Knowledge-based e-Contract Negotiation among Agents Using Semantic Web Technologies

Kalliopi Kravari, Christos Papavasileiou, and Nick Bassiliades

Dep. of Informatics, Aristotle University of Thessaloniki, GR-54124, Thessaloniki, Greece

{kkravari, cpapavas, nbassili} AT csd.auth.gr

Abstract: E-Commerce enabled new ways of transactions. Companies and individuals negotiate and make contracts every day. Practically, contracts are agreements between parties that must be kept. These agreements affect the involved parties irretrievably. Hence, negotiating them efficiently is proved vital. To this end we propose the use of intelligent agents, which benefit from Semantic Web technologies, such as RDF and RuleML, for data and policy exchanges. Each agent encounter is characterized by the interaction or negotiation protocol and each party's strategy. This study defines a knowledge-based negotiation procedure where protocols and strategies are separated enabling reusability and thus enabling agent participation in interaction processes without the need of reprogramming. In addition, we present the integration of this methodology into a multi-agent knowledge-based framework and next a use case scenario using the contract net protocol that demonstrates the added value of the approach.

Keywords: Semantic Web, Agents, e-Contract negotiation, Reaction RuleML.

1 Introduction

The massive growth of e-Commerce [13] is indisputable, being clear that it will continue to grow. However, e-Commerce is rather complicated, since the parties involved have to collect information, negotiate and safely execute transactions. Although companies and individuals interact every day, they face difficulties in reaching agreements; namely contracts that create relations and obligations that must be kept. Negotiating efficiently such a contract is important since the decisions made during negotiation affects the parties irretrievably. To this end we propose the use of Intelligent Agents (IAs) [8]. IAs benefit from Semantic Web (SW) technologies [3], performing unsupervised complex actions on behalf of their users, reflecting their specific needs and preferences. While SW full vision may be a bit distant, there are already capabilities that can make software more interoperable and cheaper to maintain. For instance, the use of SW technologies, such as RDF and RuleML, for data and policy exchanges maximizes interoperability among parties. Hence, as IAs are gradually enriched with SW technologies their use is increasing.

Each agent is able to manage a private policy (or strategy), a set of rules representing requirements, obligations and restrictions, and personal data that meet its user's

adfa, p. 1, 2011.

interests. Sophisticated tasks, such as negotiation and brokering services, are already carried out efficiently by IAs. On the other hand, each transaction among parties is strictly specified by an interaction protocol, a set of rules that specify among others guidelines and restrictions on the parties involved. Hence, using IAs could be the answer for a flexible but efficient (contract) agreement procedure management. However, in order to reach SW agents' maximum efficiency, a well–formed modeling framework is needed. It should be re-usable, easily comprehensible by the user (promoting agent usage) and easily analyzable by the agent (promoting automatization).

Usually both private negotiation strategies and public interaction protocols are jointly hard-coded in each agent. Undoubtedly, it is a common and convenient practice, however it is inflexible. Yet, separating policies (personal strategies) from protocols is imperative. Each agent's policy is private and any disclosure of it could lead to incalculable loss. On the other hand, each protocol should be a common resource since agents must comply with the same interaction protocol in order to interact. Hence, this study attempts to define the necessary requirements and procedures that will let agents interact without the need of reprogramming. The proposed knowledge-based approach enables agents to choose the appropriate protocol (e.g. library of re-usable protocols) and combine it with their personal strategy by using SW technology.

Hence, this article proposes the use of SW languages for expressing both protocol and strategy, in addition to separating them. Separating strategy from the protocol and automatically combining them will let agents to modify their behavior while remaining compliant to the protocol with no extra programming cost. Hence, a fully automated procedure for agent transactions (here in contracts) is presented. Nevertheless, an appropriate framework providing enough compliance with the proposed SW technologies should be used, thus, an integration of the above methodology into EMERALD [11], a knowledge-based MAS, is presented. The rest of the paper is structured as follows: Section 2 gives an overview of the approach, Section 3 briefly overviews EMERALD, while Section 4 illustrates a Contract Net use case, which better displays the potential of the approach. The paper is concluded with references to related work, conclusions and directions for future improvements.

