Using Rules and R2ML for Modeling Negotiation Mechanisms in E-Commerce Agent Systems*

Costin Bădică[1], Adrian Giurca[2], and Gerd Wagner[2]

[1] Software Engineering Department, University of Craiova,
Bvd.Decebal 107, Craiova, 200440, Romania
badica_costin@software.ucv.ro
[2] Internet-Technology Department,
Brandenburg University of Technology at Cottbus,
Walther Pauer Str. 2, 03046 Cottbus, Germany
{Giurca, G.Wagner}@tu-cottbus.de

Abstract. With the spread of e-commerce on a global scale, the development of truly open semantic descriptions of negotiation mechanisms for agent systems generated a lot of interest in the research community. This paper proposes the use of the REWERSE rule-markup language R2ML for semantic modeling of negotiation mechanisms to enable agents to engage in more flexible and open negotiations. Rules are developed on top of an ontology of negotiation concepts and define a *lingua franca* for all software agents participating in negotiation.

1 Introduction

Global information networks are described as open collaborative environments hosting intelligent and autonomous services that are able to dynamically discover each other and engage in business transactions, possibly involving automated negotiations. E-commerce is seen as a key service of modern information society and therefore, the ability of software agents to discover remote markets and engage in commercial transactions governed by market mechanisms unknown in advance, is of primary importance.

We understand automated negotiations as a process by which a group of software agents communicate with each other to reach a mutually acceptable agreement on some matter [11]. In this paper we focus our attention on *auctions* – a particular form of negotiation that spread during the last years with the advent of the Internet and the Web. Auctions are negotiations where resource allocations and prices are determined by bids exchanged between participants according to a given set of rules [15].

In automated negotiations (including auctions) it is important to distinguish between *negotiation protocols* (or *mechanisms*) and *negotiation strategies*. The protocol comprises public "rules of encounter" between negotiation participants by specifying the requirements that enable them to interact and negotiate. The strategy defines the private behavior of participants aiming at achieving their desired outcome. This behavior must be consistent with the protocol and is chosen to optimize participant welfare ([26]).

* Work of A. Giurca and G. Wagner was partially funded by European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme projects REWERSE (IST-2004-506779) cf. http://www.rewerse.net.

A key aspect that generated a lot of interest in the research community is the development of a truly open semantic description of negotiation mechanisms [2, 1, 17, 16, 19, 18]. As our literature overview indicates, we are still quite far from that vision of software agents needing only little compiled knowledge to enable ”sensing” the negotiation mechanism and ”tuning” the negotiation strategy accordingly. As an attempt to narrow this gap, in this paper we propose the use of R2ML rule-markup language for semantic modeling of negotiation mechanisms in agent systems. Our proposal builds over existing works [27, 2, 1] on rule modeling of agent-based auctions and therefore it is expected to cover at least the types of auctions discussed there.

Before proceeding let as note that the use of semantic markup languages for modeling negotiation mechanisms is not entirely new; several approaches have already been proposed in the literature ([17, 16, 19, 18]).

The proposal for formalizing negotiations introduced in [18] goes beyond the generic software framework of [2] and implemented in [1]. Its authors suggest the use of an ontology for representing negotiation protocols. Whenever an agent is admitted to negotiation it also obtains a specification of the negotiation mechanism in terms of the shared ontology. The ontology approach introduced in [18] is taken further in [19] by investigating how the ontology can be used to tune the negotiation strategy of participant agents. Note that authors of [19] point out that the ontology approach is still far from the vision where agents need only little hard-coded knowledge about the negotiation mechanism and this is due to the limitations of ontology languages to capturing explicitly the semantics of the rules that govern the negotiation. In this paper we address this issue thus making our work different from existing related works [19, 18].

The open environment for automated negotiations specifically targeted to auctions ([16, 17]) comprises: i) the *auction reference model* – ARM and ii) the *declarative auction specification language* – DAL. Note that, while not explicitly using rules, a DAL specification models in fact the auction flow using a rule-based approach. DAL uses the following constructs: views, validations, transitions and agreement generators ([16]). Views are analogous to visibility rules, validations are analogous to bidding rules, transitions are analogous to update rules and agreement generators are analogous to clearing rules. Finally, DAL provides also an explicit, implementation-level separation of the specification of auction flow from the auction data. For this purpose, a DAL specification comprises a set of SQL queries that provide access to the market data. While SQL has a declarative semantics and it is useful for the implementation side of DAL, we believe that this feature is less significant as concerning the portability of the language, as compared with the rule-based representation using R2ML.

2 Negotiation Model and Vocabulary

The starting point of our work is the rule-based framework for enforcing specific negotiation mechanisms proposed by [2]. Note that details of its implementation using JADE [5] and JESS [8] including initial experimental results for English auctions were reported in [1]. So, our work can be also seen as an attempt to provide a portable R2ML representation of the auction mechanism that could be reused by that implementation.

