INTELLIGENT AGENTS – A TOOL FOR MODELING INTERMEDIATION AND NEGOTIATION PROCESSES

Amelia BĂDICĂ, Associate Professor, PhD.
University of Craiova
Georgeta ŞOAVĂ, Professor, PhD.
University of Craiova

Keywords: Business processes, Intelligent Agents, Intermediation and Negotiation, Formal Models.

Abstract. Many contemporary problems as encountered in society and economy require advanced capabilities for evaluation of situations and alternatives and decision making, most of the time requiring intervention of human agents, experts in negotiation and intermediation. Moreover, many problems require the application of standard procedures and activities to carry out typical socio-economic processes (for example by employing standard auctions for procurement or supply of goods or convenient intermediation to access resources and information). This paper focuses on enhancing knowledge about intermediation and negotiation processes in order to improve quality of services and optimize performances of business agents, using new computational methods that combine formal methods with intelligent agents paradigm. Taking into account their modularity and extensibility, agent systems allow facile, standardized and seamless integration of negotiation protocols and strategies by employing declarative and formal representations specific to computer science.

1. Introduction

The globalization of the business environment and the increasing of the competitiveness in the new economy make it necessary to develop new devices/ means/ tools/ instruments of fundamental research in order to better exploit the emergent technologies that will represent the basis of the emerging global informational networks and the infrastructure on the large scale for a global access to the information, resources, and services.

There is a certain need to develop the knowledge of the negotiation and intermediation processes in order to improve the quality of the services and to optimize the economic agents achievements. In this purpose new computational methods for studying, modeling, classifying and analysing the negotiation and intermediation processes have to be used. The major prerequisite of this paper is that many current problems of our society and economy require advanced capabilities both for accessing information and resources and also of analyzing and assessing the situations and the alternatives. The solutions often imply the action of the human agents with expertise in intermediation and negotiation areas. In many such problems the solution is to apply new standard procedures and activities for carrying out of socio-economic processes (for example, the use of auctions for purchasing or selling goods, or appropriate intermediation for the access to information and resources). From this perspective, our paper aims to study the intermediation and negotiation processes in order to propose new computational models, based on intelligent agents paradigm. Due to their modular and extensible character, agent systems allow an easy inclusion, in a standardized and

flexible way, of diverse protocols and strategies of interaction, that use formalized and precise representations, specific to the computer science.

2. Intermediation and Negotiation Processes

Business success depends, in a higher degree, of the way negotiations are carried out. In a market, competing and dynamic economy and that is full of problems, almost everything is negotiated. Thus, negotiation is one of the most important characteristics of cotemporary life. The rules governing preparation, organization and the development of negotiations, also the business and negotiation protocol, must be known by those who negotiate [Pistol, 2002]. Negotiation science may help people to substantially improve the ability to conduct business, both on economical and psychological aspects. Negotiation is the main instrument of communication and influence inside and outside the organization, is an interpersonal decision process that is needed every time we cannot reach the objective alone. The analysis can be focused on the negotiator mind, because it is necessary to design negotiation strategies, to reach the heart of the negotiator, because finally, it is the relationship and confidence that counts [Thompson, 2006]. Negotiation is not only philosophy, but also it does not remain at the level of simple intuition. All the agreements accomplished are of win-win type. The parties would not accept an arrangement, if it would not be more advantageous than the lack of agreement. A confident attitude is necessary to reach the objectives, based on sound and tested knowledge referring to the negotiation process. However, before getting an agreement, people inevitably undergo a state of disagreement, more or less a conflict one.

Intermediation and negotiation are fundamental processes that allow human and/or intelligent software agents to access services offered by other agents, and to set up the terms and conditions of use of these services.

Intermediation is the process of solving the connection problem among the provider agents and the consumers of information, resources and/or services in a global environment, such as the global business environment and the Internet. This process is based on matching between requestors' preferences and providers' capabilities and it is carried out by specialized agents, known as *middle-agents*. The seminal paper [Decker, 1997] proposed a fundamental classification of middle-agents based on what it is initially known about preferences and capabilities within the interaction between middle-agents, provider agents and requestor agents, as follows: *Broadcaster, Front-Agent, Matchmaker* (also known as *Yellow-pages), Anonymizer, Broker, Recommender, Blackboard, Introducer, and Arbitrator*.

