Specification of Trade-Off Strategies for Agents: A Model-Driven Approach

René Schumann[1], Zijad Kurtanovic[2], and Ingo J. Timm[3]

[1] HES-SO,
Rue de Technopôle 3, 3960 Sierre, Switzerland
rene.schumann@hevs.ch
[2] University of Hamburg
22527 Hamburg, Germany
kurtanovic@informatik.uni-hamburg.de
[3] University of Trier
54286 Trier, Germany
ingo.timm@uni-trier.de

Abstract. Negotiation is an important part of today's business processes on an inter-enterprise level. Agent research offers a variety of approaches and tools to automate negotiations. Currently these technologies have the drawback that the human manager retains the responsibility for the outcome of a negotiation, but this person most often does not have the required knowledge to define the agent's behavior by himself. To increase acceptability of automated negotiation approaches, we consider it necessary that human negotiators can specify the strategies for the agents. In this article we present a meta model, which enables human negotiators to specify trade-off strategies. Trade-off strategies are a key concept in negotiation in general. This meta model is based on the Ecore meta model. The specified trade-off strategies can automatically be transformed into representations that can be used by an agent. Our meta model provides a model and a graphical notation that allows it to create graphical editors for trade-off strategies. Therefore, it becomes possible to specify trade-off strategies without any programming knowledge.

Keywords: automated negotiation, trade-off strategies, MDD.

1 Introduction

Negotiation is an important part of today's business processes on an inter-enterprise level. Agent research offers a variety of approaches and tools to automate negotiations. To enable agents to act on behalf of humans, several challenges have to be dealt with. From our perspective we see a particular challenge in the fact, that the human principal of an agent retains responsibility for the outcome of an automated negotiation. Thus, to establish agent-based negotiations it is required to a) give quality guarantees to the human principal, or b) allow the principal to specify the negotiation strategy of the agent. Independently of the chosen approach it has to be discussed if humans are willing to delegate negotiation tasks to agents.

J.P. Müller and M. Cossentino (Eds.): AOSE 2012, LNCS 7852, pp. 40–55, 2013.

Also it seems reasonable that automated negotiation can be support for humans, and that the human negotiator has in the end to accept the proposed deal of the system, before it becomes legally binding. In this article we restrict ourself to discuss the technique how negotiations can be realized, that follow a given negotiation strategy. How these techniques afterwards can be integrated into systems designed to support human negotiators is beyond the scope of this paper.

In real-world negotiation, multiple parameters as well as non-rational behavior of actors have to be considered. Thus, formal guarantees, i.e., proof of optimal behavior does not seem to be feasible. Therefore, we are focusing here on the second approach, enabling the human principal to specify her strategy. Such an approach includes means for the acquisition of negotiation knowledge from the human principal [1] of the agent, typically this is a manager or a negotiator. The negotiator has implicit knowledge about the negotiation process and related negotiation strategies. The knowledge on the process will be encoded in protocols. The knowledge on strategies is essential to automated negotiation in practical settings and has to be encoded, too. In this article we focus on trade-off strategies. In particular we present a meta model for trade-off strategies, and how strategies that have been defined based on this model, can be automatically transformed into a format that can be used by a negotiating agent.

A trade-off between two negotiation attributes specifies a preference among pairs of assignments to both variables. The idea of a trade-off is to improve one attribute while worsening the other in return [1]. Trade-offs are an important aspect of negotiations in human behavior [2,3] and have been adopted for negotiations among software agents, see e.g. [4,1].

A current problem in the application of autonomous agents as negotiators, or agents that support human negotiators, is that the human negotiator is not capable of designing or programming the agent, even though he remains responsible for the outcome of the negotiation. While on the other side the programmers can have problems understanding the strategy used during the negotiation.s In our research we want to provide means to bridge this mismatch, by providing means for specification of trade-off strategies that enables human negotiators to specify their trade-off strategies in a comprehensive way. Of course, trade-off strategies are only one part of negotiation strategies, which comprises also other aspects, like the protocols or the effects of time passing during the negotiation, e.g. if a deal has to be reached within a given time.

We use a model-driven development (MDD) approach to automatically translate a specification of a trade-off strategy into a representation that can be used by a software agent. Thus, the person specifying the strategies is not required to have knowledge about software agents or programming. Consequently, we empower the person responsible for the negotiation, the principal of the agent, to specify the strategy of the agents by himself. The need for encoding these strategies by hand becomes obsolete.

The rest of this article is structured as follows. In the next section we outline related work. Afterwards we describe formally the concept of trade-off strategies (Section 3.1). Based on this background we present our meta model (Section 3.2)

and how it can be used to specify a trade-off strategy (Section 3.3). The transformation process of a strategy is outlined in Section 3.4. In Section 4 we present an example how our meta model can be used to specify trade-off strategies in a show case. Finally, we summarize our work and outline future research.

