"A Study on Strategies of Multi Agent System for Decision Support System"

Paritosh Kumar Bansal

Research Scholar, CMJ University, Shillong, Meghalaya, India

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.

We are investigating techniques for developing distributed and adaptive collections of information agents that coordinate to retrieve, filter and fuse information relevant to the user, task and situation, as well as anticipate user's information needs. In our system of agents, information gathering is seamlessly integrated with decision support. The task for which particular information is requested of the agents does not remain in the user's head but it is explicitly represented and supported through agent collaboration. In this paper we present the distributed system architecture, agent collaboration interactions, and a reusable set of software components for structuring agents. The system architecture has three types of agents: Interface agents interact with the user receiving user specifications and delivering results. They acquire, model, and utilize user preferences to guide system coordination in support of the user's tasks. Task agents help users perform tasks by formulating problem solving plans and carrying out these plans through querying and exchanging information with other software agents. Information agents provide intelligent access to a heterogeneous collection of information sources. We have implemented this system framework and are developing collaborating agents in diverse complex real world tasks, such as organizational decision making, investment counseling, health care, and electronic commerce.

The Combination between Web services and software agents provides a promising computing paradigm for efficient service selection and integration of inter-organizational business processes. This paper proposes an agent-based Web DSS; the main contribution of our study is to provide an efficient tool that helps users find information resources available as an online service within Intranet. The decision-making is not only guided by the information provided by DSS but rather than the Web technology, the process is entirely based on communication between ISP Agents and Web agent. While negotiating compromises for conflict solving to share common resources, decision centers use Web service to conduct various complementary tasks. To illustrate the idea, a simple case study is given.

----------♦----------

INTRODUCTION

Successful Group DSS (GDSS) acts intelligently and cooperatively in a complex domain with potentially high data rates and makes judgments that model the very best human technicians. 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 judgments rely. In this way, the intelligent GDSS is able to capture the domain knowledge and provide intelligent guidance during the process. 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; 2) group dynamics is difficult to understand and consequently to support in an adequate way; 3) group behavior 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 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.

Current networking technology and the ready availability of vast amounts of data and information on the Internet-based Info sphere present great opportunities for bringing to decision makers and decision support systems more abundant and accurate information. The use of the Internet has accelerated at an unprecedented pace.

However, effective use of the Internet by humans or decision support machine systems has been hampered by some dominant characteristics of the Info sphere. First, information available from the net is unorganized, multi-modal, and distributed on server sites all over the world. Second, the number and variety of data sources and services is dramatically increasing every day. Furthermore, the availability, type and reliability of information services are constantly changing. Third, the same piece of information can be accessible from a variety of different information sources. Fourth, information is ambiguous and possibly erroneous due to the dynamic nature of the information sources and potential information updating and maintenance problems.

Therefore, information is becoming increasingly more difficult for a person or machine system to collect, filters, evaluate, and use in problem solving. As a result, the problem of locating information sources, accessing, filtering, and integrating information in support of decision making, as well as coordinating information retrieval and problem solving efforts of information sources and decision-making systems has become a very critical task.

The notion of Intelligent Software Agents has been proposed to address this Although a precise definition of an intelligent agent is still forthcoming, the current working notion is that Intelligent Software Agents are programs that act on behalf of their human users in order to perform laborious information gathering tasks, such as locating and accessing information from various on-line information sources, resolve inconsistencies in the retrieved information, filter away irrelevant or unwanted information, integrate information from heterogeneous information sources, and adapt over time to their human users' information needs and the shape of the Info sphere. Most current agent-oriented approaches have focused on what we call interface agents a single agent with simple knowledge and problem solving capabilities whose main task is information filtering to alleviate the user's cognitive overload7;8. Another type of agent is the Soft Bot 9, a single agent with general knowledge that performs a wide range of user-delegated information finding tasks. We believe that such centralized approaches have several limitations.

