Cooperating for assisting intelligently operators

P. Brézillon [1] and E. Cases [1, 2]

(1) LAFORIA, case 169, University Paris 6
4, place Jussieu, 75252 Paris Cedex 05, France
Tel: 33 1 44 27 70 07 - Fax: 33 1 44 27 70 00
E-mail: {brezil, cases}@laforia.ibp.fr

(2) DDC/DRDD/SSP/LIA, CEA BP 171
30207 Bagnols-sur-Cèze

Abstract:
We are in the process of design of an intelligent cooperative system in a nuclear plant application. The system must cooperate with an operator who accomplishes a task of supervision of a real-world process. We point out in the paper that a cooperation between a cooperative system and an operator has two modes: a waking state and a participating state. During the waking state, the system observes the operator's behavior and the consequences on the process. During the participating state, the cooperative system builds jointly with the user a solution to a problem. In our approach, the cooperation depends of the system capabilities to explain, to incrementally acquire knowledge and to make explicit the context of the cooperation. We develop these ideas in the framework of the design of the cooperative system in the nuclear plant application.

Keywords: intelligent cooperative system, cooperation, explanation, incremental knowledge acquisition, context

Résumé:
Nous sommes en cours de conception d'un système coopératif intelligent pour une application dans le domaine du nucléaire. Le système doit coopérer avec un opérateur qui accomplit une tâche de supervision d'un processus industriel. Nous montrons dans ce papier qu'une coopération entre un système coopératif et un opérateur présente deux modes : un mode de veille et un mode participatif. En mode de veille, le système observe le comportement de l'opérateur et les conséquences sur le processus industriel. Dans le mode participatif, le système va construire conjointement avec l'opérateur la solution au problème qui se pose. Dans notre approche, la coopération du système dépend essentiellement de ses capacités à expliquer, à acquérir incrémentalement des connaissances et à expliciter le contexte de la coopération. Nous développons ces idées dans le cadre de la conception de notre système coopératif dans le domaine du nucléaire.

Mots clés: système coopératif intelligent, coopération, explication, acquisition incrémentale de connaissances, contexte

In the Proceedings of the International Workshop on the Design of Cooperative Systems, January 1995, Juan Les Pins, France, pp. 370-384.

1.0 INTRODUCTION

After the initial interest for Knowledge-Based Systems (KBSs), one notes that KBSs have not respected their promise. Studies on the use of KBSs in real-world applications permit one to point out five main failures [Karsenty et al. 94].

The first failure concerns the non intervention of the user in the problem solving. This relies on the fact that KBSs are supposed to be oracles and users as novices. However, unexpected situations are rather the norm than the exception, and KBSs cannot solve such situations when users, with their practical experience, may solve them. This failure is due to the lack of consideration of the user-system cooperation and the context in which a problem is solved.

The second failure arises when KBSs do not use correctly their knowledge. Knowledge being acquired from human experts has a high contextual component that is generally not acquired with the knowledge. This failure may be overcomed if the user may intervene in the problem solving to cooperate in the definition of the limits of use of knowledge.

The third failure is because KBSs cannot initially have all the needed knowledge. Any KBS has bounded resources for problem solving and bounded influence: One can never anticipate or "design away" all the misunderstandings and problems that might arise during the use of such systems [Fischer 90]. This implies that knowledge must be acquired incrementally from users when needed, i.e., in a given context of use.

The fourth failure is that KBSs cannot generate relevant explanations for users because they do not know what the user's problem solving context is. The unique way to generate an explanation is that the KBS and the user jointly construct the explanation [Karsenty et al. 94]. This implies that one must take into account the context in which a KBS and a user cooperate for problem solving.

The fifth failure is the impossibility of KBSs to deal with tasks where alternative and unanticipated solutions are the norm rather than the exception. Random in the choice of a given solution does not lead to the better solution. Generally, users may choose among alternative solutions and may use their experiences to face unanticipated solutions. Thus, a KBS must cooperate with users to efficiently solve problems.

The hypothesis of our work is that for intelligently support users, KBSs must be able to cooperate. Among other things, this implies that acquiring knowledge incrementally and explaining must be considered as intrinsic part of any problem solving where the user has a crucial role to play. The goal of the paper is to specify the needed elements of an Intelligent Cooperative System (ICS) to support operators facing real-world processes.

Three types of cooperation are considered in the literature: human-human cooperation, machine-machine cooperation and human-machine cooperation. In each domain, the cooperation is modeled on the basis of various elements. Some elements are considered in different domains, others are specific of a given domain.