Authors of [2] sketched a software framework for implementing agent negotiations that comprises: (1) negotiation infrastructure, (2) generic negotiation protocol and (3) taxonomy of declarative rules. The *negotiation infrastructure* defines roles of negotiation participants (eg.*buyer* or *seller* in an auction) and of a negotiation host. Participants exchange proposals within a negotiation locale managed by the host.

According to the *generic negotiation protocol* ([2]), negotiation is seen as the process of exchanging proposals (or bids) via a common space or blackboard (also known as market [16, 17]) that is governed by an authoritative entity – the negotiation host (or market maker). Status information describing negotiation state and intermediary information is automatically forwarded by the negotiation host to all entitled participants according to the information revealing policy of that particular negotiation ([2, 1]).

Negotiation rules are used for enforcing the negotiation mechanism. Rules are organized into a taxonomy: rules for participants admission to negotiations, rules for checking the validity of proposals, rules for protocol enforcement, rules for updating the negotiation status and informing participants, rules for agreement formation and rules for controlling the negotiation termination.

We model the basic negotiation vocabulary with the class diagram from Figure 1.

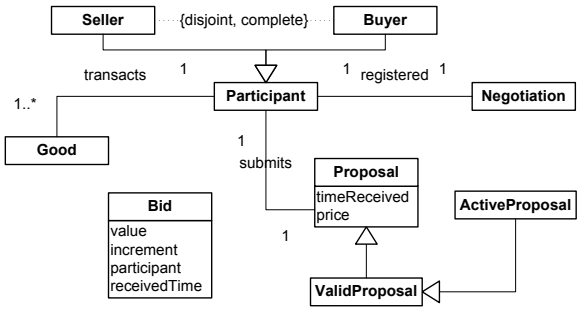

Fig. 1. An excerpt of the negotiation vocabulary

This vocabulary corresponds to an OWL [22] ontology that is used by agents involved in negotiations. *Participant*, *Seller*, *Buyer*, *Negotiation*, *Good*, *Proposal*, *ValidProposal*, *ActiveProposal*, and *Bid* are OWL classes. *transacts* is an OWL object property corresponding to the many-to-many association between classes *Participant* and *Good*, and *registered* is an OWL object property corresponding to the inverse functional association between classes *Participant* and *Negotiation*. Note that this vocabulary addresses the Platform-Independent Model[1] of a business system and therefore it is independent of the specific technological platform used to implement it.

[1] The term platform-independent model (PIM) is most frequently used in the context of the Model Driven Architecture (MDA) approach which corresponds the Object Management Group (OMG) vision of Model Driven Engineering (MDE). The main idea is that it should be possible to use a model transformation language (MTL) to transform a Platform-Independent Model (PIM) into a Platform-Specific Model (PSM).

3 Rules in Agent Negotiation

The aim of this section is to discuss the main types of rules needed to parameterize a negotiation mechanism with a focus on auctions. Our approach is exemplified with a sample set of rules that we have devised for describing single-item English auctions. We have chosen English auctions because they are a non-trivial and easy to understand auction mechanism that became popular because of the establishment of many online auction houses like `eBay`.

In order to make this presentation independent of a particular rule representation formalism, we have chosen to express our rules in an informal pseudo-code notation. The description is supplemented with a discussion of the intended semantics of the rules that govern a typical single-item English auction.

Technically, English auctions are single-item, first-price, open-cry, ascending auctions ([10],[26]). In an English auction there is a single item sold by a single seller and many buyers bidding against one another for buying the item until the auction terminates. Usually, there is a time limit for ending the auction, a seller reservation price that must be met by the winning bid for the item to be sold and a minimum value of the bid increment. A new bid must be higher than the currently highest bid plus the bid increment in order to be accepted. All the bids are visible to all the auction participants, while seller reservation price is private to the auction.

3.1 Categories of Negotiation Rules

Based on analysis performed in [2, 28, 27] and our own experience [1] we have concluded that the following categories of rules are necessary for configuring a negotiation mechanism (auction in particular): *bidding rules*, *information rules* and *clearing rules* (terminology is borrowed from [28, 27]). Rules are activated when certain events occur during the negotiation (eg.when a participant proposal is received by the host or when a given time period without any bidding activity is observed).

Bidding rules. These rules are responsible for handling proposals submitted by negotiation participants to determine if these proposals are correct according to the syntactical and semantical requirements of the negotiation mechanism.

This is a two-step process. Firstly, it involves checking if a proposal is valid – i.e. if the proposal is syntactically correct (for example if it specifies an amount to be paid and a transacted product). This check is performed by *rules for proposal validity*.

