A Multi-agent Framework for Group Decision Support System: Application to a Boiler Combustion Management System (GLZ)

Noria Taghezout[1], Abdelkader Adla[1,2], and Pascale Zaraté[2]
[1] *Department of Computer Science, University of Es-Senia Oran, BP 1524, El-M' Naouer, 31000, Oran, Algeria*
[2]*IRIT – INPT – ENSIACET, 118 route de Narbonne 31062 Toulouse Cedex 9, France*
[1]*taghezoutnour@yahoo.fr,* [2]*{adla, Zarate}@irit.fr*

Abstract

Agents are designed to be autonomous problem-solvers, possibly communicating with other agents and users, and are therefore equipped with sufficient cognitive abilities to reason about a domain, make certain types of decisions themselves, and perform the associated actions. In this paper, we propose to integrate agents in a Cooperative Intelligent Decision Support System. The resulting system, called MACIDS is designed to support operators during contingencies. During the contingency, the operators using MACIDS should be able to: gather information about the incident location; access databases related to the incident; activate predictive modeling programs; support analyses of the operator, and monitor the progress of the situation and action execution. In MACIDS the communication support enhances communication and coordination capabilities of participants. A simple scenario is given, to illustrate the feasibility of the proposal.

1. Introduction

Successful Group DSS (GDSS) acts intelligently and cooperatively in a complex domain with potentially high data rates and makes judgements that model the very best human technicians[7]. It is crucial that human technicians maintain control over the final judgments, either by focusing the system on particular reasoning goals, or by modifying the basic knowledge on which the systems judgements rely. In this way, the intelligent GDSS is able to capture the domain knowledge and provide intelligent guidance during the process [1] [6] [9]. While the data and model data manipulations are done through the DSS, decision makers can focus solely on the process issues.

Due to the inadequate support from GDSS to model group members commitment to achieve a common goal, the incompleteness and rigidity of decisional models used, and the uncertainty carried out in meeting planning, it becomes inevitable that: 1) GDSS design is complicated enough to discourage wide spreading of the system as long as users are different in background, roles and interest [4] [5]; 2) group dynamics is difficult to understand and consequently to support in an adequate way; 3) group behaviour is not generalized to other groups being highly dependent by the context of use.

Fortunately, the multi-agent system (MAS) paradigm represents one of the most promising approaches to address such kinds of problems. It offers a new dimension for GDSS integration with complementary services making easier to build complex and flexible architectures suitable for organizational settings.

In this paper, we propose to model a group decision support system based on a multi-agent architecture. The use and the integration of software agents in the decision support systems provide an automated, cost-effective means for making decisions. The agents in the system autonomously plan and pursue their actions and sub-goals to cooperate, and Coordinate to respond flexibly and intelligently to dynamic and unpredictable situations.

We experiment our system on a case of boiler breakdown to detect a functioning defect of the boiler (GLZ : Gas Liquefying Zone) to diagnose the defect and to suggest one or several appropriate cure actions. Managing this process is a complex activity which involves a number of different sub-tasks: monitoring the process, diagnosing faults, and planning and carrying out maintenance when faults occur.

In this regard, this paper applies the multi-agent system paradigm to cooperative decision support in a global contingency management. Multi-agent computational environments are suitable for studying a broad class of coordination issues involving multiple autonomous or semiautonomous problem solving agents.

In this study, we examine the potential integration of agent technology into a Cooperative Intelligent Decision Support System. Taking into account that the MAS paradigm represents a feasible way to address some of the problems encountered in GDSS theory and practice, this paper is organized as follow: Once the context of our study specified. Section 2 describes our contribution. This followed by a section (section 3) that covers the integration of agent technology into a GDSS. Section 4 describes the application area. Then, the multi-agent architecture for group decision support systems and the corresponding coordination protocol are described in section 5. We also present an example of a scenario to illustrate the feasibility of the idea in section 6. Finally, we conclude with future research direction in section 7.

2. Contribution

In their research, Limayem et al. [8] identify two gaps from their review of past research on impact of GDSS in group decision settings: (1) There is little knowledge on how groups use a GDSS (the process) and thus, little is known about how? and why? facilitation impacts decision outcomes in group decisional settings with decisional guidance. (2) Due to the lack of understanding of the process of GDSS use, the appropriateness of the process used to arrive at decisions is not known. Thus, the objective of their work was to address these two gaps.

