Cooperating knowledge-based systems for environmental decision support

N M Avouris

The paper discusses the applicability of cooperating knowledge-based system (CKBS) techniques in environmental decision support. The reasons for using CKBSs are given first. Existing environmental CKBSs are discussed, with special emphasis on a typical example, the Distributed Chemical Emergencies Manager (DCHEM). The methodological framework applied to the building of DCHEM, the knowledge acquisition technique used, and the architecture of the developed system are also described.

Keywords: distributed artificial intelligence, cooperating knowledge-based systems, environmental management

Environmental management is a human activity that has the objective of reducing broadly understood risks resulting from the interaction between our social system and its natural environment, and in particular mitigating the impact of technology on this environment. In the frame of this activity, environmental managers require sophisticated tools with which to solve the multifaceted critical environmental management problems that face all levels of our societies. This is particularly important as we rapidly move into an era of limits.

Knowledge-based systems can play an important role in this context. Some of the reasons are as follows:

- The high complexity of environmental problems. Environmental decision makers often need to understand, in a limited time, physical and biological processes in relation to socioeconomic conditions and applicable legislative frameworks. Decision support tools, incorporating past experience and relevant expertise, can be useful in this context.
- The multidisciplinary nature of environmental problems. This has as a consequence the unavailability of some necessary expertise when it is required. Knowledge-based systems, incorporating this missing expertise, can support decision makers.
- The 'objective' power of knowledge-based systems, in a highly subjective problem solving context, can provide systems of a regulatory nature, which are often needed.

Despite the urgency and the importance of the problems, the number of reported knowledge-based environmental decision support systems remains relatively low, not matching the advances of knowledge-based techniques in areas like medicine, manufacturing, business, finance and industrial control. The reasons should be sought partly in the inherent difficulties of the domain and partly in the inadequacy of the widely used knowledge-based techniques. In a survey[1] of 181 expert systems applications, environmental decision support systems were noticeably missing. One reason for this, as argued by Hushon[2] in a survey paper, is 'that environmental expert systems require expertise in a number of areas and that the heuristics for environmental decision making are not codified'. The number of reported systems has improved recently, but the great majority of the developed prototypes cover specific, narrow domains, and are based on traditional rule-based approaches[2-4].

In this paper, it is argued that techniques proposed by the emerging field of distributed artificial intelligence (DAI) are suitable for tackling environmental decision support problems and can improve the effectiveness of environmental decision support systems. This is demonstrated first through a discussion of some typical cases of the application of DAI techniques in the field and second

Computer Laboratory, Electrical Engineering Department, University of Patras, GR-265 00 Rio-Patras, Greece
Paper received 5 August 1993. Revised paper received 24 February 1994. Accepted 25 March 1994

SSDI 0950-7051(94)00289-4

through a detailed description of a particular example which demonstrates the benefits and the limitations of this approach.

In the following section, the key features of environmental problem solving are described. Subsequently, an overview of DAI is provided, with an emphasis on cooperating KB systems. The main characteristics of these systems are described and matched to the features of environmental problems. In the subsequent section, a survey of existing environmental CKBSs is included and a discussion on the reasons for the use of CKBS techniques in each case is also provided. The following sections of the paper contain a more detailed description of the DCHEM system. The CKBS shell CooperA used for its development, the knowledge acquisition process, the system architecture and the performance are also discussed. General remarks on the applicability of the DCHEM system experience to other complex applications are also included in the last section of the paper.

ENVIRONMENTAL PROBLEM SOLVING CHARACTERISTICS

The key characteristics of environmental problem solving (not necessarily unique to this field) are as follows:

- Environmental problems are multidisciplinary by nature. In a definition of environmental science[5], it is stated that 'a continuous compromise is required in the tension between the inter-disciplinary breadth and the depth of disciplinary knowledge demanded for understanding and solution of environmental problems'. As a consequence, in most environmental management situations, a single expert who can solve the problem entirely does not exist. Cooperative problem solving is therefore most widely used.
- Conflict is inherent in environmental problem solving[6]. The complexity of the fields and the multiplicity of views and interests involved call for mechanisms of reconciliation of disparate, often conflicting goals and contradictory information. Sociologists[7,8] modelling the process of environmental decision making, identify the existence of conflict and advocate the importance of negotiation and consensus making in this process.
- Physical systems are hard to model and understand. Human-made systems can be described through a finite number of states, and therefore deep models of their behaviour can be produced. In contrast, our physical environment is dynamic in nature. Attempts to model it can only be based on gross simplifications or shallow models.
- Problem solving heuristics are highly subjective and thus hard to generalise and codify, while many alternative methodologies are used by different domain experts, owing to their diverse backgrounds.
- There is often a strong spatial distribution involved. The scale can differ widely, from the globe, which is studied in the case of global environmental change, to the problems of a narrow geographical area. This dimension often necessitates splitting the problems into subareas and subsequently combining the results.
- Information collected from the environment is often imprecise, uncertain or erroneous. The application of multiple complementary problem solving techniques can often reduce this uncertainty.

Many of these characteristics, e.g. the third, fourth and sixth, are strong arguments in favour of the application of KB techniques. However, others, like the first, second and fifth, require the adaptation of the existing approaches to distributed problem solving paradigms.

It should be observed that, since the described features are not unique for this class of problem, the subsequently proposed techniques are of more general applicability.

OVERVIEW OF DISTRIBUTED ARTIFICIAL INTELLIGENCE

Distributed artificial intelligence is a relatively new field of research and technology concerned with the study and construction of automated systems that support coordinated intelligent behaviour in a group of semiautonomous computational elements, called *agents*. 'These agents, when faced with a problem that can be solved effectively through cooperation, work together by identifying the sub-problems each should solve, possibly from different viewpoints, solving them concurrently and integrating their results'[9].

A brief overview of this field is provided here; for more details, consult References 10–14.

The research work on the theoretical foundations of DAI is inspired by fields like sociology, group interaction and coordination theory[15–17], which provide paradigms for distributed problem solving. Traditional application areas of DAI techniques have been speech and language processing, like the HEARSAY II experiment[18], manufacturing and robotics[19], air traffic control[20], distributed sensing and interpretation, like DVMT[21,22], industrial monitoring and control[23]. However, as the power of distributed processing systems increases and more advanced cooperative algorithms and languages emerge, it is expected that DAI techniques will spread into new complex application areas.

Environmental applications have not been the focus of the DAI field so far. However a limited number of experiments have been reported, which are briefly discussed in this paper.

The reasons for selecting a distributed problem solving approach in a given application, can be the following among others[9]:

- an endeavour to reduce a high problem complexity by functionally decomposing problem solving,
- the physical distribution of the problem, which makes a centralised solution not feasible,

- a need to facilitate knowledge acquisition and modelling from multiple experts,
- a need for improved robustness and reliability, which can be achieved by applying multiple complementary problem-solving methods by different agents,
- the reusability of heterogeneous preexisting components,
- a graceful degradation of the system response when there is degradation of input data or failure of portions of the system,
- increased performance due to parallelism,
- flexibility in user interaction and support for user understanding[24],
- extendability, i.e. as the system grows and increases in complexity, a distributed system is more adaptable to change.

Many of these reasons can be found in environmental applications, as described above. It seems therefore that distributed problem solving is a natural paradigm to be considered for many complex environmental support systems.

Cooperating knowledge based systems

A DAI system is described according to a number of parameters. These include the number of the agents in the system, the degree of agent *heterogeneity* in terms of knowledge and functionality, the *granularity* (size and complexity of the single agent), the *degree of interaction*, the task decomposition and result synthesis technique used, the negotiation and conflict resolution protocols etc.

A cooperating knowledge-based system, or cooperating expert system, is a special case of a DAI system, in which the complexity of the individual agents is high; agents can be heterogeneous, independent problem solving modules, usually based on KB architectures.

In a typical scenario, a number of possibly preexisting expert systems, representing distinct areas of expertise and narrow problem solving domains, are needed to accomplish a task. Through the CKBS, they are incorporated in distinct agents. These agents model the capabilities of the expert systems and provide them with social behaviour. Interactions among the CKBS agents permit cooperative building up of the final solution, which it would otherwise not be possible to obtain with the independent modules.

Communication and distributed control in CKBSs is effected through alternative mechanisms (see *Figure 1*):

- *Shared global memory* (a well known example is the *blackboard architecture*[25,26]): Blackboard systems have evolved from the production system approach of the first generation expert systems where the individual production rules have been replaced by bigger knowledge modules called *knowledge sources* (KSs). These are typically small-to-medium granularity CKBSs.

