SART: AN INTELLIGENT ASSISTANT SYSTEM FOR SUBWAY CONTROL

P. Brézillon[1]
R. Naveiro[2]
M. Cavalcanti[2]
J.-Ch. Pomerol[1]

(1) LIP6, case 169, Université Paris 6
4, place Jussieu,
75252 Paris Cedex 05, France
Tel.: +331 4427 7008 – Fax: +331 4427 7000
{brezil, pomerol}@poleia.lip6.fr

(2) COPPE/UFRJ, Universidade Federal do Rio de Janeiro
Caixa Postal 68507
21945-970 – Rio de Janeiro – Brasil
tel/fax: +55 (21) 590-8817
{ricardo, marcos}@pep.ufrj.br

Abstract

One of the main characteristics of a subway line is its large transport capacity (e.g., about 60000 travelers per hour in the Parisian subway) combined with a regular transport supply. The regularity is particularly important at rush time – peak hours – when an incident can provoke important delays. Experience shows that the consequences of an incident are highly dependent on the context in which the incident occurs (e.g., peak hours or not). The decisions taken by the operators are heavily relied on the incident context, and operators often make different decisions for the same incident in different contexts. The project SART (French acronym for Support system for traffic control) aims at developing an intelligent decision support system able of helping the operator in making decisions to solve an incident occurring on a line. This system relies on the notion of context. Context includes information and knowledge on the situation that do not intervene directly in the incident solving, but constrain the way in which the operator will choose a strategy at each step of the incident solving. The paper describes the SART project and highlights how Artificial Intelligence (AI) techniques can contribute to knowledge acquisition and knowledge representation associated with its context of use. Particularly we discuss the notion of context and show how we use this notion to solve a real-world problem.

Keywords: intelligent decision support system, context, multi-agent system, subway control.

ISSN 0101-7438

1. Introduction

One of the main characteristics of a subway line is its large transport capacity (about 60 000 travelers per hour in the Parisian subway) combined with a regular transport supply. The regularity is particularly important at rush time – peak hours – when an incident can provoke important delays. Thus, the transport supply must always remain compatible with the users demand by keeping the time interval between trains close to the scheduled one given in a timetable.

The regularity of the service is obtained by a complex association of automatic communication systems that allow on-line operations and permit operators to be aware of which events are occurring at a given moment. The control of the train timing aims to check if each train follows the theoretical timetable.

A model of a subway line contains three types of information: a model of the line sections, theoretical and practical timetables and regulation algorithms. A model of the line sections contains static and dynamic information about the track and equipment of the line. The static information (e.g. a station, the architecture of the lines) does not change rapidly over the time. The dynamic information concern data like trains (how many trains are in operation at a given time), travelers, train drivers and so on.

The timetables give the interval between trains and account for the hour of the day (peak or off-peak period), the day of the week (working day or week-end), the season, holidays, etc. Most of the work done by the operators is to ensure that trains follow the timetable which has been established on the prior experience basis – the Parisian subway is in operation for one hundred years. The timetable therefore appears to be a compiled expression of a number of contextual information concerning train regulation (number of trains, moment of the day, driver availability, etc.) The regulation algorithms are in charge of helping the operator to restore the regularity of the service when some incident occurs. Indeed, operators spend a great deal of their working time in managing incidents.

An incident may concern any of the three types of information, particularly the static elements of the line (e.g., a station), trains (e.g., a door blocked), travelers (e.g., a suicide), train drivers (e.g., starting too late from the terminal of the line), others (e.g., a dog in a tunnel). When an incident appears, the operator must make predictive or corrective actions to avoid, if possible, a more critical situation that may disrupt the normal supply of transport on the line (especially at peak hours), and rapidly return to a normal situation after the incident (at least partially).

Experience shows that the consequences of an incident ***are highly dependent on the context*** in which the incident occurs (e.g., peak hours or not). The decisions taken by the operator heavily rely on the incident context; and operators make different decisions for the same incident in different contexts. For example, the operator will consider that a rush of persons will be soon in the subway because a football match will be ending within few minutes, even if it is in the evening at an off-peak time. Thus, for identifying an incident, one needs to know its context, its origin and the consequences on the traffic.

The high dependency on context is particularly important with heavy traffic. In this case, the number of incidents grows and rapidly becomes difficult to the operator to manage all the incidents. So, when the number of trains in circulation (and thus the number of users) is at its highest level, each incident can lead rapidly to a situation very far from the normal operational conditions. For example, on the French regional express subway (RER in French),

it has been shown that a delay of 10 seconds at a station is propagated all along the RER line and may imply in a delay of 30 min several stations beyond the initial station; in less than one hour. However, an incident occurring during off-peak hours may also lead to a peak-hour situation if (1) the incident solving process during off-peak hours cannot be solved before the following peak hour time, and (2) an external event suddenly transforms the off-peak hour situation in a peak hour situation. An example of the latter situation is an incident occurring at off-peak hour when the rain abruptly falls outside; a number of persons change their mind and take the subway to continue their travel.

Thus, elements intervening in an incident situation are: the state of the line, the incident identification and the former strategy applied, the action carried out by the operator, and the *context* in which the incident occurred.

The project SART (French acronym for Support system for traffic control) aims to developing an intelligent decision support system to help the operator who controls a line in taking decisions to solve an incident. In order to be a real intelligent assistant system, SART has to accomplish several functions (Brezillon & Cases, 1995) such as acquiring knowledge from operators; simulating the traffic on the line starting from the station where occurred an incident; changing the model of the line under operator's request in testing alternative issues; proposing alternatives for an incident solving; training a new operator not familiar with a given line; etc.

