HAL
archives-ouvertes.fr

Exploring the Design Space of a Declarative Framework for Automated Negotiation: Initial Considerations

Alex Muscar, Costin Bădică

▶ To cite this version:

Alex Muscar, Costin Bădică. Exploring the Design Space of a Declarative Framework for Automated Negotiation: Initial Considerations. 8th International Conference on Artificial Intelligence Applications and Innovations (AIAI), Sep 2012, Halkidiki, Greece. pp.264-273, 10.1007/978-3-642-33409-2_28 . hal-01521394

HAL Id: hal-01521394
https://hal.inria.fr/hal-01521394

Submitted on 11 May 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

Exploring the Design Space of a Declarative Framework for Automated Negotiation: Initial Considerations*

Alex Muscar and Costin Bădică

University of Craiova, Blvd. Decebal, nr. 107, RO-200440, Craiova, Romania,
{amuscar,cbadica}@software.ucv.ro

Abstract. In this paper we present our results on the exploration of the design space of a declarative framework for automated negotiation by: (1) identifying a minimal yet viable generic negotiation protocol and a declarative way of representing rules and constraints specific to negotiation mechanisms and strategies; (2) identifying the need of a set of basic concepts for describing negotiations that form a core negotiation ontology that the agents can use to reason about the negotiations; (3) proposing Belief-Desire-Intention (BDI) agents as an implementation model of our framework. We introduce both a conceptual framework for declarative specification of automated negotiations as well as a prototype implementation using the Jason agent programming language.

1 Introduction

An essential feature of agents is their ability to communicate by exchanging meaningful information for improving their own goals and society goals. Sometimes agents behave selfishly and their goals are only partly overlapping or can even be in conflict. In such cases agents must interact to reach an agreement between all or some of the involved agents. These interactive processes are broadly called negotiations. Negotiations carried out between computer systems are known as automated negotiations.

There are two components of negotiation: mechanism and strategy. The *negotiation mechanism* states the interaction rules that must be obeyed by the participants in order to meet some objectives. The mechanism, sometimes known as protocol, is public. The *negotiation strategy* describes the behavior of a negotiation participant and it is directed towards reaching his or her private goals, usually to maximize his or her gain [1], [2].

Many models of automated negotiations were proposed by researchers. The models can be parameterized according to different criteria, like: number and roles of negotiation participants, structure and properties of the negotiation subject, presence or not of a mediator, a.o [3]. Some well-known automated negotiation models are: bargaining, auctions [4], multi-criteria negotiation [5], multi-commodity negotiations.

Based on our literature survey, we observed an important deficiency of existing research approaches to automated negotiation: *they do not properly address the reusability*

*This work was supported by the strategic grant POSDRU / CPP107 / DMI1.5/S/78421, Project ID 78421 (2010), co-financed by the European Social Fund – Investing in People, within the Sectoral Operational Programme Human Resources Development 2007 – 2013, as well as by the *K-SWAN: An Interoperable Knowledge-based Framework for Negotiating Semantic Web Agents* Bilateral Research Project Greece – Romania, co-financed by UEFISCDI.

and extensibility of the research results which are so much required for their usefulness in open environments such as the Internet and the Web. Specific research questions immediately derived from this general problem are: (1) how can we define a reusable representation of automated negotiation protocols and strategies?; (2) how can agents define and interpret their behaviors based on the formal representation of a negotiation mechanism?; (3) how can we capture and encode agents' knowledge about negotiation mechanisms such that they can adjust dynamically their strategies depending on needs?

In this paper we propose the following initial answers: (1) inspired by [6], we identify a minimal (our prototype protocol allows only six actions, see Sec. 3.3) yet viable generic negotiation protocol (the protocol actions can be used to model various negotiation types), as well as a declarative approach of representing rules and constraints specific to negotiation types such that by combining them we can derive the basis of a generic, reusable negotiation framework. This would allow agents to specify and understand customizable negotiation protocols and strategies; (2) we identify a set of core concepts related to negotiations that should be part of a core negotiation ontology that can be used by agents to reason about negotiations; (3) we propose the Belief-Desire-Intention (BDI) model [7] as the basis for the design of our solution due to its suitable level of abstraction that can appropriately incorporate declarative specification of negotiation agents' behavior, thus offering a richer framework than existing proposals [8].

While we acknowledge that our goal is very ambitious, we think that the research effort is worthwhile, as our endeavor can clearly contribute to the reusability of negotiation protocols and strategies, an issue that in our opinion was insufficiently explored by the research community.

