Framework for Performance Evaluation of Service Negotiations in Agent Systems*

Mihnea Scafeş and Costin Bădică

University of Craiova, Software Engineering Department
Bvd. Decebal 107, Craiova, RO-200440, Romania

Abstract. We propose a framework and method for evaluation of performance of multi-issue, collaborative service negotiations in multi-agent systems. The framework uses the following evaluation criteria: computation complexity, communication complexity, and scalability. Our proposal is applied to analyze a version of Contract Net protocol that we use in an agent system for disaster management. In particular we discuss how agents compute the additional effort they must deploy to meet conditions of negotiation issues. Two negotiation issues specific to disaster management problems are given as example: *location* and *time*.

1 Introduction

Currently, complexity of real-world problems demands for special support for collaborative problem solving in distributed environments. Each node of a distributed system runs a set of processes that gather and generate information, either from other nodes or from the environment. This information can be integrated and combined to create more complete views of the environment, as well as to improve the problem solving process.

Multi-agent systems are a special class of intelligent distributed systems that combine intelligence with distribution of computation to improve problem solving processes. Each node of the system is represented by an agent with local domain knowledge and a set of processing abilities. Multi-agent systems have been successfully applied for solving problems involving distributed reasoning, decentralization and collaboration. An example is a collaboration system for helping human experts and population to fight against disaster situations, like those addressed by DIADEM project[1] (more specifically, DIADEM project targets potential disasters caused by chemical incidents in industrial and urban areas). Agents in DIADEM support human experts as well as other stakeholders that participate in the process of responding to a chemical incident.

DIADEM is based on Dynamic Process Integration Framework – DPIF [1] multi-agent platform. DPIF agents run reasoning processes to decide for the best course of action that fits their own objectives. Agents can be fully-autonomous or human-controlled

*Mihnea Scafeş was partly supported by IOSUD-AMPOSDRU contract 109/25.09.2008 and partly supported by DIADEM project. Costin Bădică was supported by DIADEM project. Diadem project is funded by European Union under Information and Communication Technologies (ICT) theme of the 7th Framework Programme for R&D, ref. no: 224318.

[1]DIADEM project – Distributed information acquisition and decision-making for environmental management: http://www.ist-diadem.eu/

ScafeÅ§ M. and BÄČdicÄČ C.
Framework for Performance Evaluation of Service Negotiations in Agent Systems.
DOI: 10.5220/0003024700190029
In *Proceedings of the 8th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems (ICEIS 2010)*, page
ISBN: 978-989-8425-12-6

(i.e. a sort of partially-autonomous). This means that humans can intervene into or completely replace the agent's reasoning processes. Reasoning processes of the agents are made available into the system as services. Services have input and output parameters. Input parameters consist of the necessary information for the agents to provide their services (e.g. the location at which an agent should measure gas concentrations). Output parameters are results of service execution and can be input parameters of other services. Therefore, each service may depend on other services in the system.

During an incident response, agents get connected through services and create complex workflows [1]. They execute local processes, request information from other agents (request services from provider agents) and supply information to other agents (provide services to requester agents). There might be multiple agents that are simultaneously able to provide the same service, but usually some agents promise better results than others. Alternatively, agents could be too busy and thus not able to meet the conditions of requested service. We propose the use of service negotiation for dynamic selection of optimal service providers. Whenever an agent requests a service, it negotiates with other agents about the provision of the required service. On the other hand negotiation brings a computational overhead into distributed reasoning processes. Therefore we must assure that this overhead is not a performance hit to the system, as it must handle complex situations for which time and human effort resources are important.

