INTEGRATING AGENTS INTO COOPERATIVE INTELLIGENT DECISION SUPPORT SYSTEMS

Abdelkader Adla
IRIT, Paul SABATIER University, 118 Route de Narbonne, Toulouse, France
Department of Computing, Oran University, Algeria

Keywords: Decision Support Systems, Cooperative Intelligent DSS, Agent-based DSS.

Abstract: In this paper, we propose to integrate agents in a cooperative intelligent decision support system. The resulting system, ACIDS (Agent-based Cooperative Intelligent Decision-support System) is a decision support system designed to support operators during contingencies by giving them detailed, real-time information, allowing them to integrate and interpret it and then transmit and monitor their decisions through the chain of incident command. During the contingency, the operator using the ACIDS should be able to: gather information about the incident location; access databases related to the incident; activate predictive modelling programs; support analyses of the operator, and monitor the progress of the situation and action execution. The decision making process, applied to the boilers management system, relies in ACIDS on a cycle that includes recognition of the causes of a fault (diagnosis), plan actions to solve the incidences and, execution of the selected actions.

1 INTRODUCTION

Decision Support Systems (DSS) were designed to resolve ill or non-structured decision problems. Problems where priorities, judgements, intuitions and experience of the decision-maker are essential, where the sequence of operations such as searching for a solution, formalization and structuring of problem is not beforehand known, when criteria for the decision making are numerous, in conflict or hard dependent on the perception of the user and where resolution must be acquired at restricted time.

Successful cooperative intelligent DSS and their subsystems act intelligently and cooperatively in a complex domain with potentially high data rates and make judgements that model the very best human technicians. It is also 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 cooperative intelligent DSS is able to capture the domain knowledge and provide intelligent guidance during the process. While the data and model manipulations are done through the DSS, decision makers can focus solely on the process issues.

In this paper, we propose a cooperative intelligent decision support system based on a multi-agent architecture. The use and the integration of software agents in the decision support systems provides an automated, cost-effective means for making decisions

The rest of the paper is organized as follows: First we present a literature review of some related work in section 2. Then we propose a multi-agent architecture for cooperative intelligent decision support systems in section 3. We also present an example application to illustrate the feasibility of the idea in section. Finally, we conclude with a summary and future research direction in section 4.

2 LITERATURE REVIEW

2.1 Decision Support

Decision support systems (DSS) are computer-based systems designed to support and enhance managerial decision making. Since 1970s, the field has evolved

Adla A. (2007).
INTEGRATING AGENTS INTO COOPERATIVE INTELLIGENT DECISION SUPPORT SYSTEMS.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - AIDSS*, pages 440-446
DOI: 10.5220/0002388904400446
Copyright © SciTePress

from the disciplines of management science and management information systems.

It has come to include personal decision support systems, group decision support systems, negotiation support systems, intelligent decision support systems, knowledge management based DSS, executive information systems/business intelligence systems, and data warehouses (Power, 2000).

According to Turban and Aronson (1998), the central purpose to a DSS is to support and improve decision making. Zarate (2005) defines DSS as a "model-based set of procedures for processing data and judgements to assist a manager in his decision making". He argues that to be successful such a system needs to be adaptive, easy to use, robust and complete on important issues. These features are desired but not required in a DSS. Holtzman (1989) defines a DSS as a computer-based system consisting of three interacting components: a language system, a knowledge system and a problem processing system. This definition covers both old and new DSS designs, as the problem processing system could be a model-base or an ES or an agent-based system or some other system providing problem manipulation capabilities.

While each type of DSS varies in their technologies, their common purpose is to aid human judgement in decision making. A DSS might achieve this through advanced capabilities in information storage and retrieval. Using mathematical modelling techniques, a DSS may also provide forecasting capabilities, including calculations of the best solutions to "what-if" scenarios.