The main results consist of a conceptual framework for development of automated negotiations, as well as an initial prototype developed using Jason agent programming language [9]. While the implementation still lacks some of the features of our proposed conceptual framework, we believe it is a good starting point for the further exploration of the design space of declarative specifications of automated negotiations. Our choice for Jason as implementation language is motivated by: (i) Jason is probably the best known example of the BDI camp in the agent programming community; (ii) Jason supports a declarative programming style, closer to logic programming. We believe that the mix of declarative, goal-oriented, knowledge representation and meta-programming features of Jason make it a good candidate implementation language for our prototype.

The paper is structured as follows. In Sec. 2 we set the stage by looking at some of the related research in this area. In Sec. 3 we outline our system architecture and go into more details regarding our proposed approach and at the same time illustrate our presentation with code samples from a prototype implementation. If Sec. 3 looks at the system as a whole, Sec. 4 is dedicated to exploring the details of single agents in our framework. We conclude in Sec. 5 and we also outline future research directions.

2 Background and Related Work

Albeit the plethora of formal representations originating from researches in Semantic Web and Software Engineering ([10], [11]), the current results do not go beyond simple XML-based representations of specific negotiation mechanisms.

Authors of [6] proposed a generic software framework for automated negotiations. Although very interesting for our research, this proposal only addressed the problem from the perspective of the authority that controls the negotiation, i.e. the *Auction Host*. This approach is limited because, unlike a human negotiator, an artificial negotiator would have to be *a priori* designed to understand certain negotiation mechanisms. This is a considerable limitation of artificial agents as compared to humans, which leads to the impossibility of an artificial agent to act on a market whose negotiation mechanism is not known and understood before the agent design. Therefore, the perspective of the artificial agent acting as a participant in a negotiation should be also taken into account.

Authors of [8] proposed the AB3D software framework for auction development. The AB3D system provides an auction specification framework, as well as a runtime system for the agents that enact the auction specifications. While this approach is interesting as it acknowledges the importance of generic, parameterizable auction specifications, we believe that the AB3D scripting language lacks with respect to flexibility when specifying participant strategies. The question of how a participant agent should dynamically understand the specification of a negotiation protocol (auction in particular) in order to define his strategy is not addressed by that work.

A recent and interesting example of BDI being used for automated negotiations is presented in [12]. The author proposes a multi-strategy automated negotiation framework based on the BDI model. While the presented approach is somewhat similar to our own in that it uses a specialization of the BDI model for the agents in the system, it differs in that the author mostly focuses on individual agent's decision-making mechanism. While we acknowledge that this is an important aspect of an automated negotiation framework, we believe that the holistic approach we propose can lead to a truly generic solution. Nevertheless, the paper remains interesting, because it goes into much more detail w.r.t individual decision-making than the approach we are proposing and it might prove a valuable source of inspiration in our further efforts.

3 Negotiation Model and System Architecture

An automated negotiation can be observed and analyzed from two perspectives: the perspective of the authority that controls the negotiation and the perspective of the negotiation participants, their preferences and their private strategies. Each perspective addresses one of the two components of an automated negotiation: negotiation mechanism and respectively negotiation strategy.

The initial source of inspiration for our negotiation model is [6]. Its authors focused on auctions and performed a deeper analysis of the parametrization of the negotiation protocol by conceptually decomposing it into several rule sets for: admission of participants to negotiation, proposal validity, protocol enforcement, updating negotiation status and informing participants, agreement formation, and controlling the lifecycle of the negotiation process. Although very interesting, their proposal is biased towards the perspective of the auction authority, while the implications of this classification onto the participating agents is omitted from their analysis. In order to understand this point, it is useful to observe that a declarative specification can be utilized in two ways: (i) as an enforcer, to constrain the agents as well as (ii) a generator of agents' permissible actions

depending on the specific negotiation context. For example, let us consider the rules for proposal validity that constrain the submitted proposals to be consistent with a given template. According to [6], the auction authority will perform a consistency check of each proposal submitted by an auction participant. However, a participant agent can use this template to generate, whenever this is allowed by the protocol rules, a new proposal that is compliant with the proposal validity rules.