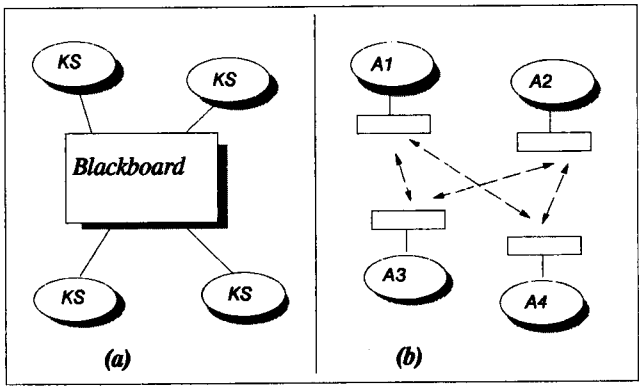

Figure 1 Communication; (a) blackboard, (b) message passing

- *Message passing* (e.g. the actor model[27,28]): Messages are exchanged asynchronously among the agents. Complex protocols based on message passing can be developed for task allocation, permitting contracting and multiagent negotiation[29], or multiagent planning[12].

Often a combination of blackboards and message passing is used[22]. It should be observed that a single blackboard architecture may be both vulnerable and a potential bottleneck[30], while in a multiple blackboard architecture the semantics of keeping them consistent reverts to message passing.

Recently, research work in concurrent object based systems has produced a number of message-passing languages[31] which are proposed as powerful tools for building CKBSs. Well known pioneering examples of message-passing CKBS architectures have been proposed by Gasser *et al.* (MACE)[32], Lesser *et al.* (DVMT)[21], and the ARCHON project[23,33].

The CooperA platform[34] described in some detail in this paper, belongs in this class of experimental prototypes. It has been built for studying multiagent systems and has been used as an implementation platform for our case study, the DCHEM environmental decision support system.

The main features of a message-passing CKBS are described below in terms of a typical model for the individual agent and an interagent cooperation support mechanism.

CKBS agent description

A CKB agent is a semiautonomous computing element with its own closed knowledge world model, reasoning and communication mechanisms.

Each individual agent consists of two parts, as shown in *Figure 2*:

- the problem solving part (PS), containing a goal hierarchy, which can be controlled by the overlaying Cooperation Layer; the PS is typically a knowledge-based system,

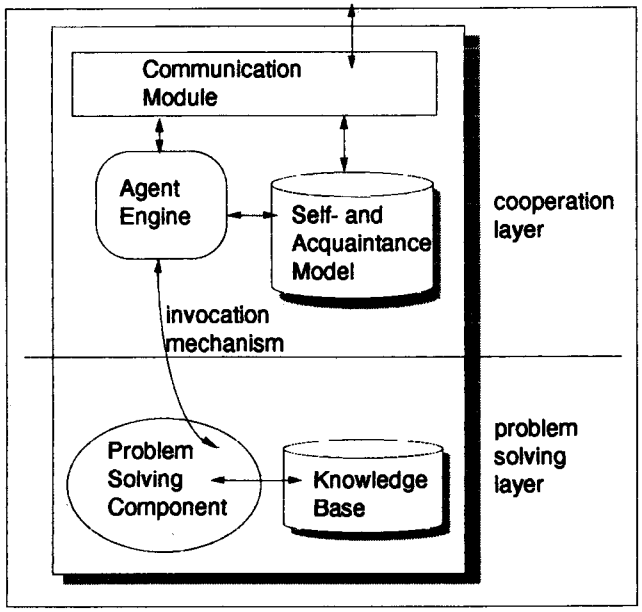

Figure 2 CKB agent

- the cooperation layer (CL), of a similar structure in all agents, which controls the problem solving part in such a way that the individual agent demonstrates cooperative behaviour towards the other agents; the CL provides the necessary services for the integration of the PS component into the multiagent environment.

The cooperation layer manages interaction with the other agents and relates the local activity of the intelligent system to the global problem solving. This is achieved through a control mechanism (the agent engine (AE)) which uses a set of data structures:

- a model of the other agents (agent acquaintances model (AAM)) used for planning cooperation and interaction,
- a model of the underlying problem solving component (self model), used for planning and controlling local activities,
- a communication module which, by using the AAM, effects interaction with the other agents.

The AE operates asynchronously and at higher level of control than the inference engine of the problem solving part which acts on domain-specific knowledge.

A communication mechanism and an interagent cooperation language, based on a typology of exchanged messages, need to be defined in a CKBS. A common data model is also required in which the cooperation layer concepts are expressed.

The coherent behaviour of the multiagent system is guaranteed by either the explicit or the implicit representation of the common system goals and provision for negotiation and conflict resolution mechanisms.

Cooperation control in CKBS

One issue of particular importance in CKBSs is the coherency of distributed problem solving, that is, the effect that the local actions have on the global convergence towards the solution. In CKBSs reported in the literature, this issue is tackled through various mechanisms. The main ones are as follows:

- *Hierarchical or centralised organisations*: The agents have specific 'roles' in a hierarchical structure. As a result, within a specific context, the control relations among them have been decided by their roles. These systems do not have high flexibility, but usually have good performance in engineering or business domains. An example of such a system is the organisation implementation of MACE[32] and ARCHON[33].
- *Negotiation based interaction*: The agents compete for resources and task execution through bidding and contractual mechanisms. A well known example is the Contract Net protocol[29]. Other examples are negotiation-based cooperation, implemented in argumentation and industrial conflicts resolution[35], and the concept of the domain-independent conflict resolution heuristic knowledge, advocated by Klein[36] and applied in cooperative design.
- *Functionally accurate cooperation*: This is an approach, often used in physically distributed problems, like distributed sensory networks, where the solution has to be built incrementally by dislocated agents. According to this approach, agents need not have all the necessary information locally to solve their subproblems. They coordinate their activity by exchanging partial solutions (functionally accurate, but not exact) of their subproblems. The distributed vehicle monitoring experiment (DVMT) is a well known example of this approach[21,37].

In environmental decision making, one has to study the specific problem characteristics in order to select the appropriate cooperation control regime. Often flexibility in distributed problem solving is needed, since, as described in the introduction, the resolution of conflicts is necessary. In this case, the second approach has advantages. Many environmental monitoring systems, on the other hand, have many common characteristics with the third approach. Finally, in many cases, the robustness of the first approach is required. The CooperA shell, described in more detail below, does not explicitly support either of these control mechanisms.

SURVEY OF CKB SYSTEMS IN ENVIRONMENTAL APPLICATIONS

A number of cooperative knowledge-based environmental decision support systems have been reported in the literature. They are surveyed in this section, in relation to

the CKBS features presented above, and the domain characteristics that have led to their design decisions.

- A CKBS developed in the domain of air quality control is the DUSTPRO system[38]. According to its developers, the reason for opting for such a paradigm is the high complexity of the domain. A number of expert systems have been built which are concerned with specific possible sources of dust in an indoor industrial environment (a mine). One expert system deals with improving the cutting technique, according to the specific minerals and conditions, another expert system deals with the possible malfunctioning of the ventilation equipment etc. The system is built around a blackboard architecture and specific modules have been introduced for the housekeeping of the system, e.g. for keeping track of a session.
- Cooperating KB systems techniques have also been implemented in the STORMCAST system[39,40], which has been built for supporting storm forecasting over a wide area (the Scandinavian peninsula). STORMCAST is based on multiple similar KB systems which reside on distinct processing nodes and are geographically dislocated. The nodes of the system cover separate overlapping geographical areas; however, the exchange of data and solutions is necessary, especially among adjacent nodes. The distribution of problem solving is a result of the inherent physical distribution of the problem and requirements of robustness and redundancy in a problem solving capacity. The characteristics of STORMCAST are typical of a class of distributed environmental monitoring applications. Other examples of CKBSs are a distributed air pollution monitoring system for a metropolitan area[41] and a river monitoring system[42].
- A system of particular interest is the Low Clouds Prediction System (LCPS)[43], based on a blackboard architecture, which was developed for the Atmospheric Environment Service of Canada. In this case, multiple complementary methodologies are used. The available reasoning components are (a) a system based on the physical rules that govern meteorological phenomena, through the so-called physical principles forecaster flowchart, which establishes the conditions necessary for the presence of low clouds by the thermal and humidity profiles of the atmosphere, (b) statistical prediction models, and (c) heuristic approximations, applied when it is necessary or more efficient. When the data necessary for applying a are unavailable, incomplete or erroneous, modules b and c act in support. This characteristic of the system, to reduce the uncertainty of the problem by applying alternative reasoning techniques or knowledge bases, is a typical use of CKBSs. It simulates the problem solving behaviour of a group of experts who apply various problem solving methodologies and expertise in relation to a given formulated problem. The coupling of numeric and symbolic components, necessary in many environmental management problems, can be achieved by this type of CKBS.
- The PHOENIX system is a fire-fighting simulator[44] developed for managing forest fires at the Yellowstone National Park, USA. PHOENIX is a multiagent planner, the agents of which cooperate through a message passing protocol. A number of agents have been defined, each with a specific role, in relation to a simulated forest fire. One agent is the fire boss, who constructs the top level plan and coordinates activities. Other agents are the fire fighters (e.g. bulldozers) who analyse the high level plan delivered to them by the fire boss and plan their activity in more detail, while they interact with the fire boss and the environment (the fire). PHOENIX agents can have both cognitive and reflective behaviours, and so they can build a long-term plan for cutting a fire line while moving in a reflective way, for instance moving back in order to avoid the advancing fire. The reason for building PHOENIX as a distributed system is that the fire-fighting problem is spatially distributed and complex. Most fires are too big for a single agent to control. This is simulated in the system, since each fire fighter has a limited view and knowledge of the situation; it needs therefore to coordinate its activity and plans with the others and the fire boss in order to accomplish the fire-abatement task. Simulators are tools commonly used for environmental decision support. Apart from the coarse-grain cognitive agents, like the PHOENIX ones, purely reflective, fine-grain multiagent systems have been reported in experiments of the simulation of ecological modelling and artificial life[45], which, however, fall outside the scope of the CKBSs described here.
- Finally, a CKBS based on a multiagent message passing architecture is the DCHEM (Distributed Chemical Emergencies Manager) system[34], which supports decision making in relation to environmental emergencies in installations involving toxic substances. The reason for selecting a CKBS approach in DCHEM is the high complexity and the existence of multiple experts in the domain, with distinct areas of expertise. DCHEM is discussed in more detail in this paper and is used as a test case in our presentation of environmental distributed problem solving.