A number of frameworks or typologies have been proposed for organizing our knowledge about decision support systems (Power, 2000). The two most widely implemented approaches for delivering decision-support are Data-Driven and Model-Driven DSS. Data-Driven DSS help managers organize, retrieve, and synthesize large volumes of relevant data using database queries, OLAP techniques, and data mining tools. Model-Driven DSS use formal representations of decision models and provide analytical support using the tools of decision analysis, optimization, stochastic modelling, simulation, statistics, and logic modelling. Three other approaches have become more wide spread and sophisticated because of collaboration and web technologies: Communication-Driven DSS rely on electronic communication technologies to link multiple decision makers who might be separated in space or time, or to link decision makers with relevant information and tools. Knowledge-Driven DSS can suggest or recommend actions to managers. Finally, Document-Driven DSS integrate a variety of storage and processing technologies to provide managers document retrieval and analysis. Classic standalone DSS tool design comprises components for: (1) database management capabilities with access to internal and external data, information and knowledge; (2) powerful modelling function accessed by a model management system; and (3) user interface design that enable interactive queries, reporting and graphic functions.

2.2 Intelligent Decision Support Systems

Intelligent decision support systems (IDSSs) are interactive computer-based systems that use data, expert knowledge and models for supporting decision makers in organizations to solve complex, imprecise and ill-structured problems by incorporating artificial intelligence techniques. They draw on ideas from diverse disciplines such decision analysis, artificial intelligence, knowledge-based systems and systems engineering. In general, the need for IDSS derives from: (i) the growing need for relevant and effective decision support to deal with a dynamic, uncertain and increasingly complex management environment, (ii) the need to build context-tailored, not general purpose systems, and (iii) standard support technology is becoming obsolete as a way to improve decision quality and work productivity (Ribeiro et al., 2006).

Intelligent decision support systems (IDSSs) use Expert Systems (ES) technology to enhance the capabilities of decisions makers in understanding a decision problem and selecting a sound alternative. Because of the people-centred focus of such technologies, it is important not only to assess their technical aspects and overall performance but also to seek the views of potential users. Turban and Aronson (2001) suggested two fundamental ES/DSS integration models: (1) ES is integrated into DSS components, and (2) ES is a separate component in the DSS. In (Power, 2000), the second model is used, where the DSS is responsible for both data and model manipulation, while the ES provides domain knowledge and recommends resolutions during the planning the process. The proposed architecture signifies the integration of a DSS and an ES. During the process, data and models are manipulated through the DBMS and model base management system (MBMS), respectively. Instructions for data modifications and model execution may come from the ES interface directly. The MBMS obtains the relevant input data for model executions from the

DBMS and, in return, results generated from model executions are sent back to DBMS for storage. The database also provides facts for the ES as part of the knowledge base. Using these facts together with the predefined rules, the inference engine of the ES performs model validations and planning evaluations, according to what a domain expert is supposed to do. Conclusions and recommendations are then passed to the interface, where they are displayed to the decision maker and transferred to the MBMS and DBMS, in the form of procedures calls for various actions.

However, making a simple machine act intelligently may be much less useful or important than being able to cooperate in an environment with the users. It is found that “lack of attention to the human and organizational aspects of IT is a major explanatory factor and is manifest in a failure to involve users appropriately (Guerlain, 2000)

Indeed, despite their impressive functionalities, DSS of all of types are focused on supporting, not replacing, a human decision maker for important decision tasks, as many of the problem situations faced by managers are unstructured in nature and require the use of reasoning and human judgement. Therefore, as articulated by Lévin and Pomerol (1995), “the DSS and the decision maker form a united problem solver”. In DSS, the user is defined by physical and purposeful interaction with the system. Therefore, a system might have one, or many users, each interacting with the system in different ways, for different purposes, and with varying frequencies. As Keen (1981) stated, decision support systems “support, rather than replace, judgement in that they do not automate the decision process nor impose a sequence of analysis on the user”. Therefore, judgement and decision making must occur throughout the entire problem solving process, that is, during the user’s physical interaction with the system, and as the final human decision is being made. Because of this, the user’s decision processes must be factored into the design process of successful cooperative intelligent decision support systems.