We address human-machine cooperation to support an operator who has in charge a real-world process. Operators are not novices but have a practical experience that may be complementary of those of the expert with which the KBS has been developed (i.e., a type of theoretical experience). In all situations, operators have the final decision, and, often, several operators successively intervene on the real-world process during a day. This implies some constraints on the type of cooperation between a human and an ICS. For instance, the system must be able to observe the operator's work and only intervene on operator's request. We claim that cooperative systems in that situation must present some specificities that are not considered in other domains: one must deal with a human, a system and a real-world process.

Cooperative systems are based on a successful combination of human skills and computing power in carrying out a task which cannot be done either by the human or by the computer alone [Fischer 90]. The goal is to construct systems that augment and amplify human intelligence in problem solving, decision making and information management rather than just replacing it. Such systems must be active agents working in real-time and in synchronization with the user [Clarke et al. 93]. They appear as a generalization of the intelligent interfaces that are considered in the literature, although the ICS has bounded resources for problem solving and bounded influence: One can never anticipate or “design away” all of the misunderstandings and problems that might arise during the use of such systems [Fischer 90].

This paper relies on a bibliographical report on cooperation in various domains [Brézillon 94c]. In particular, the report discusses various works which emphasize on systems, methodologies, and languages. We pay a particular attention at the role of explanations in cooperation because it has been shown that explanations represent about one third of human-human dialogues [Karsenty et al. 94].

Hereafter, we only consider cooperation in the framework of the design and development of an intelligent cooperative system to support an operator that has in charge a complex real-world process. Thus, Section 2 presents the framework in which we consider cooperation. In Section 3, we discuss the main elements of cooperation emphasizing the role of explanation and context for ICSs. Section 4 presents the way in which the design and development of an ICS is considered in our application.

2.0 CONTEXT OF THE STUDY

Real-world processes are increasingly complex and techniques coming from various domains are needed. Nuclear and chemical plants are such processes. Operators who have in charge such processes can not tackle alone the supervision of them and computer systems support them in their task. Such computer systems contain models of the real-world processes and all the heterogeneous knowledge that is needed. The operator and the system have complementary knowledge and their competences must be assembled at the level of the joint cognitive system composing of the human and the system [Woods et al. 90]. We plan to develop an intelligent cooperative system to cooperate with operators in order to relieve operators of some tasks and provides them with fruitful information for the final decision they have to take. Several applications are developed in this framework [Brézillon 94a].

The term “operators” must be distinguished from the term "users" in that the former implies a greater involvement with the system, presumably one where the person is more uniquely assigned to the device. Experienced operators are competent agents with a high level of expertise: They have in charge the survey of the real-world process. Their interventions can not be limited to solution proposals. They also are able to evaluate, plan, generate partial solutions, try to explain the problem and identify a solution [Karsenty et al. 94]. Moreover, they may volunteer information, i.e., they can make statements about the domain which are out of context with respect to the current step of the problem solving. Volunteering information allow operators to focus the attention of the system on the information which they feel relevant. Thus, operators must take an active role in deciding what the system is reasoning about. Generally, complex real-world processes require a 24-hours supervision by different operators. The recognition of different needs and characteristics of operators (and even for a given operator at different times of a day) must be taken into account at the early steps of the design of such systems.

An ICS is like Janus with two faces (i.e., two interfaces): one towards the operator, the other one towards the real-world process. It must observe together the process behavior, the operator's behavior and the interaction among them. The cooperation for an ICS corresponds to two states: a waking state and a participating one. In the waking mode, the ICS observes operators' actions on the process and only intervenes when a crucial situation occurs. In this case, the ICS first analyze the operator's actions on the process and determine if the problem can be ruled out. If so, the ICS does not intervene, otherwise, the ICS alerts the operator and enters in a participating phase on operator's request. Thus the participating phase is triggered by either the detection of an abnormal state of the real-world process or a request from the operator. For the waking and the participating modes, the ICS needs an operator's model. Such a model must permit the ICS to identify the operator's intentions from sequences of actions. The ICS is mainly concerned with operator's intentions, i.e., possible action sequences, in order to analyze their behaviors, compare them to the process behavior, interpret the consequences, and take the initiative to alert operators when a misfit is detected.

3.0 SOME ELEMENTS OF COOPERATION

In this section, we present the main elements of cooperation encountered in the literature in order to discuss them in the framework of the design of ICSs. We focus after three elements that are seldom evoked in the literature on cooperation, namely explanation, incremental knowledge acquisition and context.

