THEORIES AND METHODOLOGIES
THÉORIES ET MÉTHODOLOGIES

CONTEXTUAL KNOWLEDGE SHARING AND COOPERATION IN INTELLIGENT ASSISTANT SYSTEMS

by P. BRÉZILLON* and J.-Ch. POMEROL*

RÉSUMÉ

PARTAGE DES CONNAISSANCES CONTEXTUELLES ET COOPÉRATION DANS LES SYSTÈMES D'ASSISTANCE INTELLIGENTS

Nous nous efforçons de montrer, dans l'optique du développement de systèmes d'assistance intelligents, que la coopération est indissociable des explications et de l'acquisition incrémentale de connaissances et que le contexte joue un rôle essentiel dans ces processus. Les explications sont une partie intrinsèque de toute coopération car elles permettent le partage de connaissances en étendant le contexte de la coopération et donc, de renforcer cette coopération. Inversement, en explicitant le contexte de la coopération, il est possible de produire des explications plus pertinentes, ce qui a pour but de renforcer la coopération. L'acquisition incrémentale de connaissances permet au système de s'améliorer au cours d'une résolution de problème (les connaissances étant alors acquises dans leur contexte d'utilisation) et donc, au cours de toutes ses interactions ultérieures avec un utilisateur. L'acquisition incrémentale de connaissances est importante dans la génération d'explications car le système doit pouvoir accepter des explications de l'utilisateur. Pour ces deux aspects de la coopération, il est nécessaire de trouver une représentation explicite du contexte. Partant de notre expérience dans le développement de systèmes, en particulier, dans le système SART d'aide à la résolution d'incidents sur une ligne de métro, nous illustrons notre conception du contexte dans les systèmes d'assistance. A partir d'exemples, nous proposons quelques pistes pour définir le contexte. Nous montrons qu'il existe différents types de contexte et qu'il est plus judicieux de parler de connaissances contextuelles et de contexte procéduralisé, le statut des différents types de connaissances évoluant au cours de la résolution d'un problème. Cette explicitation du contexte rend la coopération plus efficace, les échanges plus pertinents et donc moins nombreux.

Mots-clés: *Système d'assistance intelligent, Coopération, Contexte, Explication, Acquisition incrémentale des connaissances.*

* Université Paris 6, LIP6, Case 169, 4 place Jussieu, 75252 Paris Cedex 05, e-mails: Name.Surname@lip6.fr

Le Travail Humain, tome 62, n° 3/1999, 223-246

I. INTRODUCTION

Many authors have already raised the issue of context use within interactive systems (Bainbridge, 1997; Cahour & Karsenty, 1993; Hollnagel, 1993; Mittal & Paris, 1993). In the light of some practical experience and especially of a recent implemented system for the interactive control of a subway line, we propose some ideas on this topic, and we focus on the problem of context sharing. We begin with some important issues pointed out by Brézillon and Pomerol (1996, 1997) concerning man-machine interaction in the specific field of Knowledge-Based Systems (KBSs). Among these concerns are :

(1) The user is excluded from the problem solving process, when problems should be jointly solved by the user and the system, the former knowing the context in which the problem occurs.
(2) KBSs do not use their knowledge correctly because this knowledge has some strong contextual dimensions that are generally not acquired with the knowledge.
(3) KBSs cannot initially possess all the required knowledge. Knowledge must be acquired incrementally when needed, together with the context of use.
(4) KBSs cannot generate relevant explanations. The KBS and the user must jointly construct the explanations by sharing knowledge.

Not surprisingly, these issues are more or less at the core of the design of any *cooperative system.* Artificial Intelligence has moved from the idea of Expert Systems replacing people to Joint Cognitive Systems (JCSs) working in a symbiotic manner with the users as quoted by Woods, Roth and Benett (1990). This evolution was already stated, for example, in the book of Winograd and Flores (1986) in which the last chapter is an illustration and defense of interactivity very similar to that put forward by DSS designers ten years ago and since ever. For example, Keen and Scott Morton (1978, p. 2) wrote that "The relevance for managers is the creation of a supportive tool, under their own control, which does not attempt to automate the decision process, predefine objectives, or impose solutions".

Woods, Roth and Benett (1990) coined the expression "Joint Cognitive System" to stress the fact that neither the system nor the user is able to solve alone the problem at hand. An important word in *Joint Cognitive Systems* is the word "cognitive" that gives emphasis to the cognitive aspects of task solving. In a broad sense, this implies that the system and the user have their own cognitive systems (Hollnagel & Woods, 1983) and are able to understand each other when solving a problem. This may be rather utopic because it is unlikely that a machine understands users. However, in a metaphoric sense, this entails some very strong requirements about the design of the system such as: *the sharing of the cognitive representations, an agreement about the contextual knowledge of problem solving, the ability to follow each other's reasoning, the ability to exchange expla-*

nations and the control of the interaction by the user. Karsenty and Brézillon (1995) pointed out that a realistic approach is to provide the system with the means of making its representations (possibly partial representations) compatible with those of the user. The system and the user then build their interpretations with their own cognitive representations. In the case of divergence in their interpretations, they can compare and subsequently update the contextual elements on which their interpretations rely. This implies for the system to acquire knowledge incrementally and provide explanation according to the interaction context.

The cognitive aspect preponderates in several studies where people have gathered cognitive data about process control (Amalberti, 1996; Bainbridge, 1997; Hoc, 1996; Hoc & Amalberti, 1994, 1995). Our contribution, on the other hand, does not rely on a cognitive background as in the preceding references but on our experience of system development. We bring the engineer's view of those specialized in building knowledge-based and interactive systems. However, even before acquiring contextual knowledge, the designer faces the representation problem and a problem of frontier. The representation problem is not only a technical problem of finding the best model and computer data structure to represent context. It is also a problem of choice of what to model: data, preferences, beliefs, etc. This problem of representation and representation sharing has deserved many discussions in the literature on man-machine cooperation. For instance, the notion of "common framework" was introduced by de Terssac and Chabaud (1990). Millot (1998) and Millot and Hoc (1997) stress the joint building of this "common framework". The second choice concerns the limits of contextual data. For example, sociological, historical data may involve a huge volume of information. Even if this not so dramatic in technical systems, it is unlikely to represent everything that may influence the system, including contextual data with very small or rare influence. The designer can be tempted to exclude weakly influential information but he must capture all the knowledge that may be consequential for operations.

This raises the difficult question of the operating limits of a system. What does the system know? In which context is it safe to use it or not? Amalberti (1996), Boy (1991) and Hoc (1996) show that this latter question is very important in preventing accidents. Beyond the notion of "common framework", Hoc (1998) also emphasizes the role of plan sharing and of the understanding of the dynamics of the decision.

Provisionally, if we rely on a loosely defined notion of context, we might agree that contextual knowledge sharing is one of the prerequisites for man-machine cooperation, but so far, we have not defined what we mean by context among the proposals already made and, secondly, we have not said how this contextual knowledge can be acquired and managed. We want to address these two questions in this paper by stating and illustrating our proposals in the light of our experience in system development.