A single general agent would need an enormous amount of knowledge to be able to deal effectively with user information requests that cover a variety of tasks. In addition, a centralized system constitutes a processing bottleneck and a \single point of failure". Furthermore, unless the agent has beyond the state of the art learning capabilities, it would need considerable reprogramming to deal with the appearance of new agents and information sources in the environment. Finally, because of the complexity of the information finding and filtering task, and the large amount of information, the required processing would overwhelm a single agent.

Another proposed solution is to use multi-agent computer systems to access, filter, evaluate, and integrate this information 6;10. Such multi-agent systems can compartmentalize specialized task knowledge, organize them to avoid processing bottlenecks, and can be built expressly to deal with dynamic changes in the agent and information-source landscape. In addition, Multiple Intelligent Coordinating Agents are ideally suited to the predominant characteristics of the Info sphere, such as the heterogeneity of the information sources, the diversity of information gathering and problem solving tasks that the gathered information supports, and the presence of multiple users with related information needs. We therefore believe that a distributed approach is superior, and possibly the only one that would work for information gathering and coherent information fusion.

The context of multi-agent systems widens the notion of intelligent agent in at least two general ways. First, an agent's \user" that imparts goals to it and delegates tasks can be not only a human but also another agent. Second, an agent must have been designed with explicit mechanisms for communicating and interacting with other agents. Our notion is that such multi agent systems may comprise interface agents tied closely to an individual human's goals, task agents involved in the processes associated with arbitrary problem-solving tasks, and information agents that are closely tied to a source or sources of data.

Computer technology progress has led to widespread use of computerized support in various activities. Particularly, traditional decision support systems (DSS) focus on computerized support for making decision with respect to managerial problems.

There is an emerging and fast growing interest in computerized support systems in many other domains such as information retrieval support systems, research support systems, teaching and learning support systems, computerized medical support systems, knowledge management support systems, and many more. The recent development of the Web generates further momentum to the design and implementation of support systems.

Obviously enough, there is a strong trend for studying computerized support systems especially on Web platforms. Research on information retrieval support systems, research support systems, teaching and learning support systems, decision support systems, computerized medical support systems, and knowledge management support systems are just some of their representatives.

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. 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 centralized systems.

Several agent-based systems have been developed to support a smooth integration of software agents into human teams. For example, Miller et al. 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.

Decision Support System and Web Based Decision Support System Decision Support System

Before we start with detailed aspects of the issue, it is important to tackle the definition of decision support systems. Decision Support Systems can be defined as computer technology solutions that can be used to support complex decision making and problem solving. To account decision problems complexity and uncertainty, we understand the DSS as a set of computer-based tools that

provide decision maker with interactive capabilities. It aims to enhance his understanding and information basis about considered decision problem through usage of models and data processing. The latter, in turn, allows reaching decisions by combining personal judgment with information provided by these tools. The classic DSS tool design is comprised of the components for:

- Database management capabilities with access to internal and external data, information and knowledge;

- Powerful modelling functions accessed by a model management system; and

- User interfaces that enable interactive communication between the user and system.

Decision Support Systems (DSSs) are interactive computer-based systems intended to help decision makers utilize data and models to identify and solve problems and make decisions. The "system must aid a decision maker in solving UN programmed, unstructured (or 'semi structured') problems...the system must possess an interactive query facility, with a query language that ...is ...easy to learn and use". DSSs help managers/decision makers use and manipulate data, apply checklists and heuristics, and build and use mathematical models.

According to Turban, a DSS has four major characteristics: it incorporates both data and models; it is designed to assist managers in their decision processes in semi structured (or unstructured) tasks; it supports, rather than replaces, managerial judgment; and its objective is to improve the effectiveness of decisions, not the efficiency with which decisions are being made.

According to, decision support systems fall into five categories:

- Communications-Driven DSS – uses network and communications technologies to facilitate collaboration and communication;

- Data-Driven DSS – emphasizes access to and manipulation of a time-series of internal company data and sometimes external data;

- Document-Driven DSS – integrates a variety of storage and processing technologies to provide complete document retrieval and analysis;

- Knowledge-Driven - intended to suggest or recommend actions to managers. These DSSs are personal computer systems with specialized problem-solving expertise;

- Model-Driven DSS or Model-oriented DSS – emphasizes access to and manipulation of a model, e.g. statistical, financial, optimization and/or simulation. Simple statistical and analytical tools provide the most elementary level of functionality.

