Integrating a voting protocol within an argumentation-based BDI System

Cecilia Sosa Toranzo, Federico Schlesinger, Edgardo Ferretti, and Marcelo Errecalde

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional
Universidad Nacional de San Luis. San Luis, Argentina.
{ceciliasosatoranzo,fedest}@gmail.com,
{ferretti,merreca}@unsl.edu.ar

Abstract. BDI models are powerful tools that can play a fundamental role in implementing intelligent systems for complex business and industrial problems. Besides, it has also been recognized the benefits achieved when integrating BDI models with different reasoning formalisms, like for instance argumentation or case-based reasoning. Several multi-agent systems have been proposed where voting-based protocols have proven to be efficient mechanisms to lead to a coordinated social result. Likewise, there are several works where these protocols have also been applied as internal processes that arise in one agent's mind. Following these trends, the main contribution of this work it to integrate voting jointly with argumentation into a BDI system to implement the agent's deliberative aptitudes. All the concepts involved in this proposal are exemplified by working with a *travel assistant* agent developed with freely available technologies.

1 Introduction

Within the field of Computer Science, decision making problems have been mainly tackled from the research field of *Artificial Intelligence* (AI). As stated in [1], it is very relevant to develop and formalize decision-making mechanisms but these should not exist in a vacuum, and its is highly recommendable to integrated them in coherent ways into agent models developed in the broader field of AI. Likewise, in [2] two main trends which are currently influencing decision-making research in AI were distinguished: *Classical Decision Theory* on the one hand, and cognitively-oriented approaches such as *Practical Reasoning* or *Beliefs-Desires-Intentions* (*BDI*) settings on the other hand.

The BDI model have gained considerable attention as a powerful tool to build artificial intelligent systems, and is considered as one of the most influential *practical reasoning architectures*. It has very solid theoretical foundations [3] and its use is not only restricted to the academic area. Recent works have shown its potential to deal with very complex business and industrial problems [4, 5]. Besides, different studies [6–8] have recognized the benefits achieved when integrating BDI models with different reasoning formalisms, like case-based reasoning [9] or argumentation [10].

Several platforms [11–14] have been proposed to develop systems with multiple cognitive agents using the BDI approach, as well as others that allow to develop hybrid systems [15–17]. Independently of the agents type, when more than two agents interact in a common environment, it is necessary to have mechanisms that allow that their interactions lead to a coordinated social result. In this regard, the development of artificial multi-agent systems is usually based on high-level interaction mechanisms frequently used by humans beings, like voting, negotiation, auction among others. These mechanisms differ in the social result achieved as well as the applicable interaction protocols.

Several of these mechanisms originally conceived for multi-agent systems, have also been applied as internal processes that arise in the agent's mind. One approach that has received growing attention is the one that considers the agent's decisions as the result of a voting process among its inner components [18–20]. In this case, the agent's possible actions (or decisions) become the alternatives and its components act as voters. Following these trends, in this paper a formerly developed BDI system [21, 22] is extended by integrating voting to implement the agent's decision-making policy.

In [21, 22] the *WADEX* (*W*eb services, *A*rgumentation and (BDI) Ja*dex*) generic framework was proposed to provide support to integrate argumentation-based inference and web service technologies into the design of a BDI system. In this framework, argumentation was used in two of the three main processes of its control structure; the alternatives' acceptability filter and in the selection process. The feasibility of this integration was shown by implementing a *travel assistant* agent intended to help a human user that wanted to go on a trip to an undetermined destination in USA. A device called *decision rules* [23] was used to perform the decision process. Although this device was used together with an argumentation formalism [24] which has an optimized interpreter [25], the response times noticeably increase with respect to the number of alternatives to be considered by the agent and the number of preference criteria it has been provided. In this regard, other works [20, 26] have shown that voting-based decision making is a quite efficient mechanism to manage a great number of alternatives to be compared according to large number of preference criteria. In this way, implementing the selection process of WADEX agent's control structure by using a voting-based mechanism, combines the advantages in efficiency of this approach and it also maintains the ease of argumentation to handle the alternatives' acceptability.

As mentioned above, BDI models have been largely integrated with different reasoning engines to implement the agent's deliberative aptitudes. As far as we know, this is the first work which particularly uses voting jointly with argumentation as such a mechanism.