2 Overview

E-Contracts are the most common procedures in everyday life. The main differentiation is that they are modeled, specified and executed by a software system, overcoming the delays and drawbacks of the manual process. Hence, here, we study contract protocols and more specifically the FIPA Contract Net Interaction Protocol.

2.1 FIPA Contract Net Interaction Protocol

Although, contract interaction protocols are acknowledged as vital, their modeling is a really challenging task. Yet, several domain-depended interaction protocols have already been developed. The Contract Net Protocol (CNET), for instance, is a noteworthy and probably the most widely used protocol, firstly introduced by Smith [15].

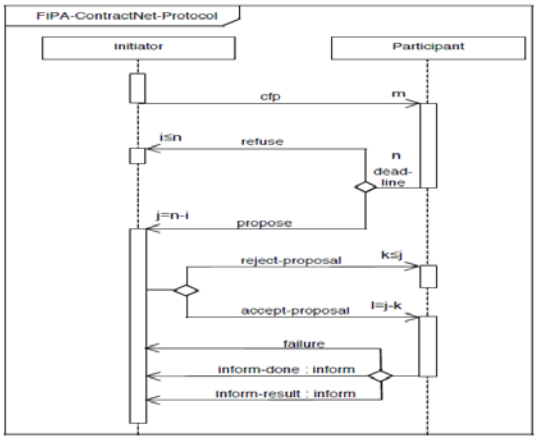

Fig. 1. FIPA Contract Net Interaction Protocol

In CNET negotiation is considered as a two-way communication in which an agent evaluates the offer of assigning a contract or receiving one from its own perspective. Since then, much research has been done, successfully resolving important issues in the field. Although CNET was proved valuable in a variety of situations, it had to be modified in order to reflect changes in agent technology. In this direction, FIPA, an IEEE Computer Society standards organization, provides among others the FIPA Contract Net Interaction Protocol [16]. This standardized protocol has been used over time as the basis for a variety of cases. According to the FIPA specification (Fig. 1) in the contract net interaction protocol, one agent (the Initiator) takes the role of manager which wishes to have some task performed by one or more other agents (the Participants) and further wishes to optimize a function that characterizes the task. This characteristic could be for instance the minimum price. For a given task, the Initiator has to send a call for proposal message communicating its request. Next, any number of the Participants may respond positively; the rest must refuse. Negotiations then continue with the Participants that accepted the call. A positive response however is not a strict acceptance but rather a counter proposal. Hence, the Initiator has to evaluate the offers and ignore the refusals. Finally, it has to accept the best offer by sending back an acceptance messages whereas reject messages should be send to the rest.

2.2 Separating Interaction Protocol from Agent Strategy

However, having an appropriate protocol is not enough, flexibility and reusability is also needed, which actually can be obtained by separating each agent's private policy from the protocol. To this end, in the proposed approach two main rulesets were defined. The first one is related to the Strategy (agent's personal policy), while the second one is related to the Protocol. The Strategy ruleset defines the agent's personal preferences whereas the Protocol ruleset mainly orchestrates message exchange among agents. It is obvious that the message exchange, defined by the protocol, is the key for a successful transaction. Hence, providing appropriate message structures is

vital. To this end, Reaction RuleML was chosen for expressing both the protocol and the strategy rules [4], [7]. This rule language was chosen for two major reasons. Firstly, it is flexible in rule representation and secondly its syntax supports a message structure that can include all message modules provided by the FIPA specifications. The message structures in RuleML (Fig. 2) and FIPA are similar since they both contain predicates for *Sender*, *Receiver*, *Content (Payload)*, *Protocol* and C*onversation Identifier* (called *conversation-id* in FIPA and *oid* in RuleML).

```
<Message mode="outbound" directive="CFP”
  <oid> <!-- conversation ID--> </oid>
  <protocol> <!-- transport protocol --> </protocol>
  …
</Message>
```