Web-based Decision Support System - Web used technologies are employed to improve the capacity of decision support systems through decision models, On-line Analysis Processing (OLAP) and data mining tools that allow "standardized" publishing and sharing of decision resources on the Internet. In a web-based decision support system, all decision support related operations are performed on a network server n order to benefit from platform independence, shorter learning curves for already familiar users with the Web tools and web navigation, lower software distribution costs, ease of performing system updates and “reusability” of decision modules and information on the Internet through standardized protocols and formats.

According to, the importance of using Web-based DSS originates from the growing amount of available information that should be identified, controlled and accessed remotely using web based tools to support reusability of integrated decision modules. Using such systems, an enterprise can create survey software, Web based forms, build document-driven DSS for requests and approvals. They help global enterprises manage and improve decision processes through improved efficiency, better process control, improved customer service, more flexible re-design, and streamlining and simplification of business processes. Using Web-based DSS, decision-makers can share open decision modules on the Internet using standardized protocols such as HTTP, and a standardized format like XML or DAML.

According to, Web-based systems are regarded as «platforms of choice” for delivering decision support while taking into account many technical, economic and social considerations. The migration towards web based DSS denotes a shift from DSS generators (that allow users to develop specific applications characterized by limited deployment, inflexibility) to integrated cross application orientations that emphasize the reuse of applications and components. By deploying Web capabilities, multiple knowledge bases and knowledge processing techniques can be used. The design of decision support systems has been affected by the availability of a wide range of web based tools, techniques and technologies. The use of web tools are reshaping the description of relations between information components and decision modules in a way that affects both the physical and logical design of the DSS, model visualization, sharability of decision modules and the development life cycle of DSS.

As a result, the underlying architecture for Web-based DSS has moved from mainframes, to client–server systems, to Web and network technology based distributed systems that enable the integration of large amounts of data and decisions support tools originating from heterogeneous multidisciplinary sources for the provision of value-added information using knowledge discovery and data mining tools.

Agent Engineering

As our point of departure in structuring an agent, we use the Task Control Architecture (TCA) 18 which we extend and specialize for real-time user interaction, information gathering, and decision support tasks in the Info sphere. The control constructs available in TCA are used to integrate, coordinate, and monitor planning and plan execution, and to incrementally improve the efficiency and robustness of the multi-agent information system. These control constructs are part of the reusable agent architecture. The overall architectural design of a TCA-based agent is shown in Figure 1.

The planning module takes as input a set of goals and produces a plan that satisfies the goals. The planning module of the task agents can be a full-edged planner, whereas the planning module of the interface agents and the information agents is much simpler consisting of retrieval and instantiation of plan templates. In our initial implementation, the information agent planning component is a simple plan retrieval mechanism that instantiates a new task structure for each goal. Thus it is extremely fast but lacks edibility. Every plan step has an (optional) execution deadline.

The key component of this architecture is a hierarchical representation of task/subtask relationships, on which we rely heavily in our information software agent architecture. This representation, called a task tree, has goals as non-terminal nodes, and executable actions and execution monitoring mechanisms at the leaves. Temporal constraints between nodes are used to schedule task planning and execution: actions are queued until their temporal constraints are satisfied. For example, a sequential-achievement constraint between two nodes implies that all actions associated under the first node must be handled before any of those under the second node; whereas a parallel-achievement constraint allows that the actions under the first node can be parallelly executed along with the actions under the second node. This combination of hierarchical task decomposition and temporal constraints form the agent's representation of plans. Either a first principle general planner or a plan retrieval component plus domain-specific plan fragments can be used to generate plans. We adopt the plan retrieval approach in our implementation because of efficiency considerations.