First of all, we should stress a particularity of man-machine cooperation that leads us to include our thinking within the framework of Intelligent Assistant Systems (IASs): There is an asymmetry between the

respective roles of the human being and the machine. The user makes the final decision and the system provides intelligent support by proposing satisfying (or "satisficing" in Simon's sense) alternatives and arguing over them, recalling points in the previous interaction, having a model of the user and a model of the process on which the user works, and within which he thinks. This asymmetry arises from the different responsibilities of the system and the user and from the fact that it is only the latter who, besides responsibility, has intention and commitment (see, for example, Langley *et al.*, 1995 and the references therein). To emphasize the asymmetry between the leading role of the operator and the subsidiary role of the systems, we call such systems Intelligent Assistant Systems.

This paper is organized as follows. In Section II, we review the desired characteristics of IASs. Section III presents the role of context in a real-world application for traffic control and incident management in subways. Based on this application, we identify, in Section IV, different types of context. Section V then deals with how context must be taken into consideration in cooperation, especially context sharing, and emphasizes some issues that we consider important, namely, explanation and incremental knowledge acquisition.

II. INTELLIGENT ASSISTANT SYSTEMS

An Intelligent Assistant System has two sides: the process side and the human side. This implies that an IAS has to understand: (a) the process; (b) the operation model; and (c) the operator's behavior (Brézillon & Cases, 1995). This constitutes three interdependent knowledge bases as represented in Figure 1. Let us identify more precisely these three main knowledge sources.

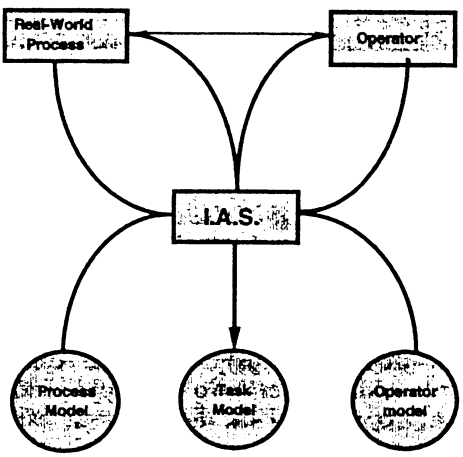

Fig. 1. — General architecture of an IAS

Architecture générale d'un IAS

(a) *The process model:*

This model is deduced from the model used by the designers and the engineers who built the system. The model theoretically allows the IAS to anticipate, by simulation, the process behavior, to check the effects of the operator's actions, and to enable the operator to simulate alternative solutions before making a decision. This process model relies mainly on conceptual knowledge (Hoc, 1996) or deep knowledge, two more or less equivalent concepts.

(b) *The operation model:*

The operation model theoretically allows the IAS to control the system. In other words this knowledge base contains the information that is thought, by the designers, to be necessary to understand and control the process. With this knowledge, we can hope to identify the operator's intentions, to check the coherence of operator's actions and to generate explanations. Using this knowledge, the IAS can eventually correct the operator and suggest alternative sequences (*e.g.*, shortcuts). This knowledge is often referred to in the literature as proceduralized knowledge (Bainbridge, 1997) or rule-driven knowledge (artificial intelligence tradition). The building of the operation model is in two steps. The first step consists of merging theoretical knowledge about the process with the operator's knowledge. The theoretical knowledge comes from designer's and developer's specifications, while the operator's knowledge generally comes from a knowledge acquisition process during operations. This results in a first model of the task. Then, a second step of incremental improvement of the model starts. This can be done either "on-line" when the operator intervenes on the process model or "off-line" when the operator is not under time pressure. This may be called the model of the prescribed task (Guillermain & Salazar-Ferrer, 1998).

(c) *The operator model:*

From the operator's actions and from knowledge in (a) and (b) the IAS can infer the operator's intentions and preferences by observing his choices during the problem solving. By recognizing situations already met, the IAS might propose a solution by similarity, as in case-based reasoning. Remember that cooperating with experienced operators having to make the final decision, the operator model is necessarily a reactive action-oriented model.

The IAS may learn (*i.e.*, acquire knowledge) by observing the behavior of the operators and the process. This is an incremental knowledge acquisition process where knowledge is acquired when needed and, above all, in its context of use. Only a kernel of knowledge has to be elicited directly, mainly to select the right formal representation for the knowledge; depending on the operator's experience, this cooperation can take one of two forms: a waking state if there is no discrepancy between the operator's actions and those of the system, and a cooperative state if a difference is noted relative to either the operator's action or its possible consequences in the future. Thus, there is an interference of goals

between the operator and the system as illustrated by Hoc (1996). The goal of such an intelligent assistant system is to prevent any operator actions that may be a problem sooner or later. Hoc (1996) also evoked such a problem when a delay exists between an action and its consequences (for the case of an action at the helm of a boat and its consequences). This is a very important issue, because delayed feedback impedes or prevents learning.

Before describing an operating system designed according to IAS philosophy, we should say a little about the implementation techniques. In the case described below, we choose a multi-agent approach to allow an evolutionary architecture in which other agents, *e.g.* a training agent, can be added after the first phase of development, in order to expand the functions of the system. Another example of IAS design (Brézillon & Cases, 1995) convinced us of the prominent importance of two issues in cooperation, namely explanation generation and incremental knowledge acquisition. Explanation is an effective way of convincing others (Karsenty, 1996). Operators really do want to follow the system reasoning to feel comfortable with its recommendations. This is a matter of confidence. (It is not surprising that the question of confidence in system design is increasingly deserving attention, as stated by Moray, 1993.) The necessity of incremental knowledge acquisition is obvious to the extent that in real cases, the burden of data introduction is very heavy. Another less obvious reason is that automatic knowledge acquisition is the best way to acquire the context of use and avoid misuse of the system as we will show.

III. CONTEXT IN THE SART PROJECT

III.1. CONTEXT

Before going on to our definition and proposals about context sharing, we present a control system that helped us to set some problems and illustrates some of our ideas. At the very beginning we thought that it would have been possible to give some crisp, intangible definition of context but it appears that knowledge that can be qualified as "contextual" depends on the context! So, before going into details, we take a brief look at SART. In this section we will define the context as overall knowledge that is never explicitly used in operator's reasoning but which, in a very broad sense, sets a resolution space.

III.2. PRESENTATION OF THE SART PROJECT

The SART project (French acronym for support system for traffic control) aims at developing an intelligent decision support system to help the operators who control a subway line to react to incidents that occur on the line. In order to be an IAS, SART has to accomplish several functions

such as acquiring knowledge from operators; simulating train traffic on the line, possibly with incidents; changing the model of the line on operator's request for helping the operator to test alternative issues; proposing alternatives for an incident solving; training a new operator not familiar with a given line; etc.

As already mentioned, we use a multi-agent approach in SART development. To date, our group has developed three agents in the last two years, namely a line-configuration agent, a traffic-simulation agent and an incident-management agent. Specifications are now completed and we are in the programming phase. An extended presentation can be found in the paper of Brézillon, Gentile, Saker and Secron (1997).

III.3. Various context representations in SART

We consider three types of knowledge in a given decision making step: (1) knowledge that is shared by those involved in the problem and is directly but tacitly used for the problem solving, (2) knowledge that is not explicitly used but influences the problem solving, and (3) knowledge that has nothing to do with the current decision making step but is known by many of those involved. We call these three types of knowledge respectively: *proceduralized context, contextual knowledge* and *external knowledge.* We start by giving some insights about these three types of contextual knowledge in the framework of SART before stating more precise definitions in the next section.