3.1. Usual elements evoked in the literature

Various elements are evoked to specify cooperation: decomposition and allocation of tasks, coordination, collaboration, organization, negotiation, conflicts, initiative, control, dealing with alternative solutions, etc. Generally, any combination of some of these elements leads to a cooperation model that generally depends on the nature of the application. Moreover, a partial result at a step of the problem solving may imply, for instance, a re-ordering of domain and task knowledge of all agents, a turn of initiative, and a change of the control. This sequence of interventions leads to an iterative refinement of the solution to satisfy the constraints managed by the operator [Zacklad 93; Darses et al. 93].

The task analysis aims to determine tasks that each agent may accomplish alone, in cooperation or alternatively with the other agent. The analysis may be made from books and direct elicitation with the operators. Each task can be decomposed into a number of plans, each plan being a hierarchy of goals with actions as terminal nodes. Such an analysis must be done along two dimensions: a functional one and a cognitive one. Moreover, the asymmetry of the agents (i.e., the human and the machine) must be exploited. The human and machine agents contain partial and overlapping expertise that, if integrated, can result in better joint system performance than is possible by either agent alone [Woods et al. 90]. For instance, the machine may compute large quantities of calculus, and the human may produce interpretations of data provided by the machine [Fischer 90]. This implies creating an environment where the refinement of solutions can be based on argument and the resolution of differing viewpoints [Clarke et al. 93].

A coordination mechanism must set up a relationship between agents wherein one agent can “count on” the actions of the other. To coordinate its actions with those of the user, a system needs to represent and reason about the knowledge, actions, and plans of the user. The coordination may be simplified by using standard communication protocols and by relying upon global viewpoints [Bond et al. 88]. Shared (or mutual) knowledge and plans are important factors in coordination, as it helps an agent to anticipate the other agent's behavior, minimizing explicit dialogue about their shared plan. Sharing of knowledge permits to increase the shared context of the cooperation, and, then, decrease the need to be verbose.

The agents have different goals, different ways of achieving similar goals, knowledge, constraints, resources and evaluation criteria. Mainly, a conflict may occur when they have different interpretations on the solution that is built, mainly when the agents' competences overlap each other. Generally, further action in the problem solving can not be considered before an agreement is found between them.

For insuring an efficient communication between the user and the system, one must design a kind of interlingua, even if each agent possesses an internal language [Davis et al. 88]. The notion of communicative acts permits to coordinate actions and exchange knowledge among agents [Wong 93]. Tasks of information transfer (and communicative acts) are different of the other tasks of the problem solving. Communication protocols are sequences with precisely defined steps to convene more complex intentions than single communicative acts. They impose the sequences of interactions among cooperating agents to solve specific classes of problems. Effective human-computer communication requires providing the system with a considerable body of knowledge about the world to agree on terms of reference to be adopted during the process [Fischer 90; Clarke et al. 93]. Establishing this "common ground" in the form of shared understanding and agreement between them is a crucial aspect of communication. (Knowing what to do not communicate also is important.)

3.2. Explanation

Studies of human-human dialogues show that about one third of dialogues are explanations, and most of them are uttered spontaneously without questions [Karsenty et al. 94]. Explanations may influence how the problem solving proceeds as partial solutions are negotiated among agents. They may be a condition to go to the next action and modify the direction taken by the cooperation [Karsenty et al. 94]. The two major effects of explanations on cooperation are: (i) explanations may modify the agents' representation of the problem, and (ii) explanations may modify the agents' problem solving knowledge.

The user and the system may have different interpretations on the current state of the problem solving. The differing interpretations will be compatible if the user and the system: make proposals, explain their viewpoints and spontaneously produce information [Karsenty & Brézillon 94]. In order to align the system's reasoning with that of the user and vice versa, the user and the system must co-construct the explanation in the current context of the problem solving. Thus, explanations become an intrinsic part of the problem solving and the line of reasoning of the system may be modified by explanation. It appears necessary to explain for cooperate and cooperate to explain.

In human-machine cooperation, the machine and the user may have to explain.The role of an explanation is to persuade the other agent that a partial solution is correct, not just to provide a final justification of an already determined completed solution. For instance, an explanation permits the user to know: (1) what the system can or cannot do; (2) what the system has done; (3) explain what the system is trying to do; (4) why it is doing what it does by responding to the user's clarification questions. Explanations aim to modify the knowledge of the other agent (bringing new pieces of knowledge or revising existing ones); to clarify the context of the shared knowledge; to convey new information pieces unknown by the explainee; to reach a common agreement; to coordinate the agents' activities; to rethink our own implicit assumptions, forcing us to trace our own reasoning process and revealing alternative ways to think about a problem.