Additionally, considering the rules for protocol enforcement, they might state that at some point a participant can either bid or leave the auction. The auction authority will use this information to check the messages received from a participant, recognizing and allowing only the actions of bidding or withdrawing (while an action for updating a previously submitted bid would be rejected as not allowed), thus acting as enforcing mechanism for the auction semantics. The participant, on the other hand, will consider its options and, based on its custom strategy, it will pick up one of the two possible actions. From its perspective the negotiation mechanism will act as action generator. Incidentally, the custom strategy will act by additionally filtering the generated available options. We will get back to this point in Sec. 4.

3.1 System architecture

We are going to use the architectural model introduced in [13] as a foundation for our system architecture. The framework proposed by the authors features several types of agents (note that the initial framework targeted auctions, so we have used the term 'auction' interchangeably with 'negotiation'):

Auction Service (AS) Usually it implements a specific type of auction. Its main purpose is to manage auction-related activities (e.g. creation, termination) and to coordinate the auction participants. AS registers with an Auction Service Directory (ASD) that can be queried for a certain AS. An AS contains an Auction Directory (AD), which keeps track of all the ongoing auctions which are represented by Auction Instances (AI);

Auction Host (AH) It is an arbitrator agent responsible with coordinating the agents participating in a single AI. In order to accomplish this task is uses a specific mechanism for each auction type (e.g. English, Dutch).

Auction Participant (AP) This agent participates by bidding in an AI. Out of all the participants one is distinguished as the Auction Initiator Participant (AIP).

We envision that a negotiation agent will use the framework to either initiate a new negotiation or to register for bidding in an existing negotiation. For achieving these functions, the agent can be pre-designed to understand the negotiation mechanism (we already claimed that this is a considerable limitation) or it can download the declarative description of the negotiation mechanism from the *Auction Host* and apply it to define its private strategy. This second option is more interesting for the purpose of this paper.

For a more detailed presentation of this architecture please consult reference [13].

3.2 Conceptual model

We propose to conceptualize the declarative specification of a negotiation as composed of three essential ingredients: *Generic Negotiation Protocol* (GNP), *Declarative Negotiation Mechanism* (DNM) and *Custom Negotiation Strategy* (CNS).

GNP Defines and governs the interaction between the participant agents and the host agent that are part of the system. It is the same for all participants independently of their specific role in the negotiation (i.e. buyer, seller). However, we will have a GNP part for the AH role, as well as a GNP part for the AP role; see Sec. 3.3 for more details);

DNM Is specific to a given negotiation type (e.g. English Auction, Continuous Double Auction, Iterative Bargaining). The DNM is usually defined using a declarative, Prolog-like language. The DNM serves to customize the GNP for representing the conditions and events that enable the permissible actions of negotiation agents (see Sec. 3.4 for more details); and

CNS Is specific to a given AP and must be consistent with the DNM. It is used by the AP to select and configure a specific negotiation action that could be most useful in a given negotiation context (see Sec. 3.5 for more details).

Based on the three features described above we can define the following metaphorical equations that more succinctly describe the agents present in our framework:

$$AH = GNP_{role-host} + DNM_{host}$$
$$AP = GNP_{role-participant} + DNM_{participant} + CNS$$

These ingredients can be consistently bound into descriptions of AH and respectively AP behaviors by referring a common core vocabulary of terms for defining and parameterizing the space of negotiation types. This vocabulary is called *Core Negotiation Ontology* – CNO. The CNO can contain generic parameters, and actions. The former can be used to customize the auction, while the latter can be used by APs to choose corresponding actions with a certain semantics for the auction. Another element of the CNO could be the types of rules used to guide action generation/selection during the negotiation process. The development of a comprehensive CNO is a complex task that is part of our future research and it is outside the scope of this paper.

We are going to use the English Auction as a running example for the following subsections, to give more insight into the three components introduced here. Therefore, for the purpose of this paper we are going to use a sample set of terms, mainly referring to negotiation actions for English Auctions, that will be introduced in Sec. 3.4.