2 Related Work

In this paper, we focus on means of modeling trade-off strategies to enable humans to specify their strategies with the goal to support them by (partially) automate the negotiation process.

Automated negotiation is an established and active research topic in multiagent systems research, see e.g. [5,6]. To increase user acceptance, agents which act on behalf of humans in negotiations require knowledge from the human expert. Surprisingly, it has to be stated that only little research has been done in investigating how to acquire negotiations strategies from human experts in the field of automated negotiations, the work by Lou et al. [1] is one of those rare exceptions. Only very few work has been done to provide means that would allow to specify negotiation strategies in a declarative way.

One exception is the work by Chiu et al. [7]. The author present an e-Negotiation process based on an ontology. The process should support human negotiators to specify their negotiation strategies. The improvement of our work is, that we have combined means to specify trade-off strategies with MDD techniques to automatically generate a representation that can be used by agents, without any additional human effort.

Benyoucef and Rinderle [8] have presented a model-driven approach for developing service-oriented negotiation systems. Their specification of the negotiation behavior uses also a declarative approach for the specification. In our paper we strictly focus only to trade-off strategies.

In [1] user's trade-off strategies and preferences are acquired by using the *default-then-adjust* method. This approach is based on the use of a preexisting default knowledge with the aim to assist the user and reduce their workload. On the one side, such knowledge can be an important assistance for the user, on the other side the access to relevant expertise is limited and often not available [9].

Our work is based on the definition of trade-off- and preference-functions presented in [1], we have modified them slightly as follows:

- In [1] it is assumed that the domains of negotiation attributes are all continuous and numeric. We have relaxed this assumption and can handle domains with symbolic values and also attributes with discrete domains, too. This is done by mapping symbolic values to numeric values.
- To avoid formal case-based considerations and to be able to base a trade-off strategy on a pair of attributes with heterogeneous value sets, the domains of all negotiation attributes are represented as numeric and continuous. In the agent's reasoning his proposals are based on this assumption and are then approximated to become conform with the actual domains.

Another field in multiagent system research that is considered as related work here is the application of MDD techniques to multiagent system development. The probably most widely known system for the model-driven development of multiagent systems (MAS) is the INGENIAS system [10], which can be considered as a forerunner for MDD development of MAS. Recently, Hahn et al. have developed a coherent modeling framework for MAS. In their effort Hahn et al. have developed a platform-independent meta model for MAS [11], and afterwards detailed their model by adding modeling capabilities for interaction protocols [12]. In current research efforts the development of MDD tooling for MAS is top-down. Meta models are presented for MAS in general, as pointed out above. Their goal is to allow the modeling of entire MAS and their generation. In the approach we present here we only cover one aspect, namely the modeling of trade-off strategies. This bottom-up approach tries to come up with dedicated aspects of a MAS, that can be put in use early on. Also it would be interesting to see to what extend the work proposed in this article can be used in existing approaches. The negotiation meta model is not designed to be exclusive to other approaches, but allows for a more detailed modeling of a specific aspect of a MAS.

3 A Meta Model for Trade-Off Strategies

Before we are going to present the meta model for trade-off strategies (Section 3.2) we provide a brief definition of the concepts used. Afterwards, we detail how a trade-off strategy can be specified (Section 3.3) and transformed (Section 3.4) to be used by an agent during a negotiation.

3.1 Trade-Off Strategies

A trade-off strategy specifies what combination of attributes' values form an acceptable deal for the user. Within a trade-off strategy all information about the trade-offs is encoded. Formally, a trade-off strategy contains a set of trade-off relations and a set of *independent* attributes. Independent attributes are not member of a trade-off relation with other attributes.

A trade-off is a relation between two negotiation attributes. It defines that in favor for worsening one attribute the other one has to improve. Note that we focus here on binary trade-off relationships, since these are the most common ones [3]. According to [1] a trade-off function can be formalized as follows:

Definition 1
Let the domain of the attribute x be $X = [l_x, r_x]$, and the domain of y be $Y = [l_y, r_y]$. Note the domains values are ordered. A function $f : X \to Y$ is a trade-off function if it is continuous, monotonic *and met the* boundary condition. *The boundary condition ensures, that if one attribute is assigned to the best value, the other attribute has to be made worse [1].*

If a trade-off function exists between two attributes they are also called to be a trade-off pair. For each trade-off a preference function $p : A \times B \rightarrow [0, 1]$ is defined, which define the preference over the trade-off alternatives. It reflects a trapezoid formula of three segments (analogue to the preference function in [1]) to describe the increasing, steady and decreasing preference over trade-off alternatives. Trade-off and preference functions can also be specified graphically. An example is shown in Figure 1. In Figure 1 right hand side, a trade-off function between price (on the x-axis) and accuracy (on the y axis) is shown. A preference function indicates the degree of satisfaction (Figure 1 left hand side) of the negotiator, expressed in terms of an interval between [0,1]. Thereby, 1 is indicating high satisfaction and 0 is indicating dissatisfaction.