Based on our literature review, we observed that there are quite few works addressing evaluation of performance impact of negotiations in multi-agent systems ([2–5]). So we propose a framework and method for evaluation of negotiations in the context of the DIADEM project. We first define a set of evaluation criteria (computation complexity, communication complexity and scalability) and then analyze a version of negotiation that we use in our system with respect to these criteria. For computation complexity we analyze the complexity of the decision steps of the negotiation (creating task announcements, making proposals and task awarding). While analyzing the phase of making proposals, we discuss how agents compute the additional effort they will have to deploy by taking into account the negotiation issues (two issues are given as an example: *location* and *time*). For communication complexity we analyze the communication overhead (the number of messages). For scalability we study if the system can effectively support a large number of negotiations about *location* and *time*. The results show that service negotiation has an affordable computation complexity, while communication complexity can be quite large.

2 Negotiation Framework

This section briefly introduces our negotiation framework [6]. The framework is generic and addresses negotiation protocols, negotiation subject and decision components (how agents make proposals and how they select the best proposals).

The protocol supports one-to-many negotiations and therefore it defines two agent roles: *manager* and *contractor*. The *manager* is the agent that requests a service and thus initiates the negotiation. The *contractor* is the agent that is able to provide the service requested by the manager. A set of generic negotiation steps have been defined: (i) the initial step consisting of subject identification and negotiation announcement

(initiation of negotiations), (ii) the process of making proposals and counter-proposals, (iii) deciding whether an agreement or a conflict has been reached, and (iv) termination.

Agents negotiate about terms and conditions of service provision. A service takes a set of input and output parameters. The input parameters represent necessary information for the agents to provide their services. Output parameters represent new information produced by services.

Negotiation subject comprises the service description and a subset of the service parameters that are important decision factors during negotiation (i.e. they are taken into consideration when selecting the appropriate service providers). During negotiation, these parameters are considered negotiation issues. Thus, when the negotiation designer configures the service, he also defines the negotiable parameters of the service (negotiation issues). Issues are described by properties like: *name*, *data type*, *weight*.

Note that in our current model negotiation issues are considered independent. This means that total utility of a negotiation proposal is the weighted sum of partial utilities of each independent negotiation issue [6].

The default negotiation in DPIF is an implementation of Contract Net (CNET in what follows [7]). A manager agent starts a negotiation for a service by sending a call for proposals (task announcement) to contractor agents. Contractor agents then decide whether to engage in negotiation or not. If their decision regarding engagement in negotiation is positive, they must set the conditions under which they will be able to provide the required service (compute proposals). Contractors evaluate possible proposals using a utility function that takes into consideration the effort needed to provide the service under the conditions in the proposals. After all the proposals have been received or after a given deadline has passed, the manager selects the best proposal received (i.e. the proposal that returns him the highest utility) and informs the winner agent. The manager uses a weighted sum utility function to evaluate proposals. A connection is created between the manager agent and the contractor that is awarded the service. At this point negotiation ends. Service provisioning will be done at a later step, following the connection created between manager and the winner contractor.

Although the default negotiation protocol is CNET, the system allows the use of arbitrary negotiation schemes, supporting domain specific one-to-many negotiation approaches where protocol, negotiation subject and strategies are specified during the configuration phase. Our model supports two levels of configuring negotiations: (i) *negotiation type* level where designer defines the negotiation protocol, identifies the negotiation issues and sets default values for their properties; (ii) *negotiation instance* level where a human expert can tune the issue properties for a certain situation.

3 Evaluation Criteria

We introduce the following performance measures for evaluation of DIADEM service negotiation mechanisms [8]: (i) *Computation complexity*. Represents the amount of time that execution of computations required for service negotiations add to the total system execution time. Agents must make certain decisions during negotiations (e.g. compute offers/counter-offers, decide if to submit a bid or not, a.o.) [9]. These decisions consume time that can be estimated using measures of computation complexity, derived

using asymptotic analysis of the computation time; (ii) *Communication complexity*. Represents the amount of communication activities incurred during service negotiation processes. Negotiation assumes interaction between participants. In multi-agent systems, interaction between agents is done using messages that transport information or commands. A large number of messages represents a performance hit to the system; (iii) *Scalability*. This performance measure tells us if the system can effectively support a large number of negotiations. It would be difficult to discuss scalability without taking into consideration the other performance measures. Therefore we will analyze the scalability after we analyze computation and communication complexities. Because computation, communication and scalability are typical problems of multi-agent systems that use negotiation for dynamic service discovery and provisioning, we consider that our type of analysis can be also applied to similar systems.