3.3. Incremental knowledge acquisition

The generation of explanations by the user to the system implies that the system may incrementally acquire knowledge, assimilate it in its knowledge base, and propagate the consequences of the new knowledge. Knowledge in current Knowledge-Based Systems (KBSs) is now acquired in a monolithic manner beforehand the use of the KBS. For instance, the faults in a diagnostic hierarchy are defined and their basic form of knowledge representation (frames, meta-rules, etc.) is used as a template to acquire knowledge. The knowledge is tested only when the system reasons. If a problem with the knowledge is detected, the developer is forced to modify the knowledge, re-load (re-compile or re-interpret) and then test the reasoning again. Capturing and using knowledge in the context of use greatly simplifies knowledge acquisition because knowledge provided by experts is always in a specific context and is essentially a justification of the expert's judgment in that context. It would be much more useful if, first the system acquired knowledge when needed, and, second, had the ability to incrementally modify the acquired knowledge.

Through the cooperation between the system and the user and the explicit definition and use of context, new knowledge is acquired and validated. This knowledge is acquired through clarification dialogue. If the system can identify gradually through its dialogue with the user, the target knowledge, it can verify if this knowledge is missing or not. Incremental knowledge acquisition plays an important role in two situations:

- First, the knowledge missing in the current context must be acquired when needed in order to permit the problem solving to resume. For missing knowledge in the system, the user should be provided with the capability to add new knowledge through explanation. Here, explanations permit to make explicit the context of the knowledge use. In the case of interactive knowledge acquisition through explanation, a user introduces a pseudo-formal specification for a change in the knowledge that is evaluated with respect to the other knowledge in the system according to the four-step process described above.

- Second, a chunk of knowledge may be known by the system but used incorrectly, i.e., a link for the current context is missing. This link could be added in the form of a consequence of the current context. Here, incremental knowledge acquisition will focus on the refinement of the context of use of that chunk of knowledge, and explanations must support such a refinement.

For the user to interactively add knowledge, the system must be able to acquire the knowledge in a structured and general manner. For this reason, it is essential that the knowledge in the system be structured and tractable. Structured and tractable systems are key in traditional software engineering systems which use modules to compartmentalize information (whether it is data or procedures) and localize side effects in executing software. Indeed, the transfer of information between the user and the system has a high contextual nature. All KBS practitioners have yet to accept the obvious need for adopting some of the traditional and very successful software engineering methods. However, some KBS developers have argued for the explicit and structured representation of knowledge [Chandrasekaran et al. 83].

3.4. Context

Context is what must be shared by each agent in a cooperation at each step of the problem solving [Cahour 93]. This permits the adjustment to face all the implicit things in dialogue, extend the possibilities of the common language between agents, and have multiple representation of the same underlying data sets [Branki et al. 93a]. Cooperation thus must imply a transfer of knowledge for the problem solving and the transfer of contextual information between agents.

The cooperation context contains items like the dialogue memory, the task at hand, the spatio-temporal situation (place, moment, elements, ...), and the psycho-social situation (user model, roles and social status, ...). Making explicit a context in an operator question, the generated explanation should reference it much as a human explaining within a particular context would [Abu-Hakima 93].Contextualization has the advantages of: (i) Reducing the explainer's effort and allowing to reach a point of mutual understanding more rapidly if the answer confirms the explainee's assumption; and (ii) Allowing the explainer to predict an explanation need if the answer differs from the explainee's assumption.

Acts of explaining convey contextual information that is missing from the explainee's context. When two agents disagree, each one may offer explanation to try and get the other participant to change their beliefs, so that agreement may be achieved. In such a situation, explanations aim to adjust both agents' contexts to reach compatible interpretations by an incremental acquisition of the missing knowledge.

As a first approximation, the context of the cooperation may be assimilated to the screen that is a visual field shared by the user and the system. On the screen, there may have as much possible contexts as windows. The current context is the activated window where both the user and the system may add, change, delete information. For instance, pressing F1 within the context of an explanation reasoning screen, provides context sensitive reasoning explanations (procedural knowledge) and the possibility to navigate to different explanation strategic levels. Pressing F1 within a data entry context provides context-sensitive explanations for the required data and its use (declarative knowledge), along with the navigation facility.