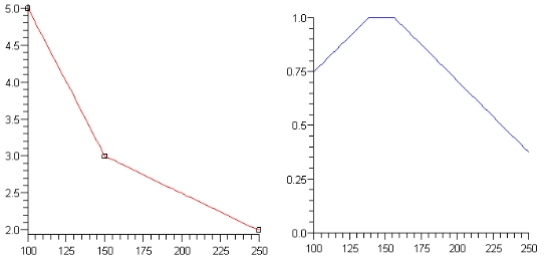


Fig. 1. Left: Example of a trade-off function between price (x-axis) and accuracy (y-axis), Right: Preference function about prices. Prices x-axis, Satisfaction degree y-axis.

Preference functions are also defined for independent attributes as $p : A \rightarrow [0, 1]$, with A: $\forall a, b \in A : \ a \preceq b \Leftrightarrow p(a) \leq p(b)$.

Following the previous definition a trade-off strategy can be represented as a forest, as illustrated in Figure 2: the nodes represent negotiation attributes and the edges trade-off relations.

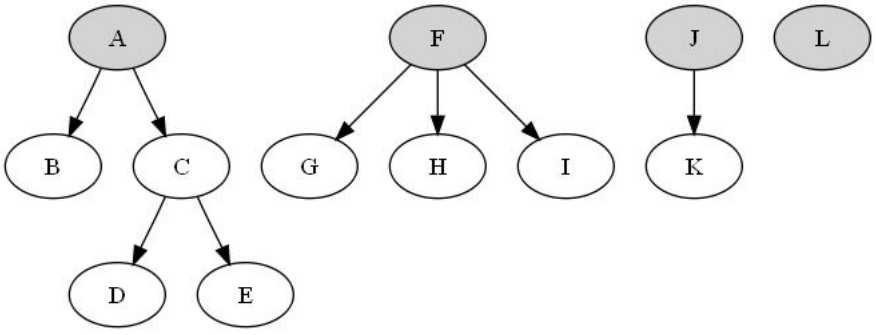


Fig. 2. Representing a trade-off strategy as a forest. L is an independent attribute.

This ensures some formal but also informal benefits. A trade-off strategy can be visualized in a *clear* and *accustomed* way to the users. Due to the acyclic structure there cannot exist inconsistencies, which may be introduced by cycles. This reduces the complexity for specifying and validating these strategies. Moreover, the forest structure allows to use of efficient algorithms for reasoning about the trade-off strategies [13].

3.2 Meta Model

We base our *negostrategy*-meta model on the meta model Ecore of the Eclipse Modeling Framework (EMF)[1]. As shown in Figure 3 the meta model for trade-off strategies has been modeled in two packages. Within the `base` package

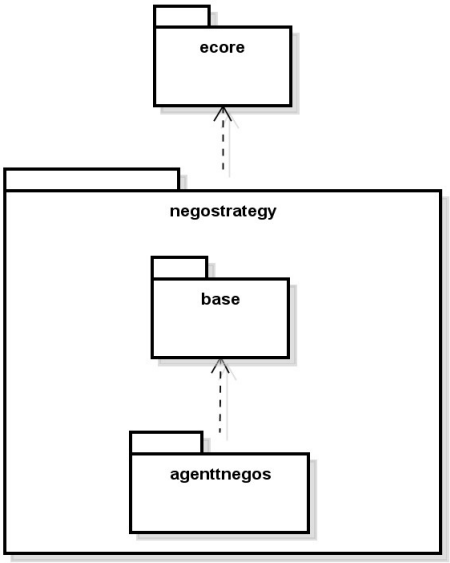


Fig. 3. Package diagram of the trade-off meta model

`negotiation attributes` are defined that can be used to define trade-off strategies. Also basic operators and other term are defined in the `base` package. The `base` package is detailed in Figure 4.

Trade-off relations are defined in the `agentnegos` package (Figure 5). Within this package we distinct between attributes that describe the context, in which a trade-off relation is valid, and attributes used within a trade-off relation. A trade-off strategy is formed by a set of trade-off relations and their context. The context allows to specify when a particular relation is applicable.

[1] see `http://www.eclipse.org/modeling/emf`, accessed at 27.10.2012.