Negotiation is referred to as the process of interaction among diverse groups of human and/or intelligent agents in order to achieve a mutual agreement about a certain problem [Lomuscio, 2003]. Multi-agent negotiations were also defined as processes carried out in order to allocate different categories of resources (products, services, money, access to information, band-width of a communication channel) to agents participating to a negotiation [Chevaleyre, 2006 and Wurman, 2001]. The outcome of a negotiation process is a contract, i.e. a mutual agreement among a set of participants, at least two. A contract is made up of the following: i) a description of the contracting parties: companies, persons and/or individuals, intelligent agents; ii) a lot of mutual promises, also called stipulations of the contract, which define the rights and obligations of the parties. When the participants to a negotiation are intelligent agents, this process is called an automated negotiation. When the participants are human

agents, but the negotiation is carried out with the help of digital technologies, then is the case of an electronic negotiation, called e-negotiation.

In this paper we propose to treat the negotiation as a particular case of the intermediation process executed with the help of a specialized agent from the category *Arbitrator*. The advantage of this approach is its integrated and, at the same time, general characteristics of the model proposed for the analysis of intermediation and negotiation processes. So, the paper suggests a new computational model based on the extension of the results obtained by our previous research regarding the use of process algebra for modeling and qualitative analysis of the business processes.

3. State-of-the-art of research in the area

The last decay demonstrated a significant interest for the area of modeling the organizational process, inside the community of business management [van Loon, 2007 and Ould, 2005] as well as in the community of computer science [Phalp, 1998]. While the interest of management community is specially orientated to the use of the models in order to evaluate and improve the quality of the processes within an organization, in accordance with the standards and the practices in force (for example ISO IEC 1550427), the computers science community is preoccupied in particular to study formalisms and languages of representation and qualitative evaluation of the process. The point of convergence of these two orientations is modeling and precise analysis of the process [Anderson, 2005 and Janssen, 1999]. Our paper tries to reach this aim conceiving new information models to study and analyze the middle-agents negotiation agents. Research literature points out the fact that, although the interest for the subject of the process of intermediation is increasing, only a very few papers address the problem of a concise definition of the middle-agents in terms of their interaction capabilities with providers and requesters. Citing from [Klusch, 2001], '"notions of middle-agents, matchmakers, brokers [...] are used freely in the literature without necessarily being clearly defined".

The topic of middle-agents was recently brought to the attention of scientific community by the work [Fasli, 2007]. The author of work [Fasli, 2007] presents in an informal way the following types of middle-agents: *Matchmaker, Broker, Broadcaster and Recommender*. The presentation is specially oriented to the interaction protocols, representation languages and matching techniques of providers' capabilities and requesters preferences.

The work [Klusch, 2001] shows only models of *Matchmaker* and *Broker* agents using input/output automata, while the work [Hristozova, 2002] presents the interactions between middle-agents, providers and requesters as sequences of message exchanges presented natural language. Some recent approaches propose the use of the process algebras for precise modeling of the interactions in multi-agent systems. The work [Esterline, 2006] proposes the use of *pi-calculus* and presents models of a prototype system of agents for an unattended grounds operation-center using a fault resolution scenario in theorz, without any results. There are also approaches to modeling and formal checking the agent systems using formal specification languages that are different from process algebras. Thus, [Albrecht, 2003] proposes the use of the situation calculus, a formalism to model the dynamic systems in artificial intelligence. The work [Podorozhny, 2007] shows an interesting approach to modeling and formal verification of negotiation agents using the software tool Alloy, based on the modeling technique and not on experimental outcome.