Secondly, the process involves checking if the bid is in accordance with the semantical requirements of the negotiation mechanism. This check is performed by *rules for protocol enforcement*. For example: i) *posting rules* check the conditions when a participant is allowed to submit a bid; ii) *improvement rules* check if a participant's proposal is an improvement over its own previous proposal or over the proposal that is currently revealed by the negotiation; iii) *withdrawal rules* check if and when a proposal can be withdrawn (for example a proposal can be active only a fixed amount of time or until it is explicitly withdrawn by a participant).

Information Rules. The negotiation host is essentially a data processor. It is responsible with processing proposals submitted by participants, with updating the state of the negotiation process and with informing participants according to the information

revealing policy of the negotiation. *Information rules* govern the policies for generating all this intermediate information that is necessary for running the negotiation. Typically, this information includes negotiation state information (eg.negotiation stage or round, currently highest price, etc.) and information revealed to participants.

For example: i) *update rules* specify how negotiation data (including negotiation parameters or negotiation stage) is updated in case certain events occurred; ii) *visibility rules* specify what negotiation information is visible to which participants; iii) *display rules* specify if and how a specific information about the negotiation should be notified to (some of) the participants.

Clearing Rules. The negotiation goal is to produce one or more deals between the negotiation participants. Clearing rules are responsible with detecting and computing negotiation deals and controlling negotiation life-cycle.

For example: i) *agreement formation rules* determine when an agreement can be reached and what is the corresponding set of deals made; ii) *termination rules* specify when the negotiation terminates.

Rule Activation. Rule activation is triggered by the occurrence of certain events during the negotiation. Usually, the activation of bidding rules is triggered when the negotiation host receives a new proposal. However, information and clearing rules can be triggered by other events, as well, including: lack of bidding activity for a given time, timer events, admission of a new proposal, certain updates of the negotiation state (like changing the round), etc. Note that by combining negotiation activities using associated triggering events and conditions may result in a great variety of negotiations.

3.2 Intended Semantics of Negotiation Rules for English Auctions

In this section we describe a sample set of negotiation rules for single-item English auctions. Rules are written using an intuitive pseudo-code notation, while their intended interpretation is described in natural language.

Bidding Rules for English Auctions handle proposals submitted by negotiation participants and check their correctness according to the English auction mechanism.

VALIDITY rule checks if a proposal is well formed, i.e. if it specifies transacted good and amount to be paid and if it comes from a registered participant (*seller* or *buyer*). In case of success the proposal is recorded as valid together with the time it was received by the negotiation host – *submission time*.

VALIDITY
IF
S is a participant registered with negotiation **AND**
S transacts good A **AND**
A new proposal Pr was submitted by S **AND**
S has role $R \in \{buyer, seller\}$ **AND**
Proposal Pr contains amount to be paid P
THEN
Proposal Pr is valid **AND**
Submission time T is recorded with proposal Pr

Posting rules check if a valid proposal can be posted depending on the type of proposals that were previously posted by the other participants. POSTING-BUYER

rule specifies that a *buyer* participant can post a proposal whenever there is a matching offer already posted by a *seller* participant. POSTING-SELLER rule specifies that the *seller* must be the first participant that posts a proposal. Therefore the *seller* is called *market initiator*. Every negotiation mechanism usually specifies a market initiator that is responsible with the initiation of a negotiation process. Posting rules collectively specify that in an English auction the participant with role *seller* must be the first to submit a proposal (with the intention to sell) and only then participants with role *buyer* will submit their proposals (usually called bids, with the intention to buy).

POSTING-BUYER

IF

There is a valid proposal Pr of a participant with role *buyer* on good A **AND**
There is an active proposal of a participant with role *seller* on good A

THEN

Proposal Pr is posted

POSTING-SELLER

IF

There is a valid proposal Pr of the participant with role *seller* on good A **AND**
There are no active proposals on good A

THEN

Proposal Pr is posted

Improvement rules check if a valid proposal can be posted depending on the content of proposals that were previously posted. IMPROVEMENT-BUYER enforces a new valid proposal to specify a price higher than the currently highest bid plus a give increment. ACTIVATE-SELLER just activates a valid bid posted by the *seller* (note that this rule was added to preserve the symmetry of treating *buyer* and *seller* proposals).

IMPROVEMENT-BUYER

IF

Negotiation is on good A **AND**
Bid increment is Inc **AND**
Currently highest bid is B **AND**
Proposal Pr on good A with amount to be paid P was posted by this *buyer* **AND**
$P \geq B + Inc$

THEN

Proposal Pr is active

ACTIVATE-SELLER

IF

Proposal Pr was posted by this *seller*

THEN

Proposal Pr is active

Note that posting and improvement rules actually check dynamic constraints of the negotiation mechanism, i.e. what sequences of proposals are allowed. Also note that a proposal that passed the validity tests is called *valid*, a proposal that passed the posting tests is called *posted* and a proposal that passed the improvement tests is called *active*.