3.3 Generic Negotiation Protocol

The GNP describes the permissible negotiation conversations involving a set of agents comprising the following two generic roles: (i) role of AH; there is a single agent playing this role in a negotiation; and (ii) role of AP; there are one or more agents playing this role in a negotiation. Thus the GNP is agnostic of any details that are specific to a particular negotiation type. Consequently, it must only expose the basic negotiation

```
state(uninitialised).

// When registering an agent
+register[source(A)] : can_register(A)
    // check if the agent can register
    // (e.g. the negotiation hasn't started)
    <- +registered(A);
       // get the current quote
       ?quote(_, Quote);
       // and send it to the freshly added
       // participant
       .send(A, tell, quote(Quote)).

// When receiving a bid check if bids
// are accepted and if the bid is an
// improvement
+bid(Offer)[source(A)] : can_bid(A) &
    check_progress(Offer)
    // Update the current quote
    <- -+quote(A, Offer);
       // and notify all the other participants
       .findall(X, registered(X), Registered);
       .send(Registered, tell, quote(Offer)).

// When the negotiation finishes
+!close
    // and we have a winner
    : check_winner
    <- -+state(closed);
       ?initiator(I);
       ?quote(A, Offer);
       // notify the initiator
       .send(I, tell,
             winner(A, Offer));
       // and the participant
       .send(A, tell, winner).

// Otherwise notify the initiator
// that there is no winner
+!close
    <- -+state(closed);
       ?initiator(I);
       .send(I, tell, no_winner).
```

Listing 1: The *Auction Host* agent which implements the generic negotiation protocol.

actions that are common to as many negotiation types, leaving unspecified the specific details of a certain negotiation type. Those details can be captured and declaratively represented with the help of the DNM.

GNP is inspired by our previous work [13]. As part of the GNP we identified a set of actions as sufficient for introducing the proof-of-concept implementation:

register used by an AP to register itself with a specific auction host
bid used to place a bid
tell/ask depending on the semantics of the push/pull semantics desired the protocol can expose information to the participants via tell or ask actions
fold used by the participant to get out of an auction
close used by the auction host to end the auction
winner used by the auction host to notify the winning participants

The Jason implementation of the AH role of the GNP is presented in listing 1. We refer the reader to [9] for details on the syntax and semantics of the language.

The code in listing 1 closely follows the definition of the GNP outlined above. Very briefly, the plan for `+register` registers a participant and sends it the current *quote* (i.e. the value of the outstanding bid) – this presumes push semantics; the `+bid` plan updates the quote and pushes messages with the new value to *all* the participants; the `+close` plan ends the auction and notifies any winner if there is one. All the plans keep track of the current state of the agent.

Note that the plans make use of rules that are included in the definition of the DNM (this will be covered in the next subsection). While an AS conceptually implements a type of auction, an AH is spawned for each new auction started by the AS. During its initialization the AH loads the DNM specific to the negotiation type supported by the AS and initialized with the parameters received from the AIP.

```
// A negotiation can be initialized
// only if it hasn't been previously
// initialized
can_init[state(uninitialised)] :-
    state(uninitialised).

// A participant can register only
// after the negotiation was initialized
// and if it hasn't already registered
can_register(A)[state(bidding)] :-
    state(bidding) &
    not(registered(A)).

// Only registered agents can bid
can_bid(A)[state(bidding)] :-
    state(bidding) &
    registered(A).

// An offer needs to be an inprovement of
// the current quote
check_progress(Offer)[state(bidding)] :-
    increment(Increment) &
    quote(_, Quote) &
    Offer >= Quote + Increment.

// The winner needs to offer more
// than the reserved price
check_winner[state(bidding)] :-
    quote(A, Offer) & A \== "" &
    min_price(MinPrice) &
    Offer >= MinPrice.
```

Listing 2: *Declarative Negotiation Mechanism* for English Auction.

3.4 Declarative Negotiation Mechanism

We chose a rule-based representation for our DNM, which builds on our previous approach presented in [14]. Moreover, this representation was mapped naturally to Jason.

Listing 2 shows the rules that we used to customize the *Auction Host* for our running example. Note that these rules illustrate the DNM from the AH perspective.

Besides their formal parameters, the rules are also implicitly parameterized by the current state of the negotiation process – here we are using the annotations mechanism from Jason to achieve this, e.g. the `[state(bidding)]` annotation on the `can_bid` rule makes it applicable only when the agent is in the `bidding` state.

One important aspect, that we haven't detailed yet, is the sample CNO for this particular example. This is implicitly represented both in the rules presented in listing 2 and in the *Auction Host* presented in listing 1. `min_price`, `increment`, `quote` and `state` are part of the CNO. They are either parameters that must be filled in by the AIP – the first two – or state variables that are maintained by the *Auction Host* – the former two. In our prototype implementation the CNO is expressed as s set of beliefs. Note, though that alternative approaches are possible, like explicitly specifying the ontology with the aid of a specialized language like OWL. With such an approach the OWL specification of the CNO must be compiled into agent beliefs before the start of a new negotiation.