The rest of the paper is organized as follows. Section 2 briefly introduces the *voting-based protocol* used in this work. In Sect. 3 the WADEX framework is proposed to integrate argumentation and voting in the framework's decision component. Then, an example showing a concrete application of the framework is explained in Sect. 4. Finally, in Sect. 5 some general conclusions are drawn and possible future work is put forward.

2 Voting Overview

In a classic voting situation, there exist a set $\mathscr{C}$ of candidate alternatives and a set A of agents. Each agent $i \in A$ has a rational preference relation $\succsim_i$ defined over $\mathscr{C}$. The aim of a voting-based mechanism is to determine a social choice rule that given as inputs the agents' individual preferences computes a social preference relation $\succsim_A$, with certain desirable properties. These properties[1] were initially stated by Arrow [27] who proved from his *Impossibility theorem* that there was no social choice rule capable of simultaneously satisfy them all. In turn, Straffin [28] put forward desirable criteria (Pareto, Condorcet winner, Condorcet loser, Monotonicity and Majority) to be fulfilled by the results obtained with different voting rules.

An interesting voting rule proposed by Borda in 1781, known as *Borda count*, specifies that voters should provide a preference ranking for the n considered alternatives. An alternative receives 0 points if it is the last one in the voter's ranking, 1 point if it is the second to last until $n-1$ points if it is the first one. Then, for each alternative all the voters' points are added, and the alternative with highest score becomes the winner. This rule does not satisfy the Condorcet winner and Majority criteria formerly mentioned. To solve this matter, Duncan Black proposed a simple rule which states that the Condorcet winner should be chosen, if it does exist, otherwise the Borda count should be computed. This simple approach known as *rule of Black* satisfies all the above-mentioned criteria and is the voting protocol integrated in this work to the WADEX framework, as described in the following section.

3 The WADEX Framework

The *WADEX* framework [21, 22] conceptually consists of: (a) an *integration architecture* which describes how the software components implementing these models interact, and (b) a *general control structure* that provides some general guidelines about how the main steps involved in a particular application can be programmed.

The *integration architecture* is depicted in Fig. 1, where the agent platform and the communication channels between the different elements involved are shown. Most of the application logic is encapsulated in a Jadex agent. This agent runs on top of a deployed JADE platform. There are other agents also running on the same JADE platform, which can be of diverse nature and can communicate through messages with the main Jadex agent, since Jadex provides facilities for sending and receiving messages from plans, whether in a synchronous or asynchronous way. A JADE agent with special characteristics is the Web Services Integration Gateway (WSIG) agent, which allows to extend these advantages of messaging to the use of invoking web services as well.

A web service invocation process starts with a lookup for the service in the *Directory Facilitator* (*DF*), a JADE special agent implementing the *yellow pages* service. Once the service has been located, the agent formats a request message in the body of a plan and sends it. Information obtained from the web service invocation can be stored in the agent's belief base. Moreover, our Jadex agent can make queries to a DeLP-Server. The DeLP-Server runs a DeLP program containing part of the facts and decision logic which are supposed to be static. The possibility of including contextual information in the queries allows to have a piece of the DeLP program in the Jadex agent and / or dynamically generate it (or a part of it). In this case, queries are directly made from the

[1]Pareto efficiency, Nondictatorial existence, Independence of the irrelevant alternatives, among others (see [27] for a detailed description).

executing code of the Jadex agent sending queries (via sockets) to the DeLP-Server, and receiving responses in the same plan without any agent messages involved.

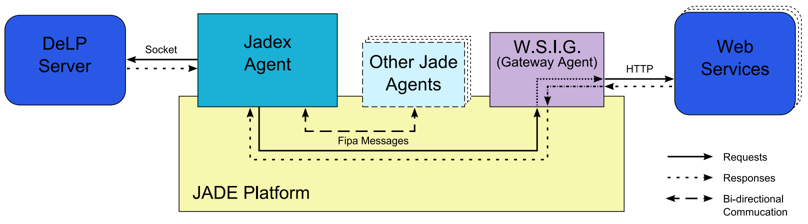


Fig. 1. The integration architecture

As the general application logic is controlled by a (BDI) Jadex agent which acts as initiator of all the interactions requiring distinct web services and inference processes, is important to analyze the three generic processes (the *acceptability filter*, the *selection process*, and the *execution stage*) involved in the *WADEX's general control structure*.