Note that our problem deals with service negotiation between cooperative agents. Negotiations for certain service provision are carried out only with agents that are able to provide the required service (i.e. that posses the domain knowledge or physical capabilities – in the case of humans – that are needed to provide the service). Provider agents will usually accept to provide their services if they are currently able to do so (i.e., for example, they have all the necessary resources), without asking in compensation for payments from requester agents. In our model payments are not necessary and consequently are not taken into consideration. Additionally, as agents are only able to provide certain types of services, tasks cannot be exchanged between agents as it happens in task-oriented domains [10]. Role of negotiation is to select the optimal provider of requested service in a specific context without considering the problem of distributing tasks evenly between available agents. Under these assumptions, criteria like *load balancing*, *fairness* and *utilization of resources* are out the scope of our discussion.

Note also that we evaluate only the performance of negotiations, rather than performance of the whole system.

4 Sample Evaluation

The protocol that we use for service negotiation is a variant of Contract Net (CNET) Protocol [7]. We motivate our choice by the fact that CNET has been widely used for distributed problem solving [7, 10]. Other suitable negotiation protocols were introduced in [11, 12]. There is a key difference between our version and other versions of CNET that is driven by the nature of our problem: it does not involve payments.

Our CNET version follows FIPA specifications (see sequence diagram from [13]) up to the point when interaction for service provision starts (see Section 2). In our model, service provision is handled separately in the system framework and it is not considered part of negotiation. Negotiation is used only to create connections between agents [1]. In other words, messages with information about the progress of the task are not sent.

Before proceeding, we first introduce some notation. Let A be the set of agents in the system. Let A_j be the set of agents involved in negotiation j. We denote the task that is currently negotiated by T. Let us assume that T_c is the set of currently active tasks of the contractor and T_{pos} the set of tasks that are currently being negotiated (except for

T) and that might be awarded to the contractor if T is also awarded. The subset of T_{pos} that will be actually awarded to the contractor is denoted as T_f (future tasks).

4.1 Computation Complexity

There are 3 important decision points in our negotiation process: (i) when the manager creates task announcements; (ii) when a contractor creates proposals; and (iii) when the manager awards a task to the winning contractor.

Service announcements are created whenever an agent (the manager agent) requires a service. This may happen when an agent starts a new workflow instance by invoking a top level service or when an agent starts a service that is required by another service [1]. Top level services are normally started by human agents, i.e. human experts. They choose what service to start based on the context of their work.

Services are created offline (at design time), resulting a set of service descriptions [1] that are stored into a service repository. Human experts initially define the service concept (usually represented by the service name) and the system framework automatically searches the service repository for a suitable service to help a human to configure the proper service description. Task announcements for required services are created by humans in a similar way. The system framework automatically determines the services required by a certain service and creates task announcements. As services are defined offline and stored in a data structure for fast retrieval (e.g. hash table), we can assume that task announcements are created in constant time, thus having constant complexity.

The next decision point of the protocol is part of the contractor role. As soon as the contractor receives a task announcement, he must determine: (i) if he should respond with a proposal; (ii) if he decides to submit a proposal then he must determine suitable values for the parameters of the proposal. The contractor will decide about and compute proposals by taking into consideration the effort that he will have to deploy for providing the service in order to meet the conditions specified in the proposal.

Contractors compute marginal efforts [10] and then decide whether to provide the service or not. Marginal efforts represent the lower and higher boundaries of the effort the contractor will have to deploy in order to provide the task.