Information Rules for English Auctions specify the processing applied to an active proposal. This usually results in updates of the negotiation state and notifications sent by negotiation host to negotiation participants.

Update rules specify the necessary updates of the negotiation state when a new active proposal is posted. UPDATE-BUYER rule performs the update of the currently highest bid after a new active proposal was posted by a *buyer* participant (note that rule IMPROVEMENT-BUYER only checks the *buyer* proposal, but does not update the negotiation state). UPDATE-SELLER rule initializes the negotiation state when an active proposal with an offer was posted by a *seller* participant.

UPDATE-BUYER

IF

There is an active proposal Pr posted by participant S with role *buyer* **AND**
Proposal Pr has price P and was received at time T **AND**
Currently highest bid is B

THEN

Currently highest bid becomes P and was submitted by S at time T

UPDATE-SELLER

IF

There is an active proposal Pr posted by participant with role *seller* **AND**
Proposal Pr has price P and refers to good A **AND**

THEN

Negotiated good are set to A **AND**
Seller reservation price is initialized to P **AND**
Currently highest bid is initialized to a default value (0) **AND**
Termination time window is initialized to a default value

INFORM rule specifies that whenever the currently highest bid is updated, all the negotiation participants must be notified accordingly. Usually this notification contains the value of the highest bid, the identity of the submitter and the time when it was submitted (actually received by the negotiation host).

INFORM

IF

Currently highest bid has been updated

THEN

Notify accordingly all the negotiation participants

Visibility rules specify what negotiation information is disclosed to which participants, and what negotiation information is private to the negotiation.

VISIBILITY-SELLER-PROPOSAL rule specifies that good, submission time and participant name of an active proposal submitted by a *seller* are public to all *buyer* participants, while the price is private to the unique *seller* participant.

VISIBILITY-SELLER-PROPOSAL

IF

There is an active proposal submitted by participant S with role *seller* **AND**
This proposal is on good A and was recorded at time T

THEN

S, A and T are visible to all participants

VISIBILITY-BUYER-PROPOSAL rule specifies that all the parameters of an active proposal submitted by a *buyer* (i.e. participant name, price, good and submission time) are public to al negotiation participants.

VISIBILITY-BUYER-PROPOSAL
IF
The currently highest bid is B and is on good A **AND**
The currently highest bid was submitted by a participant S at time T
THEN
S, A, T and B are visible to all participants

Clearing Rules for English Auctions determine negotiation outputs and control negotiation termination.

AGREEMENT-FORMATION rule specifies that whenever agreement formation is triggered, if the currently highest bid is greater than the *seller* reservation price, an agreement is formed between the submitter of the highest bid and the *seller*.

AGREEMENT-FORMATION
IF
The currently highest bid is B and was submitted by *buyer* $S1$ **AND**
There is an active proposal of *seller* $S2$ with price P **AND**
Negotiation is on good A **AND**
$B \geq P$
THEN
An agreement of $S1$ with $S2$ to transact good A at price $P1$ is formed

TERMINATION rule dictates auction termination whenever a given period of bidding inactivity is observed.

TERMINATION
IF
Termination time window is W **AND**
Active proposal that generated currently highest bid was recorded at time Ta **AND**
Current time is Tc **AND**
$Tc > Ta + W$
THEN
Negotiation is declared terminated **AND**
Negotiation participants are notified accordingly

4 Representing Negotiation Rules in R2ML

Representing negotiation rules in a global information network (eg. an agent environment) requires a commonly agreed rule interchange format. This format must be able to support different rule languages within a single representation framework shared by all parties.

General purpose rule interchange formats, such as RuleML [21] and R2ML [20], address the Platform-Independent Model level (PIM) of a software or business system. One of their goals is to support a PSM[2] to PSM rule interchange via the PIM level.

[2] PSM stands for Platform-Specific Model i.e. a business system level that is dependent of the specific technological platform used to implement it.

Expressing negotiation rules at PIM level is a significant advantage since the business system does not require any conceptual changes when it is implemented in different specific technological platforms.

RuleML Initiative [21] aims at providing such a general purpose format. The SWRL [3] rule language tries to combine the rule concept from RuleML with the knowledge representation support of OWL [22]. However, both languages have limitations regarding the representation of well known concepts from software engineering: data types, operation calls, etc that are usually needed in real applications. Moreover, none of them supports Event-Condition-Action (ECA) rules that are basic kind of rules in agent negotiations (as seen in the previous section of this paper). The first ideas of a general rule language that will support not only the power of logic programming concepts, but also the widely used object oriented programming paradigm come from 2003 (see [23]). Following this work, proposal of R2ML rule markup language was recently launched [20]. R2ML supports ECA rules and provides markup for rules written in various rule languages including: Prolog, F-Logic [9], SQL, OCL [12], Jena [7], Jess [8], ILR [4], RuleML [21], SWRL [3].