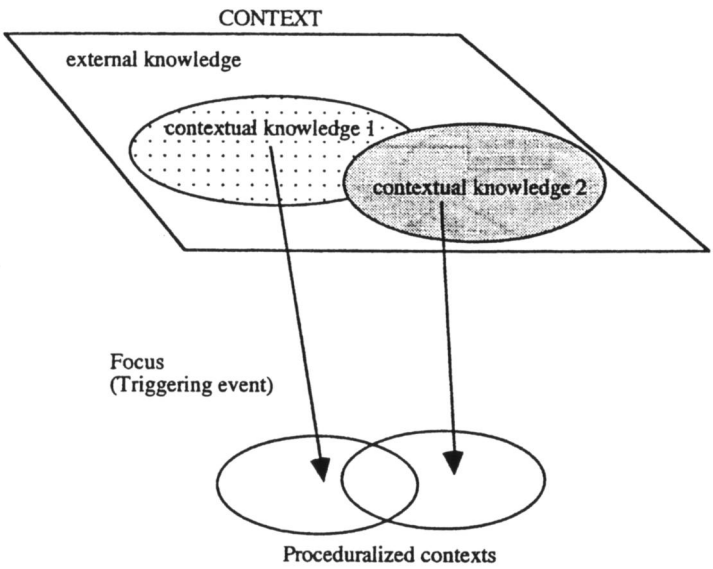

Fig. 2. — Different types of context

Différents types de contexte

For simplicity's sake, we now outline our ideas before illustrating them with SART. We define the context as the sum of all the knowledge possessed by the operators on the whole task. At a given step of a decision process, we separate the part of the context that is relevant at this step of the decision making, and the part that is not relevant. The latter part is called external knowledge. The former part is called contextual knowledge, and obviously depends on the agent and on the decision at hand. A part of the contextual knowledge is, as explained below, proceduralized at each step of the problem solving. We call it the proceduralized context. Figure 2 illustrates the three types of context.

III.3.A. *A representation of proceduralized context*

Figure 3 gives a very partial view of the handling of the incident "Sick traveler in a train" (Brézillon *et al.*, 1997). Incidents are represented by ovals and actions by rectangular boxes (*e.g.*, "Answer Alarm signal"). Consider the partial reasoning resulting in the action "Stop at the next station." This reasoning stems from some chunks of implicit knowledge, not represented here, which are imposed on the driver because they correspond to mandatory procedures. Procedures are part of the operation model acquired from operator's experience during similar incidents and fixed by the company. An implicit piece of knowledge is that travelers are safer in a station than in a tunnel. At a deeper level, the driver has to avoid stopping the train a long time in a tunnel because some

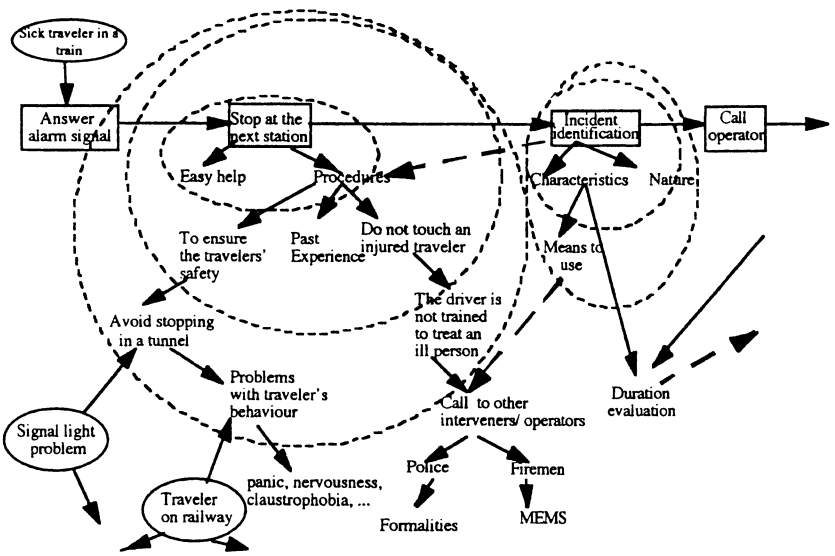

Fig. 3. — Context-based representation of the incident "Sick traveler in a car"

Représentation basée sur le contexte de l'incident "Voyageur malade dans un train"

travelers may have behavioral troubles such as claustrophobia and could leave the train to wander about on the railway (and thus may generate another type of incident such as "Traveler on the railway"). *These pieces of knowledge, which are not necessarily expressed, result in more or less proceduralized actions that are compiled as the proceduralized context.* Very often many pieces of proceduralized context are structured together in comprehensive knowledge about actions. Using a computer metaphor, we can say that some parts of the proceduralized context in working situations or problem resolution are compiled.

Not all the pieces of the proceduralized context are at the same distance of the resulting action "Stop at the next station." For instance, "Procedures" is proceduralized knowledge that is close to the incident-solving step, while "Avoid stopping in a tunnel" is another piece of the proceduralized context that is further away. Distances of this kind enable pieces of proceduralized context to be ordered in layers around an action like the skins of an onion. We call this the *onion metaphor.* Layers of proceduralized-context pieces are represented by stippled circles in figure 3.

Context modeling according to the onion metaphor reveals three interesting results for the generation of explanations:

(1) Proceduralized context is necessary to understand the reasoning.
(2) Pieces of proceduralized context may be partially ordered by a kind of reasoning distance. In other words, the knowledge that is closest to the action is that which give the first rationales and, at the second layer, the rationales for the first rationales and so on. If we consider the step "Stop at the next station," we observe that some knowledge pieces of its context (*e.g.,* "Easy help") are closer to the action than others (*e.g.,* "Do not touch an injured traveler") because the reasons for stopping in a station are immediately understood without any further explanation. It is thus possible to give explanations at different levels of details.
(3) Proceduralized context appears to be obvious for the actors of the system, but it is so obvious that it is generally left implicit and sometimes forgotten. This proceduralized context should be elicited in the operation model.

III.3.B. *A representation of contextual knowledge*

Contextual knowledge is more or less similar to what people generally have in mind about the term "context". It contains some general information about the situation and the environment of the problem. Contextual knowledge implicitly delimits the resolution space (this idea is evoked in Bainbridge, 1997). As already mentioned, this knowledge is a subset of the overall context. This subset is always evoked by a task or, in our case, an event. Contextual knowledge does not focus on a task or on the achievement of a goal but is mobilized, even though it has not yet been proceduralized for use.

How are layers of contextual knowledge used by the operators in a diagnosis process? For instance, in a normal situation, the control operator faces the following concern:

F0: the "normal" focus of attention is to see that schedules and intervals between trains are respected.

This task *F0* can be regarded as routine and does not require special attention. Nevertheless, contextual knowledge about control is involved. In this task, the word "normal" has different meanings according to the context; let us look at some possible contexts.

C0: the normal context associated with *F0* involves:
- k1: type of day (*e.g.*, working day, Saturday, Sunday, Holidays),
- k2: period of the day (morning, afternoon, evening),
- k3: traffic state (rush hours, off-peak hours),
- k4: the section load (very busy, few people).

All these pieces of knowledge are some of the elements defining the contextual knowledge describing the environment of the problem with which the following pieces of knowledge are associated:

- k5: the interval between trains according to the situation,
- k6: the stopping time in stations, etc.