The boundaries for additional effort that contractor has to deploy for a task T are:

$$effort^{-}_{add}(T) = \min_{T_f \subset T_{pos}} [effort_{total}(T \cup T_c \cup T_f) - effort_{total}(T_c \cup T_f)] \quad (1)$$

$$effort^{+}_{add}(T) = \max_{T_f \subset T_{pos}} [effort_{total}(T \cup T_c \cup T_f) - effort_{total}(T_c \cup T_f)] \quad (2)$$

Note that these formulas are very similar to cost computation in [10]. The additional effort contractor will have to deploy if he adds task T to its set of tasks represents the difference between the total effort deployed to execute the active tasks, future tasks and task T and the total effort deployed to execute only the active tasks and the future tasks. How exactly this difference is computed depends on the task T and the sets T_c and T_f and it is described in what follows.

As we presented in Section 2, we deal with multi-issue service negotiation. We consider that issues are independent (a change in the value of one issue does not affect the values of the other issues). The issues represent conditions under which the service

is provided and a change in the value of one issue involves a change in the amount of effort an agent will have to deploy in order to provide the service. Therefore it seems logical to split the effort for one task into multiple parts, one for each issue. We must also consider the effort required to compute the task after all the conditions have been met (perform calculations, measurements etc). Thus, the general formula for computing the additional effort for task T is:

$$effort_{add}(T) = \sum_{x_i \in input(T)} effort(x_i) + comp(T) \tag{3}$$

where $input(T)$ is the set of input parameters/negotiation issues of task T and $comp(T)$ is the additional computation effort required to perform the task.

As previously discussed, tasks can be performed manually by human experts or automatically by software agents. Performing a task may involve either a simple or a more complex operation, but analyzing this complexity will depend on the task in hand so it is outside the scope of this paper. Here we analyze only the complexity of the decision processes during negotiations. Therefore, for simplicity we shall consider here that $comp(T)$ has constant complexity ($O(1)$).

The complexity of $effort(x_i)$ depends on the type of negotiation issue x_i. Two issues that are typically being negotiated in our problem are *location* (i.e. geographical location) and *time*. *Location* is important because it might require physical mobility, i.e. agents might have to physically move from one location to another. *Time* is very important in complex situations in order to specify deadlines for achievement of various tasks. So agents will often have to reserve a time period required to perform a service.

Location. Regarding the requirement of performing a task related to a given location, there are two types of agents: (i) agents that have to move to the required location in order to perform the task (e.g. measure gas concentration at a certain location) and (ii) agents that do not have to move, but that still have to perform some reasoning about a certain remote location (e.g. human experts in control centers that must decide whether to evacuate residents of a certain city region). Agents that do not have to move do not need to deploy additional effort for task execution, so in this case $effort(location) = 0$. For agents that do have to move there is an effort associated to agent migration between physical locations. The minimum additional effort is computed as:

$$\begin{aligned} effort^{-}_{add}(loc_T) &= \min_{c \in T_c}(|loc_c - loc_T|) + \min_{f \in T_f,\, T_f \subset T_{pos}}(|loc_T - loc_f|) \\ &= \min_{c \in T_c}(|loc_c - loc_T|) + \min_{f \in T_{pos}}(|loc_T - loc_f|) \end{aligned} \tag{4}$$

while the maximum additional effort is computed as:

$$\begin{aligned} effort^{+}_{add}(loc_T) &= \max_{c \in T_c}(|loc_c - loc_T|) + \max_{f \in T_f,\, T_f \subset T_{pos}}(|loc_T - loc_f|) \\ &= \max_{c \in T_c}(|loc_c - loc_T|) + \max_{f \in T_{pos}}(|loc_T - loc_f|) \end{aligned} \tag{5}$$

Here loc_c is location required by a currently active task, loc_T is location required by the negotiated task and loc_f is location required by a possible future task, currently in negotiation. The minimum additional effort corresponding to the location of task T is the effort deployed to move from a location required by one of the current tasks (loc_c) to loc_T and then to the closest location required by one of the future tasks. The minimum

effort corresponds to the optimal ordering of the current and future tasks with respect to the location. The maximum effort corresponds to the worst ordering of the tasks.