Let us note that our negotiation rules are reaction rules (ECA rules) that follow the event-condition-action model. Therefore, we start with a brief description of the R2ML model of ECA rules and then we provide details of our proposed mapping.

4.1 R2ML ECA-Rules

A R2ML reaction rule is a statement of programming logic that specifies the execution of one or more actions in the case of a *triggering event* occurrence and if its *conditions* are satisfied. *Post-conditions* may be optionally required to be satisfied after the *action* execution. Reaction rules therefore have an operational semantics (formalizing state changes, e.g., on the basis of a state transition system formalism). The execution effect of reaction rules may depend on the rules order (note that the order is defined by the rule execution mechanism or by the rules representation).

The R2ML Events Metamodel specifies the core concepts required for dynamic behavior of rules and provides the infrastructure for more detailed definition of this behavior. Basic properties of an R2ML event expression are: *startDateTime*, *duration* (defines a value specification of the temporal distance between two time expressions that specify time instants) and *occurDateTime* (a derived property given by the addition of duration to the existent start date time).

For the purpose of encoding the agent negotiation rules we utilize only message event expressions. A *message event expression* is an atomic event described by two properties: i) *sender* which is the same with the actor (inherited from *ActionEventExpr*) and ii) *receiver*, an URI reference describing the receiver of the event. See [20] for more details on the R2ML event model.

4.2 Mapping Examples

This section is devoted to the description of the mapping of agent negotiation rules presented in Section 3 to R2ML. Because of space limitation, the mapping is illustrated

by means of few examples involving R2ML representations of vocabulary, rules, events, conditions and actions.

As R2ML is a rule-based language (other examples are Jess [8], JBoss Rules [6], Oracle Business Rules [14]), it provides the concepts of *ruleset*. Recall that our negotiation rules are based on the vocabulary described in Section 2. Vocabularies can be referred in R2ML rule sets and, moreover, for simplicity of implementations, R2ML provides its own markup for vocabularies. At the *ruleset* level a *specific vocabulary* for the entire set of rules can be encoded.

For example, the *Bid* class from our vocabulary can be represented as:

```
<r2mlv:Class r2mlv:ID="v:Bid">
 <r2mlv:Attribute r2mlv:ID="v:value">
  <r2mlv:range>
   <r2mlv:Datatype r2mlv:ID="xs:positiveInteger"/>
  </r2mlv:range>
 </r2mlv:Attribute>
 <r2mlv:Attribute r2mlv:ID="v:increment">
  <r2mlv:range>
   <r2mlv:Datatype r2mlv:ID="xs:decimal"/>
  </r2mlv:range>
 </r2mlv:Attribute>
 <r2mlv:Attribute r2mlv:ID="v:receivedTime">
  <r2mlv:range>
   <r2mlv:Datatype r2mlv:ID="xs:time"/>
  </r2mlv:range>
 </r2mlv:Attribute>
 <r2mlv:ReferenceProperty r2mlv:ID="v:participant">
  <r2mlv:range>
   <r2mlv:Class r2mlv:ID="v:Participant"/>
  </r2mlv:range>
 </r2mlv:ReferenceProperty>
</r2mlv:Class>
```

The *registered* association is represented as:

```
<r2mlv:ReferenceProperty r2mlv:ID="v:registered">
 <r2mlv:domain>
  <r2mlv:Class r2mlv:ID="v:Participant"/>
 </r2mlv:domain>
 <r2mlv:range>
  <r2mlv:Class r2mlv:ID="v:Negotiation"/>
 </r2mlv:range>
</r2mlv:ReferenceProperty>
```

Notice that `r2mlv` is the standard namespace notation for R2ML vocabulary[3] and `v` is a user-defined notation for his specific namespace of concepts[4].

[3] The R2ML vocabulary schema URL is `http://oxygen.informatik.tu-cottbus.de/R2ML/0.4/Vocabulary/r2mlv.xsd`

[4] For illustration purposes the vocabulary can be found at `http://www.example.org/ecommerce/agents/negotiation/vocabulary`.

We detail the mapping of the VALIDITY rule as example. The reader may consult Appendix 5 for the complete R2ML markup of another example rule.

The *triggering event* of this rule is the submission of a new proposal by a registered participant. We consider this event to be *atomic* i.e with *no duration*. This is represented in R2ML by an `MessageEventExpression`:

```
<r2ml:triggeringEvent>
 <r2ml:MessageEventExpression r2ml:sender="http://www.example.org/eshop"
        r2ml:startTime="2006-04-21T09:00:00"
        r2ml:duration="P0Y0M0DT0H0M0S"
        r2ml:eventType="e:submitProposal">
  <r2ml:arguments>
   <r2ml:ObjectVariable r2ml:name="N" r2ml:classID="v:Negotiation"/>
   <r2ml:ObjectVariable r2ml:name="S" r2ml:classID="v:Participant"/>
   <r2ml:ObjectVariable r2ml:name="Pr" r2ml:classID="v:Proposal"/>
  </r2ml:arguments>
 </r2ml:MessageEventExpression>
</r2ml:triggeringEvent>
```

Note that object variables `S` and `Pr` are instantiated by matching with the content of the incoming event and they are bound when rule conditions are evaluated.