The work [Merayo, 2007] introduces an extension of the a formalism of the machines with finite state - *Extended Utility State Machines* (EUSM) for strategic aspects modeling of intelligent agents. EUSM allows a more detailed representation of the intelligent agents behavior as compared with FSP using state variables and utility functions. The paper [Miller, 2007] proposes the RASA approach based on process algebra extended with constraint-based reasoning capabilities for the representation of semantic aspects of interaction protocols. However, the work [Miller, 2007] presents only a simplistic example, and it is not obvious when and how the RASA formalism can practically be applied.

The problem of formal modeling of business processes was addressed by the works [Badica, 2005], [Badica, 2004], [Badica, 2003] and [Dogaru, 2006]. In paper [Badica, 2003] we addressed the problem of formal modeling using FSP of the processes captured as diagrams of roles and activities. In work [Dogaru, 2006] we have shown how an agent system, also employing a *Matchmaker* middle-agent, can be used for matching requests and offers within a system for news syndication in electronic commerce. In works [Dobriceanu, 2007] we have proposed a generic framework for designing and implementation of negotiations in agent systems, using an *Arbitrator* middle-agent as a host coordinating the negotiation process. Work [Badica, 2007] outlines the general framework of modeling that will be analyzed and further extended in this project. Works [Badica, 2008] and [Badica, 2007] present preliminary results for modeling and checking negotiation processes, using an example of the English auction.

4. Types of Middle-Agents

We focused on a concise natural language description of the middle-agent interaction patterns.

Broadcaster. A *Broadcaster* middle-agent assumes that requester preferences are initially known only to the requester and provider capabilities are initially known only to the provider. In our opinion the main characteristic of a *Broadcaster* is that it broadcasts requests to available providers, but note that, at least in theory, the inverse situation is conceivable, i.e. advertisements of provider capabilities may be broadcasted to available requesters. Considering the first case, this basically shows that a *Broadcaster* does not have the necessary knowledge for determining a matching provider, and consequently it broadcasts the request to let providers decide themselves if the request matches or not their capabilities (but it has knowledge about what providers are available in the system). The following scenarios are conceivable: i) matching providers respond to *Broadcaster*; ii) matching providers respond directly to requester. Assuming that the main function of a *Broadcaster* is to discover matching providers (broadcasting is frequently used by discovery services), for efficiency reasons we rule out the first scenario.

Front-agent. A *Front-agent* middle-agent (also known as *Proxy*) assumes that requester preferences are initially known only to the requester, while provider capabilities are initially known both to the provider and to the middle-agent. This means that a provider will have to advertise its capabilities with the *Front-agent* and the *Front-agent* has the responsibility to match a request with registered capabilities advertisements. Additionally, as the provider capabilities are not initially known to the requester, the *Front-agent* also has the responsibility of intermediating the transaction between the requester and the matching provider (this is why often this type of middleagent is called *Broker* rather than *Front-agent*).

Matchmaker. A*Matchmaker* middle-agent (also known as *Yellow-pages*) assumes that requester preferences are initially known only to the requester, while provider capabilities are initially known to all participants in the interaction. This means that, as with the *Front-agent*, a provider will have to advertise its capabilities with the *Matchmaker* and the *Matchmaker* has the responsibility to match a request with registered capabilities advertisements. However, differently from the *Front-agent*, the fact that provider capabilities are initially known also by the requester means that the result of the matching (i.e set of matching providers) now goes back from *Matchmaker* to requester, and the choice of the matching provider is the responsibility of the requester. Another difference is that the transaction is not intermediated by the *Matchmaker*; rather, requester sends request to the chosen matching provider, which in turn returns result.

Anonymizer. An *Anonymizer* middle-agent assumes that requester preferences are initially known both to the requester and middle-agent, while provider capabilities are initially known only to the provider. This situation is symmetric with *Front-agent*, i.e nowthe requester has to register his preferences with the *Anonymizer*, while the *Anonymizer* has the responsibility to match a provider capability with registered requester preferences. Additionally, as the requester preferences are not initially known to the provider, the *Anonymizer* also has the responsibility of intermediating the transaction between the matching requester and the provider.