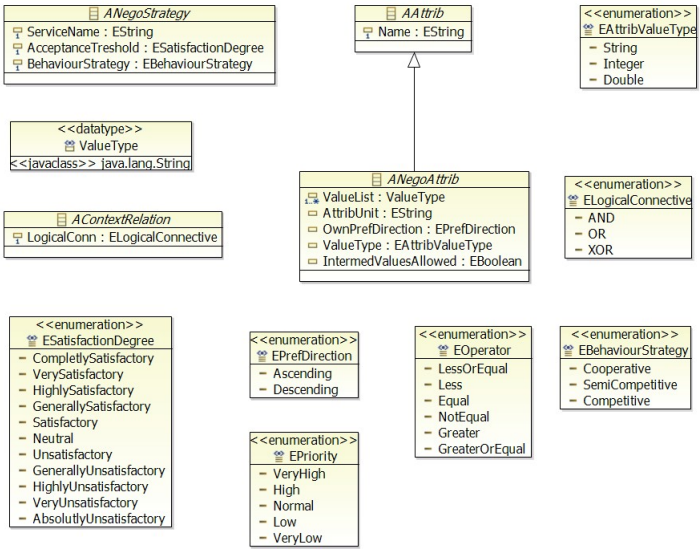


Fig. 4. The `base` package of the trade-off strategy meta model

In the following we highlight the three major concepts of the proposed meta model in more details, these are *negotiation strategies root*, *trade-off negotiation attribute*, and *trade-off relation*. The entire meta model can be found in [14]. For most of the concepts we have also defined a graphical notation, to allow further extensions, like a graphical editor for defining trade-off strategies, following our vision that the agent behavior can be defined by the principal of the agent, and not by a programmer.

The *negotiation strategy root* contains all negotiation attributes and their trade-off relationships. It can be seen as the artificial root node, for the entire forest representing a trade-off strategy. Each tree in the trade-off strategy has a priority, so the relative value between the trees can be encoded. Also an acceptance threshold is stored in the strategy. The acceptance threshold specifies a value that the agent use to a) generate an offer that is acceptable or b) decide on acceptability of an offer for the user. The graphical representation for this concept is a trapezoid containing the name of the service, as shown in Figure 6(a).

A *trade-off negotiation attribute* represents a negotiation attribute used in a trade-off strategy. It has a name, a domain, which can be an continuous interval([]) or a discrete enumeration({}), and a preference function over it's domain, plus a list of trade-off relationships in which this attribute is involved. The graphical representation is shown in Figure 6(b).

A *trade-off relation* encodes the trade-off function between two attributes. A relation defines the optimal combination of values between the two attributes. Given the optimal combination between the attribute values of the trade-off function and the preference functions, the trade-off combinations can be computed and ranked. A *trade-off relation* is represented as a labeled edge connecting the

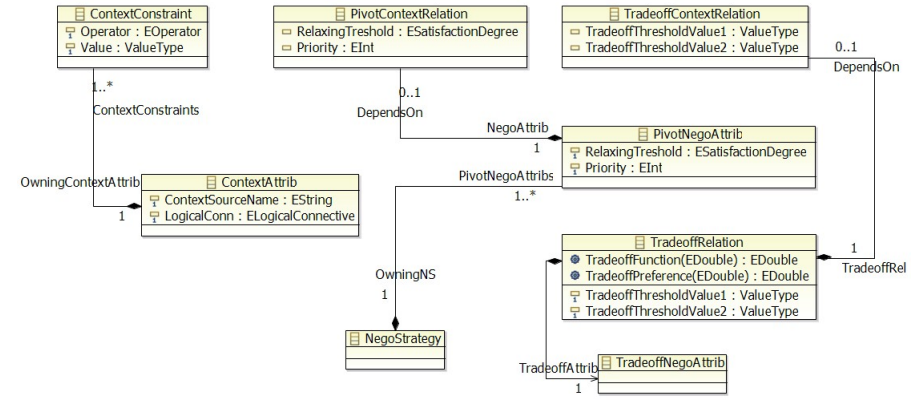


Fig. 5. The `agentnegos` package of the trade-off strategy meta model

(a) Negotiation strategy root

(b) Negotiation attribute

(c) Trade-off relation between two negotiation attributes

Fig. 6. Graphical representation for negotiation strategy concepts

two negotiation attributes. The label is the optimal value combination of both attributes. An example is shown in Figure 6(c).

Additionally, it is possible to specify the context for which the trade-off relations are defined. For instance, it is possible to differentiate the strategies depending on with whom the agent negotiate.

3.3 Specification of Trade-off Strategies

As we use the EMF as a base for our meta model we have the option to provide a graphical editor for negotiation strategies in the near future. As outline above, this would enable strategy specification by non IT-experts, which most often comprises the persons responsible for the outcomes of a negotiation, e.g. a manager.

Currently, a negotiation strategy is specified using an Eclipse widget that has been generated automatically based on the specification of the meta model.