It can easily be observed that computation of the location effort depends on the current location and the locations required by the current and future tasks. The reason for the simplification in (4) and (5) is that the minimum and maximum distances can be computed without the need to generate all the possible subsets T_f of T_{pos}. As $T_f \subseteq T_{pos}$, we only have to consider tasks $f \in T_{pos}$ and thus the complexity of the computation of the location effort is $O(|T_c| + |T_{pos}|)$.

Time. $Time$ is usually represented either as deadlines or as time durations. A deadline represents the latest moment in time when the results of the service execution must be provided or the associated physical activities must terminate. A time duration represents the amount of time taken by service execution. When they are projected onto the provider agent timeline (i.e. attached to a certain moment of time), time durations become equivalent to deadlines, so in what follows we shall only consider time modeled as deadlines. If $time$ represents the deadline of providing results of service execution, the associated agent effort to meet the deadline is:

$$effort^{-}_{add}(time_T) = effort^{+}_{add}(time_T) = comptime(T) \tag{6}$$

The explanation is that independently of how tasks are arranged or how many future tasks are considered, the agent will spend the same amount of time with the execution of task T. As the computation of time effort does not depend on the current tasks and on the future tasks, the resulting complexity is $O(1)$.

Note that while computation of time effort is very simple, computation of the location effort takes more steps. Overall, the step of creating proposals has $O(l_j \times (|T_c| + |T_{pos}|))$ complexity, where l_j is the number of location issues of negotiation j. Typically in a system like ours there is one *time* issue and one *location* issue, so we can consider $l_j = 1$ and state that computation complexity is $O(|T_c| + |T_{pos}|)$ for negotiations that involve one issue *time* and one issue *location*. In Section 4.3 when we deal with scalability, we consider that a significant percent of the total number of negotiations involve *time* and *location* issues.

Awarding phase is available only to manager agent. He uses a weighted sum utility function to select the best contractor. Utility computation takes $O(m_j \times (|A_j| - 1))$ steps, where m_j is the total number of issues of negotiation j.

4.2 Communication Complexity

Because our version of CNET omits service provision messages and uses only call for proposal, propose, reject, accept and refuse messages (see Section 4), a negotiation instance j brings an overhead to the system consisting of maximum $3 \times (|A_j| - 1)$ messages, where A is the set of agents in the system and $A_j \subset A$ is the set of agents involved in negotiation j. First, manager sends a call for proposal to the rest of agents in A_j, then these agents reply either with a proposal or a reject message and finally manager sends acceptance or refusal messages to the participants. The number of messages may be smaller because the manager might not wait for all the replies (when the deadline occurs) and it does not send final messages to agents that did not send proposals.

4.3 Scalability

Scalability measures system adaptability to a large number of agents, services and negotiations. We denote with: (i) SA (services per agent) the average number of services an agent is able to provide, (ii) RS the average number of services a certain service requires, (iii) I the total number of incidents the system is able to handle concurrently, and (iv) SI (service invocations) the average number of times a service of an agent can be negotiated per incident. As previously mentioned, the total number of agents in the system is $|A|$ and the number of participants in negotiation j is $|A_j|$.

An agent can act simultaneously both as manager and contractor. The average number of times an agent is manager during all incidents is $n_{man} = SI \times RS \times SA \times I$ and the average number of times an agent is contractor is $n_{con} = SI \times SA \times I$. Then, the total number of negotiations an agent is involved in is $n_{agent} = n_{man} + n_{con} = SI \times SA \times I \times (1 + RS)$. Additionally, the total number of current and possibly future tasks of an agent is less than or equal to the total number of tasks an agent is capable of simultaneously handling for all the incidents, i.e. $|T_c| + |T_{pos}| \leq SA \times SI \times I$. We conclude that computation complexity of an agent for all negotiations about *time* and *location* for all incidents is:

$$\begin{aligned} &cc_{agent} = n_{con} \times cc_{con} + n_{man} \times cc_{man} = \\ &(SI \times SA \times I) \times (|T_c| + |T_{pos}|) + (SI \times RS \times SA \times I) \times m_j \times (|A_j| - 1) \leq \\ &(SI \times SA \times I) \times [(SI \times SA \times I) + RS \times m_j \times (|A| - 1)] \end{aligned} \tag{7}$$