Contextual knowledge is therefore quite large and not focused. Many "normal" contexts are contained in this contextual knowledge. Assume now that an incident occurs on the subway line; the pieces of knowledge k1 to k4 are (or should be) immediately invoked. This results in k5 and k6 being invoked. We will say that k5 and k6 are invoked, they become a part of the proceduralized context in which the incident is resolved; contextual knowledge appears back-stage, whereas the proceduralized context is front-stage in the spotlights (Fig. 3). It is noteworthy that, as far as engineering is concerned, only the proceduralized context matters, but contextual knowledge is necessary because this is the raw material from which proceduralized context is made. *One can say that contextual knowledge is proceduralized, not necessarily explicitly, to become the proceduralized context.* In a sense, the proceduralized context is the contextual knowledge activated and structured to make diagnoses or decisions.

Note also that in our work "contextual" is not opposed to "linear" reasoning as in Bainbridge (1997); see also Hoc and Amalberti (1995) for a criticism of "linear" models of diagnosis. As Pomerol (1997) stated, the analysis of decision systems which separate diagnosis from anticipations is just an engineer's simplification and not a cognitive model, however many observations about what is called "decision bias" in decision theory may be due to the wrong intertwining of diagnosis, anticipation and preferences. It is also noteworthy that during the transformation of contextual knowledge into proceduralized context, some interpretations and rationalizations occur; this is reminiscent of the transformation of data into facts in the sense of Hoc and Amalberti (1995). Note also that our views about context are not inspired by a holistic point of view as opposed to a decomposition of the actions and reasoning (see Hollnagel, 1993, for his holistic argumentation).

IV.C. Dynamics of the different contexts and scenario thinking

Decision making is a dynamic process. From one step to the next one, a piece of contextual knowledge either enters the proceduralized context or become external knowledge. Conversely, a piece of proceduralized context may become either contextual knowledge or external knowledge. Thus, the content of the context evolves continuously all through the decision making. Once the first pieces of contextual knowledge are mobilized, some other pieces of contextual knowledge, such as the position of the incident on the line, also enter the focus of attention and are proceduralized. The proceduralized context may also evolve to integrate some knowledge that, up to now, has neither been proceduralized nor is contextual (*i.e.*, external knowledge) such as maintenance activity on the line, the number of trains on the line, the available help in the control room or the experience of the train driver. Thus, the diagnosis context evolves jointly and continuously with the reasoning process.

Operators solve an incident by choosing a scenario, which is a sequence of actions conditional on possible events. The choice of a scenario greatly relies on contextual knowledge. One operator said us: "When an incident is announced, I first look at the context in which the incident occurs." The reason is that operators want to have a clear idea of future events; the purpose of this look-ahead reasoning (Pomerol, 1997) is to reduce, as far as possible, the uncertainty in the scenario. The problem for operators is that many scenarios are similar at the beginning and then diverge according to the context. Thus, a scenario is a sequence of actions intertwined with events that do not depend on the decision makers but that result in a limitation of their actions. For instance:

Focus of attention: Removal of a damaged train from the line.
Contextual information: Level of activity on the line.
Action: Lead the damaged train:
- — to the terminal if the line activity is low
- — to the nearest secondary line if line activity is high.

As most of the contextual elements may intervene in several scenarios (*e.g.*, traffic activity, position of the next train), operators prefer to take them into account as soon as possible to get a general picture of the best path to choose. At this step, contextual knowledge is proceduralized and in the meantime operators postpone action. The main objective is to eliminate event nodes. By grouping together a set of actions in a macro-action, operators hope to make the following step easier. Figure 4 gives a simplified example of a decision tree drawn from the SART project.

Without entering into details, *macro-actions are a way of proceduralizing contextual knowledge* and introducing modularity into the diagnosis process by managing different modules that accomplish the same function in different ways according to the context. However, action postponement is not always possible, and it is preferable to try to prune the decision tree in

some situations (Brézillon, Pomerol, & Saker, 1998). In Figure 4, several macro-actions existing on different branches can be identified (*e.g.*, A6-A7-A9). Such macro-actions are a kind of compilation, originating from experience, of several actions. In this compilation, a part of the knowledge on each action becomes implicit in the proceduralized context. This is close to Edmondson and Meech's view (1993) on context as a process of contextualization. However, to explain a macro-action (and thus the whole reasoning involved), an operator needs to decompile the macro-action to retrieve the rationales. Such an operation is not always easy, especially when experience comes from a previous generation of operators.

At RATP (the French metro company), most of the incidents have been well-known for a long time (object on the track, lack of power supply, suicide, etc.). Thus, the company has established procedures for incident solving on the basis of their experience. However, each operator develops his own practice to solve an incident, and one observes almost as many practices as operators for a given procedure because each operator tailors the procedure in order to take into account the current proceduralized context, which is particular and specific. In many working processes human beings can be observed to develop genuine procedures to reach the efficiency that decision makers intended when designing the task. Some parts of this practice are not coded (Hatchuel & Weil, 1992). Such know-how is generally built up case by case and is complemented by "makeshift repairs" (or non-written rules) that allow the operational agents to reach the required efficiency. This is a way of getting the result

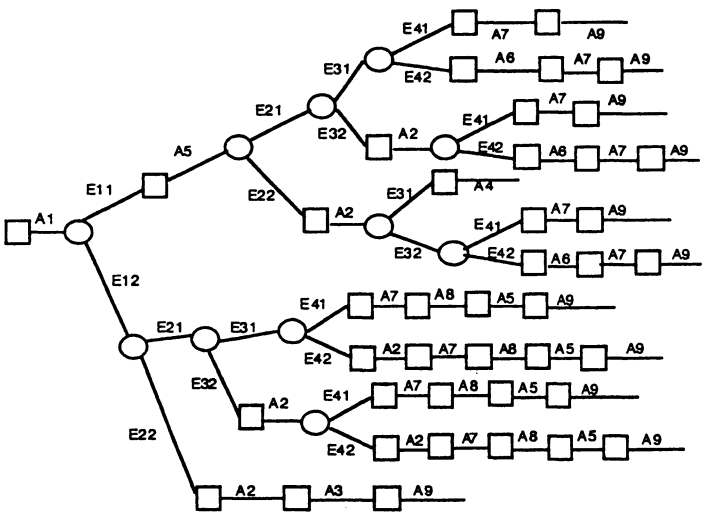

Fig. 4. — A tree in which many actions are postponed to the end of the branches (the symbols are described in the following table)

Un arbre où beaucoup d'actions sont repoussées à la fin des branches (les symboles sont définis dans le tableau suivant)

whatever the path followed. The validation of unwritten rules is linked more to the result than to the procedure to reach it. De Terssac (1992) spoke of logic of efficiency.

Some Actions (A) and Events (E) in case of incident on a metro line.