Making explicit the context in cooperation permits to generate explanations cooperatively, because the context of the cooperation determines the explanation needs. Explanations must be given at the right level of detail and with the right level of assumed shared understanding about the other agent's knowledge [Fischer 90]. For the system to acquire knowledge incrementally, the context of reasoning must be made explicit to be acquired and explained. This context represents, e.g., the traversal of nodes through various actions in a particular context. In this manner, the system is given the ability to acquire knowledge incrementally. New knowledge is acquired and validated through user-system cooperation and the explicit definition and use of context. Explanations are a means to facilitate the acquisition of the missing knowledge that will be acquired when needed.

Conversely, explanations are a mean for anticipating the explanation needs of the other when they are produced spontaneously. They permit to enrich the context that is shared by both agents, leading to a mutual understanding and a mutual acceptance, and thus facilitating their cooperation. Spontaneous explanations have numerous advantages: they make understanding easier, reduce misunderstandings, insure that the interpretation of the agents' solutions was the one they wanted to communicate, they decrease the number of dialogue exchange needed to achieve the interaction goal, they increase the shared knowledge. However, an ICS would have to decide when it is necessary to volunteer explanations. We claim that the main goal of spontaneous explanations is the validation of the cooperation context.

3.5. Conclusion

In this section, we mainly point out the need to make explicit the roles of explanation, incremental knowledge acquisition and context, and their relationships. If explanation is an important part of cooperation, it implies a revision of the design and the development of cooperative systems. Generation of explanation must be understood from the system to the user, and from the user to the system. The former has been largely studied in early knowledge-based systems. The latter is something relatively new. This supposes that the system may incrementally acquire knowledge when needed, i.e., in given contexts which also are to be acquired.

4.0 THE INTELLIGENT COOPERATIVE SYSTEM (ICS)
4.1. Elements of an ICS

A cooperative system has two sides: the real-world process side and the human side. One change the system (process side) at the design time and the interface (human side) through training and experience. This implies that an ICS has to understand: (a) the real-world process; (b) the tasks on the process; and (c) the operator's behavior. This constitutes three inter-dependent knowledge bases.

The Figure 1 presents the three elements, namely the operator, the ICS and the real-world process, and the three types of knowledge that are necessary at the ICS to cooperate. The three types of knowledge that are necessary to an ICS, are:

(a) The real-world process (Process model).
This type of knowledge corresponds mainly to a model of the process. The model permits the ICS to control the process behavior by a comparison with the simulated behavior under the same inputs given to both the process and its model. The three goals are: to control the evolution of the process; to verify the coherence between variation of the process behavior and operator's actions; and to permit the operator to simulate alternative solutions before to take a decision. Models of real-world processes generally exist beforehand because they are important tools to control the process by simulation. We use such models in our application to design rapidly a pseudo-process and a model of this pseudo-process.

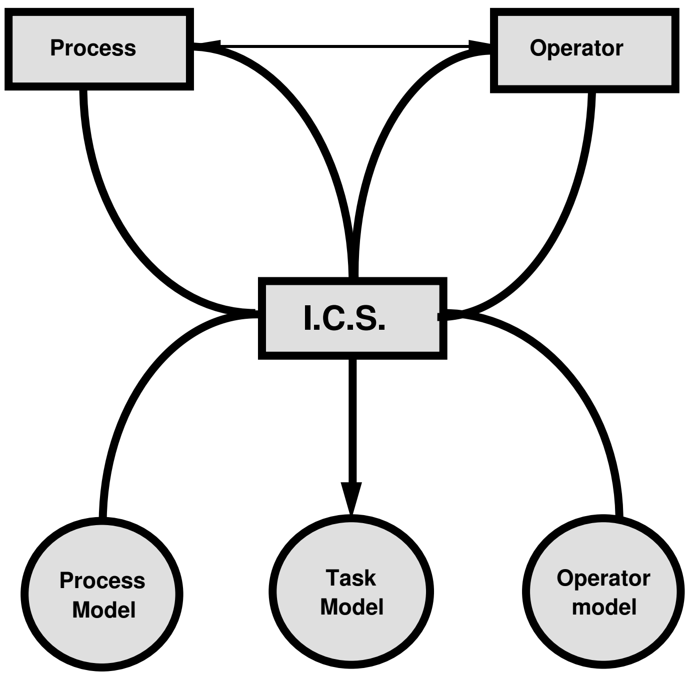

FIGURE 1: Relationships of an ICS with other actors and knowledge source

(b) The supervision task (Task model).
Knowledge on the supervision task permits the ICS, first to model the task, and, second to simulate the operator's activity. The goals are threefold: to identify the operators' intentions from

their action sequences, to control and explain the process behavior. Eventually, the ICS may correct the operator and suggest alternative sequences (e.g., short-cuts). The task analysis may be established in two steps. The first step corresponds to an elicitation of knowledge from operators and a use of reports, books and related matter. This permits to develop a first model of the task that operators can use, among other things, to choose the best representation of this knowledge, and to validate the approach. The second step starts with the model obtained at the first step. This may be done either "on-line" when the operator intervenes on the real-world process or at a differed time when the operator is not under time pressure. Note that the ICS is then only an observer and stays in an attentive waking state.