Some other advantages proposed by Marakas (2003) gives the advantages of using intelligent components with DSSs as opposed to plain DSSs as increased timeliness in making decisions, improved consistency in decisions, improved explanations and justifications for specific recommendations, improved management of uncertainty, and formalisation of organisational knowledge. The most useful of these advantages is the improved explanations and justifications which is an extremely useful feature particularly in the fields like medicine, etc. where it helps if the real expert can validate the machine reasoning.

2.3 Multi-Agent Systems

Using agent provides a means of modelling the various information flows and interactions with the user and within the environment. The working definition of an agent is adapted from (Jennings, 1996): an agent is an artificial, computational entity that can perform certain tasks with a certain degree of autonomy or initiative whilst intelligently adapting to its environment. Note that a human is not an agent in this definition.

The definition of multi-agent systems (MAS) is well known and accepted as a loosely coupled network of agents that work together to find answers to problems that are beyond the individual capabilities or knowledge of each agent and there is no global control system. An agent’s architecture is a particular design or methodology for constructing an agent. Wooldridge and Jennings refer to an agent’s architecture as a software engineering model of an agent (Jennings, 1996). Using these guidelines, agent architecture is a collection of software modules that implement the desired features of an agent in accordance with a theory of agency. This collection of software modules enable the agent to reason about or select actions and react to changes in its environment.

A broad range of architectures for agents (including reactive, deliberative…) have been studied. Properties that distinguish the version agent architectures include reasoning capabilities, resource limitations, control flow, knowledge handling, autonomy, user interaction, temporal context, and decision making.

Integrated architectures that include both deliberative or classical planning and reactive components can support a more autonomous agent than either architecture can support alone. An integrated architecture provides a mechanism that enables an agent to both deliberate and respond efficiently to exogenous events.

3 AGENT-BASED COOPERATIVE INTELLIGENT DSS

3.1 Cooperative Intelligent DSS

To cooperate, particularly means distributing tasks to be carried out among both the system and the user. Sharing tasks is a condition to implement cooperation between the two agents. The task that is the subject of cooperation is decomposed in consistent subsets. The decision to be made is modelled in tasks and sub-tasks as well as the associated methods to achieve them. This modelling is based on a hierarchy (tree) of tasks and sub-tasks introducing then a relation of order between the different tasks to be achieved.

The task distribution between the system and the user is dynamically made, according to the performances of the couple man/machine and of the workload of the user. Competences of the user and the system are sometimes complementary, sometimes “redundant”. In the latter case, user and system are often able to play the same role. The choice question of the appropriate agent which will have to play one role settles therefore. According to the context, different indications could be made to direct this choice. The set of indications on the manner to allocate different roles to the agents defines the cooperation modes.

For the implementation of the system/user cooperation, we use a structure based principally on conceptual models of expertise: Domain Conceptual Model Task Conceptual Model (of the application for the system, but also of the users). The proposed architecture (Adla, 2006) for the design of a cooperative intelligent decision support system extends that of Soubie (1998) developed for cooperative knowledge-based systems.

The proposed architecture is composed of the following main components:

Data Base Management System (DBMS): mainly contains a relational database which is managed by a software program called the database management system, and which provides speed data retrieval, updating, and appending. The data in a DSS database are usually extracts or copies of operational databases, so using a DSS does not interfere with critical operation systems.

Model Base Management System (MBMS): The model base subsystem includes many statistical, management scientific models, or other quantitative models that offer the system’s analytical or forecasting capability to solve future outcomes. There are many types of models: Statistical models generally contain the full range of expected statistical functions including means, medians, deviations and scatter plots, Optimization models, such linear programming and dynamic programming, are often adopted to determine the optimal resource allocation to maximize or minimize an objective function.