In our study we add another gap to the ones mentioned in [8]: the coordination problems when they occur have several causes. Most of them are a consequence of limitations in both the decision making processes and the technological support for communication. For this reason, the information and tasks related to the decisions made in GDDS have to be visible to other organizations to keep the relief effort coordinated between the agents.

In addition, the quality of support received during the decision making processes is the key to reaching optimal decisions. Decisional guidance mechanism provides the decision makers with step-by-step guidance throughout the decision-making process and allows them to evaluate more alternatives. As a result, DSS users with decisional guidance can easily come up with better decisions than those with no decisional guidance.

Mahoney et al. [12] pointed out that when faced with complexities in a decision situation, decisional guidance helps users to choose among and interact with a system's capabilities. They argued that in less structured tasks that deal with uncertainty and risk, users need more guidance to choose among competing solution techniques or among

alternative methods of processing information to structure an appropriate decision-making process using the GDSS.

We argue that using a single central controller for a large group of agents has an obvious problem. As the group size increases it becomes very difficult for the central controller to be informed of all the agents' beliefs and intentions. Also such a controller can become a severe communication bottleneck and would render the remaining components unusable if it fails. Agent technology provides a natural way of overcoming this problem. We use a coordinator agent to execute an important task. We are exploring ways of solving multi-agent plan coordination problems in a distributed manner.

3. Agent integration in GDSS

There is a wide range of existing application domains that are making use of the agent paradigm and develop agent-based systems, for example in software technology, robotics, and complex systems. Luke et al. [14] make a distinction between two main Multi-Agent System (MAS) paradigms: multi-agent decision systems and multi-agent simulation systems. In multi-agent decision systems, agents participating in the system must make joint decisions as a group. In this study we focus on the first paradigm and here in particular on the modelling of organizations and agent communication.

MAS are software systems composed of several autonomous software agents running in a distributed environment. Beside the local goals of each agent, global objectives are established committing all or some group of agents to their completion. Some advantages of this approach are: 1) it is a natural way for controlling the complexity of large and highly distributed systems; 2) it allows the construction of scalable systems since the addition of more agents become an easy task; 3) MAS are potentially more robust and fault-tolerant than centralised systems [15].

Several agent-based systems have been developed to support a smooth integration of software agents into human teams. For example, Miller et al. [17] developed a virtual environment for battle staff training using a knowledge-based approach to encode the roles of team members, as well as goals, capabilities, responsibilities, needs, situations, and activities of the entire team, sub-teams, and individuals in the team. To describe team structures (roles and responsibilities), teamwork process knowledge (e.g., work flows, team plans), collaborative decision making knowledge, communication strategies and protocols they use a logic-based representation language called MALLET. A complementary approach has been proposed in the ELEVES project formerly used to host a visiting researcher. The approach emphasizes the need to adjust the autonomy of agents when acting as proxies for the corresponding humans. Focusing on interaction aspects between agents and humans, COLLAGEN was used to build a collaborative interface agent for an air travel application [16].

We consider that agents are designed to be autonomous problem-solvers, possibly communicating with other agents and users, and are therefore equipped with sufficient cognitive abilities to reason about a domain, make certain types of decisions themselves, and perform the associated actions. Integrated in DSS in general and GDSS in particular, they offer the potential to automate a far wider part of the overall problem-solving task than was possible with classical DSS or Expert System DSS [11]. In this regard, this paper applies the multi-agent system paradigm to cooperative decision support in a global contingency management. The application details are given in the next section.

4. The Boiler Combustion Management System

Usually, in a situation of contingency (breakdown of a boiler), the exploiting engineer (the process administrator and the direct operator), tent to identify the breakdown, to analyze and diagnose it on the local site, to make contact with other exploiting engineers of the parent company and send for the technicians of the boilers constructor company, in general located abroad. This type of situation, compel the plant to work in degraded functioning if not to stop the process (case of shutdown alarm) waiting for the problem solving. Different sensors are set up to detect anomalies at different stages of the process. Breakdown can be automatically signposted by means of an alarm or intercepted by the exploiting engineers (case of defectiveness of the sensor where no alarm is triggered off but the boiler does not work). If there is a defect, an alarm will be triggered off. In case an alarm is signposted to the operator: the flag (the reference given to every alarm) is pointed out on the board (control room). It acquaints with an alarm and locates the defect. To solve this problem, diagnosis and actions of cure are generated by the system. Otherwise, a breakdown is directly raised by the operator (not triggered off alarm). This scenario occurs when a sensor defect doesn't allow to automatically signpost the breakdown. In this case, the operator must explore a large research space of potential defects with a series of tests. In both cases, the operator tries to solve the problem. Managing this process is a complex activity which involves a number of different sub-tasks: monitoring the process, diagnosing faults, and planning and carrying out maintenance when faults occur. In the next section, the most important agents of the system are detailed and their general overview is given in Figure 2.