3.5 Custom Negotiation Strategy

The CNS must be implemented by each individual AP agent. In particular, this component allows each AP to adopt different risk attitudes toward the auction participation (e.g. eager, aggressive, neutral). As mentioned before, this is a key aspect of our approach, since it allows to model much richer negotiation scenarios. So, in principle, CNS should be not only understandable and adjustable by the AP itself, but also externally configurable by a human user which can be present behind a certain AP.

Conceptually, the CNS is built on top of the GNP, so the strategy designer is not exposed to the lower level encoding of the negotiation process. Instead he or she can

```
product(macbook).
amount(1200).
current_offer(0).

!start.

+!start : product(Product)
    <- .wait(100);
       // Look for an auction featuring
       // the product
       .send(auction_service, tell,
             find_auction(Product)).


+quote(Quote) : current_offer(Offer)
    <- !bid(Quote).

+winner <- .print("I won!").
```

```
+!bid(Quote) : amount(Amount) & Amount > 100
    <- ?auction_host(AuctionHost);
       // Eagerly bid as long as the
       // agents has money left
       -+current_offer(Quote + 100);
       -+amount(Amount - 100);
       .send(AuctionHost, tell,
             bid(Quote + 100)).

+auctions_for(Product, [auction(_, AH)|_])
    <- +auction_host(AH);
       .send(AH, tell, register).
```

Listing 3: *Auction Participant* that acts as a buyer.

focus on the more relevant aspects of the definition of an appropriate strategy – like improving his or her outcome from the negotiation participation.

Listing 3 shows an eager buyer for our running example of an English auction. Conceptually, the plans on the left side of the listing would be the only ones that the auction designer has to write. The plans on the right would be part of the framework, and as such, it would not be exposed to the implementer. For our prototype, this could be achieved by customizing the Jason agent to internally handle the GNP related events (e.g. `+quote` messages) and triggering the relevant GNS plans (in our case `+!bid`) which would act like "hooks" into the framework for the auction designer.

4 Agent Architecture

The GNP, DNM and CNS fit naturally to the generic BDI model, and in particular to the Jason programming model. We chose BDI as our underlying architecture because it offers certain advantages:

- The GNP, DNM and CNS map naturally to components of the BDI agents' architecture: beliefs, goals, events, and plans, that are also supported by Jason.
- The BDI model is an established paradigm for the development of autonomous rational agents. So we expect that it can be useful for more complicated negotiation scenarios, for example involving agents that simultaneously participate in multiple auctions possibly sharing partly or entirely the negotiation subject.

Other recent approaches of developing automated negotiations using BDI agents were reported by [15, 12].

The reasoning cycle of a BDI agent [7, p. 7] can be utilized to materialize the *enactment* and *enforcement* of the GNP for both AH and AP agents. The option generation and selection phase corresponds to finding all possible actions at each step of the negotiation. Selecting an intention to be executed based on the available options corresponds to executing a step of the negotiation. Both steps use the DNM to filter out actions that

are not achievable. Additionally, APs use the CNS to further constrain the actions that are going to be performed during each step of the negotiation process. Finally, getting the new external events maps to receiving messages that form the GNP.

Note that although our conceptual model is mature enough, the proposed implementation using Jason is just an initial prototype. Its further extension and evaluation are required as part of our future work. Nevertheless, the decision to use Jason as an implementation language has proven a good choice for the following reasons.

First, Jason's Prolog-like rule language fit well with the DNM specification. It can be used to write declarative rules that serve to enforce negotiation protocol semantics on top of the GNP. This aspect of Jason's rule language corresponds to checking generic parameters defined in the CNO. By using Jason's built-in inference engine, the rule language could be used to guide the generation/selection of actions defined in the CNO. This approach is appealing because it would allow a complete (it supports both generic parameters and negotiation actions) and uniform (the same language is used for both generic parameters and actions) representation of the DNM. Due to these reasons this is one of the next steps we are going to take in further developing our prototype.

Second, we can leverage Jason's features (e.g. first class beliefs and goals, BDI architecture) to build our framework. Since our abstract framework could be seen as a specialization of BDI, this is a natural approach. Nevertheless, more investigation is required to check the appropriateness of the matching between Jason's internal architecture and our proposed conceptual framework.

Third, Jason was engineered as an extensible system, offering many customization points. Together with its metaprogramming capabilities this reduced the development time of our prototype considerably. We will further investigate using these customization mechanisms to address the problem mentioned earlier.

Overall, the development of the prototype on top of the BDI model has proven the viability of our proposal. It remains to be seen if the final representation language will be Jason or another – possibly custom – language. Ideally, programs should read like an executable specification of the implemented negotiation.