Fig. 2. (Call-For-Proposal) Message structure in Reaction-RuleML syntax

Using appropriately the above message structures, agents are able to exchange from simple facts to rulebases (sets of rules). Even more useful is the fact that due to the conversation-id module, agents will be also able to get involved in longwinded and usually asynchronous communications, and thus being flexible. In this context, Reaction RuleML is expedient since it provides specific predicates (`SendMsg` and `rcvMsg`) which are appropriate for message exchange in any agent interaction. However, having both protocol and strategy rules expressed in Reaction RuleML is not enough. Agents' final behavior, the combination of protocol and strategy, should be executed in a compact way that will let them represent both their environment knowledge and their behavior patterns quite easily. To this end, the JESS execution engine and language was chosen [9]. JESS is considered as a very expressive language that can express complex logical relationships with very little code, while it is commonly used by agent programmers. Additionally, it is also FIPA compliant.

2.3 Combining Protocol with Strategy

Practically, there is a public repository where protocols are stored with their rulesets expressed in Reaction RuleML. Additionally, there are private repositories, one or more, for each agent where their personal strategy rulesets are stored, also expressed in Reaction RuleML. For instance, assume that there is a number of agents (e.g. four) interacting according to the FIPA Contract Net Interaction Protocol. One of them should be the initiator of the process and the rest (participants) will respond according to their personal intentions. Each of them will retrieve the protocol ruleset from the appropriate repository (Fig. 3), which is common for all of them, whereas they have their private repository for their strategies. In other words, the protocol will define how they will communicate whereas their strategy will define what to share. What is important here is the fission of the protocol rules to initiator and participant rules. This is essential because although there is a single protocol, it defines rules for both roles (initiator and participant) since each of them have to act from another perspective. How these sets of rules should be combined and executed is an important issue.

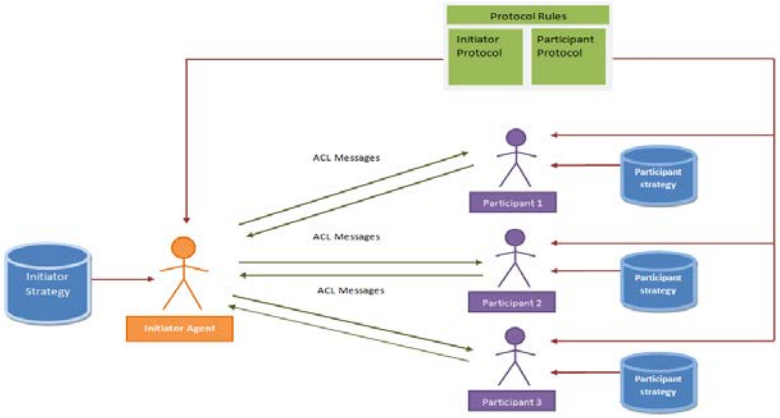

Fig. 3. Procedure overview

Fig. 4 presents the process; domain depended XSLT transformations will use both RuleML sets of rules (for protocol and strategy) and transform them to an executable ruleset (e.g. in Jess) which is fused inside each agent in order to participate in the transaction effectively. Notice that using a high-level, declarative description of the protocol and strategy no extra programming cost is needed by the agent owners.