A1	Disconnect regulation function of damaged train	E11	Damaged train in a station
A2	Emergency exit of travelers from all trains on the line	E12	Damaged train in the tunnel
A3	Make damaged train reach next station	E21	Immediate repair possible
A4	Lead the damaged train to the terminus	E22	Immediate repair impossible
A5	Exit of travelers from the damaged train	E31	Few trains in section line
A6	Lead next train to the nearest station	E32	Many trains in section line
A7	Emergency exit of travelers from the next train	E41	Next train in a station
A8	Push damaged train to the next station	E42	Next train in tunnel
A9	Push damaged train to the terminus		

As we have seen, it appears that, for many decision makers, the most urgent need is, on the one hand, to reduce the number of events to envisage and, on the other hand, to undertake robust actions and, if possible, macro-actions. At that level, contextual knowledge enables the uncertainty in the decision making to be reduced. However, some contextual knowledge may be very far removed from the incident solution. For example, all the paths in Figure 4 conclude with the same action (except one): evacuate the damaged train to the terminus, and not evacuate the damaged train on a secondary line. The reason is that the workshops for repairing trains in Paris are in the terminus. (The strategy for the metro in Rio de Janeiro is different because workshops are in the middle of the lines.)

IV. DIFFERENT KINDS OF CONTEXT

IV.1. What is context?

Before examining context sharing and the use of contextual information in cooperation we shall propose different kinds of context and the articulation between them. There is already an abundant literature on context. From a review of the literature, Brézillon (1999) shows that context has played an important role in a number of domains since a long

time, especially for activities such as foreseeing context changes, explaining unanticipated events and helping to handle them, and helping to focus attention. Adopting an engineering stance, we will define context as the set of all the knowledge that could be evoked by a human being facing a situation, assuming that he has an unlimited time to think about it. Brézillon and Abu-Hakima (1995) show that context has different meanings depending on the background. On the one hand, the cognitive viewpoint is that context is used to model interactions and situations, and as such it is dynamic and evolves continuously. This *a posteriori* view implies that context can be elicited only during the task or problem solving. On the other hand, engineers tend to want to represent context and consider it as a raw material for problem solving. They think that context has an *a priori* existence and can be represented.

Moreover, context possesses a time dimension that poses some problems in modeling. Some have suggested that context is related to the interactions among agents, as opposed to context as a fixed concept relative to a particular problem or application domain (Maskery & Meads, 1992). That is, without interacting agents, there would be no context. In communication, the context is considered as the history of all that occurred over a period of time, the overall state of knowledge of the participating agents at a given moment, and the small set of things they are expecting at that particular moment. Context appears as a shared knowledge space. However, each entity involved in an interaction has its own context, which may or may not be consistent with some parts of the contexts of the others. One important point underlined by Mittal and Paris (1995) is that communication, including explanations, and context interact with each other: the context of the situation triggers some actions, and this in turn modifies the context of the situation. These authors bring together different notions of context such as the elements of a global picture that might be taken into account by an explanation module, depending on the needs of the application.

In a knowledge engineering setting, it has been argued (Grant, 1992), that the term "context" has some features in common with scripts (Schank & Riesbeck, 1989), frames or schemata as developed in human cognition. The context here is a candidate for something that is stored in long-term memory, and recalled as a whole, as a viable unit of a task appropriate to some step in a decision making. As regards Schank's theory, assuming that context is the set of all the possible stories, the case corresponds approximately to our proceduralized context while contextual knowledge is the set of paradigmatic cases.

IV.2. Defining contextual knowledge and proceduralized context

It is in fact difficult to define the concept of context without considering the people involved in a situation because, at a first glance, context involves knowledge that is not explicit. This "explicitness" depends on the actors. Some common knowledge is implicit but well-

known, for example the fact that it is easier to organize emergency operations in a station than in a tunnel. When the reasoning yields this type of knowledge, it is easily proceduralized and becomes an implicit part of the reasoning that can be elicited by knowledge engineers and finally included in the operation model.

The second fact is that each person involved uses a large amount of knowledge, different from one person to another, to picture the situation. We can define the contextual knowledge as all the knowledge that is relevant and can be mobilized to understand a given situated decision problem. By "situated" we mean in given, dated, well specified circumstances. The word "situated" was introduced into artificial intelligence by Clancey (1991). In its artificial intelligence sense, "situated cognition" emphasizes the role of interaction and context in human behavior. This weak situated cognition hypothesis (Menzies, 1997), which links knowledge, interaction and context, provides a good background for our views.

Contextual knowledge is *personal to a participant* and it has *no clear limit* (MacCarthy, 1993). Contextual knowledge is evoked by situations and events, and loosely tied to a task or a goal. However, when the task becomes more precise, a large part of this *contextual knowledge can be proceduralized according to the current focus of the decision making.* Although the contextual knowledge exists in theory, it is actually implicit and latent, and is not usable unless a goal (or an intention) emerges. When an event occurs, the attention of the actor is focused and a large part of the contextual knowledge will be proceduralized. In our definition, the contextual knowledge is dependent on the situation (date, location, participants); it is a sub-part of the overall context (Fig. 2). Thus, the rest of the context, which is not relevant for a given situation, is called *external knowledge.*

The proceduralized context is a part of the contextual knowledge that is invoked, structured and situated according to a given focus and which is common to the various people involved in decision making. The proceduralized context may be compiled but can generally be elicited with the usual techniques of knowledge acquisition, it is limited and should be a part of the operation model. For the model to be usable people have to define a finite number of situations, and diagnosis consists mainly of trying to determine in which situation those involved find themselves.

IV.3. Interaction and movement between proceduralized context, contextual and external knowledge

The dynamics of the process is very important in diagnosis as well as control of complex systems (Hoc, 1996 ; Hoc, & Amalberti, 1995). We think that it is not only important to understand the dynamics of planning and action but also the dynamics of knowledge management. This is a twofold phenomenon that consists of focusing on some stimuli and, on moving contextual information from back-stage to front-stage. Interaction

between agents appears to be a privileged way for moving a contextual knowledge into and out of the proceduralized context.

At the interaction level, making context explicit enables knowledge to be shared with others. For example, in the case of "Sick traveler on a train," the change from the step "Answer the alarm signal" to "Stop at the next station" was surprising to us (as knowledge engineers). The triggering of the alarm signal implied for many years an immediate stop of the train, even in a tunnel because an alarm signal needed immediate attention. To explain the skipping of the action "Answer the alarm signal", the operator said that, based on company experience, they have decided to stop only at the next station for several reasons (see Fig. 3). Some reasons are easy to understand (*e.g.,* rescue is easier in a station than in a tunnel). Other reasons needed additional knowledge (*e.g.,* drivers follow procedures because a traveler may be sensible to claustrophobia in a tunnel). Once we were able to gather all these reasons into a coherent proceduralized context, we understood the change. Note that other reasons are left implicit (*e.g.*, if the stop will last a long time in a station, other travelers may leave the train to go by bus, reducing the number of travelers waiting on the platform).