We are developing this project by using a multi-agent approach. In such an approach, each agent proceeds a function of SART. To date, our group has developed three agents in the last two years, namely a configuration agent, a simulation agent and an incident-management agent. We develop SART according to an off-line approach. This means that the operator provides the data, but an on-line use of the simulator can be considered later, with a data acquisition system by the simulator directly from the subway line. Specifications are now completed and we are in the programming phase.

The multi-agent approach was chosen to permit SART to have an evolutionary architecture in which other agents – such as a communication agent, a training agent and an incident-analyzer agent – will be added later to increase SART functionality. However, we limit our objective now to the line control and regulation, but the structure of SART should be reused easily for other tasks, say, for line maintenance. Thus, we progressively develop a complex system that will accomplish different tasks. Moreover, the architecture can also be reused for other subways in the world because all the subways rely on approximately the same philosophy and architecture.

The SART project must be considered at different levels. First, it is the object of two conventions, one between the University Paris 6 and RATP (the French company who has in charge the subway in Paris, France), and the second between the Federal University and the Metro company in Rio de Janeiro (Brazil). At an upper level, an international convention concerns the University Paris 6 and the Federal University of Rio de Janeiro with the support of two governmental agencies: CAPES (Brazil) and COFECUB (France). About thirty persons have been working on the SART project in these two countries.

This paper describes the SART project. In this first section we present our objectives, the domain chosen for this application – namely the line control in subways –, the problem to solve and the context in which is embedded the SART project. The second section discusses the importance of distinguishing clearly different types of knowledge to introduce the need of making context explicit in the knowledge representation and in case-based reasoning.

Section three focuses on the context modeling in our application and gives an example of context-based representation of the domain knowledge. In Section four, we present the general characteristics of an Intelligent Assistant System (IAS), the architecture of SART and the multi-agent approach that we have chosen. We describe also the specificities of a SART agent, and each agent that we are developing, namely the line configurator, a traffic simulator and the incident manager. The paper ends by providing the perspectives on the project.

2. Knowledge Types and Case-based Reasoning

2.1 Different types of knowledge

Many authors have attempted to distinguish different types of knowledge. Worden *et al.* (1987) distinguish declarative domain knowledge (stored as rules and facts), dynamic knowledge (stored in datasets), procedural knowledge (stored in tasks and subtasks), task division knowledge (stored as rules), system self-knowledge about the structure and quality of its own knowledge (dependency rules, timestamps, sources of updates and measures of reliability), and "how to be an assistant" knowledge. Stothert & McLeod (1997) bring another distinction with a priori and operational knowledge. A priori knowledge defines what is known about the plant. It is used to build a framework that facilitates a solution for the control problem being addressed. Operational knowledge is the knowledge available during plant operation to determine future control actions. Reatgui *et al.* (1997) consider specific and general knowledge in a reasoning. Specific knowledge is represented in the form of cases while general knowledge is represented in the form of category descriptors.

When talking of control of the line, we mean regulation, and control of the operation from train driving to security of the travelers at an integrated level. This is different from process control in which low level processors can be used. In our system, information pieces are so numerous and heterogeneous that only composite and integrative methods can be used and moreover automation is not at hand.

These different views on knowledge show that one cannot speak about knowledge separately from its use. We will distinguish static knowledge from dynamic knowledge. Dynamic knowledge depends on the environment in which the problem occurs because this environment evolves dynamically. As a consequence of the changing environment in real-world applications, a given problem may be similar to one already treated, but never identical. This point meets Suchman's point arguing that human actions are situated and are not controlled by prior plans in the same way that a program controls a computer (Suchman, 1987).

2.2 Knowledge evolution

Part of the domain knowledge is obtained through captors providing data which are further processed by reasoning. Aamodt & Nygard (1995) underline a distinction between data, information and knowledge and envision the consequences on the development of integrated systems. For the authors, data acquisition depends on prior knowledge. An specific problem solving episode or case, mobilizes data, information, and/or knowledge, accordingly to the place and moment in the decision-making process.

The knowledge is framed by technology changes but also by specific problems. For instance, Degani & Wiener (1997) distinguish procedures, practices and techniques. Procedures are specified beforehand by developers to save time during critical situations. Practices

encompass what the users do with procedures. Ideally, procedures and practices should be the same, but the users either conform to the procedure or deviate from it, even if the procedure is mandatory. Techniques are defined as personal methods for carrying out specific tasks without violating procedural constraints. Techniques are developed by users over years of experience (Brezillon, 1996). Knowledge acquisition focuses on procedures and, eventually, practices, but rarely on techniques. Moreover, in most real-world applications, a decision maker faces ill-defined situations where the form of the argumentation rather than the explicit decision proposal is crucial (Forslund, 1995b). This implies that it would be better to record advantages and disadvantages with the final decision.

2.3 Difficulties in using knowledge

Identifying problems (diagnosis) is the first step of any troubleshooting. However, except in simple cases, it is not possible to establish a predefined table of solutions adapted to each case. This raises the difficulty to plan everything beforehand and the need to make context explicit in any application. Xiao *et al.* (1997) study anesthesiologists' work for which each patient can represent a drastically different "plant" and thus there are relatively few well-defined procedures stipulated either from inside by professional communities or from outside by regulatory agencies. It is claimed that the planning process is necessarily fragmentary and nonexhaustive. The same conclusion is observed in other domains too (e.g., see Hoc, 1996; Debenham, 1997; Bainbridge, 1997). The main reason is the impossibility of making explicit the relationships between a system and its changing environment. In an application in air control, Degani & Wiener (1997) have shown that the descent phase is highly context-dependent due to the uncertainty of the environment (e.g. air traffic control, weather), making it quite resistant to procedurization. Hollnagel (1993) proposes a contextual control model that distinguishes between a competence model (actions, heuristics, procedures, plans) and a control model (mechanisms for constructing a sequence of actions in context), which are both heavily influenced by context (skills, knowledge, environmental cues, etc.).