The conditions part of the rule is a conjunction of three atoms and it can be expressed in R2ML as follows:

```
 <r2ml:conditions>
  <r2ml:ReferencePropertyAtom r2ml:referencePropertyID="v:registered">
   <r2ml:subject>
    <r2ml:ObjectVariable r2ml:name="S"/>
   </r2ml:subject>
   <r2ml:object>
    <r2ml:ObjectVariable r2ml:name="N" r2ml:classID="v:Negotiation"/>
   </r2ml:object>
  </r2ml:ReferencePropertyAtom>
  <r2ml:ReferencePropertyAtom r2ml:referencePropertyID="v:transacts">
   <r2ml:subject>
    <r2ml:ObjectVariable r2ml:name="S"/>
   </r2ml:subject>
   <r2ml:object>
    <r2ml:ObjectVariable r2ml:name="A" r2ml:classID="v:Good"/>
   </r2ml:object>
  </r2ml:ReferencePropertyAtom>
  <r2ml:AttributionAtom r2ml:attributeID="v:price">
   <r2ml:subject>
    <r2ml:ObjectVariable r2ml:name="Pr" r2ml:classID="v:Proposal"/>
   </r2ml:subject>
   <r2ml:dataValue>
    <r2ml:DataVariable r2ml:name="P" r2ml:datatypeID="xs:positiveInteger"/>
   </r2ml:dataValue>
  </r2ml:AttributionAtom>
 </r2ml:conditions>
```

First atom (lines 14–21) is a R2ML *reference property atom* that models the condition "S is a participant registered with negotiation N". This atom is true if participant denoted by variable S is registered with the current negotiation denoted by variable N.

The second atom (lines 22–29) is also a *reference property atom* describing the condition "S transacts good A".

The third atom (lines 30–37) is a R2ML *attribution atom* implementing the condition "Proposal Pr has price P". The execution model consists in computing the value of the attribute price in the context of the object variable Pr (the proposal).

The reader may notice that the condition "*S* has role $R \in \{buyer, seller\}$" is already implemented at the vocabulary level (classes Seller and Buyer are a complete partition of Participant).

Since the action part of the rule ("Submission time *T* is recorded with proposal *Pr*") denotes an update that invokes a "recording operation", it will go into an R2ML *invoke action expression*. The action receives as argument a R2ML *attribute function term* that evaluates to the value of the attribute v:timeReceived of proposal Pr. Note that this corresponds to an UML-like operation call recordSubmissionTime(Pr.timeReceived). The resulting R2ML markup of the action is:

```
39 <r2ml:producedAction>
40  <r2ml:InvokeActionExpression r2ml:operationID="a:recordSubmissionTime">
41   <r2ml:arguments>
42    <r2ml:AttributeFunctionTerm r2ml:attributeID="v:timeReceived">
43     <r2ml:contextArgument>
44      <r2ml:ObjectVariable r2ml:name="Pr" r2ml:classID="v:Proposal"/>
45     </r2ml:contextArgument>
46    </r2ml:AttributeFunctionTerm>
47   </r2ml:arguments>
48  </r2ml:InvokeActionExpression>
49 </r2ml:producedAction>
```

The VALIDITY rule has also a postcondition – "Proposal *Pr* is valid". This postcondition corresponds to a R2ML *object classification atom*:

```
50 <r2ml:postcondition>
51  <r2ml:ObjectClassificationAtom r2ml:classID="v:ValidProposal">
52   <r2ml:ObjectVariable r2ml:name="Pr"/>
53  </r2ml:ObjectClassificationAtom>
54 </r2ml:postcondition>
```

4.3 General Mapping Criteria

Business rules (including those presented in Section 3) are not usually captured using a formal representation. Instead, they are natural language descriptions based on core ontological concepts (eg. variable and class) and have the usual meaning of IF ... THEN programming constructs. It is the role of the rule engineer to map them onto a formal representation. Below we describe the general mapping criteria of such a formalization using R2ML:

1. Rule variables are mapped onto object variables or data variables according to their values types:

- *Object variables*, if they instantiate classes;
- *Data variables*, if they instantiate datatypes;

2. UML properties are mapped onto different kinds of atoms according to their ranges:
   - Attributes i.e. UML properties that have data as values are mapped onto *attribution atoms* or *attribute function terms* depending of the context of usage. For example the UML expression `Pr.price=P` is mapped onto the attribution atom from lines 29–36. See also the attribute function term from lines 41–45 in the example.
   - Object properties i.e. UML properties that have objects as values are mapped onto R2ML *reference property atoms* or *reference property function terms* depending of the context of usage. For example, the reference property atom from lines 21–28 encodes the UML expression `S.transacts=A`;
3. Actions are mapped onto one of:
   - *InvokeActionExpression*, corresponding to an operation call;
   - *AssignActionExpression*, corresponding to assignment of values to different UML attributes;
   - *CreateActionExpression*, corresponding to a constructor-call;
   - *DeleteActionExpression*, corresponding to a destructor-call;
4. Because the negotiation box rules are triggered by instantaneous events (like request/response of a message), events are mapped onto the subclass of R2ML message events that represents atomic events (events without duration).