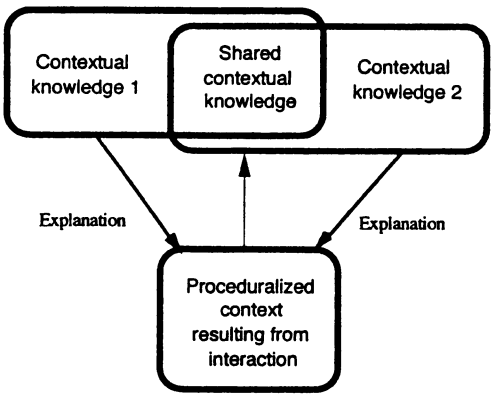

Fig. 5. — A representation of the interaction context

Une représentation du contexte d'interaction

Figure 5 represents how the proceduralized context is built from contextual knowledge during an interaction between two agents. The interaction context contains proceduralized pieces of knowledge in the focus of attention of the two agents. These pieces of knowledge are extracted from the contextual knowledge of each agent; they are structured jointly by both agents and result in a shared knowledge. Generally, the first utterance of an agent gives a rule such as "Stop at the next station" if the alarm signal is triggered. Then on the request of the second agent, the first agent may add some pieces of knowledge related to his first utterance. If this knowledge chunk belongs to the common part of the contextual kno-

wledge of the agents, the pieces are integrated into a mutually acceptable knowledge structure, and the knowledge structure may then be moved to the shared proceduralized context. Thus, the proceduralized context contains all the pieces of knowledge that have been discussed and accepted (at least made compatible) by all the agents, and these pieces of proceduralized context then become part of the contextual knowledge of each agent, even if they do not remain within the focus of the proceduralized context. Luff and Heath (in press) give such an example of the joint production of the proceduralized context by two controllers and a system.

V. CONTEXT SHARING IN COOPERATIVE SYSTEMS

We have shown that contextual knowledge depends on the agent and situation and that proceduralized context can be shared. As recalled in the introduction, there is no cooperation without knowledge sharing. In an IAS, we distinguish three models; among them at least two — the operation model and the operator model — involve shared knowledge and must contain the proceduralized context. We can imagine that the process model works as a black box for the operators, but the comparison between the operation and operator models is the cornerstone of cooperation in IASs. Thus, the latter two models must agree on the proceduralized context relative to the finite number of possible situations. This implies a possibly difficult agreement on the specification and characterization of these situations and agreement on what part of the contextual knowledge, for a given situation, must be compiled into the proceduralized context. This process of knowledge acquisition is difficult and burdensome. However we think that it is possible to alleviate this burden by using learning and explanations.

V.1. EXPLANATION GENERATION

Explanations have many roles. For Brézillon (1994), explanations allow the integration into the knowledge of one agent of some pieces of the other's knowledge, while many researchers have focused on explanations as a transfer of knowledge *from* the system *to* the user. As a consequence, users can rarely be involved in the generation of explanation. An opposite position was taken by Brézillon (1990) in SEPT by letting the user alone build his or her explanation. This was not satisfying because some users included complex commands that were not always compatible with their work and time constraints. The lesson is that the user and the system must cooperate to solve the problem jointly and to co-construct an explanation for the solution.

Karsenty and Brézillon (1995) show that explanation is an intrinsic part of cooperation. Explanation is a means of developing the shared context that is needed for a full understanding (Karsenty, 1996). Explanations are a way of making explicit the implicit knowledge in a procedure and of interpreting new knowledge pieces. Thus, explanation generation acts as a

contextualization process (Edmonson & Meech, 1993; Karsenty & Brézillon, 1995); in our framework it makes it easier to compile contextual knowledge into the proceduralized context. Conversely, the explicit use of context allows explanations to be tailored to a specific request. It is the context that supplies the explanations needed to validate suggestions (Karsenty & Falzon, 1992). Explanations improve cooperation by building the shared proceduralized context, and cooperation allows each agent to produce relevant explanations for the others. Thus, a system must be able to accept explanations from the user, and then change its knowledge organization accordingly. Explanations are a way of facilitating acquisition of the missing knowledge when needed. This assumes an incremental knowledge acquisition of knowledge and its context of use by the system.

In MYCIN, and in all earlier expert systems, the explanations are directly drawn from the rules triggered (see Fig. 6.a). In figure 6 the context is represented by rectangles (unique and fixed in 6.a) while the logic of explanation are confounded, *i.e.* the explanations are exactly similar to the trace of the system. Thus, all traces of reasoning are kept and their contents provided to the user for explanation.

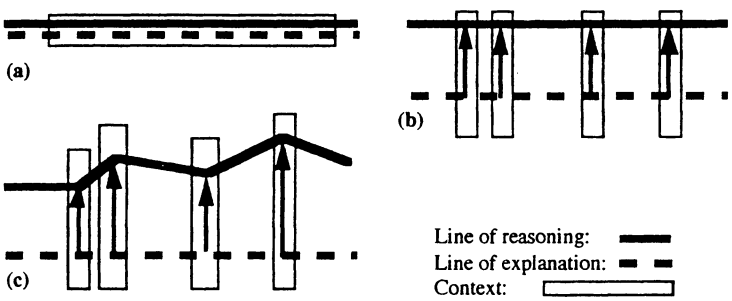

Fig. 6. — Line of reasoning versus line of explanation

Ligne de raisonnement et ligne d'explication

However, for the sake of implementation, the representation of expert's knowledge is sometimes darken. For example, MYCIN used screening clauses to control the firing of rules as described by Clancey (1983). (Screening clauses acted as pre-conditions in other programming languages.) For example, consider the following rule of MYCIN:

IF

1. The infection, which requires therapy, is meningitis,
2. Only circumstantial evidence is available for this case,
3. The type of meningitis is bacterial,
4. *The age of the patient is greater than 17 years old*, and
5. The patient is an alcoholic,

THEN

There is evidence that the organisms which might be causing The infection are diplococcus-pneumoniae (.3) or e.coli (.2).

The rule contains different types of knowledge (strategic knowledge, causal knowledge, etc.). Clause 4 was added by the developer to control the triggering of the rule. The clause acts as a screening clause and implies that the rule use is restricted to the context of adults. Such knowledge does not intervene directly in the problem solving, but merely constrains it. To explain this clause, extra knowledge is required, such as the fact that tetracycline gives a violet color to teeth, which is a problem during child growth but not for adults. With our definition some pieces of contextual knowledge have to be proceduralized to explain the clause 4.

In other systems, such as that described by Paris, Wick and Thompson (1988) and Wick and Thompson (1992), explanation is only periodically available in the reasoning of the system (see Fig. 6.b) and the context changes. In Wick's system, only selected contexts in which the system reasons are explained to the user. In this approach, additional explanatory knowledge (knowledge on the domain and the expertise that are not directly necessary for the task at hand) may be used to generate enhanced explanations. This implies that the explanation line is separated from the reasoning line to produce relevant explanations. The context is here an extended version of the previous context because it also contains domain and task knowledge not directly considered in the reasoning of the problem solving, and possibly some information on users, through a model. One problem with such an approach is that it may be unsuitable for critical applications whose results may affect the safety of processes and people; in these systems, as well as in the system proposed by Lester, & Porter (1991), the dynamics between different contexts is not clear.

Another approach to explanation is to accept that the system's reasoning is often different from that of the user. Thus, the user and the system may have different interpretations of the current state of the problem solving. The different interpretations become consistent when the user and the system make proposals, explain their viewpoints and spontaneously produce information, as noted by Karsenty and Brézillon (1995). In order to line up 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 problem solving (the proceduralized context). Thus, explanations become an intrinsic part of problem solving and, as a consequence, the reasoning line of the system may be modified by explanation (see Fig. 6.c). This leads to cooperative problem solving.

The way in which an explanation is chosen and produced depends essentially on the context in which the two participants interact. An explanation always takes place in a space of alternatives that require different explanations according to the current context. For instance, Huuskonen and Korteniemi (1992) propose following different steps to generate an explanation accounting for the context:

— find the context from previous explanations already provided;
— find the questions matching with the current context;
— refine the context if necessary;
— find the answer and display it.