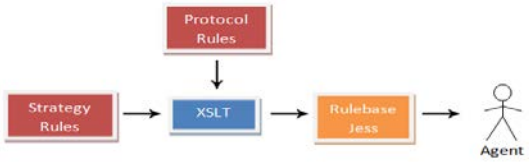

Fig. 4. Strategy – Protocol Combination

3 EMERALD

EMERALD (Fig. 5) is a multi-agent knowledge-based framework, based on SW and FIPA standards, that enables reusability and interoperability of behavior between agents. It is built on JADE [2], a reliable and widely used framework. EMERALD supported so far the implementation of various applications, like brokering and agent negotiations. This framework, in order to model and manage the parties involved in an e-Contract negotiation procedure, provides a generic, reusable agent prototype for knowledge-customizable agents (KC-Agents), consisted of an agent model (KC Model), a directory service (Advanced Yellow Pages Service) and several external Java methods (Basic Java Library). Agents that comply with this prototype are equipped with a Jess rule engine [9] and a knowledge base (KB) that contains environment knowledge (facts), behavior patterns and strategies (Jess production rules). Additionally, since trust has been recognized as a key issue in SW MASs, EMERALD adopts a variety of reputation mechanisms, both decentralized and centralized. In this study, the Jess KB is actually the agent's data and rules that comprise its policy and charac-

terize its behavior, as described above. Using the KC-Agents prototype offers certain advantages, such as modularity, reusability, maintainability and interoperability, as opposed to having behavior hard-wired into the agent's code (e.g. in Java).

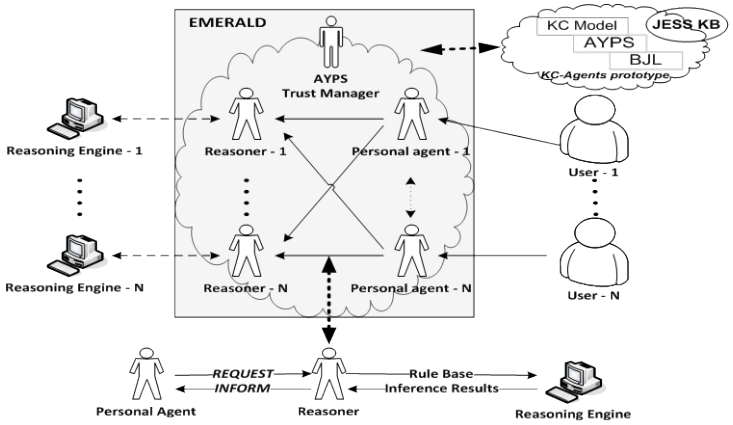

Fig. 5. EMERALD abstract architecture

Hence, since agents do not necessarily share a common rule or logic formalism it is vital for them to find a way to exchange their position arguments seamlessly. To this end, EMERALD proposes the use of Reasoners [12], which are agents that offer reasoning services to the rest of the agent community. This approach does not rely on translation between rule formalisms, but on exchanging the results of the reasoning process of the rule base over the input data. Currently, EMERALD implements a number of Reasoners that offer services in two major reasoning paradigms: deductive rules and defeasible logic. Following the above specifications EMERALD commits to SW and FIPA standards, namely, it uses among others the RuleML language [4] since it has become a de facto standard. Additionally, it uses the RDF model [14] for data representation both for the agents' private data and the reasoning results generated during the process, as used in contract agreement interactions presented in [10].

3.1 Extending KC-Agents prototype

EMERALD's KC-Agents prototype was extended in order to adapt to the new requirements; namely the initial separation of protocol and strategy definitions and the final combination in one Jess rulebase. So far, KC-Agents were limited in receiving one file containing both strategy and protocol; hence it was the programmer's responsibility to merge them. Each time a new behavior was needed, a new file containing both (new) strategy and protocol should be provided. Yet, a new agent model was added in the prototype based on this study. The extended KC-Agents agent derives two separated files one for the protocol and one for the strategy. Hence, whenever protocol or strategy is modified, no extra programming cost is needed. The agent will retrieve the appropriate (new) files from the corresponding repositories. The transformation and merge of them will be executed automatically. Following this approach

new behaviors and protocols can be added to the private and public repositories, respectively, for future use. Agents will automatically use them when needed. In this context, the new agent model contains a function that retrieves the appropriate XSLT (from another repository) and uses it in the protocol-strategy fusion process (Fig. 5).