A solution to the impossibility of a comprehensive a priori planning is the acquisition of skills for anticipating or look-ahead (see Pomerol, 1997). For Hoc (1996), the anticipative mode is the usual functioning mode of human beings: the human operator always checks, more or less explicitly, hypotheses instead of being in a situation of discovery or surprise. An anticipatory system would be a system that uses knowledge about future states to decide what action to make at the moment. An anticipatory system has a model of itself and of the relevant part of its environment and will use this model to anticipate the future (Ekdahl *et al.*, 1995). The system then uses the prediction to determine its behavior, i.e., it allows the future states affect its present state. An intelligent system should be able to predict what will probably happen and pre-adapt itself for the occurrence of a crucial or time-critical event. This implies that a simulation component must be present in an intelligent system.

One must also underline that if it is difficult to fix the status of the knowledge; the means used to represent knowledge intervenes too: the representation determines how the knowledge is manipulated and as a result affects the "intelligence" of the system.

2.4 Context and case-based reasoning

We have pointed out that two incidents are never identical because the environment is never exactly the same and it is necessary to account for the respective contexts of the incidents. This is not a problem for small domains, but for large repositories storing different

information (e.g., multifunctional information bases or federated databases), a request specification becomes a bottleneck (Jurisica, 1994). This problem begins to be addressed in different communities such as machine learning and case-based reasoning.

In case-based reasoning, Jurisica (1994) proposes a context-based similarity as a basis for flexible retrieval. The proposed similarity assessment theory represents explicitly constraints for similarity matching, and the context is defined as a set of attribute values with associated constraints. Context allows us to specify how the matching should perform, giving local-based matching criteria as proposed by Kolodner (1993). It also allows us to specify which parts of information to compare and what kind of matching criteria to use. Items are then considered relevant if they are similar to the current context. The motivation for considering context is its ability to bring additional knowledge to the reasoning process and thus focus attention on relevant details (Light & Butterworth, 1993). When retrieving items, we might want to consider all attributes used to describe them or only certain subsets of them – the relevant subsets may be different depending on the situation.

The latest point is important because it may be impossible to represent all the attributes of a real world situation, and the items in the repository are constrained by a limited number of information carried by them. We can see this constraint as a context applied to real world objects, an implicit context. Implicit context allows us to map real world objects into a particular representation (an item) and explicit context allows us to define a specific view of the item in an information base. Explicit context allows to define mapping between items defined in different schematas (Jurisica, 1994). Reatgui *et al.* (1997) points out that context may also play a role in another way. Some findings can imply the presence of other findings. Thus the system is able to realize that a finding may be present in one case and not present in a second one, but its existence implies the presence of other findings that can be observed in the second case. This type of inference permits two cases that do not contain a large number of common findings to have a good degree of similarity. Thus, similarity judgments are made with respect to representations of entities, not with respect to the entities themselves.

2.5 Appraisal

Our approach relies on a decomposition as discussed in this section. We distinguish domain knowledge, meta-knowledge to use domain knowledge and the knowledge for communication. Each of the three categories presents its own problems. The meta-knowledge is twofold. On one hand, the meta-knowledge explains "how to be an assistant," and, on the other hand, it carries out the knowledge about the interactions among the software agents. Starting with three agents inside SART, we will tackle this last type of knowledge (knowledge for communication) by designing and developing the communication agent.

This brief overview about the nature of knowledge leads us to the following conclusions concerning the SART project. A part of the domain knowledge can be represented by static knowledge, e.g. to build a line model. However, a part of the domain knowledge may change because the environment evolves dynamically. As a consequence, an intelligent assistant system must have: (1) a context-based representation of the changing knowledge (technology changes and description of incidents) and a special type of case-based reasoning; (2) the ability to assemble dynamically fragments of plans to bring an efficient decision support; and (3) simulation means for look-ahead.

3. Context and Knowledge Representation

3.1 Context in the subway application

Contextual considerations are normally drawn up on the basis of operators, drivers and maintenance staff's experience. They analyze the on-line situation when an incident occurs (time of occurrence, track loads, seriousness of the incident, etc.) to define the strategies to be adopted to restore normal service.

Representing knowledge with its context implies to rethink knowledge representation. There are two aspects of context that play an important role. One aspect is that context is composed of relationships between different knowledge pieces. This gives a static view on knowledge and the way in which knowledge pieces are related within their context of use. A dynamic view consists of considering context as a mechanism of contextualization for retrieving knowledge (Edmondson & Meech, 1993), and its links to the reasoning mechanism that associates the considered incident with incidents known by the system. We introduce these two aspects of context in SART.

