defeasible Logic in Multi-Agent Programming Languages. In: 11th International Workshop on Argumentation in Multiagent Systems. 2014.

[15] Wooldridge, M.; Jennings, N. R.; Kinny, D. A methodology for agent-oriented analysis and design. In: Proceedings of the Third International Conference on Autonomous Agents Agents 99. 1999.