The *acceptability filter*, is the process which considers the whole set of available alternatives and discards those that do not satisfy essential requirements to be considered acceptable options. The *selection process* takes as input the options that survive the acceptability filter and compares them in order to decide which alternative will be selected. Three approaches can be used to face this issue: (a) to simply delegate this responsibility to the user, (b) making decision in an automatic way, or (c) using an hybrid approach mixing manual and automatic decisions. This latter approach is the one used in this work. Finally, the *execution stage* involves deciding *how* to accomplish the selected option and executing this plan to achieve the desired goals. At present, this stage involves to select and execute plans from a library of pre-compiled plans, following the standard approach adopted in most PRS-based BDI systems.

The plan for calculating the acceptable options takes each potential destination and sets up a sub-goal for calculating its acceptability. At present, the process which considers the whole set of cities and discards those that do not satisfy essential requirements to be acceptable options, only uses a web service to obtain its weather forecast.

The web service invoked by our application corresponds to the National Weather Service of the United States' National Oceanic and Atmospheric Administration.[2] This web service is relatively straightforward to use, and provides extensive information in a well structured manner. Among the set of its available functions, it was only used the one called `NDFDgen`, which receives as parameters two real numbers representing latitude and longitude of the geographical point, two dates stating the period of time for which the forecast is required, and a series of weather parameters indicating the required information. Only the most common weather parameters, such as minimum and maximum temperature, and probability of rain, were requested.

The agent has a goal that states the need of weather forecast information for a city (given as a parameter to the goal). When the service has been located, the plan sets up

[2]http://www.nws.noaa.gov/forecasts/xml/.

the request for the web service, taking the latitude and longitude from the parameter, and sends it to the Gateway Agent. After getting forecast information from the web service, the argumentation process takes place to determine the acceptability of a city. Our application relies on a DeLP program that runs in an instance of a DeLP-Server, and encapsulates the logic to decide whether a city is an acceptable option or not. The acceptability of a city `x` in our DeLP program, will be given by a warranted argument supporting the conclusion `acc(x)`.

The candidate cities are stored in the agent's belief base, in the form of Java objects containing the city name, its coordinates and an associated hash table to store the weather-related information of that city. A default set of cities is specified in the agent's definition but more cities can be dynamically added.

When the set of acceptable alternatives has been determined, a sub-goal is set up for computing the rule of Black, the voting-based decision process which compare them among each other, as previously described in Sect. 2. When computing this rule, the Java objects related to the acceptable cities are used along with those referring to the dimensions which will be used to compare them. Both processes, filtering acceptable cities and comparing among acceptable cities to select one, have to reflect the user's restrictions and preferences. These are captured through an screen with options that is presented to the user when the agent is loaded, and they are automatically translated into their internal representations.

The necessary steps to travel to the selected city can depend on many factors. Our approach consists of having a set of plans with different courses of action to achieve this goal, and use meta-level reasoning to choose the most adequate. A central aspect of BDI architectures is the possibility of reconsidering intentions. In our example, it is assumed that once the agent has selected a city, the necessary steps for travelling to that city take a considerable amount of time, during which some conditions might change and make the agent discard the selected city and choose a different one. We consider here two possible situations: (a) the selected city ceases to be an acceptable option, and (b) a new option arrives and is considered to be a better option than the current one.

The first case could happen if for example the weather forecast changes. When a change is detected, the agent issues a sub-goal to recheck the acceptability of that city. If the city has turned unacceptable, the goal to perform the necessary steps to travel to the selected city must be dropped, and the main plan has to start over, recheck acceptability for all the cities and select a new one. The second situation might arise for example, if the agent receives information of a sale in flight tickets for a given destination different from the chosen one. In this case, the next step would be to issue a goal to check the acceptability of the new option. If the new option is acceptable, the agent issues the goal to select a city from a set of acceptable cities, but this time only the currently selected option and the new one are passed as parameters to the goal, to select among these two. If the new option is chosen over the current one, the goal for performing steps to travel to the selected city must be dropped. The main plan has to be repeated, but this time there is no need to check acceptability for the other cities and selecting one, as the selected city will now be the newly arrived option. So the main plan now begins from the moment of starting to carry out the trip (*i.e.*, the third abstract step).