Broker. A *Broker* middle-agent assumes that requester preferences are initially known only to the requester and provider capabilities are initially known only to the requester and the middle-agent. The crucial point is that, however, requester preferences are not initially known to the provider and provider capabilities are not initially known to the requester. This basically means that a *Broker* will truly intermediate transactions between providers and requesters in both directions.

Recommender. A *Recommender* middle-agent assumes that requester preferences are initially known both to the requester and the middle-agent, while provider capabilities are initially known to all participants in the interaction. First, similarly with a *Matchmaker*, a provider will have to advertise its capabilities with the *Recommender* and the *Recommender* has the responsibility to match a request with registered capabilities advertisements and return a set of matching providers to the requester. However, differently from a *Matchmaker*, if no matching provider is found the request is remembered by the middle-agent as a wish-ad. Also, differently from a *Matchmaker*, when a new capability offer of a provider is registered with the *Recommender*, it is also matched with registered wish-ads, and if a match is found, the provider id is notified by the *Recommender* to the matching requester – i.e. the middle-agent 'recommends' the provider to the requester.

Blackboard. A *Blackboard* middle-agent assumes that provider capabilities are initially known only to the provider, while requester preferences are initially known to all participants in the interaction. This means that, as with the *Anonymizer*, a requester will have to register its preferences with the *Blackboard* and the *Blackboard* has the responsibility to match a provider capability with registered preferences. However, differently from the *Anonymizer*, the fact that requester preferences are initially known also by the provider means that the result of the matching (i.e set of matching requesters) now goes back from *Blackboard* to provider, and the choice of the matching requester to serve is the responsibility of the provider. Another difference is that the transaction is not intermediated by the *Blackboard*.

Introducer. An *Introducer* middle-agent assumes that requester preferences are initially known to all participants in the interaction, while provider capabilities are initially known only to the provider and the middle-agent. With an *Introducer*, the situation is rather symmetric with the *Recommender*. The requester will have to register its preferences with the *Introducer* and the *Introducer* has the responsibility to match an incoming provider capability with registered requests and return a set of matching requesters to the provider. Additionally, (independently if a matching requester is found or not) the provider capability is registered with the middle-agent as a capability advertisement. Also, when a new request is registered with the *Introducer*, it is also matched with registered capability advertisements, and if a match was found, the requester id is notified by the *Introducer* to the matching provider – i.e. the middle-agent 'introduces' the requester to the provider (this particular behavior probably gives the name of this middle-agent).

Arbitrator. An *Arbitrator* middle-agent assumes that both requester preferences and provider capabilities are initially known to all participants in the interaction. This gives the possibility of a broad range of interaction patterns that are impossible to capture in a single model. Typical applications of *Arbitrator*s are resource allocation (negotiation) and dispute resolution. An *Arbitrator* middle-agent can be used for modeling an English auction.

5. Modeling Negotiations with Middle-Agents and FSP

Connecting requesters with providers was recognized as a crucial problem in an agent environment. Its solution requires the use of middle-agents – replacements of middlemen in a virtual environment. In this paper we considered formal models of domain independent interaction patterns between software agents involved in intermediation. the framework we used is the *finite state process algebra* – FSP modeling language.

FSP is an algebraic specification technique of concurrent and cooperating computational processes as finite state labeled transition systems (LTS hereafter). FSP allows a more compact and easy to manage description of a LTS, rather than directly describing it as a list of states and transitions between states.

A FSP model consists of a finite set of sequential and/or composite process definitions. Additionally, a sequential process definition consists of a sequence of one or more definitions of local processes. A process definition consists of a process name associated to a process term. FSP uses a rich set of constructs for process terms. For the purpose of this paper we are using the following constructs: prefix, choice, and process alphabet extension for sequential process terms and parallel composition and relabeling for composite process terms. FSP has an operational semantics given via a LTS.

In the modeling of negotiations we used the following algorithm:

i) Agents are modeled as FSP processes. As our process language is FSP, we have chosen to model agent types as sequential processes. Instantiation of an agent of a given type can be defined by invoking the associated process with a suitable renaming of its alphabet.

ii) A multi-agent system is modeled as a parallel composition of processes. It follows that communication between agents is modeled by appropriately utilizing the synchronization capabilities of FSP. In FSP, synchronization of processes of a parallel composition is done by default on their common alphabets. So, special care should be taken in order to accurately model agents communication using FSP synchronization.