(c) The operators (Operator model).
This type of knowledge corresponds to the action sequences of operators facing a problem in the process. From operator's actions, the ICS may deduce from knowledge in (a) and (b) operator's intentions, and their preferences from their choices during the solving of problems (e.g., shortcuts). Identifying later a similar situation, the ICS may propose to any operator a similar solution that is chosen in the spirit of the case-based reasoning. One may proceed in the same way as for the modelling of the process, i.e., a short elicitation phase and an incremental development of the knowledge base during its use. Recall that dealing with experienced operators having to take the final decision, the operator model only concerns their actions on the real-world process in given situations (i.e., a change of state of the process in given conditions) to support them in their task. One may enrich the operator model by adding their preferences which may be determined automatically from a series of problem solvings.

This approach presents the advantage to have rapidly a mockup that will be improve incrementally on the knowledge bases as well as on the development of the visible part of the ICS, namely the interface. The ICS may learn (i.e., acquire knowledge) from the behaviors of the operator and the real-world process. This is an incremental knowledge acquisition where knowledge is acquired when needed. Only a kernel of knowledge has to be elicited directly from operators, mainly to select the right representation formalism for knowledge. (For managing knowledge, the system must be able to accomplish tasks as: acquisition, assimilation, learning, validation-verification, information retrieval, indexing.)

The ICS must intervene spontaneously at a statement of the operator, such as: announcement of the procedure the operator intends to follow, the goal to be reached, the interpretation they make of some interface object, etc. [Darses et al. 93]. It also must initiate interaction with the operator to provide information instead of waiting for a request and offer alternative solutions to the problem being addressed. In that sense, an ICS acts as a decision support system. It also must allow the operators to return easily to previous states of their information search. For managing cooperation with the operator, a cooperative system must be able to accomplish tasks as: human-computer interaction management, dialogue management, explanation, documentation, simulation, user modelling, support all aspect of conflict management.

4.2. Cooperative aspects of an ICS

Contrarily to first knowledge-based systems, an ICS must let the operator to have an active role and lead the problem solving. This supposes that the ICS has knowledge about humans and exploit a model of operators to generate descriptions tailored to various types of operators. This permits the ICS to interpret the declared goal and intentions according to its own rules and models of the process and tasks. Two types of operator model have to be considered: a long term one for the general characteristics of the operator and a short term one for the operator current state [Cahour 93].

The task analysis plays a particular role in ICSs: it permits the ICS to determine operators' intention from their actions. A task model may be represented as an intention graph, i.e., a graph where nodes are sequences of actions. The ICS then observes a sequence of actions of the operator, identifies the operator's intention behind this action sequence (using its operator model), judges from the intention graph the way in which the operator is solving the current problem with the real-world process (using its task model) and the consequences on the process (using its process model for simulation purposes). The task analysis also may permit the ICS to benefit of the opportunity for incrementally acquiring operators' preferences (e.g., shortcuts). Beyond the allocation of tasks between the operator and the ICS, the task analysis must lead the ICS to observe directly the operator's behavior with respect to the process's behavior, and thus intervene at the right time, e.g., if the operator does not react when the process behavior moves away of its normal point of functioning. Moreover, the task analysis must be completed with secondary tasks of mediating and controlling the association of the operator and the ICS.

Coordination and control are not a crucial problem for the design of ICSs: Operators always have the control of the real-world process (and also the control of the interaction with the ICS) and the ICS only adapts its actions to operator's actions. The unique intervention of the ICS in operators' task is to alert them of a potential problem that they may not have seen. For instance, the ICS may decide to alert the operator when the ICS detects an abnormal behavior of the real-world process with respect to those of its process model if the operator does not seem recognize the problem.