5. The multi-agent Framework

Agents were integrated into the DSS for the purpose of automating more tasks for the user, enabling more indirect management, and requiring less direct manipulation of the DSS. Specifically, agents were used to collect information outside of the organisation and to generate decision-making alternatives that would allow the user to focus on solutions that were found to be significant [3]. A set of agents is integrated to the system and placed in the DSS components, according to our architecture of GDSS.

5.1. Integrating Agent in Cooperative Intelligent Decision Support System

The individual DSS, as shown in Figure 1. comprises a set of agents grouped in an agency. The agents in an agency are tightly coupled to the dominating agent (representing the decision maker). The dominating agent provides access to the world outside its agency. Different agents in an agency communicate with each other through messages. Incoming messages are selected by each agent based on the event selection mechanism such as first come first served (FCFS). The proposed architecture comprises:

The Interface Agent (IA) continuously receives data from the process – e.g. alarm messages about unusual events and status information about the process components. From this information, the IA periodically produces a snapshot which describes the entire system state at the current instant in time. It also performs a preliminary analysis on the data it receives from the process to determine whether there may be a fault.

A Decision Maker and Agent (DMA) performs most of the autonomous problem solving. It exhibits a higher level of sophistication and complexity than other agents. A DMA: (1) receives user delegated task specifications from an IA, (2) interprets the

specifications and extracts problem solving goals, (3) forms plans to satisfy these goals, (4) identifies information seeking sub-goals that are present in its plans, (4) decomposes the plans and coordinates with appropriate Information Retrieval Agent (IRA), Modelling Agent (MA), Diagnosis Agent (DA) and Action Agent (AA) for plan execution, monitoring, and results composition.

DMA has the following knowledge: 1) knowledge for performing the task (e.g. query decomposition, sequencing of task steps), 2) information gathering needs associated with the task model, 3) knowledge about relevant information, modelling, diagnosis, and action agents that it must coordinate with in support of its particular task, 4) **coordination rules** that enable coordination with the other relevant agents.

An Information Retrieval Agent (IRA) primarily provides intelligent information services. The simpler of these services is a shot retrieval of information in response to a query: A more enhanced information service is constant monitoring of available database for the occurrence of predefined information patterns. An even more advanced information agent can, in addition to communication with other agents, monitor its data base for the appearance of particular patterns. A typical information specific agent knows: 1) model and associated meta-level information of the databases that it is associated with, such size, average time it takes to answer a query, 2) procedures for accessing databases, 3) conflict resolution and information fusion strategies, and 4) protocols for coordination with other relevant software agents.

A Modelling Agent (MA) anticipates the occurrence of contingencies using mathematical and computational models. It integrates data from different sources with mathematical and computational models that model the contingency in order to predict its behaviour and consequences.

Knowledge management agent (KMA) comprises, manage and update knowledge base;

A Diagnosis Agent (DA) is activated by the receipt of information from DMA which indicates that there might be a fault. It uses IA snapshot information and KMA knowledge to update its knowledge model of the process on which its diagnosis is based. It pinpoints the approximate region of the fault then it generates and verifies the cause of the fault in the process.

The Action Agent (AA) generates a plan of action which can be used to repair the process once the cause and location of the fault have been determined. As described in Figure 3. a refined representation of DA, AA and KMA agents is given.