Note that we have omitted the steps requiring constant time (e.g. the task announcement). The resulting computation complexity (which is an upper bound) is linear with respect to the number of agents ($|A|$), but quadratic with respect to the number of services per agent (SA), number of incidents (I) and the number of times the same service of an agent can be negotiated during an incident (SI). There is no general relation between the total number of agents and the rest of the parameters. Moreover, there is no general relation between the total number of agents and the agents involved in a negotiation. The actual performance can only be evaluated through experiments by varying values of SA, I, SI, RS, $|A_j|$ and m_j.

However, based on this worst case analysis, we can get a glimpse at the performance by looking into the project's requirements. The DIADEM User Requirements Document [14] states that the system should support the collaboration of 100 individual stakeholders ($|A| = 100$), handling at least 3 simultaneous chemical incidents (I = 3). We assume that each agent is capable of providing 5 services ($SA = 5$) and that no service requires more than 3 other services ($RS = 3$). We consider that an agent can supply a service for at most 2 times during an incident ($SI = 2$). Under these conditions, the number of computation steps of one agent for all the incidents (all negotiations) is:

$$cc_{agent} = 900 + 8910 \times m_j \tag{8}$$

It is important to note that the agent will not handle all the negotiations at once. This number of steps is computed for all the negotiations, but there are delays between negotiations and the agent does not have to execute all the steps at once. We conclude that this complexity does not overwhelm the agent, as the number of steps is not large.

According to results of Section 4.2, communication complexity of a negotiation is $3 \times (|A_j| - 1)$. The traffic for one agent for all the incidents (all negotiations) is therefore:

$$traffic_{agent} = n_{agent} \times 3 \times (|A_j| - 1) = SI \times SA \times I \times (1 + RS) \times 3 \times (|A_j| - 1) \leq SI \times SA \times I \times (1 + RS) \times 3 \times (|A| - 1) \quad (9)$$

Under the conditions described above ($|A| = 100$, $I = 3$, $SA = 5$, $RS = 3$, $SI = 2$), the traffic for one agent is:

$$traffic_{agent} \leq 36000 \text{ messages} \quad (10)$$

This number of messages is quite large, even if in realistic scenarios it will be slightly lower, as not all negotiations involve all the agents. But as there is no general relation between $|A|$ and $|A_j|$, we can only compute an upper bound of the communication complexity (just like we did for the computation complexity). The actual performance will be evaluated through experiments, by varying the parameters.

5 Related Work

There is little literature on the evaluation of the impact of service negotiation mechanisms on large scale distributed systems. The few papers that exist deal with CNET protocol. In [5] the authors evaluate through experiments different types of contracts introduced in [4]. Our study is different because our problem deals with service negotiation between collaborative agents that do not accept payments. Agents are able to provide only certain sets of services (depending on their capabilities) and tasks cannot be exchanged between agents. As a result we cannot use such contracts and study concepts such as social welfare and avoiding local optima.

[3] theoretically evaluates various extensions of CNET according to a set of criteria, but without a specific problem in mind. On the other hand we focus on performance of negotiations in the context of service negotiation in agent-based disaster management systems. Moreover, our analysis considers specific features of disaster management problems that demand agents' physical mobility and working under time constraints.

A scalability analysis of CNET has been done in [2]. The authors draw the conclusion that CNET cannot be used without deadlines and that negotiations have to be repeated under heavy load. In this paper we evaluate CNET not only with respect to scalability, but complexity also. We consider scalability to depend on the results of the evaluation with respect to the other performance measures.