4 Travel Assistant Agent Domain

Let us suppose that John (our tourist) is in New York, and the weekend of November 12th wants to visit another city of USA. New York is the only city he has visited, and he has no particular preference for visiting any other specific city. Since he is leaving USA on November 14th from JFK airport, he has only two days to travel. This fact, maybe a reason for considering only New York's nearby cities. Besides, John has no much extra money to spend in expensive flight tickets or accommodation; therefore, if there is an offer of a cheap flight for a distant city, this might change his mind about travelling only to nearby cities. John is not a hard-to-please person, but as he has little time for travelling he really wants to visit a place with nice weather to be able of visiting as much as possible from that place.

As discussed in Sect. 3, a DeLP-program will be used to determine the acceptability of the cities. In Fig. 2 an excerption of this program is shown. As it can be observed, rules (1) and (2) are defeasible rules stating that, by default, a city is acceptable if it is a nearby city, and is not recommendable if it is distant. Rules (3)−(5) exemplify more complex considerations: a distant city can be acceptable if there is a cheap flight to that city, but a city with bad weather is not acceptable, whether it is a nearby or a distant city with a cheap flight to it. The definition of what constitutes a *nearby* or a *distant* city is given by rules (6) and (7) (in this case, the limit was fixed to 500 Km).

Next, rules (8)−(12) define the predicate `bad_weather`, exemplifying a way in which the agent could deal with partial and potentially contradictory information. Bad weather is first defined in terms of the forecast (information obtained from the web service) using strict rules. However, forecast information might not be available for some reason, like the date for the trip being very distant from the current date. The last two rules in the extract define bad weather in terms of climate: for example, one might suppose that there is a high probability of very low temperatures in Boston during winter, or that Miami usually has good weather. These last rules are defeasible, because forecast information could contradict them (*e.g.*, the forecast could announce a tropical storm heading to Miami).

```
[...]
(1)  acc(X) -< nearby_city(X).
(2)  ~acc(X) -< distant_city(X).
(3)  acc(X) -< distant_city(X), travel_date(Y), origin_city(Z), cheap_flight(Z,X,Y).
(4)  ~acc(X) -< nearby_city(X), travel_date(Y), bad_weather(X,Y).
(5)  ~acc(X) -< distant_city(X), travel_date(Y), origin_city(Z),
cheap_flight(Z,X,Y), bad_weather(X,Y).
[...]
(6)  nearby_city(X) <- origin_city(Z), distance(Z,X,K), K < 500.
(7)  distant_city(X) <- origin_city(Z), distance(Z,X,K), K >= 500.
[...]
(8)  bad_weather(X,Y) <- forecast(X,Y,rain).
(9)  bad_weather(X,Y) <- forecast(X,Y,snow).
(10) bad_weather(X,Y) <- forecast(X,Y,min_temp(K)), K < 0.
(11) bad_weather(X,Y) -< bad_climate(X,Y).
(12) ~bad_weather(X,Y) -< good_climate(X,Y).
[...]
```

Fig. 2. DeLP-program for determining a city's acceptability

Given the situation described at the beginning of this section, an example of contextual information might include the facts shown in Fig. 3. With this context, the set of acceptable alternatives $\mathscr{C} = \{Miami, Washington, Chicago\}$ will be used as input to the selection process.[3] *Boston* is judged not acceptable, even when it is a nearby city, since bad weather is assumed from the climate information due to the lack of weather forecast for that city. *Miami* and *Chicago* are distant cities but the agent is aware of cheap flight offers from New York to them, which is not the case with *Los Angeles*.

```
origin_city(new_york).                   forecast(miami,f11_12_10, min_temp(20)).
travel_date(f11_12_10).                  forecast(washington,f11_12_10,min_temp(12)).
distance(new_york, boston, 300).         forecast(chicago,f11_12_10,min_temp(5)).
distance(new_york, miami, 1747).         forecast(los_angeles,f11_12_10,min_temp(18)).
distance(new_york, washington, 328).     bad_climate(boston, Y) <- winter(Y).
distance(new_york, chicago, 1140).       cheap_flight(new_york, miami, f11_12_10).
distance(new_york, los_angeles, 3921).   cheap_flight(new_york, chicago, f11_12_10).
```