With such a method, the user should easily accept the idea of conversation through context refinement. Using context allows a kind of "cognitive coupling" between the user and the system as noted by Karsenty (1996). In our framework, dialogue is one of the best ways to construct the proceduralized context.

V.2. Incremental knowledge acquisition

Knowledge acquisition is a difficult and time-consuming task. In our opinion, the knowledge engineer associates a context to the couple (problem, solution) that may be different from those in expert's mind. Encoding the knowledge relative to the task at hand leads to a proceduralization of the context because knowledge is (or should be) encoded into the system with a part of the contextual knowledge. As Henninger (1992) noted, "You won't know what is really needed until you're in the design process."

Incremental knowledge acquisition plays an important role in two situations:

— When the knowledge is missing in the current context, the user adds new knowledge through explanations. Here, explanations enable the contextual knowledge to be proceduralized.

— A chunk of knowledge may be known by the system but incorrectly used, *i.e.* a link to 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 proceduralized context, and explanation will support that refinement. Thus, gathering and using knowledge in the context of use greatly simplifies knowledge acquisition because the knowledge provided by experts is always in a specific context and is essentially a justification of the expert's judgment in that context.

Several approaches have been proposed to acquire knowledge in context. We give here two examples. Compton and Jansen (1988) attempt to capture the context by entering the expert's new rule only when necessary. More precisely, when a rule fails, the expert is requested to give some extra knowledge or rule. This new rule is directly related to the rule which failed and this latter is recalled to the expert because it gives the context for writing the new rule.

Gruber (1991) considers a justification-based knowledge acquisition. The machine provides the computational medium, including the knowledge representation and the context of use, such that everything that is acquired from the user can be assimilated into the computational model. The dialogue is directed by the machine that can provide thus a frame to fill, allowing some kind of syntactic checking and ensures that all required parameters are given.

The main idea of all these approaches is that experts provide their knowledge in a specific context and that knowledge can only be relied upon with in this context. Thus, knowledge cannot be generalized when it is acquired, and it is fundamental to record the context in which the knowledge is acquired. Nevertheless, acquiring knowledge in context is still a challenge.

VI. CONCLUSION

The two questions of contextual knowledge and knowledge sharing have received wide consideration in recent years. In this paper we have addressed these two topics through Intelligent Assistant Systems and, more specifically, by using our experience in the development of IASs for process control. As a result of our analyses, we stress the dynamic aspect of the contextual knowledge. We propose defining the contextual knowledge as a possibly unlimited, personal and situated set of relevant knowledge involved in problem solving. Part of this contextual knowledge is proceduralized to enable cooperation and this results in a shared, limited proceduralized context that can be elicited, and therefore made explicit by the usual methods of knowledge engineering. There is a finite number of pieces of knowledge in the proceduralized context, each of them being related to a situation, date, locations, participants, problem, etc. One difficult question which can be the main question in diagnosis is to identify the relevant situation. The second question that we can address through case-based reasoning (Brézillon & Pomerol, 1998) is to use proceduralized context for situations which are close or similar to the reference situation (*i.e.*, a kind of context-based reasoning). Using our definition we show how contextual knowledge is partially proceduralized in cooperative settings. The proceduralization process being itself a cooperative process we show how the proceduralized context is incrementally enriched and we examine the role of explanation in this process.

We are strongly of the opinion that contextual issues cannot be addressed in a static framework and that eliciting and sharing of contextual knowledge is a key process in addressing and understanding context problems.

REFERENCES

Amalberti, R. (1996). *La conduite de systèmes à risques*. Paris: PUF.

Bainbridge, L. (1997). The change in concepts needed to account for human behavior in complex dynamic tasks. *IEEE transactions on Systems, Man and Cybernetics, 27*, 351-359.

Boy, G. (1991). *Intelligent Assistant Systems*. London: Academic Press.

Brézillon, P. (1990). Interpretation and rule packet in expert systems. Application to the SEPT expert system. In S. Ramani, R. Chandrasekar, & K. S. R. Anjaneyulu (Eds.), *Knowledge Based Computer Systems* (pp. 78-87). Heidelberg: Springer-Verlag.

Brézillon, P. (1994). Context needs in cooperative building of explanations. Paper presented at the *First European Conference on Cognitive Science in Industry*. Luxembourg, September.

Brézillon, P. (1999). Context in human-machine problem solving: A survey. *Knowledge Engineering Review, 14*,1-34.

Brézillon, P., & Abu-Hakima, S. (1995). Using Knowledge in its context: Report on the IJCAI-93 Workshop. *AI Magazine, 16*, 87-91.

Brézillon, P., & Cases, E. (1995). *Cooperating for assisting intelligently operators.* Paper presented at the *International Workshop on the Design of Cooperative Systems (COOP-95)*. Antibes, France, January.

Brézillon, P., Gentile, C., Saker, I., & Secron, M. (1997). *SART: A system for supporting operators with contextual knowledge.* Paper presented at the *International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-97)*. Rio de Janeiro, Brasil, February.

Brézillon, P., & Pomerol, J.-Ch. (1996). Misuse and Nonuse of Knowledge-Based Systems: The Past Experiences Revisited. In P. Humphreys, L. Bannon, A. McCosh, P. Migliarese, & J.-Ch. Pomerol (Eds.), *Implementing Systems for Supporting Management Decisions* (pp. 44-60). London, UK: Chapman Hall.

Brézillon, P., & Pomerol, J.-Ch. (1997). User Acceptance of Interactive Systems: Lessons from Knowledge-Based and Decision Support Systems. *International Journal on Failures and Lessons Learned in Information Technology Management, 1*, 67-75.

Brézillon, P., & Pomerol, J.-Ch. (1998). Using contextual information in decision making. In G. Widmeyer, D. Berkeley, P. Brézillon, & V. Rajkovic (Eds.), *Context-Sensitive Decision Support Systems* (pp. 158-173). London, UK: Chapman & Hall.

Brézillon, P., Pomerol, J.-Ch., & Saker, I. (1998). Contextual and contextualized knowledge: An application in subway control. *International Journal of Human-Computer Studies, 98*, 357-373.

Cahour, B., & Karsenty, L. (1993). *Context of dialogue : A cognitive point of view.* Paper presented at the *International Joint Conference on Artificial Intelligence Workshop on "Using Knowledge in its Context"*. Paris, France, August.

Clancey, W. J. (1983). The epistemology of a rule-based expert system: A framework for explanation. *Artificial Intelligence Journal, 20*, 197-204.

Clancey, W. J. (1991). Bookreview of Israel Rosenfeld, The invention of memory : A new view of the brain. *Artificial Intelligence, 50*, 241-284.

Compton, P., & Jansen, B. (1988). Knowledge in context: A strategy for expert system maintenance. In J. Siekmann (Ed.), *Lecture Notes in Artificial Intelligence* (pp. 37-49). Heidelberg: Springer-Verlag.

Edmondson, W. H., & Meech, J. F. (1993). *A model of context for human-computer interaction.* Paper presented at the *International Joint Conference on Artificial Intelligence Workshop on "Using Knowledge in its Context"*. Paris, France, August.

Grant, A. S. (1992). *Mental models and everyday activities.* Paper presented at the *2nd Interdisciplinary Workshop on Mental Models*. Cambridge, UK, March.