5 Conclusions and Future Work

In this paper we have presented our initial ideas of an original approach for the development of a generic, declarative negotiation framework. The first important aspect is the proposal of a novel dichotomy between the auction host and the auction participant when considering the declarative negotiation mechanism. The second innovative aspect is the description of the mapping of our conceptual framework to the BDI agent architecture. Our proposal is supported by a sample proof-of-concept implementation using Jason agent programming language. We claim that our initial results make the development of a generic framework for automated negotiations a feasible endeavor.

Our future directions are related to expanding the negotiation framework in order to allow it to express a richer set of negotiation types. Important aspects are related to: (i) sharing specifications of negotiation mechanisms in open environments, like the Internet and the Web; (ii) developing a proper *Core Negotiation Ontology*; and (iii)

evaluating the framework by developing specifications of different negotiation types and negotiation agents.

References

1. Kraus, S.: Automated negotiation and decision making in multiagent environments. In Luck, M., Mark, V., Åtepnkov, O., Trappl, R., eds.: Multi-Agent Systems and Applications. Volume 2086 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2006) 150–172
2. Jennings, N., Faratin, P., Lomuscio, A., Parsons, S., Wooldridge, M., Sierra, C.: Automated negotiation: Prospects, methods and challenges. Group Decision and Negotiation **10** (2001) 199–215
3. Buttner, R.: A classification structure for automated negotiations. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology – Workshops, Los Alamitos, CA, USA, IEEE Computer Society (2006) 523–530
4. Wurman, P.R., Wellman, M.P., Walsh, W.E.: A parametrization of the auction design space. Games and Economic Behavior **35**(12) (2001) 304 – 338
5. Scafeş, M., Bădică, C.: Computing equilibria for constraint-based negotiation games with interdependent issues. In: Proceedings of Federated Conference on Computer Science and Information Systems - FedCSIS 2011. (2011) 597–603
6. Bartolini, C., Preist, C., Jennings, N.R.: A software framework for automated negotiation. In Choren, R., Garcia, A.F., de Lucena, C.J.P., Romanovsky, A.B., eds.: SELMAS. Volume 3390 of Lecture Notes in Computer Science., Springer (2004) 213–235
7. Rao, A.S., Georgeff, M.P.: Bdi agents: From theory to practice. In Lesser, V.R., Gasser, L., eds.: Proceedings of the First International Conference on Multiagent Systems, ICMAS'95, The MIT Press (1995) 312–319
8. Lochner, K.M., Wellman, M.P.: Rule-based specification of auction mechanisms. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2. AAMAS '04, Washington, DC, USA, IEEE Computer Society (2004) 818–825
9. Bordini, R.H., Hbner, J.F., Vieira, R.: Jason and the golden fleece of agent-oriented programming. In Bordini, R.H., Dastani, M., Dix, J., Fallah-Seghrouchni, A.E., eds.: Multi-Agent Programming. Volume 15 of Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer (2005) 3–37
10. Tamma, V., Phelps, S., Dickinson, I., Wooldridge, M.: Ontologies for supporting negotiation in e-commerce. Engineering Applications of Artificial Intelligence **18**(2) (2005) 223 – 236
11. Dong, H., Hussain, F., Chang, E.: State of the art in negotiation ontologies for enhancing business intelligence. In: Next Generation Web Services Practices, 2008. NWESP '08. 4th International Conference on. (oct. 2008) 107 –112
12. Cao, M.: Multi-strategy selection supported automated negotiation system based on bdi agent. In: Proc. 45th Hawaii International International Conference on Systems Science (HICSS-45 2012), Los Alamitos, CA, USA, IEEE Computer Society (2012) 638–647
13. Dobriceanu, A., Biscu, L., Bădică, A., Bădică, C.: The design and implementation of an agent-based auction service. IJAOSE **3**(2/3) (2009) 116–134
14. Bădică, C., Giurca, A., Wagner, G.: Using rules and r2ml for modeling negotiation mechanisms in e-commerce agent systems. In: Trends in Enterprise Application Architecture, 2nd International Conference, TEAA 2006. Volume 4473 of Lecture Notes in Computer Science., Berlin, Heidelberg, Springer (2007) 84–99
15. Le Dinh, B.C., Seow, K.T.: Unifying distributed constraint algorithms in a bdi negotiation framework. In: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, AAMAS '07, New York, NY, USA, ACM (2007) 117:1–117:8