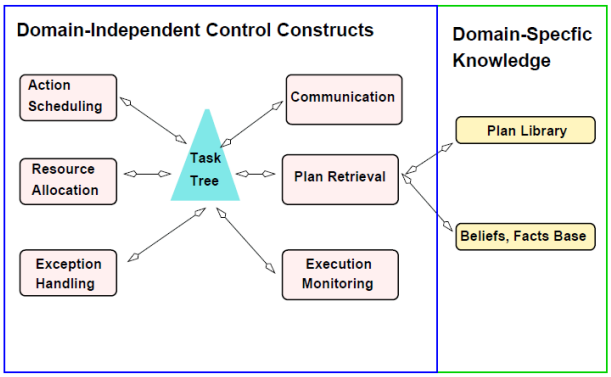

Figure 1: Agent Architecture

We have extended the original TCA architecture with a communication module that accepts and interprets messages from other agents in KQML. In addition, interface agents also accept and interpret e-mail messages. We have found that e-mail is a convenient medium of communicating with the user and/or other interface agents (e.g. agents that provide event notification services). Messages can contain request for services. These requests become goals of the recipient agent.

The scheduling module schedules each of the plan steps. The agent scheduling process in general takes as input the agent's current set of plan instances, in particular, the set of all executable actions, and decides which action, if any, is to be executed next. This action is then identified as a fixed intention until it is actually carried out (by the execution component). Whereas for task agents, scheduling can be very sophisticated, in our initial implementation of information agents, we use a simple earliest-deadline-first schedule execution heuristic.

Agent reactivity considerations are handled by the execution monitoring and exception handling processes. The agent execution monitoring process takes as input the agent's next intended action and prepares, monitors, and completes its execution. The execution monitor prepares an action for execution by setting up a context (including the results of previous actions, etc.) for the action. It monitors the action by optionally providing the associated computation limited resources e.g. the action may be allowed only a certain amount of time and if the action does not complete before that time is up, the computation is interrupted and the action is marked as having failed.

When an action is marked as failed, the exception handling process takes over to replan from the current execution point to help the agent recover from the failure.

For instance, when a certain external information source is out of service temporarily, the agent who needs data from this information source shouldn't just wait passively until the service is back. Instead, the agent might want to try another information source or switch its attention to other tasks for a certain period of time before returning to the original task. Mechanisms for reactivity in the agent architecture provide a systematic and reusable way of engineering these uncertainties handling mechanism into software agents. A simple example is a timeout mechanism. Whenever an agent fails to retrieve the information of interest from a certain source within a predetermined time limit, the agent will automatically invoke an exception handling routine, which might invoke a preplanning process or simply wait for a particular time interval before re-trying accessing the information. Upon completion of an action, results are recorded, downstream actions are enabled if so indicated, and statistics collected.

The agent's plan library contains skeletal plans and plan fragments that are indexed by goals and can be retrieved and instantiated according to the current input parameters. The retrieved and instantiated plan fragments are used to form the agent's task tree that is incrementally executed.

The multi-agent System

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 organization and to generate decision-making alternatives that would allow the user to focus on solutions that were found to be significant. A set of agents is integrated to the system and placed in the DSS components, according to our architecture of GDSS.

Integrating Agent in Cooperative Intelligent Decision Support System - The individual DSS, as shown in Figure 2. 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,

(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.

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.

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 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

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.