5.2. Agents structure

Clearly, all the modules representing the inner structure of an agent may depend on each other. This is especially true for the local problem solver and the coordination modules (as shown in Figure 3) which do not only exchange real time information but, in addition, must coordinate their decision rules and performance criteria. If we consider the relationship between the coordination module, the problem solver, and the knowledge base. We found that they have to make sure the data needed available. The communication between the agents may roughly be described by the coordination module and the interface component. They are describing the way how agents may communicate

5.3. A coordination protocol

The problem solving mechanism is based on a set of cycles until the entire problem is solved. Each cycle consists of the following steps:

1) identifying candidate methods;
2) identifying triggered methods;
3) selecting a method;
4) assigning the method to an agent;
5) executing the method; and
6) Evaluating the task state.

Before launching the problem solving process (diagnosis and actions of repair), The DMA Agent ask for help, it calls the coordinator agent which is created at the same time. The coordination protocol provides the rules for an information exchange regarding the coordination task while the communication protocol (e.g. interface) together with its management. In fact, the rules applied in the coordination module concern the coordination itself and the way how the data are made available.

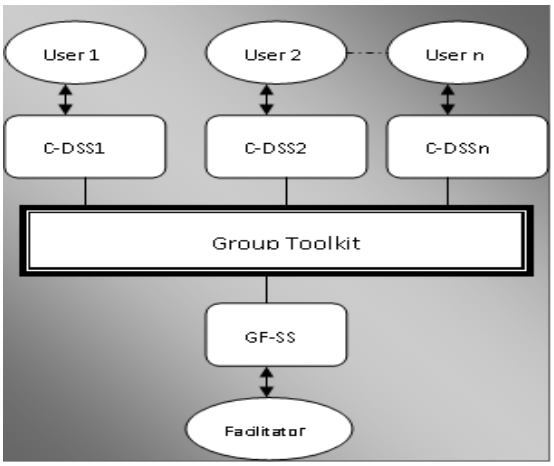

Figure 1. The group decision support system architecture

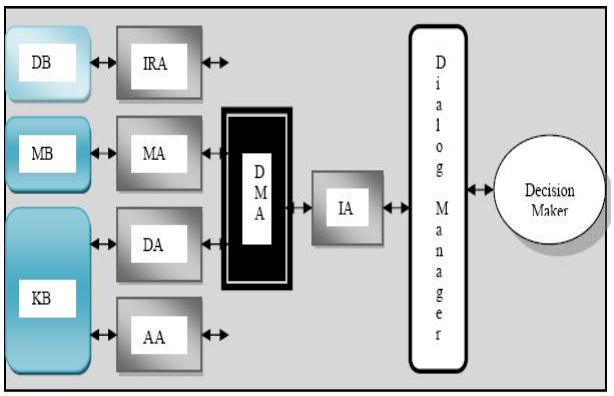

Figure 2. Agent architecture for individual DSS

5.3.1. The coordinator agent: A coordinator agent in our system exchanges plan information with task agents to help them coordinate their actions. The coordinator agent

provides two services to task agents: (i) it computes summary information for hierarchical plans submitted by the task agents, and, (ii) coordinates hierarchical plans using summary information. At the start of each episode, the coordinator waits for coordination requests from task agents for a specified interval of time. After the request phase has timed out, the coordinator examines the goals and plans of the participating agents. As discussed in the previous section, the coordinator may either coordinate the plans of the agents, or inform the agents that they can use a previously coordinated plan if the context is appropriate.

As described in Figure 4. The coordinator agent includes several types of functional modules such as: the task generation module, configuration module, a database, and an interface and coordination module. The generating task module is the core of this architecture; its role is to break up a complex problem into sub-problems. Through its participation, it offers a valuable assistance to the DMA agent. It reduces its function of handling a problem which has occurred during the management of the GLZ oil plant. The coordinator agent analyzes the input events and assigns tasks to agents in order to solve the events.