Depending on circumstances, this will require renamings and/or alphabet extensions of local processes.

iii) We assume that R is the set of requesters and P is the set of providers. Identity of requester and provider agents is explicitly represented by integer indexes $r \in R$ and $p \in P$. Additionally: 1) requests made by requester agents are indexed with the requester id; 2) requests made to the providers are indexed with the provider id. It follows that:

a) requests from requesters to middle-agents are indexed only with the id of the requester – action *request*(*r*);

b) requests from middle-agents to providers are indexed only with the id of the providers;

c) requests made by requesters directly to providers are indexed with both ids of the requester and the provider – action *request_ to_ provider*(*r*, *p*).

This assumption means that details of a request/capability are ”incorporated” in the id of the requester/provider.

iv) Details like request preferences and service capabilities are abstracted away from our models, as we consider that such details are not necessary to understand the specific particularities of interaction for each type of middle-agent.

v) The matching operation is modeled as a relation M between the sets P of providers and R of requesters, i.e. $M \subseteq P \times R$. Activity *match_req*(*r*, *P*) indexed with id *r* of a requester and set of ids *P* of matching providers will be used to model the operation of matching a request from requester *r* against registered providers, i.e. $P = M(r)$. Symmetrically, activity *match o f f* (*p*, *R*) indexed with id *p* of a provider and set of ids *R* of matching requesters will be used to model the operation of matching a capability offer of provider *p* against pending requests, i.e. $R = M^{-1}(p)$.

6. An example of a Recommender Middle-Agent in FSP

We assume that our system contains providers, requesters and a *Recommender* middle-agent. So we have *Provider*, *Requester* and *Recommender* processes (see fig.2). *Provider* agent registers its capability offer (action *offer*) with the *Recommender* and then enters a loop where it receives requests from *Requester* agents (action *receive request*) and processes and replies accordingly (action *send reply*). *Requester* agent submits a request to the *Recommender* (action *send request*) and then waits for a reply. The *Recommender* replies with a set of matching providers (action *tell* with argument *P* representing the set of matches). Then the *Requester* has the option to choose what provider from set *P* to actually contact for performing the service (action *send request to provider* with argument *p* representing the contacted provider). Finally, *Requester* waits for a reply from the contacted provider (action *receive reply*).

$$
\begin{array}{lcl}
Provider & = & offer \rightarrow ProcessRequests, \\
ProcessRequests & = & receive\_request \rightarrow \\
 & & send\_reply \rightarrow ProcessRequests. \\
 & & \\
Requester & = & send\_request \rightarrow WaitReply, \\
WaitReply & = & tell(P \subseteq \mathcal{P}) \rightarrow ContactProvider(P), \\
ContactProvider(P \subseteq \mathcal{P}) & = & send\_request\_to\_provider(p \in P) \rightarrow \\
 & & receive\_reply(p) \rightarrow Requester. \\
 & & \\
Recommender & = & (Recommender(\emptyset, \emptyset), \\
Recommender(R, P) & = & request(r \notin R) \rightarrow MatchReq(r, R, P) | \\
 & & offer(p \notin P) \rightarrow MatchOff(p, R, P), \\
MatchReq(r, R, P) & = & \textbf{if } \mathcal{M}(r) \cap P = \emptyset \textbf{ then} \\
 & & no\_match\_req \rightarrow Recommender(R \cup \{r\}, P) | \\
 & & \textbf{if } \mathcal{M}(r) \cap P \neq \emptyset \textbf{ then} \\
 & & match\_req(r, \mathcal{M}(r) \cap P) \rightarrow \\
 & & tell(r, \mathcal{M}(r) \cap P) \rightarrow Recommender(R, P), \\
MatchOff(p, R, P) & = & \textbf{if } \mathcal{M}^{-1}(p) \cap R = \emptyset \textbf{ then} \\
 & & no\_match\_off \rightarrow Recommender(R, P \cup \{p\}) | \\
 & & \textbf{if } \mathcal{M}^{-1}(p) \cap R \neq \emptyset \textbf{ then} \\
 & & match\_off(p, \mathcal{M}^{-1}(p) \cap R) \rightarrow tell(r_1, \{p\}) \rightarrow \\
 & & \ldots \rightarrow tell(r_k, \{p\}) \rightarrow Recommender(R, P \cup \{p\})) \\
 & & + \{tell(r \in R, P \subseteq \mathcal{P})\}.
\end{array}
$$