The incident-solving context contains a number of items (e.g., it is almost the end of the football match and a number of persons will come back by the subway) among which the operator will account for the most relevant ones to choose a strategy for the incident solving. Operators retrieve these items from their experience or from that of other operators. We define incident solving as a sequence of steps and distinguish two types of context: (1) the context of a step, and (2) the overall context of the incident solving. Processing the incident solving from one step to the following one implies to move from one context to another, generally after the occurrence of a new event or a new information. After this move, some pieces of contextual knowledge become uninteresting at the new step of the incident solving, and other pieces appear (knowledge pieces that are new or were previously contextualized). However, if one may speak of a discrete set of (static) contexts at the lower level as McCarthy (1993), at the level of the incident itself, there is a global continuous context (i.e., the incident-solving context) that evolves dynamically along the successive steps. An integrated view of these different contexts is now given in (Brézillon & Pomerol, 1999).

3.2 An example of context-based representation of knowledge

Brézillon *et al.* (1997) present an example (Figure 1) which gives a partial view of the reasoning induced by the elementary incident "Sick traveler in a train." Incidents are represented by ovals. Part of the steps of the incident solving is represented as rectangular boxes, e.g., "Alarm signal," "Stop at the next station," "Incident identification" and "Call operator." Consider the step "Stop at the next station." This step – contextualized knowledge – is imposed on the driver because, for example, this corresponds to procedures. Procedures arise from experience with similar incidents, and thus come from practices. For example, travelers' security is better ensured in a station than in a tunnel, employees of the subway are not allowed to take care of injured persons, etc. At a deeper level, the driver has to avoid stopping the train for a long time in a tunnel. One reason for this is that some travelers may have claustrophobia and leave the train to wander about on the railway.

All these pieces of contextual knowledge are not at the same distance of the step "Stop at the next station." Some pieces of contextual knowledge are *nearer* than others. For instance, "Procedures" is a contextual knowledge that is close to the incident-solving step, while "Avoid stopping in a tunnel" is another piece of contextual knowledge that is far from the

same step. However, both of them make this step necessary. Such a kind of distance permits to order pieces of contextual knowledge in layers around the step like skins of an onion. We call this the **onion metaphor**. Layers of contextual knowledge are represented by stippled circles in Figure 1.

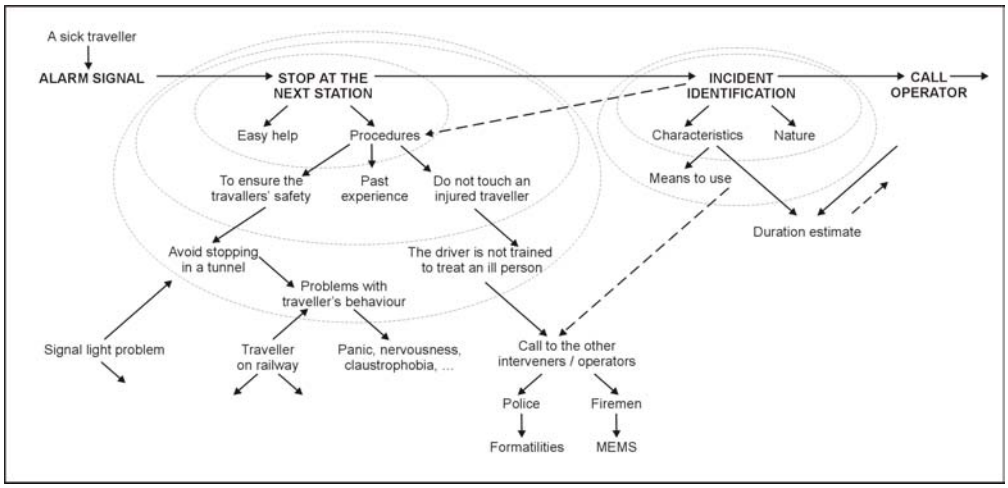

Figure 1 – Context-based representation of the incident "Sick traveler in a subway"

Tichener (cited by Jansen, 1995) also introduces the notion of a situation surrounding the organism as one of the roots of context. Tichener's definition implies that context is not part of the actual chunk of knowledge but forms a layer, or a set of layers, around the knowledge.

The onion metaphor reveals several interesting results:

(1) A step takes a meaning in a given context. Contextual knowledge does not intervene directly at this step but constrains it. For instance, in the step "Stop at the next station," the contextual knowledge "Easy help" is not the main reason for stopping the train at the station. However, it intervenes in its realization.

(2) Pieces of contextual knowledge may be partially ordered. 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 step than others (e.g. "Do not touch an injured traveler") because the constraints applied on this step are more direct.

(3) Contextual knowledge itself takes a meaning in a context. A piece of contextual knowledge "Procedures" of the step "Stop at the next station" has its own context with elements as "Past experience" and "Do not touch an injured traveler." Recursively, one can link knowledge pieces together by layers. This result is a concrete example of McCarthy's claims (1993) about the definition of a context in an outer context and about the infinite dimension of context.

(4) Pieces of contextual knowledge relate incidents together. With an association between a number of contextual-knowledge pieces, it appears that there is a relationship between a given incident and the others. Figure 1 shows how such a relationship is established between "Stop at the next station" and another such as "Signal light problem" through contextual knowledge. Establishing such an incident

net accounts for the occurrence in real-life conditions of combinations of incidents that seem apparently not related (e.g. "Sick traveler in a train" and "Signal light problem".) Common contextual components (or pieces of knowledge) of two contexts are highlighted by dotted, bold arrows in Figure 1.

We now address only one type of context, the context at the level of the knowledge representation. This type of context will have to be related to others such as the context at the level of the reasoning mechanism and the user-system interaction. The representation of contextual knowledge in SART is implemented regardless of formal considerations (logic, programming languages, expert systems, etc.).