Gruber, T. (1991). Justification-based knowledge acquisition. In H. Motoda, R. Mizoguchi, J. Boose, & B. Gaines (Eds.), *Knowledge Acquisition for Knowledge-Based Systems* (pp. 81-97). Amsterdam, The Netherlands: IOS Press.

Guillermain, H., & Salazar-Ferrer, P. (1998). Contribution à l'identification des risques facteurs humains dans la conduite des processus à haut niveau de sûreté de fonctionnement. In J.-G. Ganascia (Ed.), *Sécurité et Cognition* (pp. 19-37). Paris, Hermès.

Hatchuel, A., & Weil, B. (1992). *L'expert et le système.* Paris: Economica.

Henninger, S. (1992). *The knowledge acquisition trap.* Paper presented at the *IEEE Workshop on Applying Artificial Intelligence to Software Problems: Assessing Promises and Pitfalls (CAIA-92)*. Monterey, Canada, March.

Hoc, J.-M. (1996). *Supervision et contrôle de processus. La cognition en situation dynamique.* Grenoble: PUG.

Hoc, J.-M. (1998). Conditions et enjeux de la coopération homme-machine dans le cadre de la fiabilité des systèmes. In J.-G. Ganascia (Ed.), *Sécurité et Cognition* (pp. 147-164). Paris: Hermès.

Hoc, J.-M., & Amalberti, R. (1994). Diagnostic et prise de décision dans les situations dynamiques. *Psychologie Française, 39*, 177-192.

Hoc, J.-M., & Amalberti, R. (1995). Diagnosis : Some theoretical questions raised by applied research. *Current Psychology of Cognition, 14*, 73-101.

Hollnagel, E. (Ed.). (1993). *Human Reliability Analysis, Context and Control.* London: Academic Press.

Hollnagel, E., & Woods, D. D. (1983). Cognitive Systems engineering: New wine in new bottles. *International Journal of Human-Computer Studies, 18*, 583-560.

Huuskonen, P., & Korteniemi, A. (1992). *Explanation based on contexts.* Paper presented at the *8th Conference on Artificial Intelligence for Applications.* Monterey, Canada, March.

Karsenty, L. (1996). Une définition psychologique de l'explication. *Intellectica, 2*, 299-317.

Karsenty, L., & Brézillon, P. (1995). Cooperative problem solving and explanation. *International Journal of Expert Systems With Applications. 4*, 445-462.

Karsenty, L., & Falzon, P. (1992). *Spontaneous explanations in cooperative dialogues.* Paper presented at the *European Conference on Artificial Intelligence Workshop on Improving the Use of KBS with Explanation.* Paris, France, June.

Keen, P.G.W., & Scott Morton, M. S. (1978). *Decision Support System.* London, UK: Addison Wesley.

Langley, A., Mintzberg, H., Pitcher, A., Posada, E., & Saint-Macary, J. (1995). Opening up decision making : The view from the black stool. *Organization Science, 63*, 260-279.

Lester, J. C., & Porter, B. W. (1991). *Generating context-sensitive explanations in interactive knowledge-based systems.* Paper presented at the *American Association on Artificial Intelligence Workshop on Comparative Analysis of Explanation Planning Architectures.* Monterey, USA, July.

Luff, P., & Heath, C. (in press). *The interactional production of computer commands.*

Maskery, H., & Meads, J. (1992). Context: In the eyes of users and in computer systems. *SIGCHI Bulletin, 24*, 12-21.

McCarthy, J. (1993). *Notes on formalizing context.* Paper presented at the *13th International Joint Conference on Artificial Intelligence.* Chambéry, France, August.

Menzies, T. (to appear). Is knowledge maintenance an adequate response to the challenge of situated cognition for symbolic knowledge based-systems? *International Journal of Human-Computer Studies.*

Millot, P. (1998). La supervision et la coopération homme-machine dans les grands systèmes industriels ou de transport. In J.-G. Ganascia (Ed.), *Sécurité et Cognition* (pp. 125-145). Paris: Hermès.

Millot, P., & Hoc, J. M. (1997). *Human-Machine Cooperation: Metaphor or Possible Reality.* Paper presented at the *European Conference on Cognitive Sciences (ECCS 97)*, Manchester, UK, April.

Mittal, V. O., & Paris, C. L. (1993). *Context : Identifying its elements from the communication point of view.* Paper presented at the *International Joint Conference on Artificial Intelligence Workshop on Using Knowledge in its Context.* Paris, France, August.

Mittal, V. O., & Paris, C. L. (1995). Use of context in explanations systems. *International Journal of Expert Systems with Applications, 8*, 491-504.

Moray, N. (1993). Designing for attention. In A. Baddeley & L. Weiskrantz (Eds.), *Attention: selection, awareness, and control: a tribute to Donald Broadbent* (pp. 111-134). Oxford, UK: Oxford University Press.

Paris, C. L., Wick, M. R., & Thompson, W. B. (1988). *The line of reasoning versus the line of explanation.* Paper presented at the *American Association on Artificial Intelligence Workshop on Explanation.* Stanford, USA, April.

Pomerol, J.-Ch. (1997). Artificial Intelligence and Human Decision Making. *European Journal of Operational Research, 99*, 3-25.

Pomerol J.-Ch. (1998). Scenario development and practical decision making under uncertainty: robustness, case-based reasoning and risk control. Paper presented at the *IEEE Conference Engineering in Systems Application*. Hammamet, Tunisie, April.

Schank, R. C., & Riesbeck, C. K. (Eds.) (1989). *Inside case-based reasoning*. Hillsdale, NJ: Lawrence Erlbaum Associates.

de Terssac, G. (1992). *Autonomie dans le travail*. Série « Sociologie d'Aujourd'hui ». Paris: PUF.

de Terssac, G., & Chabot, C. (1990). Réferentiel opératif commun et fiabilité. In J. Leplat & G. de Terssac (Eds.), *Les facteurs humains de la fiabilité* (pp. 110-139). Toulouse: Octarès.

Wick, M. R., & Thompson, W. B. (1992). Reconstructive expert system explanation. *Artificial Intelligence Journal, 54*, 33-70.

Winograd, T., & Flores, F. (1986). *Understanding Computers and Cognition: A new foundation for Design*. London, UK: Ablex.

Woods, D. D., Roth, E. M., & Benett, K. (1990). Explorations in joint human-machine cognitive systems. In S. Robertson, W. Zachary, & J. B. Black (Eds.), *Cognition, Computing and Cooperation* (pp. 123-158). Norwood, NJ: Ablex.

ABSTRACT

The role of contextual information in intelligent assistant systems is controversial. In this paper, we start from our experience of Intelligent Assistant System developers to clarify some notions about context and to study the question of context sharing. Moreover, we consider two important aspects of man-machine cooperation, namely explanation generation and incremental knowledge acquisition. Making context explicit in cooperative systems is the key factor for any implementation of these two concepts. Starting from our experience in the development of knowledge-based systems, especially of an interactive system for incident management in subway control, we explain our views about context for the development of intelligent assistant systems.

Key words: *Intelligent Assistant Systems, Cooperation, Context, Explanation, Incremental Knowledge Acquisition.*

Paper received: April 1998.
Accepted in revised form: February 1999.