Summarising the characteristics of the applications discussed, it is observed that the reasons for using distributed problem solving techniques are related either to (a) the multidisciplinary nature of environmental applications, resulting in the high complexity of the problems, the modularity of the expertise, and the diversity and complementarity of the problem solving methodologies, or (b) to the geographical dislocation of the problem and the problem solving components. These characteristics have been identified earlier as being typical of many complex environmental problems.

The systems described are compared in *Table 1*.

Table 1 Distributed environmental management systems

System	Domain	Architecture	Type of distribution	Rationale for CKB system
DUSTPRO	Air quality control	Blackboard	Functional	High complexity
STORMCAST	Storm forecasting	MA negotiation	Geographical	Physical distribution
LCPS-AES	Weather prediction	Blackboard	Multiple methods	Increased reliability
PHOENIX	Fire fighting	MA planner	Distribution simulator	High complexity
DCHEM	Chemical emergencies	MA negotiation	Functional	Multiple expertise

COOPERA: A CKBS SHELL

The CooperA (Cooperating Agents) CKBS environment was built at JRC Ispra and used for the development of the distributed environmental decision support system DCHEM, discussed below. A more extensive description of CooperA can be found in References 34, 46 and 47. The CooperA shell belongs to a class of multiagent platforms developed over the last few years[48] for experimentation with cooperative problem solving. Other examples of such systems are MACE[32], DVMT[21], MICE[49] and ARCHON[33]. CooperA is an early example of such a shell that is especially suitable for building knowledge-based applications, since it is built as an extension of a KB environment supporting many knowledge representation formalisms. CooperA was mainly used for experimentation with heterogeneous KB systems, and it has many limitations in terms of the efficient use of distributed processing techniques. ARCHON was built later, implementing the main concepts of CooperA in a more efficient way. Despite the performance limitations, CooperA is a typical example of a CKBS shell, demonstrating the main characteristics of these architectures.

Overview of CooperA

CooperA has been designed and developed as a CKBS programming environment that supports the cooperation of heterogeneous distributed semiautonomous knowledge-based systems presented to the user via a customised user interface. These modules, the *application agents*, can be incrementally and selectively integrated into the system. Each expert module incorporates a self description mechanism, which allows the shell to integrate it into the agent group during the configuration phase. Thus various alternative agent configurations can be tested for the solution of a certain problem, thus simulating alternative expert groups being available for problem solving.

The user of CooperA can interact with the agents through the user interface agent. Special attention has been paid to the user interface which incorporates the active modelling of the system, so that the user can see the flows of interaction among the agents. This helps in the comprehension of the distributed problem solving activity.

CooperA has been built on top of CRL-Lisp, which is an object oriented extension of Common Lisp supporting schemata (frames). The KnowledgeCraft knowledge engineering environment is used for developing CooperA applications. This environment allows experimentation with different reasoning mechanisms and knowledge representation formalisms, integrating heterogeneous problem solving elements.

The main parts of the environment are shown in *Figure 3*, in which the software layers making up the CooperA shell can be seen:

- at the bottom layer, CRL Lisp and the reasoning mechanisms (CRL Prolog, OPS) supplied by the KnowledgeCraft environment, available for building the agent problem solving components,
- the cooperation support layer, in which agent invocation, agent scheduling and the message passing mechanism are implemented,
- the agent definition layer (agents are defined in this layer as instantiations of the agent template; each agent inherits behaviours like message sending and message receiving, and is associated with specific problem solving methods),
- the user interface layer, in which the user interface agent also resides.

A brief description of the agent structure, the cooperation control mechanisms applied and the user interface are included in the following sections.

Description of CooperA agent

A CooperA agent consists of a cooperation and a problem solving component. A number of data structures are contained in the cooperation layer of the agent, supporting problem solving and interaction activities:

- Asynchronous communication is performed through two queues: the *outgoing-messages queue* (out-q) from which the outgoing messages are transmitted, and the *incoming-messages queue* (in-q) that receives and handles incoming messages. When an agent receives a request, it queries accordingly its own local world of knowledge or activates its problem solving component.
- Each agent contains a self model containing the attributes *my-skills* and *unsatisfied-goals*. The former contains the knowledge relative to its own capacities, while the latter contains reference to unsatisfied goals with associations with other agents that can satisfy them. The attribute *status* of an agent describes the current status of its activity (*new, inactive, running* and *waiting*) during execution.

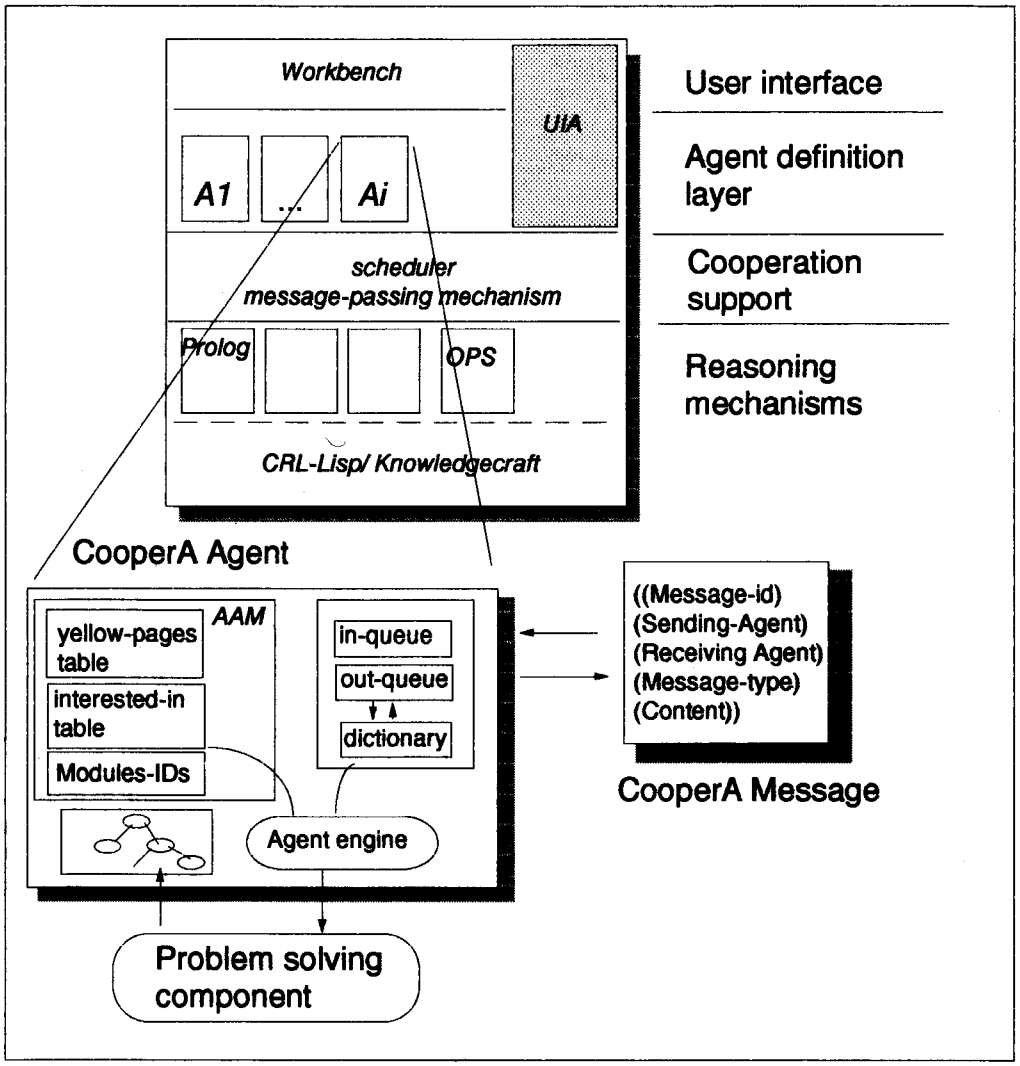

Figure 3 CooperA multiagent environment, CooperA agent and CooperA message structure

- The CooperA agent acquaintance models are structured as follows. The *yellow-pages-table* contains a description of all the acquaintances of the agent who are in a position to satisfy goals that the agent is interested in. The metaphor used is that of a yellow pages directory. The *interested-in-table* contains information about agents who are interested in the agent skills. These tables are dynamically created during the initialisation of the agents, taking into consideration the participants of the particular group of agents and the goal structure of the agent.
- The *dictionary* data structure contains a directory of associations between concepts, locally defined and used by an agent, and global terms, defined in the frame of the communication protocol. The structure of the *dictionary* for agent$_x$ is

$\{dictionary_x$
. . .
local-concept$_j$ global-concept$_j$ goal$_j$
. . .$\}$

where $\langle$local-concept$_j\rangle$ is a slot referring to a local symbol, $\langle$global-concept$_j\rangle$ is the globally known name of the local concept and $\langle$goal$_j\rangle$ represents the goal associated with local-concept$_j$. Global concepts are shared by all agents. If the same global concept is related to a $\langle$local-concept$_k\rangle$ of another agent, the two concepts are considered to refer to the same symbol in a virtual global name space of the agent's community.