4. General Characteristics of SART

4.1 What is an intelligent assistant system (IAS)?

To be an intelligent assistant, a system must accomplish many functions, firstly, for the transfer of knowledge and information to the user, and, secondly, for accepting knowledge and information from the user. The knowledge transfer from the system to the user was well studied for many years, and most expert systems and knowledge-based systems are ascribed to this realm. However, limits of such systems are now well known (e.g., see Brézillon & Pomerol, 1997 for a survey, and the Special Issue on Successes and Pitfalls of Knowledge-Based Systems in Real-World Applications. Failures & Lessons Learned in Information Technology Management, June, **1**(2), for discussions on several examples).

The main problem is that knowledge-based systems lack interactivity and acquisition skills (Pomerol & Brézillon, 1996): There is no possibility for the system to accept knowledge and information from the user or the environment, and it cannot dynamically tailor its behavior to user's needs. Frontin *et al.* (1993) points out that a system that cooperates with a user should have the possibility to adapt its behavior from the simple observations of users' actions. Aamodt & Nygard (1995) claim that a decision support system should be able to provide the user with the right information when it is needed, and provide suggestions and criticism to the user during decision making. Here, the most important point is the contextual dimension of the system's intervention (see Jones & Mitchell, 1994, for a survey). Worden *et al.* (1987) give the example of the user's overriding interventions. When a decision has been overridden for various reasons the assistant does not understand, and the same decision is required to be made in altered circumstances (i.e. other contexts), the assistant should know when it is out of its domain and ask the user for his decision again. Fischer (1990) shows that the user is able to make statements about the domain that are out of context with respect to the current dialogue between the user and the system. Volunteering information allows users to be in the speaker role and focus the attention of the system on the information that they feel is relevant. The user is no longer just answering questions, but takes an active role in deciding what the knowledge-based system is reasoning about. Thus, the system plays the role of assisting users as opposed to directing users.

However, a system cannot have all the needed knowledge at the design time. Previously, the user provided the system with the missing knowledge. However, such systems left in the hand of the end-user rapidly became intractable. The solution is that the system learns from experiences with users during the work process. Learning implies that the system is able to acquire, represent and use the knowledge in its context of use. This is an important feature, especially in a changing environment (Fischer, 1990; Aamodt & Nygard, 1995; Ekdahl *et al.*,

1995; Reddy, 1996; Bainbridge, 1997). However, effective human-computer communication requires providing the computer with a considerable body of knowledge about the world (Fischer, 1990) and integrating the functions of databases, information systems and knowledge-based systems (Aamodt & Nygard, 1995).

4.2 IAS architecture

An Intelligent Assistant System (IAS) has two sides: the real-world process side and the human side. One changes the system (process side) during the design time and the interface (human side) through training and experience. This implies that an IAS has to understand: (a) the real-world process; (b) the current tasks; and (c) the operator's behavior. This constitutes three inter-dependent knowledge bases as represented in Figure 2.

In Figure 2 it is represented the relationships between the IAS, the operator and the real-world process, and the three types of knowledge that the system needs to cooperate:

(a) The real-world process (Process model).
This type of knowledge corresponds mainly to a model of the process. The model enables the IAS to observe the process behavior by a comparison with the simulated behavior resulting of the same inputs given to both the process and its model. The three goals are: to observe the evolution of the process; to verify the coherence between variation of the process behavior and operator's actions; and to enable the operator to simulate alternative solutions before making a decision. Models of real-world processes generally exist beforehand because they are important tools for the control of the processes by simulation.

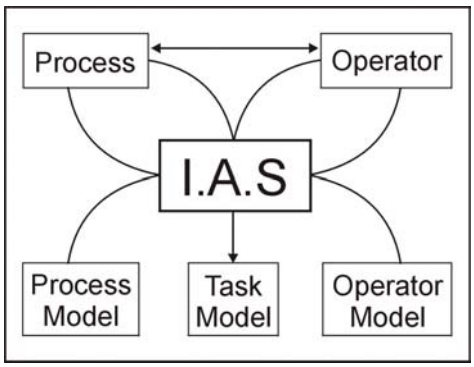

Figure 2 – General architecture of an IAS

(b) The Task model.
The task knowledge permits the IAS to simulate operators' activity. The goals are threefold: identification of operators' intentions from their action sequences, observation and explanation of the process behavior. Eventually, the IAS 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 delayed time when the operator is not under time pressure. Note that the IAS is then only an observer and stays in an attentive waking state.

(c) The operators (Operator model).
This type of knowledge corresponds to the sequence of actions of the operators facing a problem in the process. From operators actions, the IAS may deduce from knowledge in (a) and (b) operators intentions, and their preferences from their choices during solving the problems (e.g., shortcuts). Identifying later a similar situation, the IAS 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 modelling the process, i.e., a short elicitation phase and an incremental development of the knowledge base during its use. The operator model can be enriched by adding operator preferences that may be determined automatically from a series of problem solvings.

This approach has the advantage of rapidly providing a mockup that will be improved incrementally on the knowledge bases and later on the development of the visible part of the IAS, namely the interface. The IAS may learn (i.e., acquire knowledge) from the behaviors of the operator and the real-world process. This is a kind of incremental knowledge acquisition where knowledge is acquired when needed and in its context of use. 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 such tasks as acquisition, assimilation, learning, validation-verification, information retrieval, indexing.)

4.3 Role of an IAS

The IAS must intervene spontaneously at an operator's utterance, such as: announcement of the procedure that the operator intends to follow, the goal to be reached, the interpretation they make of some interface object, etc. 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 IAS acts as a decision support system. It must also 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 such as: human-computer interaction management, dialogue management, explanation, documentation, simulation, user modelling, and support all aspect of conflict management.