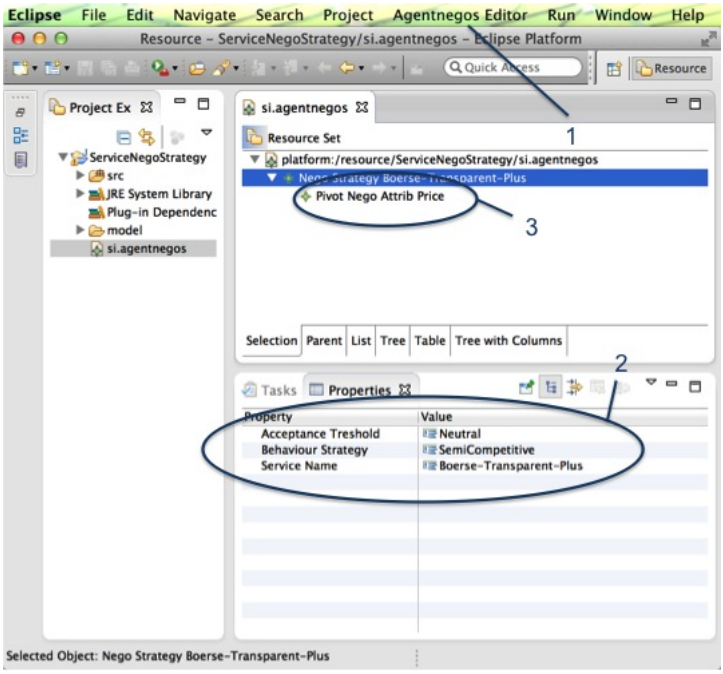


Fig. 7. Overview of the current trade-off strategy editor: 1) points at the particular tab for editing the trade-off strategy 2) panel for the editing of attribute values 3) panel for the navigation within the attribute tree forming the data structure used for the trade-off strategy

A screen shot is shown in Figure 7. Of course this form is not suitable for non IT experts. Therefore a graphical editor is needed. The strategy is specified in form of structured attribute value pairs, as shown in Figure 8. The specified strategy will be validated against the meta model and saved in the XMI format.

3.4 Model Transformation

If a trade-off strategy has been specified it needs to be transformed into a representation that can be used by a software agent. This requires platform-specific details which supplies the EMF-generator with information like the connections between multiple Ecore-models, the name of the generated files, referenced Ecore-models etc. [15]. Based on the XMI file a generator transforms a trade-off strategy into a representation that can be used by an agent. We have decided to use a relational representation. In this representation all acceptable deals, i.e., combination of values for negotiation attributes that exceed the specified acceptance threshold, are stored.

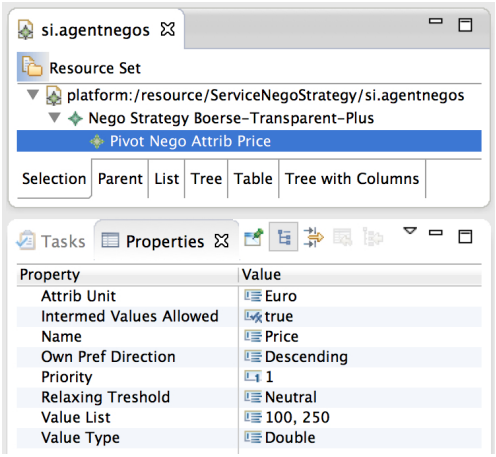


Fig. 8. Detailed view on the panel for editing the negotiation strategy

These deals are precomputed, because the problem of finding the next best offer in a negotiation process would be too time consuming, for details see Lou et al. [4]. In the same way the set of acceptable deals is precomputed. This shifts the computational efforts form the execution into the compilation phase. During the negotiation, queries are performed to retrieve elements with specific characteristics in the set of acceptable deals. This querying can be done efficiently using a relational representation, e.g., in form of SQL queries [16].

A trade-off strategy is transformed into a set of tables. For each tree of a trade-off strategy a table is generated. Each negotiation attribute is represented by a column. Each row specifies one acceptable deal. The set of tables is computed by the function `generateTables` presented in Algorithm 1. For each tree of a trade-off strategy, first a representation set of the root is generated. Afterwards the representation sets of the direct child-nodes are induced by the corresponding trade-off functions. Each directed edge in a tree represents a trade-off relation from *parent* node to *child* node.

Consequently, a resulting table has a column for the parent node's values, possible several columns holding child node's values, a column with the preference values and one column with preference values with priority degree considered. The structure of such a table is outlined in Table 1. In the resulting table preferences and priority-preferences are aggregated, e.g. using the arithmetic mean. Examples for such tables can be seen below in Figures 12 and 13. For retrieving information about a negotiation attribute it can become necessary to join several tables.

Table 1. Example of a table representing a trade-off relation between *ParentAttrib1* and *ChildAttrib1*

ParentAttrib1	ChildAttrib1	Pref	PrioPref
...	...	...	...