In Appendix 5 we present another complete example expressed in R2ML for the IMPROVEMENT-BUYER rule.

5 Conclusions and Future Work

This paper proposes the use of R2ML rule-markup language for expressing rule-based representations of agent negotiation mechanisms. Our proposal is demonstrated with an example comprising an R2ML rule model of single item English auctions.

As future work we plan to: (i) analyze how the R2ML representation of negotiation mechanisms can be implemented using a rule engine in a system for agent negotiation; (ii) asses the generality of this proposal by applying it to other price negotiations.

Appendix A

R2ML markup for IMPROVEMENT-BUYER rule:

```
<r2ml:ReactionRule  r2ml:id="IR-BUYER001">
 <r2ml:triggeringEvent>
  <r2ml:MessageEventExpression r2ml:sender="www.example.org/eshop"
            r2ml:startTime="2006-04-21T09:00:00"
            r2ml:duration="P0Y0M0DT0H0M0S" r2ml:eventType="s:submitProposal">
   <r2ml:arguments>
    <r2ml:ObjectVariable r2ml:name="S" r2ml:classID="v:Buyer"/>
    <r2ml:ObjectVariable r2ml:name="Pr" r2ml:classID="v:Proposal"/>
   </r2ml:arguments>
  </r2ml:MessageEventExpression>
 </r2ml:triggeringEvent>
```

```
<r2ml:conditions>
 <r2ml:ReferencePropertyAtom r2ml:referencePropertyID="v:registered">
  <r2ml:subject>
   <r2ml:ObjectVariable r2ml:name="S"/>
  </r2ml:subject>
  <r2ml:object>
   <r2ml:ObjectVariable r2ml:name="N" r2ml:classID="v:Negotiation"/>
  </r2ml:object>
 </r2ml:ReferencePropertyAtom>
 <r2ml:ReferencePropertyAtom r2ml:referencePropertyID="v:transacts">
  <r2ml:subject>
   <r2ml:ObjectVariable r2ml:name="S"/>
  </r2ml:subject>
  <r2ml:object>
   <r2ml:ObjectVariable r2ml:name="A" r2ml:classID="v:Good"/>
  </r2ml:object>
 </r2ml:ReferencePropertyAtom>
 <r2ml:DatatypePredicateAtom r2ml:datatypePredicateID="swrlb:greaterThan">
  <r2ml:dataArguments>
   <r2ml:AttributeFunctionTerm r2ml:attributeID="v:price">
    <r2ml:contextArgument>
     <r2ml:ObjectVariable r2ml:name="Pr"/>
    </r2ml:contextArgument>
   </r2ml:AttributeFunctionTerm>
   <r2ml:DataOperationTerm r2ml:operationID="op:numeric-add">
    <r2ml:arguments>
     <r2ml:AttributeFunctionTerm r2ml:attributeID="v:value">
      <r2ml:contextArgument>
       <r2ml:ObjectVariable r2ml:name="B" r2ml:classID="v:Bid"/>
      </r2ml:contextArgument>
     </r2ml:AttributeFunctionTerm>
     <r2ml:AttributeFunctionTerm r2ml:attributeID="v:increment">
       <r2ml:contextArgument>
        <r2ml:ObjectVariable r2ml:name="B" r2ml:classID="v:Bid"/>
       </r2ml:contextArgument>
     </r2ml:AttributeFunctionTerm>
    </r2ml:arguments>
   </r2ml:DataOperationTerm>
  </r2ml:dataArguments>
 </r2ml:DatatypePredicateAtom>
</r2ml:conditions>
<r2ml:producedAction>
 <r2ml:InvokeActionExpression r2ml:operationID="a:assert">
  <r2ml:arguments>
   <r2ml:ObjectVariable r2ml:name="Pr"/>
  </r2ml:arguments>
 </r2ml:InvokeActionExpression>
</r2ml:producedAction>
<r2ml:postcondition>
 <r2ml:ObjectClassificationAtom  r2ml:classID="v:ActiveProposal">
   <r2ml:ObjectVariable r2ml:name="Pr"/>
 </r2ml:ObjectClassificationAtom>
</r2ml:postcondition>
</r2ml:ReactionRule>
```

References

1. Bădică, C., Bădiţă, A., Ganzha, M., Iordache, A., Paprzycki, M.: *Rule-Based Framework for Automated Negotiation: Initial Implementation*. In: A. Adi, S. Stoutenburg, S. Tabet (eds.): Proc. RuleML'2005, Galway, Ireland. LNCS 3791, Springer Verlag 2005, 193–198.
2. Bartolini, C., Preist, C., Jennings, N.R.: *A Software Framework for Automated Negotiation*. In: Proc. SELMAS'2004, LNCS 3390, Springer Verlag 2005, 213–235.