In complex tasks, the real-world process is under the control of the operator that will always make the final decision. For instance, Jones (1995) presents supervisory control systems as dynamic, event-driven, worlds in which human operators are responsible for the health and safety of a system that they control. The joint cognitive system perspective defines an IAS as a resource or a source of information for the problem solver (Woods, 1995). According to this perspective, the main form of cooperation is information sharing (Jennings, 1994). Even with fewer capacities than humans, IASs are more efficient when their capacities have a nature similar to those of the humans (Millot, 1995). This implies a mutual intelligibility, a "common ground" for communication and understanding (Jones & Mitchell, 1994). Cooperative work is often a form of redundancy that helps to ensure the reliability and safety of the system; coordinated activity is frequently planned in advance and quite structured as a "standard operating procedure"; and when unexpected events arise, the composition of the team itself typically changes (e.g., senior engineers join in to assist in problem solving).

SART follows the lines described in this section with however some restrictions on the architecture for taking into account the characteristics of our application. Thus, the process model needs to be changed for a new line or for introducing changes in an existing line. This implies that a part of SART must manage the process model, when another part will use it for simulation purposes. The task model concerns mainly incident management. According to our objective to have a dynamic management of incidents, another part of SART must be devoted to this task. We have chosen the multi-agent approach to, firstly, address these different tasks (line configuration, traffic simulation and incident management), and, secondly, to have the opportunity to introduce later other agents. Thus the process model is built by the line configurator (configuration agent) and is used by the traffic simulator (simulation agent), and incidents are managed by the incident manager (incident-management agent). Once this minimal structure of SART is built, we will develop a communication agent that will have, among other tasks, to maintain an operator model. We first discuss the specificities of our agents.

4.4 The multiagent representation of SART

4.4.1 Agent specificity

In SART, each agent (e.g., the incident manager) is a specialist that interacts with specialists in other areas (e.g., the line configurator and the traffic simulator). Thus, we are not in the position where a given task can be accomplished by different agents having approximately the same competences as in reactive agents

the case of reactive agents (Drogoul, 1993). An advantage of our approach is that each agent can be designed and developed independently of the others.

We consider that an agent must be considered as a reasoning mechanism that works with three knowledge bases:

- A Personal Knowledge Base (PKB) that contains knowledge on how to communicate and how to work;
- A Static Knowledge Base (SKB) that contains the description of the domain knowledge, e.g., the physical description of what a subway line is composed of; and
- A Dynamic Knowledge Base (DKB) that contains the way of how to use domain knowledge as the assembly elements

Figure 3 gives a general presentation of an agent in SART. The head of the agent (the half circle) ensures the communication with its environment, mainly with the communication agent when this agent will be built. It receives a request (input) and provides an answer (output). The head acts as a translator between what is entered and what it must do. The body (PKB) acts as an inference engine that will solve the problem that is submitted. SKB and DKB are just described above. The DKB is required because where an expert system can be happy with just knowing what to do, an IAS should also (or rather) know why to do it (see also Forslund, 1995a). For example, the SKB of the line configurator contains items such as the definition of a station, and its DKB the general length of a station, specificities of a station in a curve. The interest to distinguish these two types of knowledge comes from the fact that a change may occur in the general strategies of the subway company. For instance, trains in RER and subway in Paris have not the same number of wagons. Moreover, DKB is context-sensitive while SKB is not context-sensitive.

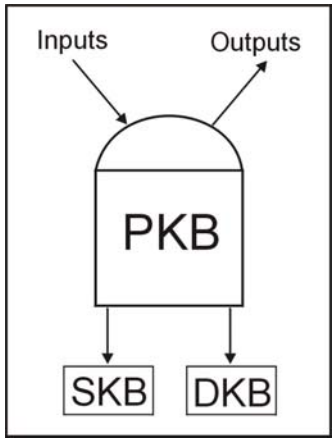

Figure 3 – A general schema for an agent.

In a multi-agent system, each agent uses a particular expression of the domain knowledge. In SART, the line configurator uses the description of equipment pieces and their relationships--the static knowledge--and produces the model of a subway line that is used by the traffic simulator as its own domain knowledge. The building of the line model is the result of the interaction of the line configurator with a user. To do that, the line configurator uses knowledge – dynamic knowledge – on constraints imposed on equipment and contextual information. A piece of contextual information may be that a part of the line is a curve in an ascent. For the traffic simulator, contextual knowledge comprises the initial conditions of the simulation and the definition of an incident during the simulation.

Our architecture of an agent is in the realm of the layered representations found in the literature. Huang *et al.* (1995) propose a layered agent architecture for decision support applications. The three layers are domain knowledge, inference knowledge, and control knowledge. Muller & Pischel (1994) structure the knowledge base in four layers that basically correspond to the structure of the agent control. The lowest layer contains facts representing the world model of the agent. The second layer defines the primitive actions and the behavior pattern. The third layer contains local plans. Finally, the layer four contains knowledge of and strategies for cooperation, e.g. beliefs about other agents' goals.

Most real world problems require the integration of different specialists, each of whom contributes to a unique point of view, and moreover one aspect of expertise is the ability to integrate specialist knowledge in some real problem context. Woods (1995) calls it the generalist-specialist problem. A three-layer model for organizing systems of interacting agents based on situation action is generally proposed (e.g., see Wawish & Graham, 1995). The top layer of the model consists of agents performing roles, the middle layer provides the skills which agents need to perform their roles, and the bottom layer consists of the behaviors that are needed to realize these skills. The model may be realized computationally by means of a production rule language or in any object-oriented programming language. Such specialists or agents may concern the simulation of a real-world process, its monitoring, its diagnosis, etc. For instance, Mentzas *et al.* (1995) describe an Intelligent Forecasting Information System, which, besides the traditional components of a decision-support system, contains four constituents that try to model the expertise required: a Process Expert, a Learning Expert, a Data Expert, and a Model Expert.