Algorithm 1. Pseudo code of the tree transformation algorithm

```
function generateTables(set of trees forming a trade-off strategy)
    for each Tree do
        generate a representation set of the root
        Call InduceRepresentationSet(root)
    end for
end function
function InduceRepresentationSet(Node X)
    if X NOT ROOT then
        Induce a representation set from X
    end if
    for each Child-Node C of X do
        InduceRepresentationSet(C)
    end for
end function
```

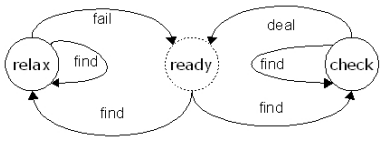
4 Automating Negotiations: A Case Study

In this section we demonstrate the specification of a negotiation by an example. We show for a simple negotiation scenario how trade-off specification can be done, and the resulting negotiation outcomes.

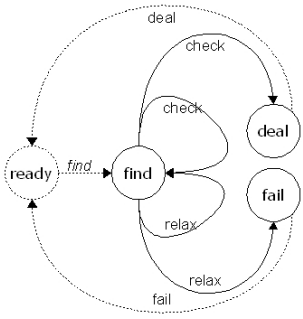
4.1 Negotiation Model

Since we focus here on the trade-off strategies, we choose an existing negotiation model presented by Lou et al. [4]. It is a simple bilateral negotiation setting. Two roles are defined: a buyer and a seller agent. Both agents negotiate about a contract with a number of attributes like price, quality, delivery or payment date. Each agent has a global preference function for all permutations of all possible outcomes of the negotiation. The seller provides access to an information service, that the buyer wants to subscribe to. Attributes of the contract are price, actuality of the data, contract duration, and accounting period.

The agents operate in a semi-competitive environment. This is reflected by their behavior strategies which are based on the *principled negotiation approach* [2]. They try to weaken their position only minimally, e.g., by minimal information disclosure, and minimal relaxation of their desires [4]. The negotiation protocol is based on the alternating offers protocol [17]. The behavior of the seller agent is presented in Figure 9(a). The `ready` state is the initial state. The states `check` and `relax` represent the allowed performatives of sending messages of an agent in a given negotiation state. The edges represent the performatives of the buyer agent that can be received during a negotiation encounter. When the performative *find* is received the negotiation is initiated and the agent can answer with the performatives *check* or *relax*. The performative *check* is used to ask the other agent to check if an offer satisfies its requirements. If no offer could be found that satisfies the published buyers constraints, the seller asks to *relax* at least one of the constraints, so that a suitable offer can be found.



(a) Seller agent's behavior protocol



(b) Buyer agent's behavior protocol

Fig. 9. Negotiation protocol for seller and buyer

The buyer agent's behavior protocol is presented in Figure 9(b). The states and edges are defined analogous to figure 9(a). The buyer starts a negotiations by sending a *find*-performative to the seller agent. Agents constrains future offers by sending constraints that all offers have to fulfill, e.g., the price should be below 340€. Constraints are published with descending priority. An offer is checked and either accepted or another constraint is published to specify the requirements more precisely, by communicating the violated constraint with the highest priority. The constraints that needs to be satisfied and the preference function among all available feature combination of negotiation attributes encodes the negotiation strategy. Both have to be defined by the principal of the agent.

4.2 Specification of Trade-off Strategies for the Example

From the seller's perspective the negotiation attributes have the following domains: The price can be in a range between [120,270] €, of course a higher price is preferred. The delivered data can have an actuality of 1,2,4 or 6 hours. As more accurate data is more expensive, older data is preferred. The seller assumes its optimal ration between profit and accuracy is€ 170,- for two hour old data. Possible contract durations are 6,12,18 or 24 month, a longer duration is preferred. Accounting periods are 1,3,4 or 6 months, shorter periods are preferred, not giving a credit to the customer. Based on the notation presented in Section 3 the resulting trade-off strategy can be modeled as shown in Figure 10. v

From the buyer's perspective the attributes have other desired values and preferences, of course. The price should be in the interval between [100,200] €, and a lower price is preferred. Actuality of the data should be between two and five hours, more accurate data is preferred and a higher price is acceptable. A fair ratio between accuracy and price for the buyer is paying € 150,- for three hours old data. The contract duration can be in the interval between [3,24] month, where a shorter duration is preferred. For a better (for the buyer a lower) price

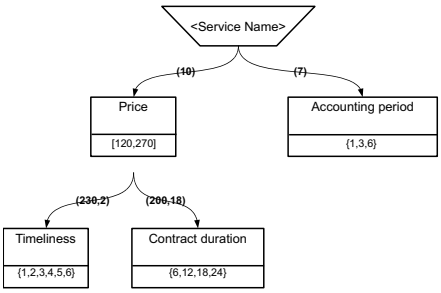


Fig. 10. Graphical representation of the seller trade-off strategy

the buyer is willing to accept a longer contract duration. Acceptable accounting periods can be one to three month. Longer periods are preferred, but for a better price, shorter periods can be accepted. The graphical model for this strategy is shown in Figure 11.