4 Use Case

A use case scenario based on the FIPA Contract Net Interaction Protocol is presented here in order to clarify why protocols and strategies should be separated and how an automated combination procedure will save time and programming effort. It is implemented in EMERALD and involves four parties; an initiator and three participants (Fig. 3) who comply with the new KC-Agents; hence protocol and strategy will combine automatically according to the XSLT transformation procedure. The Initiator agent is interested in the best offer for a laptop. It is aware of three potential e-shops, which are represented by the participant agents. First of all, it has to get the appropriate protocol in RuleML from the public repository. Hence, the Initiator following the protocol sends a CFP (Call-for-proposal) message containing the name of the product and a desired price to them in order to initiate the negotiation procedure. Next, it waits for their response, which could be either positive (PROPOSE) or negative (REFUSE). A positive response however is not a strict acceptance but rather a price proposal. Hence, the Initiator has to evaluate the offers and ignore the refusals. Next, it has to accept one by sending back an ACCEPT message whereas REJECT messages should be sent to the rest. The above rules are defined by the protocol since they refer to message exchange. The decision making however is defined in the private strategies.

The Initiator's strategy for instance determines the agent's main restriction; the price offered by a participant should be lower than the price the Initiator is willing to pay. That price for the Initiator is the firstly indicated price in the CFP (here 300 Euros) plus an amount (here 50 Euros). Hence, if none of the participants fulfill this restriction, the Initiator will reject all of them. On the other hand, each participant has a list (set of facts called Products) that contains their available products accompanied with the product type (e.g. laptop) and its price (e.g. 500 Euros). For instance, such a product is described in JESS syntax as: *(products (type laptop)(price 200))*. A comprehensible flow of rule activation, triggering and execution for this use case is presented in Fig. 7. The same principles can also be applied in other interaction scenarios. Using EMERALD and its KC-Agents prototype we activated the four agents defining two files, one for the protocol and one for the strategy. Following the CNET protocol a straight-forward procedure is performed from the first call to the final acceptance. Here, the transaction was successfully completed when the Initiator chose the participant called *p* by sending an *Accept Proposal* message, as presented in the EMERALD's execution diagram (Fig. 7). Furthermore, participant *p* sent back an *inform* message which according to FIPA specifications is needed in order to verify that the final decision is received.

Finally, it is worth mentioning that Reaction RuleML enables two types of rules, namely production and reactive. Production rules are used for agents' private strategy

allowing them to act according to their user's will whereas reactive rules are used for the protocol (Fig. 6) allowing agents to adjust to their partner's behavior based on events related to message exchanges. Below is presented shortly a reactive (protocol) rule example in JESS syntax due to space limitations, where a participant when receives a CFP message, it posts it as a fact in the agent's internal KB. Notice how the rules interact through a predefined set of fact templates that play the role of API (here *callforp*). All RuleML files used are available at http://tinyurl.com/usecase-ruleml.

```
(defrule receive-cfp
 ACLMessage(communicative-act CFP)…(protocol fipa-contract-net))
 =>
(assert (callforp (cfp_content ?c)……(cfp_cid ?cid)))
(modify ?p_start_state (state p_check_state) (state_id ?cid)))
```