With such an architecture, an agent has the properties cited by Wooldrige & Jennings (1995):

- autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their own action and internal state;
- social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;
- reactivity: agents perceive their environment (which may be the physical world, a user via a graphical interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;
- pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative.

Thus an agent may behave in a situated, efficient and goal-directed way, and be able to interact (i.e. coordinate and collaborate) with other agents.

4.4.2 Architecture of SART

The SART project has started with three agents: a line-configuration agent, a traffic-simulation agent and an incident-management agent. The fourth agent will be the communication agent. The main reason for the last agent is to limit the cognitive overload of the user that is facing several agents and communicating with different languages and interfaces. The secondary reason for this fourth agent is to answer to complex questions in which several agents must intervene in the answer building. The Figure 4 shows the architecture of SART.

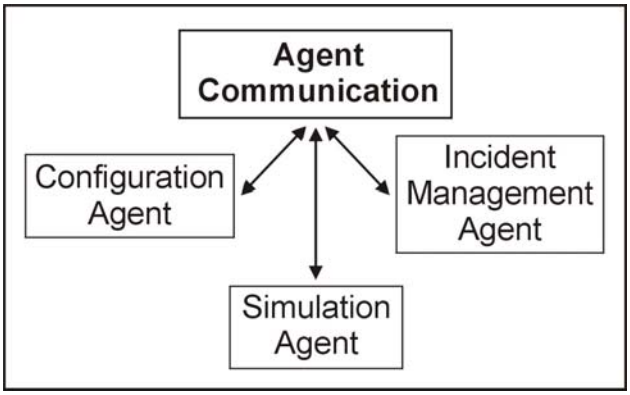

Figure 4 – SART architecture

The role of the configurator agent is to support the operator either in building the model of an specific line or for changing an existing model. The role of the simulation agent is to support the operator for the validation of the model, generating answers about an incident or analyzing past incidents. The role of the incident management agent is to record sessions during which an incident occurs and to retrieve past incidents similar to a given one, in a case-based reasoning way. We now describe each agent in more details.

4.4.3 The communication agent

The main characteristics of the communication agent are:

Tasks: management of the interaction with the operator and agents
management of a session

Input: a request from the operator or another agent

Output: the answer to the request

PKB: How to execute a subrequest sequence, to transfer information between agents, and to choose the medium for information exchanges

SKB: task definition of agents, frames of possible requests

DKB: decomposition of a complex request in subrequests manageable by the other agents, building of the answer for a complex request,

Comments:
The communication agent avoids a cognitive overload of operators

The communication agent is responsible for the communication with the user and among all the agents. In the latter case, it must be capable of managing complex requests such as: How is the traffic of this line? Which part of the line may be incident-sensitive? Is this change of the line adequate for avoiding 10% of the incidents? What is the evolution of this incidental situation? What is the traffic evolution after this incident? For a complex question asked by the line configurator such as "What are the risks for this line?", the communication agent plans its answer in the following way. First, it asks the simulator to check the line in normal situations. Second, it asks the incident manager to retrieve known incidents for this line. Third, it asks the operator for possible incidents not known of the system.

The communication agent will have a Man-Machine Interface (MMI). This interface must proceed all the inputs that come from the external environment of the application. For accomplishing its tasks, the communication agent has appropriated windows for introduction and selection of parameters to build its Static Knowledge Base (SKB). During the construction of the SKB, the communication agent is responsible for the dialogues with the operator. It analyzes the contents of the dialogue and produces a set of messages that are related to the questions. These messages are sent to the line configurator which will choose the actions to be performed. The MMI has thus functions that allow the operator to introduce, verify and modify parameters that are used on the simulation of the Dynamic Knowledge Base (DKB).

4.4.4 The line configurator

The main elements of the line configurator are:

Task: Support the operator in the building or change of a line model

Inputs: Characteristics of a line given progressively by interaction with the operator.

Output: The new model of the line

PKB: Choice of relevant alternatives, addition of elements not planned beforehand

SKB: Line sections, time tables, regulation algorithms

DKB: How to assemble elements of the SKB in the current context

Comments:
The line configurator performs two types of validation:
- an intrinsic coherence of the model with respect to the SKB and DKB
- an extrinsic coherence of the line on a set of typical situations submitted to the simulator

The role of the line configurator is to support the operator in the building of a model of a future subway line or in introducing changes in an existing model. Such a model contains information such as line sections, timetables, regulation algorithms. For accomplishing its task, the line configurator has a knowledge base containing: (1) the existing models of subway lines, (2) the basic elements for building a line (e.g., a section, a station, a train, etc.), and (3) the knowledge and the know-how about the model building according to some constraints. The development of a line model is made iteratively by interaction with the operator that knows the physical and operational data of the line.

The two first types of knowledge can be described in a structured way and constitute the SKB. For instance, a line is composed of sections, a section can be a tunnel, a station, or a switching, etc. The third type of knowledge describes how to handle information on the first two types of knowledge. For instance, if the number of section is more than 10, then it is necessary to have two switchings to insure a temporary traffic (if a serious incident occurs near the middle of the line).