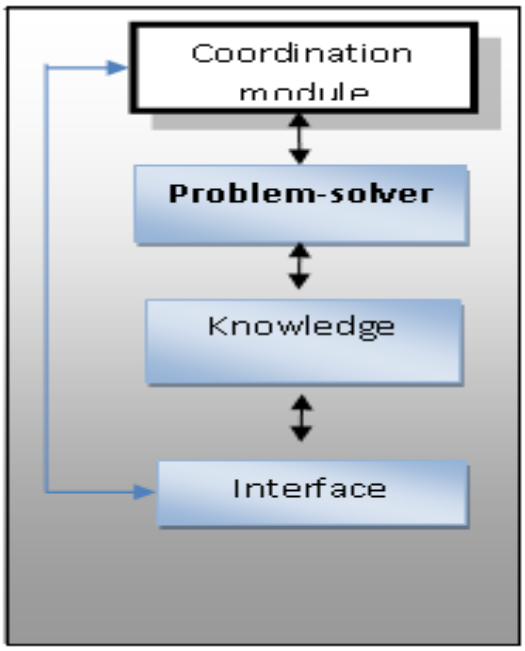

Figure 3. Representation of DA, AA, and KMA agents

The configuration module ensures the management of multiple coordination episodes (steps) and synchronizes the various obtained results. The interface module manages the information exchanges between the agent coordinator and the other agents. Finally, the coordination module contains the rules. We separated the inter-agent coordination part from the problem solving module (task generator module).

5.3.2. Communication between Agents: The requests structure presented above is also used by agents in their communications. Indeed, these requests formalize the contents of the messages exchanged by agents, each of which is structured as follows: ***m* = [id, *C*,**

sender, receiver, agenda]. Where id is the message id, *C* is the message content, sender and receiver are the sender and receiver agents' ids and the agenda term for indicating the memory of the message associated to the initial agent request.

6. A Coordination Scenario

When the task management agent (DMA) receives a task from an interface agent (IA), it decomposes the task based on the domain knowledge it has and then delegates the primitive tasks to the other agents (IRA, MA, KMA, DA or AA). The task management agent will take responsibility for retrieving data, modelling, diagnosing fault, planning action, resolving conflicts, coordinating among the related agents and finally reporting to the interface agent which conveys the results to the user. The task management agent first gets input data through the interface agent.

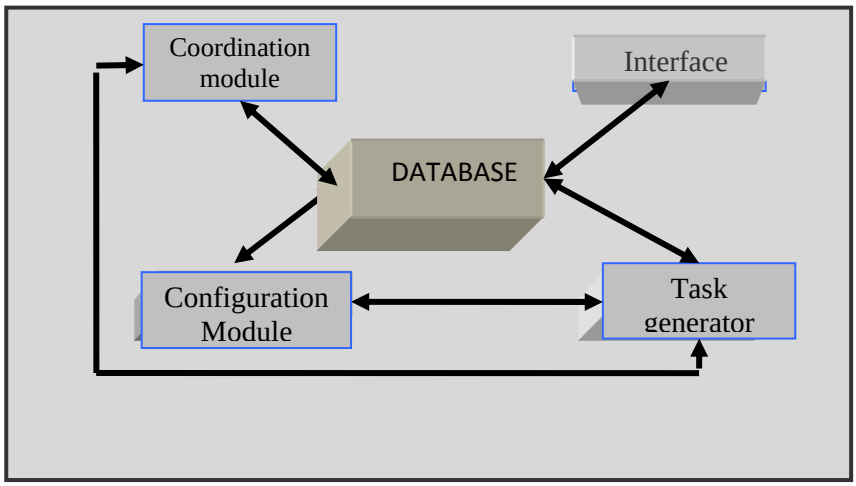

Figure 4. The coordinator agent structure

Next, the modelling agent searches for rules to select a suitable model and to execute the model to get analytical results. Additionally, all the parameters values needed by the models are retrieved from the database via the information retrieval agent. After finishing model analysis, the diagnosing and the action agents use the results of the model analysis to identify the fault causes and to perform a suggested action plan. Of course, sometimes, the diagnosis and the action agents may independently infer knowledge rules without using any model.

Different methods to achieve a task can be envisaged. Given a task, the system can then choose a method dynamically to achieve it. In order to do that, given the name of the task to be solve (wording of problem), the system constructs an action plan to be carried out (a sub-graph of tasks-methods hierarchy). To this end, first candidate actions are proposed. Next, these candidates are checked upon feasibility and relevance. Finally from the approved actions a repair plan is prepared. The execution of this plan (guided by the human operator) is monitored cooperatively by IA, which groups any alarm messages coming from the process, and DA which checks that PA's predictions about the various intermediate states of its recover plan are in fact reflected in the real process.

To validate the feasibility of the framework, a prototype system is built upon Java Agent Development (JADE) Platform. The JADE architecture enables agent

communication through message exchange based on the agent communication language (ACL). These agents work collectively to achieve the common goal by communicating and cooperating toward a better process. To illustrate this, the interface of an action agent (AA) is given in Figure 5.