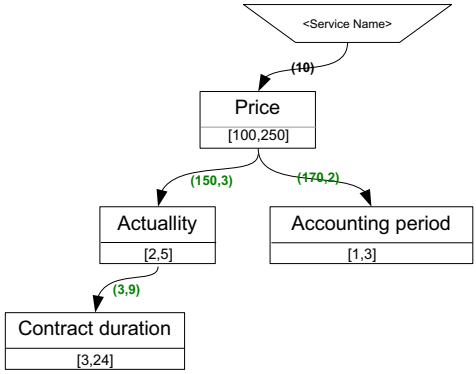


Fig. 11. Graphical representation of the buyers trade-off strategy

We have specified these two strategies with our editor and generated the relational representation for these strategies. The resulting set of acceptable deals have been generated. The relational representation for the seller's strategy is shown in Figure 12 and for the buyer's strategy in Figure 13.

We have implemented negotiating agents, based on the Jade framework[2], that use the relation representation to negotiate with each other. In Table 2 we present the negotiation process, as it has been executed by the agents for the described example. The buyer starts the negotiation by selecting the row of the table shown

[2] see `http://jade.tilab.com`, accessed at 27.10.2012.

PRICE	ACTUALITY	CONTRACTDURATION	PREF	PRIOPREF
165.0	2.4	16.5	0.96...	0.965000...
180.0	1.9	15.0	0.885	0.885
150.0	3.6	18.0	0.855	0.855
195.0	1.75	13.5	0.74	0.74

ACCOUNTINGPERIOD	PREF	PRIOPREF
1.0	1.0	1.0
1.5	0.95	0.985
2.0	0.84	0.952
2.5	0.74	0.922
3.0	0.63	0.889
3.5	0.53	0.859

Fig. 12. Relational representation of seller's trade-off strategy. His thresholds are already considered, thus only acceptable trade-offs are shown.

ACCOUNTINGPERIOD	PRICE	ACTUALITY	CONTRACTDURATION	PREF	PRIOPREF
1.86	160.0	2.9	10.5	0.97...	0.97666...
1.64	145.0	3.2	8.4	0.89...	0.89666...
2.06	175.0	2.75	12.75	0.89...	0.89666...
2.25	190.0	2.6	15.0	0.73	0.73
1.43	130.0	3.8	6.6	0.61...	0.61333...
2.44	205.0	2.45	17.25	0.54...	0.54666...

Fig. 13. Relational representation of all possible attribute combinations of the attributes price, actuality and contract duration

Table 2. Full negotiation trace of buyer and seller (PR: price, AC actuality, CD contract duration, AP accounting period)

Round 1	Buyer	Performative: Find Constraint: PR $\leq$ 160
	Seller	Performative: Check (PR:150,AC:4,CD:18,AP:1)
Round 2	Buyer	Performative: Find Constraint: PR $\leq$ 160 $\wedge$ AC $\leq$3
	Seller	Performative: Relax
Round 3	Buyer	Performative: Find Constraint: PR $\leq$ 145 $\wedge$ AC $\leq$3
	Seller	Performative: Relax
Round 4	Buyer	Performative: Find Constraint: PR $\leq$ 175 $\wedge$ AC $\leq$3
	Seller	Performative: Check (PR:165,AC:2,CD:18,AP:1)
Round 5	Buyer	Performative: Find Constraint: PR $\leq$ 175 $\wedge$ AC $\leq$3 $\wedge$ CD $\leq$ 13
	Seller	Performative: Relax
Round 6	Buyer	Performative: Find Constraint: PR $\leq$ 190 $\wedge$ AC $\leq$3 $\wedge$ CD $\leq$ 15
	Seller	Performative: Check (PR:180,AC:2,CD:12,AP:1)
Round 7	Buyer	Performative: Find Constraint: PR $\leq$ 190 $\wedge$ AC $\leq$3 $\wedge$ CD $\leq$ 15 $\wedge$ AP $\geq$ 2
	Seller	Performative: Check (PR:180,AC:2,CD:12,AP:3)
Round 8	Buyer	Performative: Deal

in Figure 13 with the most preferred combination of attributes' values according to his trade-off strategy. According to his behavior strategy the agent tries to minimize the revelation of private information, thus revealing only one constraint per round to the seller, i.e. the agent requests a deal for a price $\leq$ €160,-. The seller then queries his possible deals to find a suitable offer. The seller sends it's most preferred bid to the buyer and asks him to evaluate it. In round 2 the buyer finds that the offer is not acceptable because some constraints are violated, e.g. for the offered price a better actuality of data and shorter contract duration is expected. In consequence the buyer asks the seller to find another offer satisfying the price and another published constraint, i.e. the actuality should equal or below 3 hours. The seller agent has no fitting offer and requests a relaxation of the constraints. In doing so the buyer lowers his expected satisfaction degree he will obtain in this negotiation. Finally, after 2 more unacceptable offers from the seller in rounds 4 and 6, a deal is reached in round 8.