6 Conclusions

Negotiation components that have a large impact in the system are the decision steps and the communication. In this paper we have proposed a framework for the performance evaluation of service negotiation mechanisms and we have analyzed the computation complexity and the communication complexity of service negotiation in disaster management situations. We have used per-issue computation in order to compute the effort of one agent to provide a service. In particular, we have given examples of how the

agent computes the effort for issues *location* and *time*. Then we scaled up the analysis for one negotiation to many negotiations involving *location* and *time* in order to analyze scalability. We conclude that our service negotiation mechanism has a fair computation complexity, that the system should be capable of handling, but the communication complexity is large. As there is no general relation between the parameters introduced in Section 4.3, an experimental evaluation of the actual performance is still needed.

The analysis in this paper is preliminary. We have derived the maximum complexity of computation and communication, but it would be interesting to observe how does the system perform during realistic incidents. We plan to experimentally evaluate the performance of the negotiations by taking into consideration the average number of agents participating in a negotiation (A_j), the delay between negotiations, the average number of issues per negotiation (m_j), the average number of services an agent is able to provide (SA), the average number of services a certain service requires (RS), the total number of incidents the system handles (I) and the average number of times a service of an agent can be negotiated per incident (SI).

References

1. Pavlin, G., Kamermans, M., Scafeş, M.: Dynamic process integration framework: Toward efficient information processing in complex distributed systems. In: Intelligent Distributed Computing III, Proc.3rd International Symposium on Intelligent Distributed Computing, IDC'2009. Number 237 in Studies in Computational Intelligence, Springer Verlag (2009) 161–174
2. Juhasz, Z., Paul, P.: Scalability analysis of the contract net protocol. In: CCGRID '02: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, Washington, DC, USA, IEEE Computer Society (2002) 346
3. Bozdag, E.: A survey of extensions to the contract net protocol. Technical report, CiteSeerX - Scientific Literature Digital Library and Search Engine, http://citeseerx.ist.psu.edu/oai2 (United States) (2008)
4. Sandholm, T.: Contract types for satisficing task allocation: I theoretical results. In: AAAI Spring Symposium: Satisficing Models. (1998)
5. Andersson, M.R., Sandholm, T.: Contract types for satisficing task allocation: Ii experimental results. In: AAAI Spring Symposium: Satisficing Models. (1998)
6. Scafeş, M., Bădică, C.: Conceptual framework for design of service negotiation in disaster management applications. In: To appear in Proc. of 2nd International Workshop on Agent Technology for Disaster Management (ATDM09). Studies in Computational Intelligence, Springer Verlag (2010)
7. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Trans. Comput. 29 (1980) 1104–1113
8. Santoro, N.: Design and Analysis of Distributed Algorithms. John Wiley & Sons (2007)
9. Jennings, N. R., Faratin, P., Lomuscio, A.R., Parsons, S., Sierra, C., Wooldridge, M.: Automated negotiation: Prospects, methods and challenges. International Journal of Group Decision and Negotiation 10 (2001) 199–215
10. Sandholm, T. W.: An implementation of the contract net protocol based on marginal cost calculations. In: Proceedings of the 12th International Workshop on Distributed Artificial Intelligence, Hidden Valley, Pennsylvania (1993) 295–308
11. Scafeş, M., Bădică, C.: Service negotiation mechanisms in collaborative processes for disaster management. In Eleftherakis, G., Hannam, S., Kalyva, E., Psychogios, A., eds.:

Proceedings of the 4th South East European Doctoral Student Conference (DSC 2009), Research Track 2: Information and Communication Technologies, Thessaloniki, Greece, SEERC (2009) 407–417
12. Bădică, C., Scafeş, M.: One-to-many monotonic concession negotiation protocol. In: Proc.of the 4^{th} Balkan Conference in Informatics, BCI'2009, IEEE Computer Society (2009) 8–13
13. FIPA: Fipa contract net interaction protocol specification, identifier sc00029 (2002)
14. Damen, D., Pavlin, G., Van Der Kooij, C., Bădică, C., Comes, T., Lilienthal, A., Fontaine, B., Schou-Jensen, L., Jensen, J.S.: Diadem environmental management requirements document, issue 1.14.0. work-in-progress (2010)