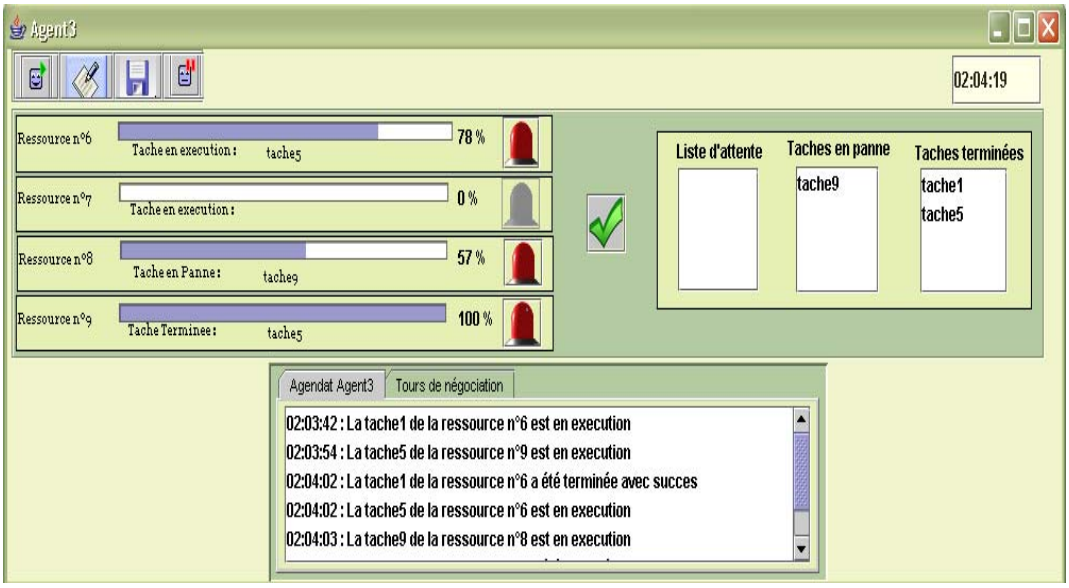

Figure 5. Action Agent interface

7. Conclusions

In the last decade the technologies used to solve complex problems has shifted from developing large and integrated software systems, to delivering small, autonomous and heterogeneous software components that can interact with humans, with other software components, and different services or data [15]. Multi agent system (MAS) paradigm represents the most natural approaches to address complex problems.

MAS may be employed in many different ways. They can be used in a purely structural way, just in splitting up a complex decision problem into simpler tasks, or they can be employed in supporting decision making in team-like systems having private information or even in principal agent settings and in negotiations[2].

In this paper, we have integrated agents into GDSS for the purpose of automating more tasks for the decision maker, enabling more indirect management, and requiring less direct manipulation of the DSS. In particular, agents were used to collect information and generate alternatives that would allow the user to focus on solutions found to be significant. Based on this, and considering that communication capabilities play an essential role in GDSS to enable ‘any-time, any-place” operation mode of the system. Further work based on coordination protocols between agents needs to be done. Particularly, the context information domain included in the software tool will be extended in order to improve the support for decision making and the coordination activities. We intend to present an efficient algorithm for multi-agent coordination based on the work presented by Cox et al. [13] in another paper. And Finally, as stated in [10]

Web technology will be ever more considered in DSS, thereby people will make co-decisions in ‘‘virtual teams’’, no matter where they are temporarily located, thus we intend to integrate Web services at the design level so that we can conduct research on the decision and collaboration behaviors of geographically dispersed teams.