The coherence of the distributed problem solving activity depends a great deal on an accurate semantic mapping of the concepts defined in the frame of the various knowledge bases by the knowledge engineers, as discussed below, in the DCHEM development description.

User interaction in CooperA

The user interface agent is a standard agent provided to application developers by the CooperA environment. The UIA has the same structure as the other CooperA agents and communicates with them through the same message passing protocol. The UIA has a dual objective: to represent the user in the agent group, and to present the agent activity to the user.

The concept of the UIA can be found in many similar systems. For more information on the design of CKBS user interfaces and the architecture of the UIA, refer to References 50–53. The UIA in CooperA represents the user as a boundless domain of knowledge in the group of agents. For this reason the UIA has some special features as far as acquaintance modelling and message structuring are concerned. There is a special type of message, *user-request*, that refers to the communication of the agents with the UIA. The knowledge necessary for supporting user dialogue is usually owned by the agent making the request. Therefore this additional knowledge is passed to the UIA, which takes care of the presentation details (graphics, windowing system etc.) for visualising the request and managing the dialogue. After an interaction, the UIA takes care of forming the reply message according to the results of the dialogue with the user, and passing it to the requesting agent or any other agents interested in it, performing the necessary message translation and transmission.

A customized user interface has also been developed (the CooperA Workbench), which permits the user dynamically to configure the system and see the flows of interaction among the agents, as shown in *Figure 4*. This special attention of CooperA to user interaction is particularly important for environmental support applications, which are characterised by strong user-interaction requirements.

CooperA has been used for the development of the DCHEM system. The main phases of this development process are described in the following sections.

DISTRIBUTED CHEMICAL EMERGENCIES MANAGER

The Distributed Chemical Emergencies Manager is a decision support system in the field of chemical accidents management. This is a typical environmental management problem, in which, following an accident in a site where suspected chemicals are stored or used, an assessment of the situation needs to be performed rapidly in order to decide on the appropriate response. Knowledge based techniques have already been proposed for this problem[54–56]. For a number of reasons, e.g. multiple expertise and high complexity, the use of CKBS techniques seems a suitable choice. The main fields of expertise involved in chemical emergencies management are chemistry, engineering knowledge of the equipment involved, accident-site geography, mitigation techniques and applicable regulations.

DCHEM is concerned with a specific class of environmental emergencies, the management of accidents involving electrical equipment containing toxic chemicals. An example is that of polychlorinated biphenyls (PCBs), a class of hazardous chemical substances used widely[57]. Accidents of this type (e.g. leaks, fires, explosions) have some common characteristics, although they can occur in diverse locations and under different circumstances. It has been recorded that most of the accidents are a consequence of overloading of the equipment. These events have resulted in a leak of the toxic substances to the working or general environment. In the case of rupturing of the container, hazardous material is spilled on the ground and can be transported via the soil, surface waters and, in extreme cases, underground waters. In the case of explosion or fire due to overheating of the toxic materials, new and more dangerous substances can be formed, such as PCB transformed in furanes (PCDFs) and dioxins (PCDDs). In this case, because of the existence of contaminated gases and dust, the main pathway can be the atmosphere.

Determining the threat of the release is often the first step in the management of these emergencies. In the process of achieving this, decision makers need expertise in different domains, and need to process information referring to the accident circumstances that is usually imprecise and incomplete.

The decision support system had to incorporate the knowledge of a group of experts in the various domains involved. Therefore DCHEM development involved a knowledge-acquisition phase, using multiple experts. This phase is briefly discussed below.

Building an environmental CKBS: issues of knowledge acquisition

Most knowledge elicitation and knowledge-base building techniques make the assumption that there is a single domain expert, whose knowledge is extracted and used. In complex domains where more than one expert is involved, there is a need to mediate among them, and take into consideration their diverse problem solving techniques during knowledge acquisition. The problem of knowledge acquisition from multiple experts has already received the attention of the AI community. Techniques and tools have been proposed in order to elicit the knowledge from multiple experts and combine the results into a single coherent expert system at design time[58–60]. Liou[61], for instance, proposes the use of consensus building mechanisms, like group repertory grid analysis, group discussion and voting. Synergy among the experts and a coherent knowledge base is claimed to be the result of this process. Ignizio[62] suggested, as a possible approach, the use of a knowledge base cloned from one expert, the key expert, to build a prototype, and then the other domain experts being allowed to critique the results. The proposed development cycle closes with obtaining the key expert's view of the comments. However, when the experts have no mastery

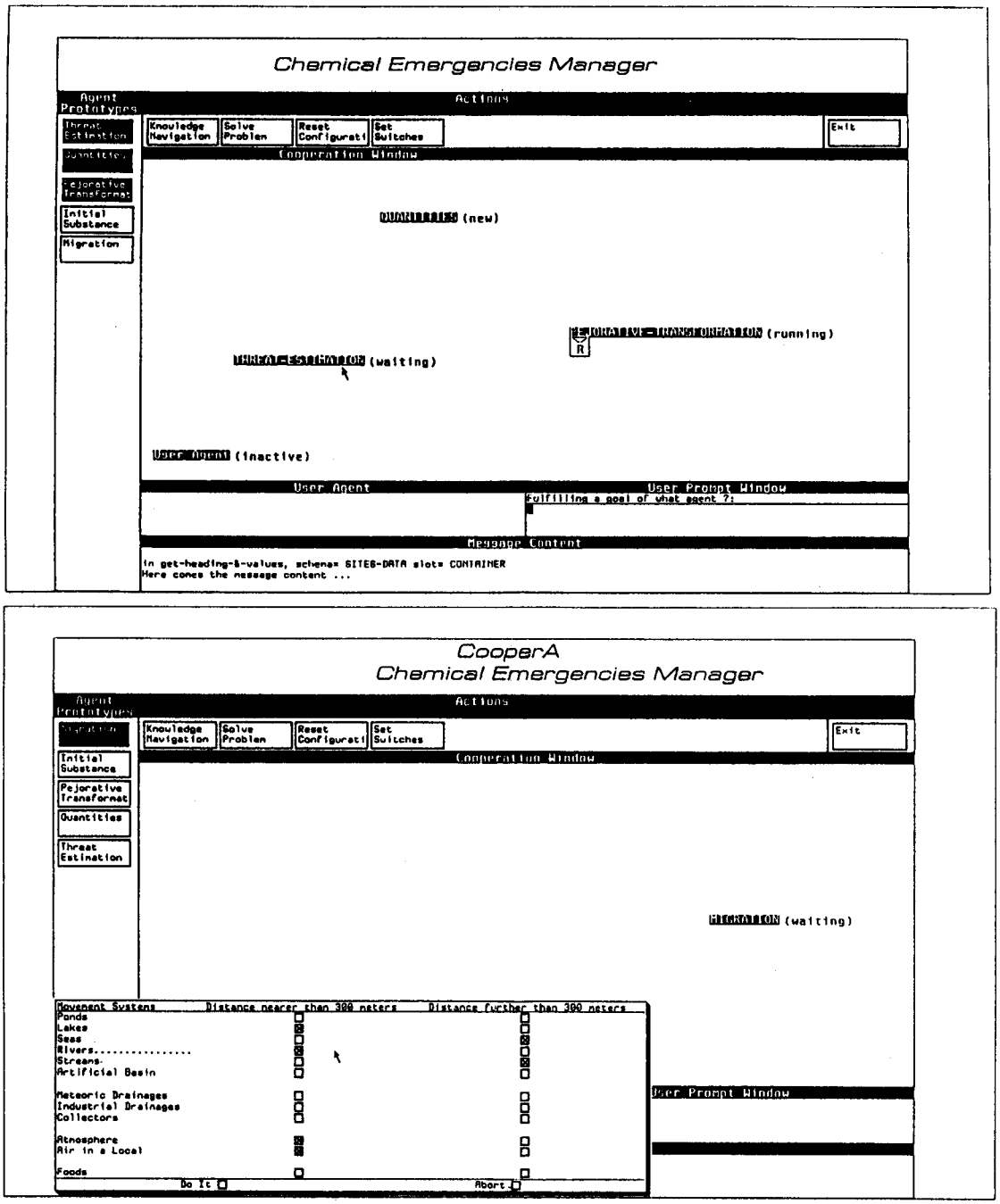

Figure 4 CooperA user interface: (top) agent's visualisation, (bottom) dialogue between user and MIG agent

across the entire domain, but their expertise covers only portions of the domain, then the use of such techniques presents difficulties.