The problems of conflict and negotiation between an operator and an ICS in our domain are not as important as in human-human cooperation: there is an asymmetry of agents because operators always will take the final decision and the ICS only must support them. The ICS would attempt to justify its position, mainly with explanations, but the operator will take the final decision. The negotiation here is a "one-turn dialogue," and, as a result, the ICS may be lead to revise and update its knowledge bases. Such an adaptation can improve coherence and coordination over time (and avoid conflict and needs of negotiation), by allowing the ICS to conform best to the requirements of changing problem situations [Bond et al. 88].

One also must distinguish agents and roles because the roles of emitter and receiver may be alternatively assigned to the ICS and the operator, roles being taken up dynamically during the problem solving [Davis et al. 88]. A role is the set of actions the agent expects to perform, and the intentions that can reasonably be inferred from these actions [Bond et al. 88; Wong 93]. Different role distributions may be chosen for the ICS depending on the operator's knowledge, the operator's goals and the task domain (e.g., manager, contractor, tutor, suggestor, advisor, critic, waking, etc.). In our case, roles are complementary (i.e., emitter and receiver): The choice of a role by an agent constraints the choice of the role of the other agent. For instance, operators may be in the receiver role, learning from the system what sort of question they can ask, or in the emitter role, either asking the system for what it wants or focusing the system attention on the information they feels relevant at the current step of the problem solving. The receiver role is the more difficult one because the receiver has to understand the problem or the element of solution of the emitter based on the emitter's description.

The ICS receives a communicative act from operators as either a direct request or their actions on the real-world processes. Operators' preferences are recorded by their choice of a particular sequence of actions among alternative sequences. As shows by Norman (1986), an operator's intention arises from a difference between the desired goal and current state. It is stated in psychological terms and translated into an action sequence, the specification of what physical acts will be performed upon the process. To go from intention to action specification requires consideration of the mapping between physical mechanisms and system state, and between system state and the resulting psychological interpretation. The existence of numerous levels of intentions leads to difficulties as a mismatch between the level at which the operator wishes to express the intentions and the level that the system requires. We claim that an efficient way to model operators is through their actions on the real-world process and the work context.

An ICS may model operators' intentions from the sequences of actions they accomplish. This is possible if a task model is available to the system. The system identifies from the action sequence the sub-task the operators intend to realize, and deduce their intentions. This is interesting when the ICS also possesses a real-world process model. Thus, it compares with the evolution of the observed and simulated behaviors of the real-world process. When the two behaviors are compatible, the system does not intervene and stays in a waking state. Conversely, if the two behaviors differ, the system may initiate a cooperation to solve the problem by alerting the operator.

4.2. Communication aspects of an ICS

Communication is an element of the cooperation model that is important in the design of ICSs. Acting as an observer during the waking state, the ICS must have access to operators' behavior, process behavior and the relationships between them. When the ICS is triggered either by a special event occurring in the real-world process or directly by the operator, the ICS must be able to find arguments to convince operators and optimize the cooperation with them. The ICS must have the possibility to provide operators with relevant information in the most useful way. For instance, the ICS may produce a simulation of the process behavior for a given action sequence of the operator.

Agents may not see in the same manner--different viewpoints--of the problem to solve, and, thus, have different interpretations on the current state of the problem solving [Bond et al. 88]. The cooperation leads to a relativization of viewpoints. Making compatible different interpretations implies that cooperating agents must make proposals, explain their viewpoint, and produce spontaneous information. They also must articulate their specific abilities, and maintain a relation of mutual dependency to exploit their complementarity (i.e., their asymmetry) [Karsenty et al. 94]. This implies that the ICS must be provided with a large body of knowledge about the world, about processes, about operators and about communication processes. At each step of the problem solving, the activity of each agent depends on the actions taken by other agents and has consequences on their activities at the next step of the problem solving. The goal of each agent is to permit a progress of both agents towards their common goal.

Two aspects of the cooperation are important for communication with a high quality, namely, the context of the cooperation and the generation of explanations. During interaction, the transfer of information between the operators and the ICS has a high contextual nature. Representing and using context in knowledge bases leads to optimize the communicative acts (i.e., reduce the needs of such acts) and minimize ambiguities in the discourse. Thus, explanations are tailored to a precise need. One problem in the design of ICS lies on the way in which an ICS may accept explanations from operators (generation of explanations by a computer system now is a well-known domain). For accepting an operator's explanation, the ICS needs to understand it, assimilates it in its knowledge bases, and propagates the consequences of the explanation integration. The ICS will need additional pieces of knowledge, and, thus, the explanation will be co-construct by the operator and the ICS.

Acknowledgments: The members of the COOP group are gratefully acknowledged for their support in the way to think cooperation.

