Using Jason to Develop Declarative Prototypes of Automated Negotiations

Alex Muscar
University of Craiova
Software Engineering Department
Bvd. Decebal 107, Craiova, 200440, Romania
silie@software.ucv.ro

Laura Surcel
University of Craiova
Software Engineering Department
Bvd. Decebal 107, Craiova, 200440, Romania
laura_surcel@yahoo.com

Costin Bădică
University of Craiova
Software Engineering Department
Bvd. Decebal 107, Craiova, 200440, Romania
cbadica@software.ucv.ro

ABSTRACT

We propose a declarative approach for prototyping automated negotiations in multi-agent systems. Our approach is demonstrated by using Jason agent programming language to describe English and Dutch auctions.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

Keywords

Artificial intelligence, computer programming

1. INTRODUCTION

Automated negotiations are common in multi-agent systems (MAS) – e.g. bargaining, auctions [8], multi-criteria negotiation [6], multi-commodity negotiations. Some of the practical applications of automated negotiations are: price negotiation for e-commerce ([2]), cooperative design, and task allocation.

Although there are many methods of automated negotiations, they generally fail to properly address the reusability and the extensibility of the proposed approaches. In most applications the negotiation mechanism is very specific and hard coded, making the reusability of research results difficult.

In order to treat these issues we have set up a research project aiming to develop a generic, reusable negotiation framework. One of the key insights of our approach is that an (automated) negotiation has two main components: the protocol which governs agents' interaction during negotiation and the strategy of participating agents. While the analysis and design of specific negotiations mechanisms and the development of appropriate strategies with desired properties (e.g. stability, equilibrium, efficiency, a.o) has received a lot of attention in the context of particular situations, they have not been properly addressed with regard to reusability and extensibility.

This paper is a follow-up of our initial proposal [5]. Our results consisted in the identification of a minimal yet viable *Generic Negotiation Protocol* (GNP) as well as of a declarative approach of representing rules and constraints specific to negotiation types. Our aim was to combine them for deriving the basis of a generic, reusable negotiation framework that would allow agents to specify and understand customizable negotiation protocols and strategies. In [5] we introduced a prototype MAS for English auctions based on Jason agent programming language [1]. Here we take a closer look at the implementation of this prototype and extend it to support Dutch auctions.

BCI'12, September 16–20, 2012, Novi Sad, Serbia.

Local Proceedings also appeared in ISBN 978-86-7031-200-5, Faculty of Sciences, University of Novi Sad.

2. BACKGROUND

The system architecture is based on our work [4], which distinguished between several types of agents:[1] (i) *Auction Service* (AS) manages auction-related activities (e.g. creation, termination), coordinates the auction participants, and acts as a yellow page service for ongoing auctions; (ii) *Auction Host* (AH) is an arbitrator agent responsible with coordinating the agents participating in a single auction instance (AI); and (iii) *Auction Participant* (AP) which participates by bidding in an AI – out of all the participants one is distinguished as the *Auction Initiator Participant* (AIP).

Conceptually, the declarative specification of a negotiation is composed of three essential ingredients: (i) *Generic Negotiation Protocol* (GNP) which defines and governs the interaction between the agents that are part of the system and it is the same for all of them, independently of their role in the negotiation (i.e. buyer, seller); (ii) *Declarative Negotiation Mechanism* (DNM) is specific to a given negotiation type and it serves to customize the GNP for representing the specific conditions and events that enable the permissible actions of specific AP agents; and (iii) *Custom Negotiation Strategy* (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.

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

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

These ingredients can be consistently bound into descriptions of AH and respectively AP behaviors by making them to refer a common core vocabulary of terms that can be used for defining and parameterizing the space of negotiation types [7]. This vocabulary is called *Core Negotiation Ontology* (CNO). The development of a more complete CNO is is part of our future research.

[1]Note that the initial framework targeted auctions, so we have used the term 'auction' interchangeably with 'negotiation'.

```
+register[source(A)]
    : can_register(A)
    <- +registered(A);
       ?buy_it_out(Sum);
       ?increment(Increment);
       ?items(Left_items);
       ?last_offer(Offer);
       ?state(State);
       .send(A, tell,
             registered(info(Sum,Increment),
       status(Offer, Left_items, State))).

+fold[source(A)]
    : registered(A)
    <-  -registered(A);
        .send(A, tell, not_registered).
```

```
+bid(Offer,Items)[source(A)]
    : check_protocol(A)
    & check_proposal(Offer,Items)
    & not(terminate(Offer,Items))
    <- -+state(processing);
       !update_status(A,Offer,Items);
       !inform_participants(A,Offer);
       -+state(bidding).

+!close
   : check_winner
   <- ?initiator(I);
      -+state(closed);
      ?bidders(A);
      .send(I, tell, winner(A));
      .send(A, tell, winner).
```

Figure 1: The AH agent which implements the GNP.

3. THE GENERIC NEGOTIATION PROTOCOL

The GNP describes the permissible conversations involving a set of negotiation agents comprising the following 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 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.