A comprehensive structured approach is proposed by Lekkas *et al.*[63] in relation to multiagent system analysis and design. According to this methodology, the domain knowledge is first modelled during the knowledge acquisition phase, following a technique based on generic tasks. The existence of one or more experts, however, does not affect this process according to the methodology. Then the problem is decomposed into a number of problem-solving tasks. A number of parameters are identified that characterise the problem and drive this problem decomposition. Subsequently, during system design, these tasks are allocated to problem solving agents. Thus there is no relation between the origin of the expertise and its allocation in the finally implemented system.

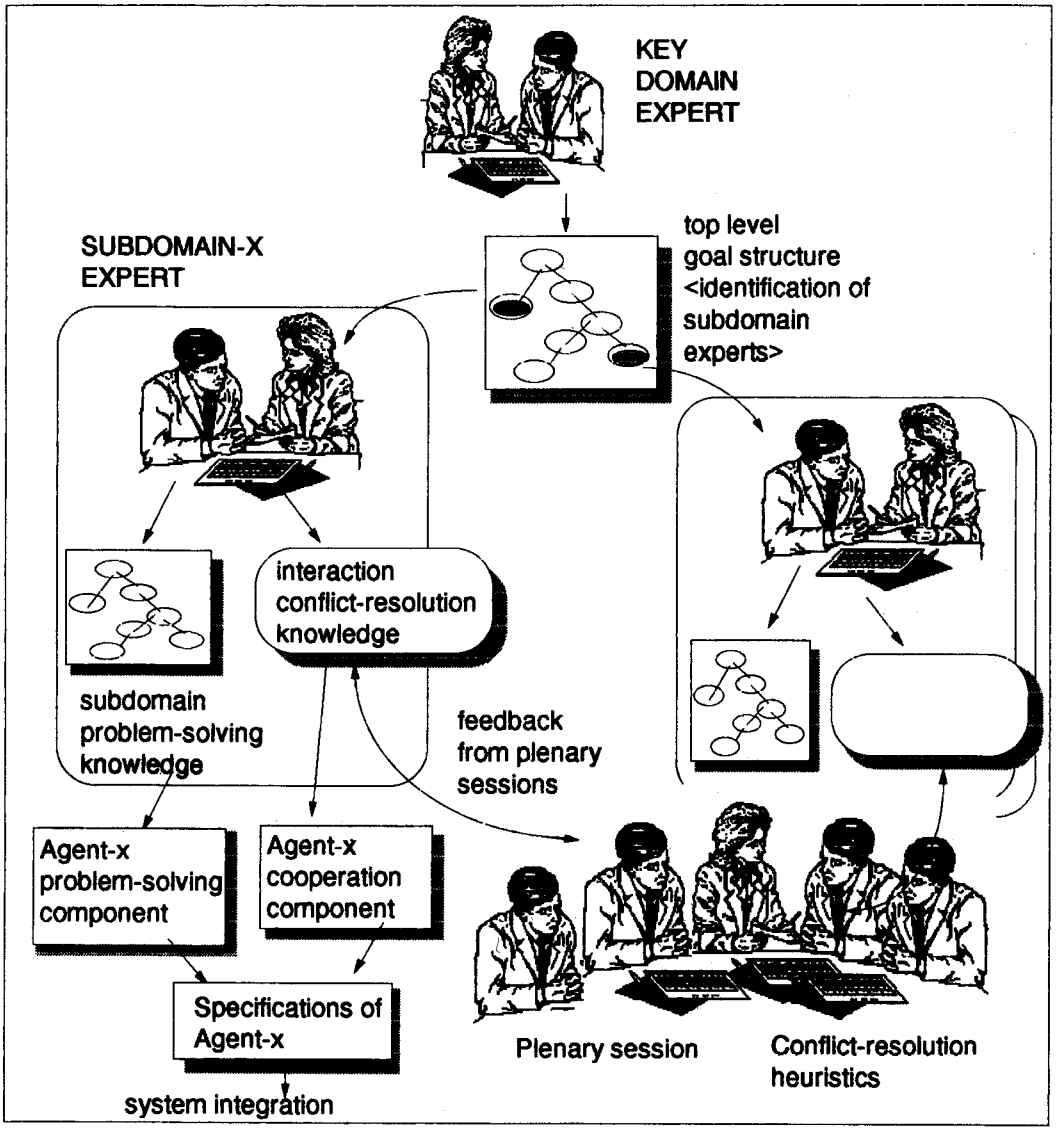

Figure 5 Knowledge acquisition in DCHEM

A novel approach has been applied in DCHEM which has been adapted from the techniques described. This approach takes into consideration the existence of multiple experts representing distinct areas of expertise. This decomposition is modelled in the designed CKB system in the form of agents. Thus the implicit assumption of this approach is that the problem decomposition reflects the domain experts. The innovative aspect of the methodology is that it is based on two interleaved phases of knowledge acquisition: the *domain-problem-solving knowledge elicitation* and the *cooperation and conflict-resolution knowledge elicitation* phases. Thus the specific characteristic of our approach is that the conflict-resolution and cooperation knowledge is also elicited and introduced separately in the system, so that it can be used to coordinate the agent's activity at runtime.

The knowledge acquisition phase is described in more detail below.

DCHEM knowledge acquisition

The top-down knowledge elicitation approach used in the DCHEM project is described in *Figure 5*. A key domain expert was initially identified who provided a task framework for the evaluation of the *threat-estimation* top goal. Through this process, identification of the subdomains of expertise was made possible. The fact that the knowledge involved could be organised in relatively independent units which could contribute to the solution in a dynamic way, according to the specific characteristics of a particular situation, led us to establish the fact that a decoupling of the various modules was necessary in the next phase of knowledge acquisition.

A number of small independent subprojects were spawned at this stage. The subdomain experts had different perspectives and backgrounds, and applied diverse problem solving methodologies. The same knowledge

engineer was not used to model all the different knowledge bases. Thus it was inevitable that the independently extracted knowledge and prototypes built presented diversities and used heterogeneous formalisms which best suited each particular subproblem. Each independent subproject had as input the specific subgoal and an overview of the context (other subdomains, top-goal structure etc.). The output was an expert system module and requirements for interaction with other parts of the system and/or the user. Interactions used a predefined protocol which was based on a set of simple primitives: (request-data ⟨agent-x⟩), (deliver-reply ⟨agent-x⟩), (request-goal ⟨agent-x⟩), (deliver-result ⟨agent-x⟩), (user-request ⟨id⟩), (user-reply ⟨id⟩).

At this stage, it was not possible to decide on the binding of all the interactions, since only 'weak' acquaintance models were available. The detailed specification of the other modules was not known, and multiple bindings were often possible and had to be decided at runtime, according to the situation, the available information and the status of the agents.

The agents were built and tested as independent prototypes, with their user satisfying the interaction with the rest of the system.

The cooperation knowledge acquisition phase followed, which involved a number of plenary sessions with the participation of the experts team, who critiqued the independent modules and detected conflicts at the points of interaction. The result of these sessions was the building of the cooperation layer primitives in each module, which took the form of the acquaintance models (*yellow-pages-table* and *interested-in-table*) in the agents and of the dictionary translation mechanism.

The interaction aspects addressed during this phase were as follows:

- It was established what information was owned by which expert module and in what form.
- The common representation and semantic mapping were performed for the parts of knowledge used in agent interactions. Examples of discrepancies which had to be solved are as follows:
  - *Syntactic heterogeneity:* If agent A_1 knows that *substance* = PCB and A_2 queries if the relation SUBSTANCE (PCB) is true, translation between the two formalisms is required.
  - *Semantic conflicts:* Conflicts, for example those due to different abstraction levels, need to be resolved. If agent A_1 knows that there is a ⟨*lake*⟩ near the accident and A_2 asks whether there is ⟨*surface-water*⟩ nearby, in order to coordinate the two agents, there is a need to introduce additional relations between the concepts ⟨*lake*⟩ and ⟨*surface-water*⟩.
- Decisions have to be made about the control relations between agents. For instance if A_1 requests the satisfaction of goal G_1 from A_2, while A_2 is busy with G_2, a decision has to be made about who exercises authority in this context.
- Conflicts and deadlocks, when identified, have to be resolved. If expert E_1 (and therefore agent A_1) establishes that under certain conditions goal $G_1 = \langle V_1 \rangle$ and expert E_2 (A_2) for the same conditions believes that $G_2 = \langle V_2 \rangle$, then the conflict has to be resolved (for instance by using a third expert's opinion) and the mechanism for resolving the conflict has to be registered in order for it to be included in the cooperation layer of the CKBS.