Fig. 3. Contextual information for the acceptability filter

In the selection process, it is considered a set of names of dimensions (preference criteria) denoted $\mathscr{D} = \{D_1, D_2, \ldots, D_n\}$. For each dimension $D_i \in \mathscr{D}$, the agent maintains the set O_{D_i} of options available to dimension D_i, and the user's preferences respect to D_i are represented by the function $P_{D_i} : O_{D_i} \mapsto [0,1]$. Each preference criterion (dimension) is represented as a Java object containing all its related information. In this particular scenario, we assume that when John interacted with the options screen he also stated that enjoys cities with vivid nightlife and nice cityscapes plenty of skyscrapers and bridges but containing galleries and museums to visit during the day, as well. Therefore, $\mathscr{D} = \{GM, C, N\}$ where GM, C and N refer to *Galleries and Museums*, *Cityscapes* and *Nightlife*, respectively. Besides, a function $P_{\mathscr{D}} : \mathscr{D} \mapsto [0,1]$ will also be computed to represent the user's preferences within dimensions. In this particular example, the values $P_{\mathscr{D}}(GM) = 0.7$, $P_{\mathscr{D}}(C) = 0.5$ and $P_{\mathscr{D}}(N) = 0.8$, are to be used to represent the above-mentioned preferences.

To determine the Condorcet winner, if such a candidate exists, it should be chosen that alternative which would beat each of the other ones in a run-off election. The first step consists of calculating the number of votes assigned to each dimension. This amount is given by the function $V_{\mathscr{D}} : \mathscr{D} \mapsto \mathbb{R}$ defined as follows:

$$V_{\mathscr{D}}(D_i) = \frac{P_{\mathscr{D}}(D_i)}{\sum\limits_{D_j \in \mathscr{D}} P_{\mathscr{D}}(D_j)} \times K \tag{1}$$

where K is an arbitrary constant used to scale the number of votes per dimension. In this way, if K is set to 100 the number of votes will be given by $V_{\mathscr{D}}(GM) = 35$, $V_{\mathscr{D}}(C) = 25$ and $V_{\mathscr{D}}(N) = 40$, respectively.

Secondly, to perform the run-off election, the numbers of votes received by the alternatives with respect to each dimension in $\mathscr{D}$, should be computed. To determine if the votes corresponding to dimension D_i are given to one alternative or the other, the

[3]Due to space restrictions, the argumentation process that determined this set is not included.

strengths of the alternatives with respect to dimension D_i are compared. This strength is defined by function $f_{D_i} : \mathscr{A} \mapsto \mathbb{R}$. In this particular setting, $f_{D_i}(a_j) = P_{D_i}(a_j(D_i))$, where alternative a_j is considered a mapping $a_j : \mathscr{D} \mapsto \mathscr{O}$, such that $\mathscr{O} = O_{D_1} \cup O_{D_2} \cup \ldots \cup O_{D_n}$ and $a_j(D_i) \in O_{D_i}$.

In this way, an alternative a_j is preferred over another alternative a_k with respect to dimension D_i, if and only if $f_{D_i}(a_j) > f_{D_i}(a_k)$, and is denoted as $a_j \succ_{D_i} a_k$. The total number of votes that an alternative a_j obtains in a run-off election with a_k is calculated as the sum of the votes associated to those dimensions where a_j is preferred to a_k. If $V_{par}(a_j, a_k)$ represents this value, and $\mathscr{D}_{a_k}^{a_j} = \{D_i \in \mathscr{D} | a_j \succ_{D_i} a_k\}$, V_{par} can be defined as a function $V_{par} : \mathscr{A} \times \mathscr{A} \mapsto \mathbb{R}$, as shown below:

$$V_{par}(a_j, a_k) = \sum_{D_i \in \mathscr{D}_{a_k}^{a_j}} V_{\mathscr{D}}(D_i) \qquad (2)$$

It may happen that two alternatives have the same strength with respect to a particular dimension, in which case none of the alternatives will receive its corresponding votes. The total number of votes regarding those dimension where alternatives a_j and a_k have the same strength is to be denoted $\tilde{V}_{par}(a_j, a_k)$. Thus, an alternative a_j is a Condorcet winner, if and only if $V_{par}(a_j, a_k) > V_{par}(a_k, a_j)$ for all alternative a_k $(k \neq j)$.