Figure 1. Agent types as FSP processes

$$
\begin{array}{lcl}
Requester(r \in \mathcal{R}) & = & Requester / \{request(r) / send\_request, tell(r, P \subseteq \mathcal{P}) / tell(P), \\
 & & request\_to\_provider(r, p \in \mathcal{P}) / send\_request\_to\_provider(p), \\
 & & reply(r, p \in \mathcal{P}) / receive\_reply(p)\}. \\
 & & \\
Provider(p \in \mathcal{P}) & = & Provider / \{offer(p) / offer, \\
 & & request\_to\_provider(r \in \mathcal{R}, p) / receive\_request, \\
 & & reply(r \in \mathcal{R}, p) / send\_reply\}. \\
 & & \\
System & = & Recommender \,||\, (||_{r \in \mathcal{R}} Requester(r)) \,||\, (||_{p \in \mathcal{P}} Provider(p)).
\end{array}
$$

Figure 2. *System* **model**

Definition of *Recommender* agent is made using a set of indexed families of local processes. Assuming that R is the set of all requesters and P is the set of all providers:

i) *Recommender*(*R*, *P*) is defined for $(R, P) \in 2^R \times 2^P$. Sets *R* and *P* represent memorized requests (i.e. not yet honored) and registered capability offers of providers. When a new request is submitted (action *request*)) the requester id *r* is passed to the local process *MatchReq* that matches *r* with registered provider capabilities. When a new provider capability is registered (action *offer*)) the provider id *p* is passed to the local process *MatchOff* to check if there are any memorized requests matching with *p*. Note the use of alphabet extension for process *Recommender* in order to assure correct

synchronization with information passing from *Recommender* to *Requester* about matching providers.

ii) *MatchReq*(r, R, P) is defined for (r, R, P) $\in$ R$\times 2^{R}\times 2^{P}$. If there are no matching providers for request r (i.e. M(r)$\cap$ P = ϕ) then this is signaled using action *no_ match_ o f f* and request r is memorized to try serving it latter. If there are matching providers (i.e. M(r) $\cap$ $P \neq \phi$) then this is signaled using action *match req* and requester r is notified accordingly using action *tell* (note the set of matching providers passed to r as second parameter of action *tell*).

iii) *MatchOff* (p, R, P) is defined for (p, R, P) $\in$ P$\times 2^{R}\times$ 2^{P}. If there are no matching requests waiting to be served for provider p (i.e. $M^{-1}(p)\cap R = \phi$) then this is signaled using action *no match req*. However, if at least one matching request is found (i.e. $\{r_1, \ldots, r_k\} = M^{-1}(p) \cap R \neq \phi$, $k \geq 1$) then the new available provider p is recommended to each of the matching requesters ri, $1 \leq i \leq k$ using action *tell*.