3. Horrocks I., Patel-Schneider P. F., Boley H., Tabet S., Grosof B., Dean M.: *SWRL: A Semantic Web Rule Language Combining OWL and RuleML.* W3C Member Submission 21 May 2004, http://www.w3.org/Submission/SWRL/
4. *The ILOG Rule Language*. http://www.ilog.com/
5. JADE: Java Agent Development Framework. http://jade.cselt.it2
6. JBoss Rules (Drools) http://www.drools.org
7. *Jena The Semantic Web Framework.* http://jena.sourceforge.net/
8. *Jess, Sandia Lab.*, http://herzberg.ca.sandia.gov/jess/
9. Kifer, M., Lausen, G., and Wu, J.: *Logical Foundations of Object-Oriented and Frame-Based Languages*. Journal of ACM, May 1995. ftp://ftp.cs.sunysb.edu/pub/TechReports/kifer/flogic.pdf
10. Laudon, K.C., Traver, C.G.: *E-commerce. business. technology. society* (2nd ed.). Pearson Addison-Wesley, 2004.
11. Lomuscio, A.R., Wooldridge, M., Jennings, N.R.: *A classification scheme for negotiation in electronic commerce*. In: F. Dignum, C. Sierra (Eds.): Agent Mediated Electronic Commerce: The European AgentLink Perspective, LNCS 1991, Springer Verlag 2002, 19–33.
12. *Object Constraint Language (OCL)*, v2.0, // http://www.omg.org/docs/ptc/03-10-14.pdf
13. Object Management Group (OMG), http://www.omg.org
14. Oracle Business Rules, http://www.oracle.com/technology/products/ias/business_rules/index.html.
15. McAfee, R.P., McMillan, J.: *Auctions and bidding*. In: Journal of Economic Literature, 1987, 25(2):699–738.
16. Rolli, D., Luckner, S., Gimpel, A.: A Descriptive Auction Language. Electronic Markets. In: Electronic Markets – The International Journal, 2005.
17. Rolli, D., Eberhart, A.: An Auction Reference Model for Describing and Running Auctions. In: 7 Internationale Tagung Wirtschaftsinformatik, Bamberg, Germany, 2005.
18. Tamma, V., Phelps, S., Dickinson, I., Wooldridge, M.: *Ontologies for Supporting Negotiation in E-Commerce.* In: Engineering Applications of Artificial Intelligence, 18, Elsevier, 2005, 223–238.
19. Tamma, V., Wooldridge, M., Dickinson, I.: *An Ontology Based Approach to Automated Negotiation*. In: Proceedings AMEC'02: Agent Mediated Electronic Commerce, LNAI 2531, Springer-Verlag 2002 219–237.
20. *R2ML - The REWERSE I1 Rule Markup Language*, http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/6
21. *The Rule Markup Initiative*, RuleML, http://www.ruleml.org.
22. Patel-Schneider, Peter F., Horroks I.: *OWL Web Ontology Language Semantic and Abstract Syntax*, http://www.w3.org/2004/OWL
23. Wagner, G.: *Seven Golden Rules for a Web Rule Language.* Invited contribution to the Trends & Controversies section of IEEE Intelligent Systems 18:5, Sept/Oct 2003.
24. Wagner, G., Giurca, A., Lukichev, S.: *R2ML: A General Approach for Marking up Rules*, Dagstuhl Seminar Proceedings 05371, In: F. Bry, F. Fages, M. Marchiori, H. Ohlbach (Eds.) Principles and Practices of Semantic Web Reasoning, 2005.
25. Wagner, G., Giurca, A., Lukichev, S.: *A Usable Interchange Format for Rich Syntax Rules Integrating OCL, RuleML and SWRL*, RoW2006, Edinburgh, UK, May 22nd, 2006.
26. Wooldridge, M.: *An Introduction to MultiAgent Systems*, John Wiley & Sons, 2002.
27. Wurman, P.R., Wellman, M.P., Walsh, W.E.: *Specifying Rules for Electronic Auctions*. In: AI Magazine, 2002, 23(3),15–23.
28. Wurman, P.R., Wellman, M.P., Walsh, W.E.: *A Parameterization of the Auction Design Space*. In: Games and Economic Behavior, 35, Vol.1/2 2001, 271–303.