5.0 REFERENCES

[Abu-Hakima 93] Abu-Hakima S., "An approach to hypermedia in diagnostic systems", "In: Intelligent Multimedia Interfaces, M. Maybury ed., AAAI Press, Fall 1993.

[Baker 92] Baker M., "Le rôle de la collaboration dans la construction d'explication", 2èmes Journées Explication du PRC-IA, Sophia Antipolis, INRIA, France, 1992, pp. 25-40.

[Bond et al. 88] Bond and Gasser L., "Chapter 1. An analysis of problems and research in DAI", Distributed Artificial Intelligence, 1988, pp. 3-35.

[Branki et al. 93a] Branki N.E. and Bridges A. "An architecture for cooperative agents and application in design", Proc. of the European Conference on Artificial Intelligence (EUropIA-93), Elsevier Science Publisher, 1993.

[Brézillon 94a] Brézillon P., "Design of an intelligent assistant system from various applications", International Conference on Expert Systems for Development, Bangkok, Thailand, 1994, pp. 228-233.

[Brézillon 94b] Brézillon P., "Context needs in cooperative building of explanations", The First European Conference on Cognitive Science in Industry, Luxembourg, September 28-30, 1994 (to appear).

[Brézillon 94c] Brézillon P. , "Bibliographical synthesis on cooperation: A point of view", Research Report of LAFORIA, 1994 (to appear).

[Cahour 93] Cahour B., "Modéliser l'utilisateur pour mieux gérer les dialogues", Cahiers de Linguistique Sociale, 1993.

[Chandrasekaran et al. 83] Chandrasekaran B. and Mittal S., “Deep versus compiled knowledge approaches to diagnostic problem-solving”, International Journal of Man-Machine Studies, 1983, 19(5): 425-436.

[Clarke et al. 93] Clarke A.A. and Smyth M.G.G., "A co-operative computer based on the principles of human co-operation", Int. J. Man-Machine Studies, 1993, 38, 3-22.

[Darses et al. 93] Darses F, Falzon P & RobertJM, "Cooperating partners: Investigating natural assistance", Proc. of HCI'93, Prlando, USA, August 8-13, 1993.

[Davis et al. 88] Davis R. and Smith R.G., "Negotiation as a metaphor for distributed problem solving", Readings in Distributed Artificial Intelligence, 1988, Morgan Kaufmann, A.H. Bond and L. Gasser (eds.), pp. 333-356.

[Fischer 90] Fischer G., "Communication requirements for cooperative problem solving systems", Information Systems, 1990, 15(1), pp. 21-36.

[Gasser 92] Gasser L., "DAI approaches to coordination", In: Distributed Artificial Intelligence: Theory and Praxis, NM Avouris & L Gasser (eds.), ECSC, EEC, EAEC, Netherlands, 1992, pp. 31-51.

[Karsenty et al. 94] Karsenty L and Brézillon P, "Cooperation problem solving and explanation", Int. J. Expert Systems with Applications, 1994 (to appear).

[Lux et al. 93] Lux A., P. de Greef, F. Bomarius and D. Steiner, "A generic framework for human computer cooperation", ICICIS'93, Roterdam, The Netherlands, 1993.

[Norman 84] Norman DA, "Stages and levels in human-machine interaction", Int. J. Man-Machine Studies, 1984, 21, pp. 365-375.

[Norman et al. 86] Norman D.A., “Cognitive Engineering”, In: User Centered System Design: New Perspectives on Human-Computer Interaction, Laurence Erlbaum Associates, London, D.A. Norman & S.W. Draper (eds.), 1986, Chapter 3, pp. 31-61.

[Wahlster 91] Wahlster W., "Chapter 3, User and discourse models for multimodal communication", In: Intelligent User Interfaces, JW Sullivan & S.W. Tyler (eds.), ACM Press, New York, 1991, pp. 45-67.

[Wong 93] Wong STC, "COSMO: A communication scheme for cooperative knowledge-based systems", IEEE Trans. on Systems, Man, and Cybernetics, 1993, 23(3), pp. 809-824.

[Woods et al. 90] Woods D.D., E.M. Roth and K. Benett, "Explorations in joint human-machine cognitive systems", In: Cognition, Computing and Cooperation, Robertson S, Zachary W & Black JB (eds.), 1990, pp. 123-158.

[Zacklad 93] Zacklad M, "Les SBC vus comme des Systèmes Interactifs d'Aide à la Résolution de problème", JAC-93, 1993.