Definition of requesters, providers and a middle agent system assumes i) instantiation of *Provider* and *Requester* agent types for each particular requester and provider created in the system using the FSP renaming operator; ii) taking the parallel composition of processes representing providers, requesters and the middle-agent (see fig.3). We have implemented the model shown in figure 3 with the help of LTSA tool for a system composed of 2 requesters and 3 providers, i.e. R = {1, 2} and P = {1, 2, 3} and the matchings M= {(1, 2), (1, 3), (2, 1), (2, 3)}. For this purpose we had to map processes indexed with sets to the FSP notation supported by LTSA. We have used the following conventions: i) indexes $r \in$ R and $p \in$ P have been encoded as [r] and [p]; ii) if |R| = m then an index $R \subseteq$ R is encoded as $[r_1] \ldots [r_m]$ such that $r_i = 1$ if $i \in R$ and $r_i = 0$ if $i \notin R$ for all $1 \leq i \leq m$; iii) if |P| = n then an index $P \subseteq$ P is encoded as $[p_1] \ldots [p_n]$ such that $p_i = 1$ if $i \in P$ and $pi = 0$ if $i \notin P$ for all $1 \leq i \leq n$; For example, local process *MatchOff* (2, {1}, {1, 3}) is mapped to *MatchOff* [2][1][0][1][0][1] and action *tell*(1, {2, 3}) is mapped to *tell*[1][0][1][1].

7. Conclusions and future work

The paper presented the necessity to enhance knowledge referring to intermediation and negotiation processes. We suggested studying, modeling, classification and analyzing of these processes using new computational methods of formal specification of multi-agent systems.We reviewed well-known middle-agent types and proposed finite state process algebra to precisely characterize their interactions. Our approach was exemplified on the example of a *Recommender* middle-agent.

As future work we plan to: i) finalize the models of all middle-agent types, ii) define and study their qualitative properties (liveness and safety), iii) investigate the use of these models for implementation of middle-agents using an agent toolkit, iv) extend the approach with more complex specifications taking into account reasoning capabilities of intelligent software agents.

REFERENCES

1. Albrecht C., Dean D. , Hansen J. V. (2003) - Using situation calculus for e-business agents. *Expert Systems with Applications*, Volume 24, Issue 4, 391-397, Elsevier;

2. Anderson B. B., Hansen J.V., Lowry P.B., Summers S.L. (2005) - Model checking for E-business control and assurance. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Volume 35, Issue 3, 445-450;

3. Badica A., Badica C.(2008) Formalizing Agent-Based English Auctions Using Finite State Process Algebra. Acceptata pt. publ. în *Journal of Universal Computer Science*;

4. Badica A., Badica C., Litoiu V. (2007) - Middle-Agents Interactions as Finite State Processes: Overview and Example. *Proc. 16th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2007)*, Paris, France. 12-17, IEEE Computer Society;

5. Badica A., Badica C. (2007) - Formal Modeling of Agent-Based English Auctions Using Finite State Process Algebra. First KES Int. Symp. on Agent and Multi-Agent Systems: Technologies and Applications, Wroclaw, Poland. 248-257, *LNCS* 4496, Springer;

6. Badica A., Badica C., Teodorescu M., Spahiu C., Fox C. (2005) - Integrating Role Activity Diagrams and Hybrid IDEF for Business Process Modeling Using MDA, In: *Proceedings 7th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC'05*, Romania. IEEE Comp.Soc. Press, pp. 71-74;

7. Badica C., Fox C. (2004) - On the Application of WF-nets for Checking Hybrid IDEF0-IDEF3 Business Process Models. In: *LNCS* 3261, pp.543-553, Springer-Verlag;

8. Badica C., Badica A., Litoiu V. (2003) - Role Activity Diagrams as Finite State Processes. In: Marcin Paprzycki (ed.): *Proc. of the International Symposium on Parallel Distributed Computing*, Ljubljana, Slovenia, pp.15-22, IEEE Comp. Soc. Press;

9. Badica C., Badica A., Litoiu V. (2003) - A New Formal IDEF-based Modeling of Business Processes, *Proc.1st Balkan Conf. in Informatics*, Thessaloniki, Greece, pp.535-549;

10. Chevaleyre Y., Dunne P.E., Endriss U., Lang J., Lemaitre M., Maudet N., Padget J., Phelps S., Rodrıguez-Aguilar J.A., Sousa P. (2006) - Issues in Multiagent Resource Allocation. *Informatica*, 30:3-31, Slovenian Society Informatika;

11. Decker, K., Sycara, K. P., and Williamson, M. (1997) - Middle-agents for the internet. In: *Proceedings of the* 15th *International Joint Conference on Artificial Intelligence IJCAI'97*, vol.1, 578–583, Morgan Kaufmann;