We have identified the following set of actions as sufficient for the purpose of introducing our proof-of-concept implementation: (1) **register** used by an AP to register itself with a specific AH; (2) **bid** used to place a bid; (3) **tell/ask** depending on the push/pull semantics, the protocol can expose information to the participants either automatically or on request; (4) **fold** used by an AP to get out of an auction; (5) **close** used by the AH to notify the APs about the auction termination; and (6) **winner** used by the AH to notify the winner participants.

4. IMPLEMENTATION DETAILS

Jason is probably the best known example of agent programming language that follows the Belief-Desire-Intention (BDI) model. It supports a declarative programming style, closer to logic programming. We chose Jason as we believe that the mix of declarative, goal-oriented and knowledge representation features of Jason make it a good candidate implementation language for our prototype. Furthermore its meta-programming features (e.g. representing programs as code) has helped with our implementation.

Our implementation provides an *internal action* that loads the rules for a specific negotiation type. This has to be called by agents before they start the negotiation in order to customize their GNP. Once loaded, the rules can be used like any other rule defined in the Jason agent source file.

The Jason implementation of the AH role of the GNP, presented 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 the agent is in.

Note that the plans make use of rules that are defined by the DNM, which we will discuss next. While an AS conceptually implements a type of auction, an AH is spawned for each new auction started by the AS. When initially started the AH loads the DNM specific to the auction implemented by the AS and a further message from the AIP is responsible for initializing the parameters of the DNM with appropriate values.

We chose a rule-based representation for our DNM, which builds on our previous approach presented in [3]. Moreover, this representation was mapped naturally to Jason.

Listing 2 shows the rules used to customize the AH for our running example, an English auction. 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 `check_proposal` rule makes it applicable only when the agent is in the `bidding` state.

5. CONCLUSION AND FUTURE WORK

We have presented our approach on developing a generic negotiation mechanism (as part of a generic negotiation framework) using the Jason programming language. We believe the solution is elegant and promising, due to its declarative nature and modular nature. Basically the GNP is independent of the negotiation type, negotiation rules are used to customize it.

One of the necessary next steps is to find a suitable serialization format for the rules used to customize the GNP, such as RuleML.[2] Another direction, as mentioned earlier, is to define a suitable CNO for the negotiation framework. Yet a third research direction is to factor out the GNP so that it can be used in multiple platforms–e.g. Jason, Java+JADE.

6. ACKNOWLEDGMENTS

The work of Alex Muscar 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. The work of Costin Bădică was supported by the multilateral agreement on academic cooperation in 2012 between Serbia (Novi Sad), Romania (Craiova), and Poland (Warsaw and Gdansk) on "Agent Technologies, Tools, Environments, Applications".

[2]RuleML – `http://ruleml.org/`

```
// Dutch auction
check_proposal(Offer,Items)
    [state(bidding)] :-
    asked_price(AskedPrice) &
    items(Left_items) &
    Left_items>=Items &
    Offer <= AskedPrice.

+!update_status(A,Offer,Items)
    <- .time(H,M,Sec2);
       ?items(I);
       ?increment(Increment);
       ?asked_price(AskedPrice);
       ?bidders(B);
        -+last_offer(Offer);
        -+items(I-Items);
        -+bidders([A|B]);
        -+asked_price(AskedPrice+Increment);
        -+last_update(Sec2).
```

```
// English auction
check_proposal(Offer,Items)
[state(bidding)] :-
    increment(Increment) &
    last_offer(Quote) &
    items(Left_items) &
    Left_items>=Items &
    Offer >= Quote + Increment.

+!update_status(A,Offer,Items)
    <- -+last_offer(Offer);
       -+bidders([A]).
```

Figure 2: DNM for English and Dutch auctions.

7. REFERENCES

[1] R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. Wiley, 2007.

[2] C. Bădică, M. Ganzha, and M. Paprzycki. Implementing rule-based automated price negotiation in an agent system. *Journal of Universal Computer Science*, 13(2):244–266, feb 2007.

[3] C. Bădică, A. Giurca, and G. Wagner. 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*, pages 84–99, Berlin, Heidelberg, 2007. Springer.

[4] A. Dobriceanu, L. Biscu, A. Bădică, and C. Bădică. The design and implementation of an agent-based auction service. *IJAOSE*, 3(2/3):116–134, 2009.

[5] A. Muscar and C. Bădică. Exploring the design space of a declarative framework for automated negotiation: initial considerations (**in press**). In *Proc.Artificial Intelligence Applications and Innovations 2012 – AIAI 2012*. Springer, 2012.

[6] M. Scafeş and C. Bădică. Computing equilibria for constraint-based negotiation games with interdependent issues. In *Proceedings of Federated Conference on Computer Science and Information Systems – FedCSIS 2011*, pages 597–603, 2011.

[7] V. Tamma, S. Phelps, I. Dickinson, and M. Wooldridge. Ontologies for supporting negotiation in e-commerce. *Engineering Applications of Artificial Intelligence*, 18(2):223–236, 2005.

[8] P. R. Wurman, M. P. Wellman, and W. E. Walsh. A parametrization of the auction design space. *Games and Economic Behavior*, 35(2):304–338, 2001.