Strong interleaving of the two knowledge acquisition phases described was necessary for there to be convergence to a coherent system performance.

DCHEM ARCHITECTURE

Five agents were built following the process described. The number of agents built is typical of many CKBSs of high granularity*. The DCHEM agents form a hierarchy in terms of the goal structure of the main threat-estimation task. The top-goal is owned by the *threat-estimator* (TE) agent, who is activated by the UIA when evaluation of the top goal is requested. Thus agent TE is at the top of the agent hierarchy controlling the other agents.

This design decision in DCHEM, which is not necessarily typical of environmental CKBSs, is worth discussing at this point. The consequence of this design is that DCHEM is strongly dependent on the TE. If the TE fails or breaks down, the system cannot produce a solution. An alternative approach could have been to distribute the top-level goal ⟨threat-estimation⟩ to a number of agents, who, according to the circumstances (accident characteristics), could dynamically decide on who would take the leading role and build up the solution, coordinating the group. For instance, in an accident with a strong toxicity threat, the substance-identifier agent would lead the group, while in a fire accident where the threat comes mostly from thermodegradation, the pejorative transformation agent would take control.

In such a scenario, a negotiation phase involving the agents and possibly the user would be required initially in order to decide the roles. Also, knowledge redundancy would have been necessary, since the top-goal structure, in a modified form, has to be owned by different agents. The result of such an approach is expected to be higher robustness, more flexibility, and adaptive behaviour of the system under different conditions.

The reason why DCHEM designers opted for a hierarchical control is partly related to the particular group of experts involved, which maintained a strong hierarchical structure. Thus the division of expertise and the control regime of the system reflects the community of the domain experts involved. It is evident that this is a side effect of the knowledge acquisition technique used.

*Most of the reported high-granularity CKBSs are made up of 2–10 agents.

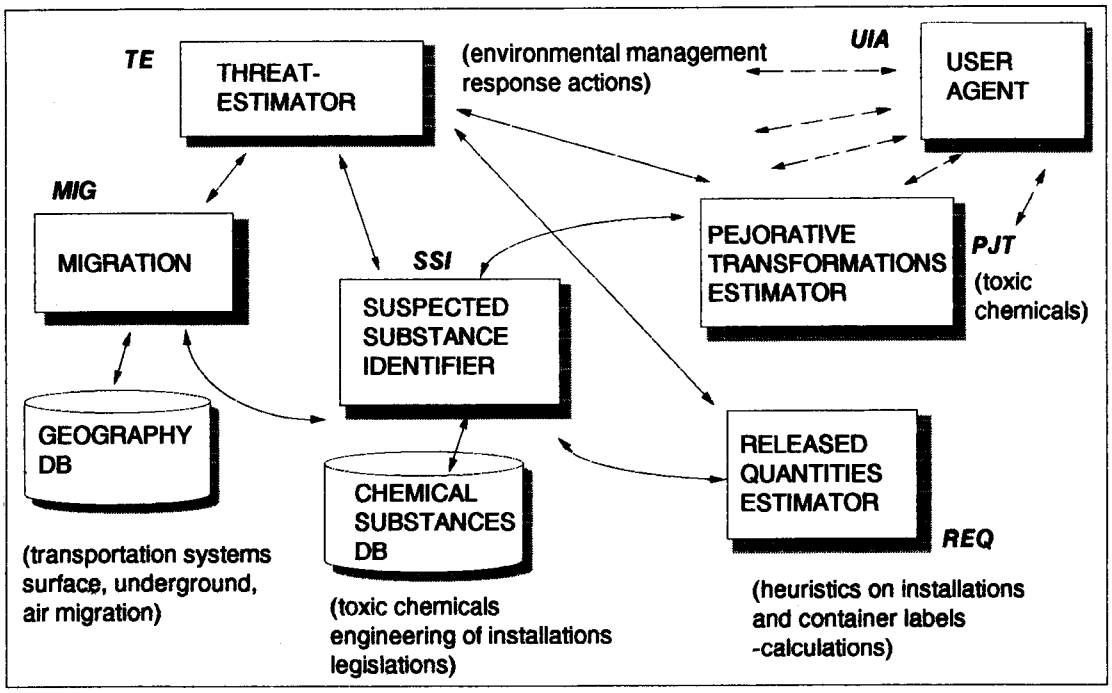

Figure 6 Organisation of DCHEM cooperating expert systems and corresponding areas of expertise

A brief description of the agents shown in *Figure 6* and their main interactions is as follows:

- The TE contains the expertise of the key-domain expert, and it can evaluate the emergency combining the high-level goals of the other specialised agents (see the following example):

  If the involved substance in the accident is ⟨PCB⟩, // provided by SSI agent and ⟨due to a fire⟩ ⟨PCB⟩ can be transformed into ⟨dioxin⟩, // provided by PJT and the amount of released substance is ⟨quite high⟩, // provided by REQ and the substance can be ⟨dispersed easily⟩ in the area, // provided by MIG
  Then it is estimated that the potential danger of the threat is ⟨high⟩ and protective measures ⟨M_1⟩ are recommended.

- The *suspected-substance-identifier* (SSI) determines what the suspected initial substance involved in the accident is. SSI bases its reasoning on information about the observational aspects of the unknown substance, description of the equipment involved and information from labels and documents. It may request the user to perform certain analytical tests or do additional observations. The knowledge contained in SSI is a combination of heuristics and knowledge of physical and chemical properties of this class of chemical substances (see the following example):

  If the substance is contained in ⟨electrical transformer of class TR_1⟩ and according to the label, the transformer is manufactured before 1981
  Then it is ⟨very-likely⟩ that the substance is ⟨PCB⟩.

  The reasoning is supported by a chemical substances database. SSI can request confirmation of certain subgoals by the PJT and REQ agents, as described in the following.
- The *pejorative-transformation agent* (PJT) establishes the likelihood of potential dangerous transformations of the chemical substances involved. The molecular structure of many substances can change over time, following complex chemical processes, like thermodegradation. PJT establishes such a danger, on the basis of a circumstantial description of the accident and the substance. PJT can make some assumptions about the class of the initial substance involved, overlapping partially with the goal structure of SSI. PJT's chemical substance identification is based on chemical classification rather than detailed information about the commercial product involved. This is a case in which the two agents can arrive at conflicting views. One possible solution to this problem is to build control in such a way that PJT starts its reasoning only when SSI has firmly established the ⟨initial-substance⟩ goal. An alternative is to, using partial overlapping, let PJT make some assumptions about ⟨initial-substance⟩ and proceed with reasoning about possible ⟨pejorative-transformations⟩. If, in the meantime, SSI's verdict on ⟨initial-substance⟩ is in conflict with PJT's assumption, negotiation between PJT and SSI needs to take place, which may result in backtracking of one of the agents. Circumstantial information needs to be shared by PJT and SSI. If, for instance, during reasoning, PJT needed to know the container type, it

would first query other agents, like SSI, and, if not satisfied, would refer to the user.

- The *migration agent* (MIG) determines the possibility of the migration of the substance via different pathways such as ground, air, drains, food, and surface and underground water in the surrounding environment. This depends on the physical properties and the quantities of the substance as well as the characteristics of the surrounding environment. MIG bases its reasoning on information supplied by SSI and PJT, with which it needs to interact, and a description of the pathways, requested from the user, or, in an alternative configuration, it can be requested by a geographical information system[64].
- Finally, the *released-quantity estimator* (REQ) estimates the released and residual quantities of the chemical substances involved. The estimation can be either numerical calculation based on measurements of the containers involved or estimations based on heuristics about the types of the containers, information volunteered by the on-site observers about released quantities etc. REQ serves agents PJT and MIG who request the result of its estimation ⟨residual-quantity⟩ and ⟨released-quantity⟩; however, MIG needs, in order to achieve its goal, information such as ⟨initial-substance⟩ and circumstantial information.

From this description of DCHEM agents, it is evident that all agents need observational and contextual information related to the accident. Since more than one of them may need the same information, they use their acquaintance models for the generation of queries. A hierarchy of possible destinations can be established so that a query of the form

(request $\langle x \rangle$ from agents $\langle A_1 \rangle$, $\langle A_2 \rangle$, $\langle A_3 \rangle$)

can be executed in the following form:

((request $\langle x \rangle$ from $\langle A_1 \rangle$)
if (not-satisfied) or (time-limit)
then
request ($\langle x \rangle$ from $\langle A_2 \rangle$) . . .) until the end of the list.

A default assumption in order to avoid ⟨null⟩ query replies is that, at the end of such a list, the UIA is always added, i.e. knowledge that cannot be provided by other agents is solicited from the user, who is considered a superexpert.

The problem solving components of DCHEM agents have been implemented in diverse languages. A limitation, however, imposed by the CooperA shell is that the only allowed formalisms are standard Lisp, Prolog and OPS-5, and their extensions CRL-Lisp, CRL-Prolog and CRL-OPS. In our case, the MIG and SSI agents were built as rule bases using CLR-OPS. The PJT agent was built in CRL-Prolog. Finally the REQ and TE agents were implemented in CRL-Lisp.