Fig. 6. Reactive Rule (Protocol): Participant receives a call

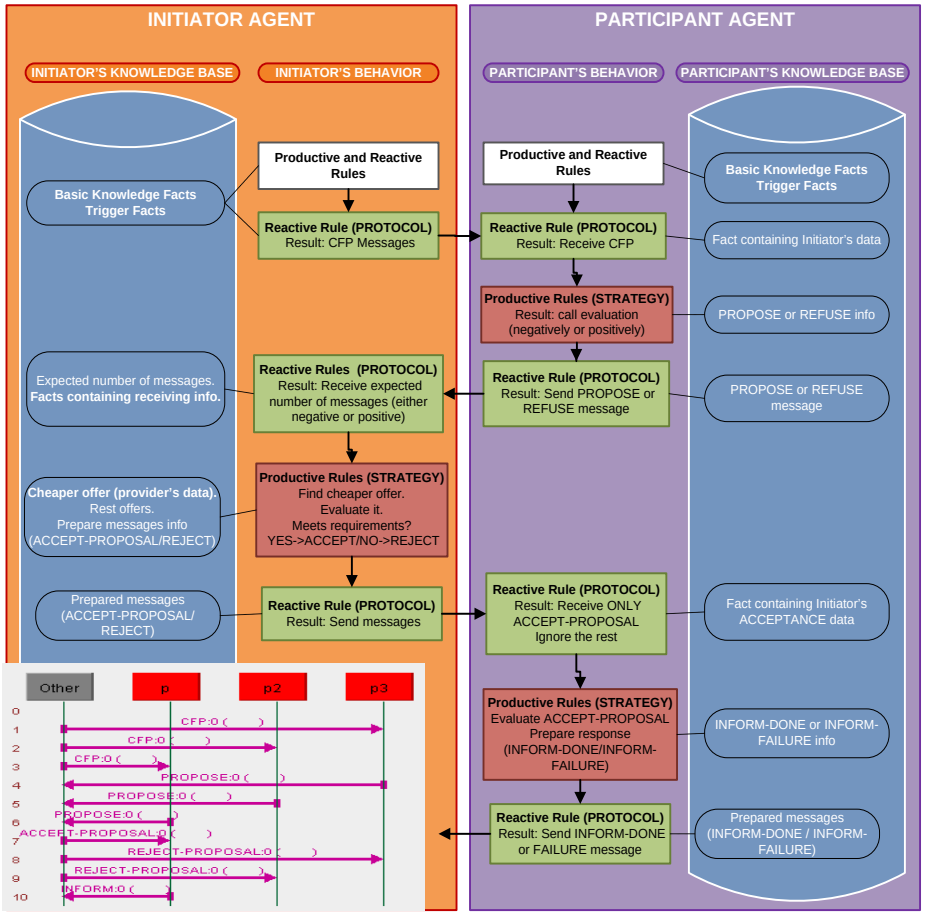

Fig. 7. Scenario overview and message exchange in EMERALD

5 Related Work

A work that automates price negotiations in e-commerce transactions using a rule-based implementation is presented in [1]. It is based on JESS and the JADE framework. It concerns only price negotiation rather than a more generic protocol. Our work on the other hand concerns modeling a reusable procedure that could be used not only in this case but also in any other protocol case. Additionally, although both approaches consider IAs and FIPA standards important for maximizing automation and efficiency in users' everyday life only our approach complies with SW standards.

Another related approach is the DR-CONTRACT [6] architecture for representing and reasoning on e-Contracts in defeasible logic. The architecture captures the notions relevant to execution and performance of e-Contracts in defeasible logics. It uses a RuleML extension and RDF/XML syntax for its exported conclusions. Hence, it uses SW standards like RDF and RuleML, similarly to our approach, whereas it is an architecture focused on e-Contract procedures omitting separation of protocols and strategies, which is an important and challenging task for the field.

Concerning interoperability, Rule Responder [5] is quite similar to EMERALD. It builds a service-oriented methodology and a rule-based middleware for interchanging rules in virtual organizations. It demonstrates the interoperation of distributed platform-specific rule execution environments, with Reaction RuleML as a platform-independent rule interchange format. It has a similar view of reasoning service for agents and usage of RuleML but it is not based on FIPA specifications. In other words, it is interested in interoperability, reusability and even protocol-strategy separation, yet it doesn't support IAs but rather web services acting like agents.

6 Conclusions and Future Work

The article argued that e-Commerce met a massive growth over the past years; with e-Contracts like ordinary contracts to be the most common procedures in everyday life. However, they are rather complicated, since involved parties have to collect information, negotiate and execute transactions. We addressed this problem by using intelligent agents acting in the Semantic Web, as agents can perform the same tasks unsupervised, relieving their users of time consuming processes. To this end, this article presented a modular and reusable framework using SW technologies, such as RuleML and RDF. We defined a knowledge-based negotiation procedure where protocols and strategies are separated enabling reusability and thus enabling agent participation in interaction processes without the need of reprogramming. In addition, an integration of this methodology into a multi-agent knowledge-based framework and a use case scenario that demonstrated the added value of the approach was also presented.