Knowledge Base Management System (KBMS): it can support any of the other subsystems or play an independent role. It suggests alternatives or actions to decision makers. Additionally, it can be inter-connected with the knowledge base.

Task Management Tool (Control): This tool has as objective to offer resolutions or parties of resolutions to the users. It insures the task decomposition in sub-tasks as well as the assignment of roles to the system or to the user. This planning tool allows collaboration between the machine and the user, and assigns the tasks beforehand modelled to them. Particularly, this planning tool is able to manage this task allocation in a dynamic way and, to change planning initially implemented according to the controls of the different tasks. This tool constitutes essential provision in cooperation management. The man-machine cooperation is possible only if this task management tool allows a quick re-planning as well as a re-counting of allocations if there is a context modification or an evolution of the problem. This approach insures the dynamic character of the tool.

3.2 Integrating Agents

3.2.1 The Boilers Management System

The management system of the boiler combustion is one of the most critical systems for the good functioning of the plant and has a high impact on the methods of cogitation and apprehension of various problems related to maintenance. The exploiting staff is often confronted with situations that impose a quick reaction of decision-making. This requires consequent human and material resources and adapted skills. We experiment our system on a case of boiler breakdown to detect a functioning defect of the boiler, to diagnose the defect and to suggest one or several appropriate cure actions.

Usually, in a situation of contingency (breakdown of a boiler), the exploiting engineers (the process administrator and the direct operators), tent to identify the breakdown, to analyse 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 by using the Agent-based Cooperative Intelligent DSS (ACIDS). 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.

3.2.2 The Multi-agent Architecture

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 de significant.

A set of agents is integrated to the system and placed in the DSS components, according to our architecture of Cooperative Intelligent DSS (figure 1). Note that agents have placed in each of the components.

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. Interface agents have the following knowledge: User models and knowledge of what must be displayed to the user and in what way. User models could be interactively updated. The main functions of an interface agent include; 1) collecting relevant information from the user to initiate a task, 2) presenting relevant information including results and explanations, 3) asking the user for additional information during problem solving, and 4) asking for user confirmation, when necessary. From the user's viewpoint, interacting only through a relevant interface agent for a task hides the underlying information gathering and problem solving complexity.

A Task Management Agent (TMA) performs most of the autonomous problem solving. It exhibits a higher level of sophistication and complexity than other agents.

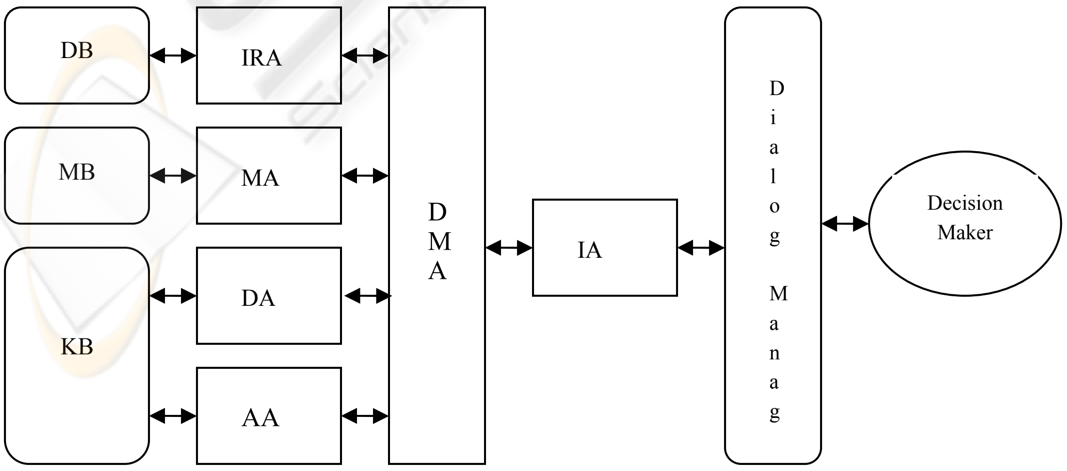

Figure 1: The ACIDS multi-agent architecture.