12. Dobriceanu A., Biscu L., Badica C., Popescu E. (2007) - Considerations on the Design and Implementation of an Agent-Based Auction Service, First International Symposium on Intelligent and Distributed Computing, IDC'2007, Craiova, Romania, 2007. *Studies in Computational Intelligence* 78, Springer, 75-85

13. Dobriceanu A., Biscu L., Badica C. (2007) - Adding a Declarative Representation of Negotiation Mechanisms to an Agent-Based Negotiation Service, *Proc. 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, Silicon Valley, USA. IEEE Comp. Soc. Press, 471-474;

14. Dogaru I., Badica A., Badica C. (2006) - Conceptual Architecture of a Multi-Agent System for News Syndication**.** *Proc. 8th Int. Symp. on Symbolic and Numeric Algorithms for Scientific Computing*, Timisoara, Romania. IEEE Com. Soc. Press, 201-204;

15. Esterline, A., Rorie, T., and Homaifar, A.: A Process-Algebraic Agent Abstraction. In: Rouff, C.A.et al. (Eds.): *Agent Technology from a Formal Perspective*, NASA Monographs in Systems and Software Engineering, 88-137, Springer 2006;

16. Fasli M (2007) - *Agent Technology For E-Commerce*. Wiley;

17. Hristozova, M., Lister, K., and Sterling, L.: Middle-agents (2002) – towards theoretical standardization. In: I. J. Timm, M. Berger, S. Poslad, and S. Kirn (Eds.): *Proc. of the International Workshop on Multi-Agent Interoperability – MAI'02;*

18. Janssen W., Mateescu R., Mauw S., Fennema P., van der Stappen P (1999) -. Model Checking for Managers, *LNCS* 1680, Springer, 92-107;

19. Klusch. M., Sycara, K.P. (2001) - Brokering and matchmaking for coordination of agent societies: A survey. In Omicini A., Zambonelli F., Klusch M., Tolksdorf R. (Eds.): *Coordination of Internet Agents. Models, Technologies, and Applications*, 197–224, Springer;

20. Lomuscio A.R., Wooldridge M., Jennings N.R. (2003) - A Classification Scheme for Negotiation in Electronic Commerce *Group Decision and Negotiation*, Volume 12, Number 1, Springer, pp. 31-56(26);

21. Loon H.van (2007) - *Process Assessment and ISO/IEC 15504: A Reference Book,* Springer;

22. Merayo M.G., Nunez M., Rodriguez I. (2007) - Formal Specification of Multi-agent Systems by Using EUSMs, *LNCS* 4767, Springer 318–333;

23. Miller, T., McBurney, P. (2007) - Using Constraints and Process Algebra for Specification of First-Class Agent Interaction Protocols, *LNCS* 4457, Springer 245–254;

24. Ould M. A. (2005) - *Business Process Management. A Rigorous Approach*. Meghan-Kiffer Press, USA;

25. Phalp, K., Henderson, P., Abeysinghe, G. and Walters, R. J. RolEnact (1998) - Role Based Enactable Models of Business Processes. *Information And Software Technology*, 40 (3). pp. 123-133., Elsevier;

26. Pistol Gheorghe (2002) - *Tehniques and strategies of negotiations*. Usages and the protocol, Editura Universitara, Bucuresti;

27. Podorozhny R.M., Khurshid S., Perry D.E., Zhang X. (2007) - Verification of Multi-agent Negotiations Using the Alloy Analyzer. Proc. Integrated Formal Methods, 6th Int. Conf. IFM 2007, Oxford, UK, 2007. *LNCS 4591*, 501-517, Springer;

28. Thompson Leigh (2006) - *Mind and heart of the negotiator. Negotiation complete handbook*, Ed. Meteor Press, Bucuresti;

29. Wurman, P.R., Wellman, M.P., Walsh, W.E. (2001) - A Parameterization of the Auction Design Space, *Games and Economic Behavior*, 35, Vol. 1/2, 271–303, Elsevier.