As for future direction, our main interest is in extending the proposed framework to model more protocols, such as brokering, price negotiations and auctions. Our final goal is to provide a general-purpose framework for protocol-strategy separation in multi-agent environments, letting agents maximize their autonomy, flexibility and efficiency. Additionally, since agents not necessarily share the same logic or rule

representation formalism, our intention is to provide a mechanism for automatic formalism transformation or interpretation.

Acknowledgments

This work is partially supported by the Greek R&D General Secretariat through a bilateral Greek-Romanian project.

Rererences

1. Badica C., Ganzha M., Paprzycki M.L: Implementing Rule-Based Automated Price Negotiation in an Agent System. J. of Universal Computer Science, 13(2), pp. 244-266 (2007)
2. Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: JADE: A white Paper. EXP in search of innovation, 3(3), 6-19 (2003)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American Magazine,284(5), 34-43 (2001) (Revised 2008)
4. Boley, H., Paschke, A., Shafiq, O.: RuleML 1.0: The Overarching Specification of Web Rules. 4th International Web Rule Symposium: Research Based and Industry Focused (RuleML'10), Springer vol .6403, pp 162-178 (2010)
5. Boley, H., Paschke, A.: Rule responder agents framework and instantiations. Semantic Agent Systems, Studies in Computational Intelligence, Vol. 344, pp. 3–23 (2011)
6. Governatori G., Hoang D. P..: A Semantic Web Based Architecture for e-Contracts in Defeasible Logic. In Int. Conf. on Rules and Rule Markup Languages for the Semantic Web (RuleML-2005), Springer-Verlag, LNCS 3791, pp. 145 - 159, Galway, Ireland (2005)
7. Governatori, G., Rotolo, A.: Modelling contracts using RuleML. In Legal Knowledge and Information Systems. Jurix 2004, pp. 141–150. IOS Press, Amsterdam (2004)
8. Hendler, J.: Agents and the Semantic Web. IEEE Intelligent Systems, 16(2), 30-37 (2001)
9. JESS, the Rule Engine for the Java Platform (2008) Available at http://www.jessrules.com/.
10. Kravari, K., Kastori, G.-E., Bassiliades, N., Governatori, G.: Contract Agreement Policy-Based Workflow Methodology for Agents Interacting in the Semantic Web. Semantic Web Rules, Proc. 4th Int. Web Rule Symposium (RuleML 2010), LNCS, Vol. 6403, 225 – 239. Springer, Berlin/Heidelberg (2010)
11. Kravari, K., Kontopoulos, E., Bassiliades, N.: EMERALD: A multi-agent system for knowledge-based reasoning interoperability in the semantic web. In Artificial Intelligence: Theories, Models and Applications, 6th Hellenic Conference on Artificial Intelligence, SETN 2010: LNCS, Vol. 6040/2010, 173-182. Springer, Berlin/Heidelberg (2010)
12. Kravari, K., Kontopoulos, E., Bassiliades, N.: Trusted Reasoning Services for Semantic Web Agents. Informatica: Int. J. of Computing and Informatics, 34(4), 429-440 (2010)
13. Laudon, K., Traver, C. G.: E-Commerce 2012 (8th ed.). Prentice Hall, New Jersey (2012)
14. Resource Description Framework (RDF): Model and Syntax Specification (2004). Available at http://www.w3.org/TR/PR-rdf-syntax/.
15. Smith, R. G.: The contract net protocol: high level communication and control in a distributed problem solver. IEEE Transactions on Computer, 29(12), 1104-1113 (1980)
16. FIPA Communicative Act Library Specification: Fipa Contract Net Interaction Protocol Specification, version H (2003). Available at http://www.fipa.org/specs/.