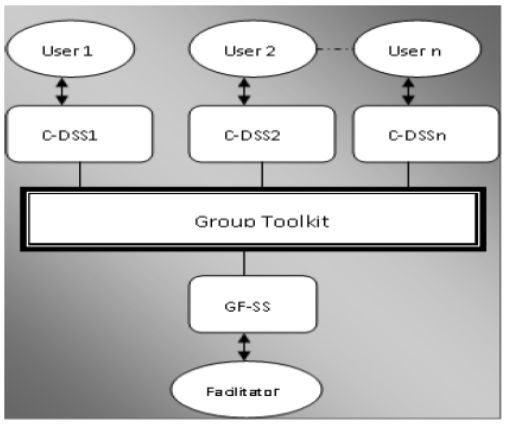

Figure 2. The group decision support system architecture

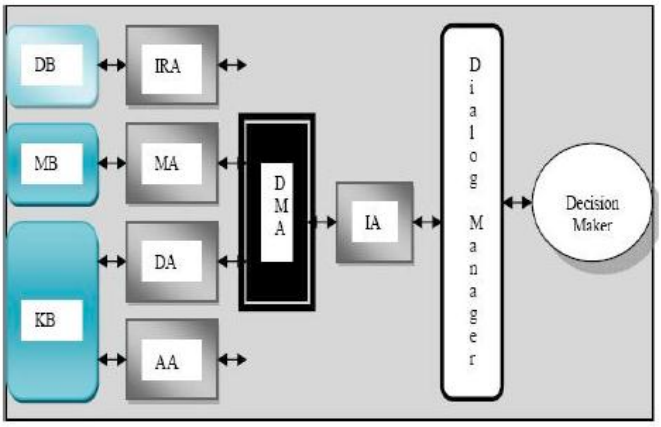

Figure 3. Agent architecture for individual DSS.

CONCLUSION

In this paper, we have described concepts and techniques for structuring and organizing distributed collections of intelligent software agents in a reusable way. We presented the various agent types that we believe are necessary for supporting and seamlessly integrating information gathering from distributed internet-based information sources and decision support, including (1) Interface agents which interact with the user receiving user specifications and delivering results, (2) Task agents which help users perform tasks by formulating problem solving plans and carrying out these plans through querying and exchanging information with other software agents, and (3) Information agents which provide intelligent access to a heterogeneous collection of information sources. We have also described and illustrated our implemented, distributed system of such collaborating agents. We believe that such flexible distributed architectures, consisting of reusable agent components, will be able to answer many of the challenges that face users as a result of the availability of the new, vast, net-based information environment. These challenges include locating, accessing, filtering and integrating information from disparate information sources, monitoring the Infosphere and notifying the user or an appropriate agent about events of particular interest in performing the user-designated tasks, and incorporating retrieved information into decision support tasks.

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. 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 negotiation.

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. in another paper. And Finally, as stated in Web technology will be ever more considered in DSS, thereby people will make codecisions 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.

A Web-based DSS uses the Web as a portal to the underlying DSS. It lets interested users access and make use of the underlying DSS through the Web. Moreover, we believe a distributed implementation of the underlying DSS is also important for a Web-based DSS presents a challenge, which needs the combination of a DSS with distributed computing technology. Our proposed multi-agent approach provides a practical way to implement a Web-based DSS.

The proposed architecture of the Web-based DSS is under development. One of our perspectives is to completely implement it, test it in a manufacturing industry in order to obtain feedback on the usability of the developed system.

REFERENCES

- Adla, J-L. Soubie, "A distributed architecture for cooperative decision support systems", Proc. of Euro Worgroup Workshop on decision support systems, London, England, 2006.
- C. Schneeweiss, Distributed Decision Making, Springer – Verlag Berlin. Heidelberg, 2003
- 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.
- Donald A. Norman. How might people interact with agents. Communications of the ACM, 37(7), July 1994.
- 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.
- E. Turban, and J. Aronson, Decision support systems and intelligent systems, Prentice-Hall International, Upper Saddle River, New Jersey, 2001.
- F.G, Filip, "Decision support and control for large-scale complex systems", Annu Rev Control, doi: 10.1016/j.arcontrol.03.002, 2008.
- Huaiqing, W., Stephen, L., Lejian, L.: Modeling constraint-based negotiating agents. Decision Support Systems, 33(2), pp. 201--217, (2002)
- 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.
- 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.
- M. R. Genesereth and S. P. Katchpel. Software agents. Communications of the ACM, 37(7):48{53,147, 1994.
- M. Limayem, P. Banerjee, and L. Ma, "Impact of GDSS: opening the black box", Decision Support Systems, Lavoisier, 2006, pp. 945- 957.
- W. Cheung, "An Intelligent decision support system for service network planning" , Decision Support Systems, Lavoisier, 2005, Vol. 39, pp. 415- 428.
- Y. Shoham. Agent-oriented programming. Arti_cial Intelligence, 60(1):51{92, 1993. 4. M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. The Knowledge Engineering Review, 10(2):115{152, 1995.