Observations on DCHEM performance

- The prototype developed had a performance similar to that of a previously developed nondistributed expert system[65] in the same domain.
- DCHEM demonstrated, however, increased flexibility and robustness, in terms of goal satisfaction in conditions of reduced reasoning capability. Thus, for instance, using the dynamic configuration feature of CooperA, reduced agent sets were used for handling the top goal. For example, when the agent set (TE, PJT, MIG, REQ) was given the ⟨threat-estimation⟩ goal, the system could arrive at the same conclusion as the full configuration, through a more detailed query to the user and a more intense interagent communication.
- The transparency of the problem solving was greatly increased by using this CKBS technique. The intermediate subgoals were clearly identified, and the interagent interactions facilitated the user's understanding of the progress of the problem solving. This was particularly supported by the dynamic visualisation feature of the CooperA workbench (see *Figure 4*).
- One limitation of this approach was the reduced coherency in the dialogue with the user. While, in a single-agent system, the user is requested to provide information which follows the line of reasoning, in a CKBS, the user can be forced to participate in interleaved dialogues with no apparent coherence, which can cause mental confusion.
- When semantic mapping was not properly performed among the various knowledge bases, effects were observed on the user dialogue such as that in the following example, in which semantically equivalent information is requested from the user by two agents:

  (Provide colour of released substance, ⟨brown, yellow, light-yellow⟩?)
  . . .
  (Please describe the suspected liquid ⟨dark⟩ or ⟨light-coloured⟩?)

  In general, the achievement of homogeneous interaction styles between the various agents and the user was found to be a particularly difficult task. A possible solution to this can be the introduction of a user modelling component at the user interface module which will 'interpret' and maintain a trace of the user dialogue, as suggested in Reference 50.
- Finally, the high modularity of the distributed system facilitates debugging and maintenance of the knowledge bases. The extendability of the system has also improved, since the introduction of new agents in the system is supported by the architecture. Thus, in the hypothetical case in which a new knowledge base relating to a different class of chemicals was made available, a new agent containing this knowledge would be introduced and the existing agents'

cooperation layers accordingly adapted in order to enable them to interact with the new component. This feature is particularly important in environmental domains, in which knowledge is highly volatile.

CONCLUSIONS

The relation between the emerging field of distributed artificial intelligence and in particular cooperating knowledge based systems and environmental decision support has been the focus of this paper. A general conclusion of this study has been that there is a strong correlation between the requirements of environmental problem solving and the proposed CKBS techniques. Thus it is expected that further introduction of CKBS in this field will increase the effectiveness of KB techniques, supporting the solution of complex environmental problems.

A survey of typical examples of existing CKBS environmental applications was undertaken. For the systems presented, the rationale for using CKBS techniques and the characteristics of the specific domain that led to such a decision have been discussed. Generalisation of the design decisions was also attempted in the frame of the survey. The conclusion is that the main reasons why CKBSs were used in environmental problem solving are related to the existence of multiple experts, a high geographical distribution of the problem-solving agents in cases of environmental monitoring systems, the need for the application of alternative problem solving methods in order to increase solution reliability, and the building of distributed environmental simulators.

A specific environmental CKBS was also described, in order to demonstrate in more detail the typical characteristics of such a solution and identify limitations and difficulties of the design process. In particular, the phases of knowledge acquisition involving multiple experts, coherency building among disparate knowledge bases, conflict identification and conflict resolution, and the selection of an adequate interagent control paradigm have been identified as key steps during system design.

DCHEM presents high modularity, extendability and maintainability. Through the architecture used, heterogeneous knowledge components are integrated into the system. However, the design and development process has increased in complexity: new phases have been introduced, such as coordination-knowledge elicitation and distributed control paradigm selection, as described in the examples discussed.

In relation to perspective work, it is felt that an area of particular importance for environmental decision making is that of negotiation protocols and consensus-building techniques. In DCHEM, the hierarchical nature of the control structure required limited use of such techniques. However in general, many emerging CKBS negotiation techniques can fall on fertile ground in complex environmental applications.

In conclusion, it is believed that the investigation of the applicability of CKBS techniques in environmental problems advocated by this paper can be beneficial to both environmental science and artificial intelligence. On the one hand, environmental researchers and system developers can be supported in seeking innovative architectures for the development of KB systems in these domains, and, on the other hand, DAI researchers can improve our methodological and architectural frameworks and distributed problem solving algorithms in order to meet the challenge of this highly demanding application area. It is hoped that, through the increased interaction of the two fields, our societies can ultimately improve performance in the tasks of environmental management and other complex problem solving domains.

ACKNOWLEDGEMENTS

The author is indebted to many researchers of the KBS Laboratory of the Joint Research Centre of Ispra, who have helped to shape up the ideas expressed in this paper over the years and contributed to building the CooperA/DCHEM system. Thanks are due to Flavio Argentesi, who inspired the whole project, Luca Bollini, who built the first CHEM prototype, Lorenzo Sommaruga and Marc Van Liedekerke, who participated in the development of CooperA/DCHEM, the domain experts Sergio Facchetti, Silvia Cerlesi, Giuseppe Belli, Walter Tumiatti and Vitto La Porta, and Esteban Gonzales, who meticulously evaluated and tested the DCHEM system. Special thanks are also due to the anonymous referees of this paper for their valuable comments on the early draft.

REFERENCES

1 Waterman, D A *A Guide to Expert Systems* Addison–Wesley, USA (1986)
2 Hushon, J M 'Expert systems for environmental problems' *Env. Sci. Tech.* Vol 21 (9) pp 838–841 (1987)
3 Conway, B J 'Expert systems and weather forecasting' *Meteorological Magazine* Vol 118 (1399) pp 23–30 (1989)
4 Lein, J K 'An expert system approach to environmental impact assessment' *Int. J. Environ. Stud.* Vol 33 (1–2) pp 13–27 (1989)
5 Weis, J S 'The status of undergraduate programs in environmental science' *Environmental Science and Technology* Vol 24 (8) pp 1116–1121 (1990)
6 Bardwell, L V 'Problem framing: a perspective on environmental problem-solving' *Environmental Management* Vol 15 (5) pp 603–612 (1991)
7 Gadomska, M 'Sociological theory perspective on computer systems design and application. Introduction to research' *JRC Technical Note TN/I.90.112* Ispra (1990)
8 Gadomska, M and Paruccini, M 'A sociological perspective on decision support systems in environmental management' *EUR Report 12583EN* JRC Ispra (1990)
9 Hern, E C L 'On distributed artificial intelligence' *Knowledge Engineering Review* Vol 3 (1) p 21–57 (1988)
10 Huhns, M N (Ed.) *Distributed Artificial Intelligence* Pitman, UK, (1987)
11 Bond, A H and Gasser, L *Readings in Distributed Artificial Intelligence* Morgan-Kaufmann, USA (1988)
12 Gasser, L and Huhns, M (Eds.) *Distributed Artificial Intelligence* (Vol II) Pitman, UK (1989)
13 Demazeau, Y and Muller, J P (Eds.) *Decentralised Artificial Intelligence* North Holland (1990) (*Proc. 1st MAAMAW*)
14 Avouris, N M and Gasser, L (Eds.) *Distributed Artificial Intelligence: Theory and Praxis* Kluwer, Netherlands (1992)
15 Werner, E 'Cooperating agents: a unified theory of communication and social structure' *in* Gasser, L and Huhns, M (Eds.)