References

[1] A. Adla, J-L. Soubie, “A distributed architecture for cooperative decision support systems”, Proc. of EuroWorgroup Workshop on decision support systems, London, England, 2006.
[2]C.Schneeweiss, Distributed Decision Making, Springer – Verlag Berlin. Heidelberg, 2003
[3] E. Jennings, “Using intelligent agents to manage business processes”. Proc. of the 1st international conference on practical applications of intelligent agents and multi-agent technology (PAAM96), B. Crabtree and N. R. Jennings editors, pp. 345 - 360.
[4] A. Adla, P. Zarate, “A cooperative intelligent decision support system”, Proc. of International Conference on service systems and service management, Troyes, France, 2006, October 25-27.
[5] A. Adla, J-L, Soubie, and P. Zarate, “A Co-operative Intelligent Decision Support System for Boilers Combustion Management based on a Distributed Architecture”, Journal of Decision Systems, Lavoisier, 2007, Vol. 16, pp. 241-263.
[6] M.Chen, “Team Spirit: Design, implementation, and evaluation of Web-based group decision support system”, Decision Support Systems, Lavoisier B.V. 2002, 43. pp. 1186-1202.
[7] W. Cheung, “An Intelligent decision support system for service network planning” , Decision Support Systems, Lavoisier, 2005, Vol. 39, pp. 415- 428.
[8] M. Limayem, P. Banerjee, and L. Ma, “Impact of GDSS: opening the black box”, Decision Support Systems, Lavoisier, 2006, pp. 945- 957.
[9] E. Turban, and J. Aronson, Decision support systems and intelligent systems, Prentice-Hall International, Upper Saddle River, New Jersey, 2001.
[10] F.G, Filip, “Decision support and control for large-scale complex systems”, Annu Rev Control, doi: 10.1016/j.arcontrol.03.002, 2008.
[11] J. Forth, K. Statis, and F. Toni, “Decision Making with a KGP Agent System”, Journal of Decision Systems, Lavoisier, 2006, vol. 15, pp. 241- 266.
[12] L.S. Mahoney, P.B. Roush, D. Bandy, “An investigation of the effect of decisional guidance and cognitive ability on decision making involving uncertainty data”, Information and Organization 13 (2), 2003 , pp. 85–110.
[13] J.S. Cox and H.Edmund, Durfee, “An Efficient Algorithm for Multiagent Plan Coordination”, Proc. Of AAMAS’05, Utrecht, Netherlands, July 25-29, 2005, pp. 828-835.
[14] M. Luck, McBurney, P., Shehory, O., & Willmott, S. “Agent technology: computing as interaction” (a roadmap for agent based computing). Liverpool, UK: AgentLink, 2005.
[15] C. Zamfiresecun. “An Agent – Oriented Approach for Supporting Self Facilitation in Group Decisions”, Studies in Informatics and Control, Vol. 12, No.2, June 2003, pp.137-148.
[16] A. Garland, N. Lesh, “Learning Task Models for Collagen”, Proc. Of AAAI Fall Systems on learning How to do things, 2000, pp. 22- 29.
[17] M. Miller, J. Yin and J. Yen, “Training teams with collaborative agents”, Proc. Of Fifth International Conference on Intelligent Tutoring Systems, 2000, pp. 63-72.

Authors

Noria Taghezout is a Doctor at the Department of Computer Sciences at Oran University, Algeria. She received a Doctorate Thesis in Artificial Intelligence and Simulation in 2008, a Master's Degree in 1998 in Computer Sciences and Automatics from Oran University, Algeria, and an Engineering Degree in Industrial Informatic 1991 from the same department. She is also with IRIT – Paul Sabatier University, Toulouse, France. She teaches human interface interaction, multi-agents systems and decision making in manufacturing systems. Her research interests include manufacturing system design, user interface, artificial intelligence, multi-agent systems and manufacturing control systems.

Abdelkader Adla is an assistant-Professor in Computer Science Department at University of Oran, Algeria. He is also with IRIT – Paul Sabatier University, Toulouse, France. He received a State Doctorate Thesis in Computer-Aided Design and Simulation from University of Oran in 2007. He has published papers on cooperative decision support systems and distributed group decision support systems. His research interests focus on Group DSS, facilitation and cooperative and collaborative systems.

Pascale Zaraté is assistant professor at Toulouse University (INPT). She conducts her researches at the IRIT laboratory. She holds a Ph.D. in Computer Sciences/Decision Support from the LAMSADE laboratory at the Paris Dauphine University, Paris (1991). She also holds a Master degree in Computer Science from the Paul Sabatier University, Toulouse, France (1986); as well as a Bachelors degree Toulouse, France (1982). Pascale Zaraté's current research interests include: distributed and asynchronous decision making processes; knowledge modeling; cooperative knowledge based systems; cooperative decision making. Since she obtained her PhD degree, she edited a book, 9 special issues in several international journals and 4 proceedings of workshops.

.