Figure 4 shows a graph representing the run-off elections among the three acceptable alternatives in $\mathscr{C}$. Arcs are labelled with the $V_{par}(a_i, a_j)$-$\tilde{V}_{par}(a_i, a_j)$-$V_{par}(a_j, a_i)$ values, while the their orientations refer to the winners (origin) and losers (end) respectively. As it can be observed in Fig. 4(b), there is no Condorcet winner, so the Borda count should be computed. In Fig. 4(a), the alternatives are presented indicating the attribute values they have for each dimension *GM*, *C* and *N*, respectively. The attributes values presented have been discretized in order to have a finite domain of options O_{D_i}. For dimension *GM*, the quality of the galleries and museums has been rated from *one-star* to *five-star* values. Similarly, for dimensions *C* and *N*, the values *regular*, *great* and *outstanding* have been used to qualify the cityscapes, as well as *good*, *groovy* and *spectacular* for depicting the nightlife of the city, respectively. Finally, the P_{D_i} values used for each dimension were:

$$\begin{array}{lll}
P_{GM}(five\text{-}star) = 1, & P_{GM}(four\text{-}star) = 0.8, & P_{GM}(three\text{-}star) = 0.6, \\
P_{GM}(two\text{-}star) = 0.4, & P_{GM}(one\text{-}star) = 0.2, & P_C(regular) = 0.5, \\
P_C(great) = 0.7, & P_C(outstanding) = 0.9, & P_N(good) = 0.6, \\
P_N(groovy) = 0.75, & P_W(spectacular) = 0.95 &
\end{array}$$

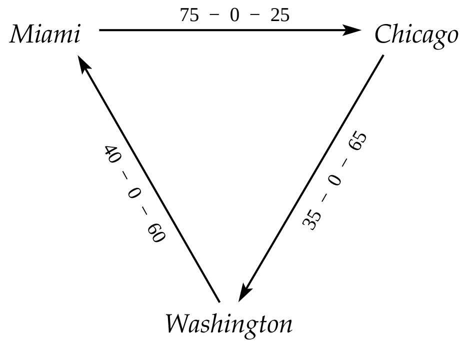
The columns BR_{GM}, BR_C and BR_N of the table shown in Fig. 4(a) contain the Borda ranking for each alternative with respect to the *Galleries and Museums*, *Cityscapes* and *Nightlife* criteria, respectively. Then, the total amount of votes received by each alternative is computed as:

$$\begin{array}{l}
V_{\mathscr{B}}(Miami) = 1 \times 35 + 2 \times 40 = 115 \\
V_{\mathscr{B}}(Chicago) = 2 \times 25 + 1 \times 40 = 90 \\
V_{\mathscr{B}}(Washington) = 2 \times 35 + 1 \times 25 = 95
\end{array}$$

In this way, the alternative selected by Borda count is *Miami*, which provides a good balance among nightlife and daylight activities, *e.g.*, visiting galleries and museums (as indicated by John when interacting with the options screen).

Alternative	*MG*	BR_{MG}	*C*	BR_C	*N*	BR_N
Miami	*four-star*	1	*regular*	0	*spectacular*	2
Chicago	*three-star*	0	*outstanding*	2	*groovy*	1
Washington	*five-star*	2	*great*	1	*good*	0

(a)



(b)

Fig. 4. Alternatives descriptions and results from: (a) Borda count and (b) run-off elections

5 Conclusions and Future Work

In this work, it was presented a proposal to effectively integrate into a BDI system, argumentation and voting-based approaches to implement the agent's deliberative aptitudes. This task was performed by using only freely available resources and other wide-spread technologies such as Web Services and FIPA-compliant agent development platforms.

All the concepts involved were exemplified by working with the *travel assistant* agent implemented in [22], so as to focuss our work on the integration of a voting-based approach to implement the agent's deliberative aptitudes in WADEX. Despite the fact that no efficiency tests were performed, a similar mechanism has already been tested within this respect in an earlier work of Errecalde *et al.* [20]. Nonetheless, as further work this performance test will be conducted. Likewise, other performance improvements of WADEX will be tested like comparing the functioning of the current acceptability filter against the approach of acceptability thresholds used in [20].

At the moment, the authors are modifying WADEX to implement hardware agents. As first step, it has been planned to work with a *Khepera* 2 robot which will use as perception the information provided by an external camera instead of web services.