A TMA (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, (5) 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. A TMA 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, 5) protocols 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.

A Diagnosis Agent (DA) is activated by the receipt of information from TMA which indicates that there might be a fault. It uses IA snapshot information 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.

The DA (respectively AA) takes as input a set of goals: faults (respectively causes) and produces a plan that satisfies the goals.

3.3 Task Resolution

When the task management agent (TMA) 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, 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. 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.

Obviously, one of the major issues involved in multi-agent systems is the problem of interoperability and communication between the agents. In our framework, we use the KQML language for inter-agent communication. Agents communicate through messages. An agent transfers its request to one or more agents and receives the information requested as a result through messages. When an action is performed, the related information is also transferred by the IA to the decision maker as messages.

4 CONCLUSIONS

The multi-agent system paradigm represents one of the most promising approaches to address decision making problems. We have integrated agents into 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 and generate alternatives that would allow the user to focus on solutions that were found to be significant.

The proposed architecture is under implementation and experimentation on the boilers management system. The next phase in our research could be the validation of the prototype and testing its value par practitioners.

Proposal for future research is to integrate this architecture in a distributed one for cooperative intelligent decision system where several decision makers are geographically dispersed and work to reach a common decision.

REFERENCES

Adla, A., 2006. Système coopératif d'aide à la décision, *IHM'06*, Montréal, Canada.

Adla, A., Soubie, J-L., Zarate, P., 2006. A distributed architecture for cooperative decision support systems. *EuroWorgroup Workshop on decision support systems*, London, England.

Arnott, D., Pervan, G. 2005. A critical analysis of decision support systems research. *Journal of Information Technology*, *20(2), pp67-87*.

Courtney, J.F., 2001. Decision making and knowledge management in inquiring organizations: toward a new decision making paradigm for DSS. *Decision Support Systems 31/1, pp17-38.*

DeSanctis, G., Gallup, B., 1997. A foundation for the study of group decision support systems. *Management Science, Vol. 13 No 12, pp1589-1609.*

Gachet, A., 2003. A Software Framework for Deloping Disributed Cooperative Decision Support Systems. *Inaugural Dissertation. University of Fribourg, Switzerland.*

Guerlain, G., 2000. Interactive advisory systems. *In Proceedings of HPSAA Conference*, Savannah, GA.

Holtzman, S., 1989. *Intelligent decision systems.* Addison Wesley.

Jennings, E., 1996. Using intelligent agents to manage business processes. *In B. Crabtree and N. R. Jennings editors, Proceedings of the 1st international conference on practical applications of intelligent agents and multi-agent technology (PAAM96), pp345-360.*

Keen, P.G.W., 1981. Value analysis: Justifying Decision Support Systems. *MIS Quaterly, 5(1), pp15-25.*

Lévine, P., Pomerol, J-C., 1995. The role of decision maker in DSSs and representation levels. *In Proceedings of Hawaii International Conference on System sciences, pp42-51.*

Marakas, G., 2003. *Decision support systems in the 21st century.* Prentice Hall.

Power D. J., 2000. Supporting Decision-Makers: An Expanded Framework, *http://dssresources.com, version 1.0, December 15.*

Ribeiro, R., 2006. Intelligent Decision Support Tool for Prioritizing Equipment Repairs in Critical/Disaster Situations. *In Proceedings of the workshop On Decision Support Systems.*

Soubie, J.L., 1998. Modelling in cooperative based systems. *In Proceedings of COOP'98, Cannes France, may.*

Turban, E., Aronson, J., 2001. *Decision support systems and intelligent systems.* Prentice-Hall International, Upper Saddle River, New Jersey.

Zarate, P., 2005. Des systèmes interactifs d'aide à la décision aux systèmes coopératifs d'aide à la décision: Contributions conceptuelles et fonctionnelles. *Mémoire HDR, INP de Toulouse*.