5 Conclusion

In this article we presented a meta model, which enables human negotiators to specify trade-off strategies. As our meta model is based on the Ecore model, we were able to define code generators that transform trade-off models into a representation that can be used by software agents. With this approach it becomes possible that, e.g., a procurement manager can specify their trade-off strategies, and software agents can negotiate on their behalf. Following the MDD principle we can avoid the expensive and possibly erroneous process of encoding the negotiation strategies by hand. We have demonstrated the feasibility of our approach in an prototype capable to perform simple negotiations as shown in the previous section.

The vision of our research is to allow a human negotiator to specify their entire negotiation strategy in a form that can be transformed automatically into reasoning knowledge of an agent. Therefore the principal of the agent is not required to have any knowledge about software agents or programming. In the future we want to realize further steps towards this vision. We will extend our tooling to cover more aspects of negotiations. As our trade-off specification meta model includes a graphical notation, we are going to develop a visual editor for the specification of negotiation strategies, to making it more convenient for humans. Moreover, we want to automate more phases of the specification of software agent negotiations using MDD principles. So further steps can be the specification and automated transformation of negotiation protocols, including the embedding of the strategy specific behavior within the executable model of the protocol.

Another important non-technical aspect that needs to be covered, is to investigate under which conditions humans could be willing to completely automate negotiations, or are willing to accept propositions made by an automated negotiation system. Thus, after a sufficient tooling has been created it is necessary to study the acceptance of such a technology. This is also necessary to adapt the methodology and tooling towards a) scenarios in which a automation is accepted by the users, and b) towards the needs of the human negotiators that are willing to be supported by negotiating agents.

References

1. Luo, X., Jennings, N.R., Shadbolt, N.: Acquiring user tradeoff strategies and preferences for negotiating agents: A default-then-adjust method. Int. J. Hum.-Comput. Stud. 64(4), 304–321 (2006)
2. Fisher, R., Ury, W.: Getting to yes: Negotiating agreement without giving in, 2nd edn. Mifflin, Boston (1991)
3. Steele, P.T., Beasor, T.: Business negotiation: A practical workbook. Gower, Aldershot (1999)
4. Luo, X., Jennings, N.R., Shadbolt, N., Leung, H., Lee, J.: A fuzzy constraint based model for bilateral multi-issue negotiations in semi-competitive environments. Artificial Intelligence Journal 148(1-2), 53–102 (2003)
5. Rosenschein, J.S., Zlotkin, G.: Rules of Encounter: Designing Conventions for Automated Negotiation among Computers, 2nd edn. MIT Press, Cambridge (1998)
6. Sandholm, T.: Distributed rational decision making. In: Weiss, G. (ed.) Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, pp. 201–258. MIT Press (1999)
7. Chiu, D.K.W., Cheung, S.C., Hung, P.C.K.: Facilitating e-Negotiation Processes with Semantic Web Technologies. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS 2005, p. 36a. IEEE Computer Society (2005) ISSN=1530-1605, doi:10.1109/HICSS.2005.269
8. Benyoucef, M., Rinderle, S.: A model-driven approach for the rapid development of e-negotiation systems. In: EMISA, pp. 80–93 (2005)
9. Wikberg, P.: Eliciting Knowledge from Experts in Modeling of Complex Systems: Managing Variation and Interactions. PhD thesis, Linköping University, Department of Computer and Information Science (2007)
10. Pavón, J., Gómez-Sanz, J.: Agent oriented software engineering with INGENIAS. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, pp. 394–403. Springer, Heidelberg (2003)
11. Hahn, C., Fischer, K.: A platform-independent metamodel for multiagent systems. International Journal on Autonomous Agents and Multi-Agent Systems (JAAMAS) 18(2), 239–266 (2009)
12. Hahn, C., Zinnikus, I., Warwas, S., Fischer, K.: Automatic generation of executable behavior: A protocol-driven approach. In: Gleizes, M.P., Gómez-Sanz, J.J. (eds.) AOSE 2009. LNCS, vol. 6038, pp. 110–124. Springer, Heidelberg (2011)
13. Russell, S.J., Norvig, P.: Artificial intelligence: A modern approach. [the intelligent agent book], 2nd edn. Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River (2003)
14. Kurtanovic, Z.: Spezifikation von Verhandlungsstrategien. Diploma thesis, Institute for Computer Science, Goethe University Frankfurt a.M. (2008)
15. Budinsky, F.: Eclipse modeling framework: A developer's guide. The eclipse series. Addison-Wesley, Boston (2003)
16. Apt, K.: Principles of Constraint Programming. Cambridge University Press, New York (2003)
17. Osborne, M.J., Rubinstein, A.: Bargaining and markets. Economic theory, econometrics, and mathematical economics. Acad. Press, San Diego (1990)