References

1. Doyle, J., Thomason, R.: Background to qualitative decision theory. AI Magazine **20**(2) (1999)
2. Amgoud, L., Prade, H.: Using arguments for making and explaining decisions. Artificial Intelligence **173**(3-4) (March 2009) 413–436
3. Bratman, M.E., Israel, D.J., Pollack, M.E.: Plans and resource-bounded practical reasoning. Computational Intelligence **4**(4) (1988) 349–355

4. Benfield, S.S., Hendrickson, J., Galanti, D.: Making a strong business case for multiagent technology. In: Proceedings of AAMAS '06, New York, NY, USA, ACM (2006) 10–15
5. Evertsz, R., Fletcher, M., Jones, R., Jarvis, J., Brusey, J., Dance, S.: Implementing Industrial Multi-agent Systems Using JACK. In: Programming Multi-Agent Systems. Springer (2004)
6. Corchado, J.M., Laza, R.: Constructing deliberative agents with case-based reasoning technology. International Journal of Intelligent Systems **18**(12) (December 2003) 1227–1241
7. Rahwan, I., Amgoud, L.: An Argumentation-based Approach for Practical Reasoning. In: Proceedings of AAMAS '06, New York, NY, USA, ACM (2006) 347–354
8. Rotstein, N.D., García, A.J., Simari, G.R.: Reasoning from desires to intentions: a dialectical framework. In: Proceedings of AAAI'07, AAAI Press (2007) 136–141
9. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. Artificial Intelligence Communications **7**(1) (1994)
10. Rahwan, I., Simari, G., eds.: Argumentation in Artificial Intelligence. Springer (2009)
11. Hindriks, K.V., Boer, F.S.D., Hoek, W.V.D., Meyer, J.J.C.: Agent Programming in 3APL. Autonomous Agents and Multi-Agent Systems **2**(4) (1999) 357–401
12. d' Inverno, M., Luck, M., Georgeff, M., Kinny, D., Wooldridge, M.: The dMARS architechure: A specification of the distributed multi-agent reasoning system. Journal of Autonomous Agents and Multi-Agent Systems **1-2**(9) (2004) 5–53
13. Winikoff, M.: Jack^{TM} Intelligent Agents: An Industrial Strength Platform. In: Multi-Agent Programming: Languages, Platforms and Applications. Volume 15. Springer (2005) 175–193
14. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. John Wiley (2007)
15. Rimassa, G., Greenwood, D., Kernland, M.E.: The Living Systems Technology Suite: An autonomous middleware for autonomic computing. In: International Conference on Autonomic and Autonomous Systems (ICAS). (2006)
16. CogniTeam: CogniTAO: A JAUS-based high-level control system for single and multiple robots (2008)
17. NASA Ames Research Center: Brahms agent environment (2005)
18. Pirjanian, P., Christensen, H.I., Fayman, J.A.: Application of voting to fusion of purposive modules: An experimental investigation. Robotics and Autonomous Systems **23**(4) (1998)
19. Rosenblatt, J.K.: DAMN: A Distributed Architecture for Mobile Navigation. PhD thesis, Carnegie Mellon University. (1997)
20. Errecalde, M., Aguirre, G., González, F.: Agentes y mecanismos de votación. In: X Congreso Argentino de Ciencias de la Computacin (CACIC). (2004) 1474–1485
21. Schlesinger, F., Errecalde, M., Aguirre, G.: An approach to integrate web services and argumentation into a bdi system. In: 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2010) 1371–1372 (Extended Abstract).
22. Schlesinger, F., Ferretti, E., Errecalde, M., Aguirre, G.: An Argumentation-based BDI Personal Assistant. In: 23rd IEA-AIE. Volume 6096 of LNAI., Springer (2010) 701–710
23. Ferretti, E., Errecalde, M.L., García, A.J., Simari, G.R.: Decision rules and arguments in defeasible decision making. In: 2nd Intl. Conference on Computational Models of Arguments (COMMA). Frontiers in Artificial Intelligence and Applications, IOS Press (2008) 171–182
24. García, A.J., Simari, G.R.: Defeasible Logic Programming: An Argumentative Approach. Theory and Practice of Logic Programming **4**(2) (2004) 95–138
25. García, A.J., Rotstein, N.D., Tucat, M., Simari, G.R.: An argumentative reasoning service for deliberative agents. In: 2nd KSEM. Volume 4798 of LNCS., Springer (2007) 128–139
26. Sen, S., Haynes, T., Arora, N.: Satisfying user preferences while negotiating meetings. International Journal of Human-Computer Studies **47**(3) (1997) 407–427
27. Arrow, K.J.: Social choice and Individual Values. John Wiley & Sons, Inc. (1963)
28. Straffin, P.D.: Topics in the Theory of Voting. Birkhauser (1980)