Distributed Artificial Intelligence (Vol II) Pitman, UK (1989) pp 3–36
16 Gasser, L 'Social conceptions of knowledge and action: DAI foundations and open systems semantics' *Artificial Intelligence* Vol 47 pp 107–138 (1991)
17 Malone, T W and Crowston, K 'Toward an interdisciplinary theory of coordination' *MIT Report CCS TR#* 120 (1991)
18 Erman, L D, Hayes-Roth, F A, Lesser, V R and Reddy, D R 'The Hearsay-II speech-understanding system: integrating knowledge to resolve uncertainty' *Computing Surveys* Vol 12 (2) pp 213–253 (1980)
19 Parunak, H V D 'Manufacturing experience with the contract net' *in* Huhns, M N (Ed.) *Distributed Artificial Intelligence* Pitman, UK (1987) pp 285–310
20 Findler, N V and Lo, R 'An examination of distributed planning in the world of air traffic control' *Journal of Parallel and Distributed Computing* Vol 3 pp 411–431 (1986)
21 Lesser, V R and Corkill, D D 'The Distributed Vehicle Monitoring Testbed: a tool for investigating distributed problem solving networks' *AI Magazine* pp 15–33 (Fall 1983)
22 Durfee, E H *Coordinating Distributed Problem Solvers* Kluwer, USA (1988)
23 Wittig, T (Ed.) *ARCHON: An Architecture for Multi-Agent Systems* Ellis Horwood, UK (1992)
24 Bobrow, D G 'Dimensions of interaction' *AI Magazine* pp 64–80 (Fall 1991)
25 Engelmore, R S and Morgan, A J *Blackboard Systems* Addison–Wesley, UK (1988)
26 Corkill, D D, Gallagher, K Q and Murray, K E 'GBB: a generic blackboard development system' *Proc. National Conf. AI* pp 1008–1014 (1986)
27 Agha, C *Actors: A Model of Concurrent Computation in Distributed Systems* MIT Press, USA (1986)
28 Agha, C 'Concurrent object oriented programming' *Communications ACM* Vol 33 (9) pp 125–141 (1990)
29 Smith, R G and Davis, R 'Frameworks for cooperation in distributed problem solving' *IEEE Trans. Systems, Man and Cybernetics* Vol 11 (1) pp 61–70 (1981)
30 Decker, K S 'Distributed problem solving techniques: a survey' *IEEE Trans. Systems, Man and Cybernetics* Vol 17 pp 729–740 (1987)
31 Gasser, I and Briot, J P 'Object-based concurrent programming and distributed artificial intelligence' *in* Avouris, N M and Gasser, L (*Eds.*) *DAI Theory and Praxis* Kluwer, Netherlands (1992) pp 81–107
32 Gasser, L, Braganza, C and Herman, N 'Implementing distributed artificial intelligent systems using MACE' *in* Huhns, M N (Ed.) *Distributed Artificial Intelligence* Pitman, UK (1987) pp 119–152
33 Jennings, N and Wittig, T 'Archon theory and practice' *in* Avouris, N M and Gasser, L (Eds.) *DAI Theory and Praxis* Kluwer, Netherlands (1992) pp 179–195
34 Avouris, N M, Van Liedekerke, M H and Sommaruga, L 'Evaluating the CooperA experiment: the transition from the expert system module to a distributed AI testbed for co-operating experts' *Proc. AAAI 9th DAI Workshop* Washington, USA (1989) pp 351–366
35 Sycara, K 'Resolving goal conflicts via negotiation' *Proc. AAAI 1988* (1988) pp 245–250
36 Klein, M 'Supporting conflict resolution in cooperative design systems' *IEEE Trans. Systems, Man and Cybernetics* Vol 21 (6) pp 1379–1391 (1991)
37 Lesser, V R 'Retrospective view of FA/C distributed problem solving' *IEEE Trans. Systems, Man and Cybernetics* Vol 21 (6) pp 1347–1362 (1991)
38 Durkin, J 'DUSTPRO: a distributed expert system for coal mine dust control' *ESD/SMI Expert Systems Proc.* Detroit, MI, USA (12–14 April 1988) pp 377–387
39 Hartvigsen, G and Johansen, D 'StormCast: a distributed artificial intelligence application for severe storm forecasting' *Distributed Computer Control Systems Proc. Eighth IFAC Workshop* Pergamon, UK (1988) pp 99–102
40 Hartvigsen, G and Johansen, D 'Co-operation in a distributed artificial intelligence environment — the StormCast application' *Engineering Applications of Artificial Intelligence* Vol 3 (3) pp 229–237 (1990)
41 Avouris, N M, Lekkas, G P and Viras, L 'Experts and systems controlling air pollution episodes. Parts II and III: Quarterly reports' *JRC Technical Note TN/I.93.03* JRC Ispra (1993)
42 Pham, H N and Wittig, T 'An adaptable architecture for river quality monitoring' *in* Assimakopoulos, D and Avouris, N (Eds.) *Environmental Informatics Applications* Kluwer (1994)
43 Hausen-Tropper, E B 'A decision and consultation tool for weather prediction' *Proc. 10th Avignon Int. Conference on Expert Systems and their Applications* Vol 2 Nanterre, France (1990) (*Publication EC2*)
44 Cohen, P R, Greenberg, M L, Hart, D M and Howe, A E 'Trial by fire: understanding the requirements for agents in complex environments' *AI Magazine* Vol 10 (3) pp 33–48 (1989)
45 Ferber, J and Drogoul, A 'Using reactive multi-agent systems in simulation and problem solving' *in* Avouris, N M and Gasser, L (Eds.) *DAI Theory and Praxis* Kluwer, Netherlands (1992) pp 53–80
46 Gonzalez, Y J E 'Distributed ChEM – ChEM on CooperA' *JRC Technical Note I.90.30* Institute of Systems Engineering and Informatics, Ispra (1990)
47 Sommaruga, L, Avouris, N M and Van Liedekerke, M H 'An environment of experimentation with interactive co-operating knowledge-based systems' *in* Shadbolt, N (Ed.) *Research and Development in Expert Systems* (Vol VI) Cambridge University Press, UK (1989)
48 Hanks, S, Pollack, M E and Cohen, P R 'Benchmarks, test beds, controlled experimentation, and the design of agent architectures' *AI Magazine* Vol 14 (4) pp 17–42 (1993)
49 Montgomery, T and Durfee, E 'Using MICE to study intelligent dynamic coordination' *Proc. 2nd Int. Conf. Tools for Artificial Intelligence* Washington DC, USA (1990) pp 438–444
50 Avouris, N M, Van Liedekerke, M, Lekkas, G and Hall, L E 'User interface design for cooperating agents in industrial process supervision and control applications' *Int. J Man–Machine Studies* Vol 24 pp 873–890 (1993)
51 Avouris, N M 'User interface design and DAI applications: an overview' *in* Avouris, N M and Gasser, L (Eds.) *DAI Theory and Praxis* Kluwer, Netherlands (1992) pp 141–162
52 Hall, L E and Avouris, N M 'Methodological issues of DAI applications interface design: transparency analysis' *in* Avouris, N M and Gasser, L (Eds.) *DAI Theory and Praxis* Kluwer, Netherlands (1992) pp 163–178
53 Hall, L E, Macaulay, L and O'Hare, G 'User interaction with distributed intelligent systems' *Proc. MAMMAW 1991* Rome, Italy (May 1991)
54 Hushon, J M 'Response to chemical emergencies' *Environ. Sci. Tech.* Vol 20 (2) pp 118–121 (1986)
55 Hushon, J M 'Expert systems to assist first responders to chemical emergencies' *Expert Systems with Applications* Vol 2 pp 129–135 (1991)
56 Michalowski, W, Kersten, G, Koperczak, Z and Szpakowicz, S 'Disaster management with NEGOPLAN' *Expert Systems with Applications* Vol 2 pp 107–120 (1991)
57 Miller, S 'The persistent PCB problem' *Environ. Sci. Tech.* Vol 16 (2) pp 98–99A (1982)
58 Boose, J H 'Rapid acquisition and combination of knowledge from multiple experts in the same domain' *Future Computing Systems* Vol 1 (2) pp 191–216 (1986)
59 Kitto, C M and Boose, J H 'Knowledge acquisition tools for different problem solving paradigms: research at Boeing Computer Services' *BCS Report* Boeing, USA (1987)
60 La Salle, A J and Medsker, L R 'Computerized conferencing for knowledge acquisition from multiple experts' *Expert Systems with Applications* Vol 3 pp 517–522 (1991)
61 Liou, Y I 'Collaborative knowledge acquisition' *Expert Systems with Applications* Vol 5 pp 1–13 (1992)
62 Ignizio, J P *Introduction to Expert Systems. The Development and Implementation of Rule-Based Expert Systems* McGraw–Hill, USA (1991)
63 Lekkas, G P, Avouris, N M and Papakonstantinou, G K 'Development of distributed problem solving systems for dynamic environments' *IEEE Trans. System, Man and Cybernetics* (to appear)
64 Avouris, N M and Finotti, S 'User interface design for expert systems based on hierarchical spatial representations' *Expert Systems with Applications* Vol 6 (2) pp 109–118 (1993)
65 Argentesi, F, Bollini, L, Facchetti, S, Nobile, G, Tumiatti, W, Belli, G, Ratti, S, Cerlesi, S, Fortunati, G U and La Porta, V 'ChEM: an expert system for the management of chemical accidents involving halogenated aromatic compounds' *Proc. World Chemical Accidents Conference* Rome, Italy (1987) pp 227–230

APPENDIX

Acronyms and abbreviations

AAM: agent acquaintances model
ARCHON: Architecture for Cooperating Heterogeneous On-Line Systems (Esprit P2256)
CKBS: cooperating knowledge-based systems
CooperA: Cooperating Agents System (JRC system)
DAI: distributed artificial intelligence
DCHEM: Distributed Chemical Emergencies Manager
DVMT: Distributed Vehicle Monitoring Testbed (University of Amherst, USA, system)
MACE: Multi-Agent Computing Environment (University of Southern California, USA, system)
MICE: Michigan Cooperation Environment (University of Michigan, USA, testbed)
MIG: (DCHEM agent) migration agent
PJT: (DCHEM agent) pejorative transformation agent
REQ: (DECHEM agent) released quantity estimator
SSI: (DCHEM agent) suspected substance identifier
TE: (DCHEM agent) threat estimator
UIA: user interface agent