The operator provides the characteristics of the line to build (number of sections, of stations, etc.). All this information defines the physical characteristics of the subway line in the model. The configurator builds the line, verifying that constraints are respected and the coherence with respect to the part of the line already built.

Indeed, we consider that the static knowledge base contains classes that are instantiated by the operator in the model for a specific line. This approach is particularly modular and the change of one instance does not imply change of the whole model. The advantages are twofold. Firstly, the operator can introduce temporary changes in an existing model either to check the elimination of a class of incidents or to simulate the consequences of alternative decisions implying a change of the line structure. Secondly, the line configurator can be applied at a low cost at any subway in the world because most of the subways are built on the same principles.

4.4.5 The traffic simulator

The main characteristics of the traffic simulator are:

Task: Provide the train movement on the line for a given situation, interact with the operator or the incident manager to simulate the traffic during and after an incident

Inputs: A set of initial conditions,
The definition and the solving of an incident by either the operator or the incident manager

Output: The traffic evolution on the line before, during and after an incident

PKB: Context-based knowledge and incident-based reasoning, simulation management

SKB: The line model, the timetables, and the regulation algorithms

DKB: Possible changes in the timetables and regulation algorithms

Comments:
The traffic simulator can be called for prediction purposes by:
- the operator, to check alternative changes on the line or of train movement
- the line configurator, to check the dynamic behavior of trains on a new line or a modified line
- the incident manager, to give the traffic evolution after an incident solving

The traffic simulator gives the dynamic behavior of the trains on the line (e.g., movement of the trains on the line, calculus of departure time of trains, etc.) for a given situation (e.g., position of trains at a given time), possibly with an incident. The main function of the traffic simulator is to give the train movement and the management of signals on the line. The traffic simulator can be called by the operator to check alternative changes on the line or in the movement of the trains, by the line configurator to check the dynamic behavior of a new line, and by the incident manager to give the behavior of the traffic during and after an incident. Thus, an important interest of the traffic simulator is the prediction of traffic behavior in different situations. The prediction can be used by the operator that wishes to intervene in the train traffic or to know the consequences of a possible decision (e.g., to suppress a train, to change the timetable).

The traffic simulator would be integrated easily with the on-line control system. Acquiring data on-line will improve notably the behavior of the traffic simulator, e.g. with a predictive view of the traffic normally not seen by the operator.

4.4.6 The incident manager

The main characteristics of the incident manager are:

Task: Management of incidents

Inputs: Definition of an incident, its context and an action to attend

Output: List of incidents in similar contexts, list of contexts for similar incidents

PKB: Context-based knowledge and incident-based reasoning

SKB: Context-based representation of incidents, their contexts of occurrence and the strategies applied to solve them

DKB: Tailoring of a similarity measure between two incidents

Comments:
At each session, the incident manager records information on the considered incident, its context and the strategy finally retained.
Contextual elements of an incident are: line topology, available means, position and state of the other trains on the line, the degree of experience of the driver, etc.
The incident manager plays the role of the corporate memory of the company.
Incidents are serious perturbations of the traffic for which regulation algorithms are insufficient to support operators in incident solving.

The operator provides SART with the characteristics of an incident, the moment of its occurrence and the traffic situation at that moment. These pieces of information permit SART to draw up, from its base of incidents (recorded with their contexts), a strategy (or even more) likely to be adopted by the operator. For each incident, SART keeps a record of events that allow a careful analysis of the events leading up to the incident, and therefore an understanding of the decision and the selection process for the solution strategy proposed by

SART. SART incrementally acquires data and knowledge on the present incident, its context and the strategy used to exploit this knowledge for solving similar future incidents.

When an incident occurs, several elements must be known or estimated carefully: the topology of the line, the means that are available, the position and state of the other trains on the line, the importance and nature of the incident, the context of the incident, the degree of experience of the operator, etc. The goal of SART then is to: acquire data and information concerning the incident situation, retrieve similar incidents in the past and similar contexts, propose an ordered list of the strategies used previously, record all the information on the incident situation. Recording information on an incident situation presents an interest in order to identify future incidents, but also for a detailed analysis of that incident and the chosen strategy, the classification of the incidents, and other operations on the incidents (e.g. training).

The incident manager keeps a trace of the corporate memory that would be reused, say, for training purpose.

4.5 Current status of the SART project

The whole complex work of acquisition and representation of the knowledge of the operators is now concluded. The three agents (excluding the communication agent) meet their final phase of development and a mock-up of the three agents will be available soon. For the current version of the system we are working under the following restrictions:

- the interaction between the line configurator and the traffic simulator will be limited, at the beginning, to the line model of the subway (the first building it and the second using it). In the future we should allow the operator to change the line model to test another strategy.
- the dynamic information on the state of the lines is introduced off-line (by the operator himself). In another version of the system it should be capable to capture this information on-line.

The main results obtained at the theoretical level are:

- a representation of the contextual domain knowledge along the onion metaphor (Brézillon *et al.*, 1997);
- a distinction in context between the external knowledge, the contextual knowledge and the procedural context (Brézillon & Pomeorl, 1999); and
- the development of a representation formalism as contextual graphs for representing the reasoning and its dynamics in incident solving (Brézillon *et al.*, 2000).

The next step is twofold. Firstly, we will develop an environment to manage the interaction among the three agents. Secondly, we will develop the communication agent to control the interaction among the three agents and with the users.

5. Conclusions

In this paper we showed how the notion of context can contribute to knowledge acquisition and to the representation of knowledge. The context-based representation of knowledge is made according to the onion metaphor that gives the possibility of organizing contextual knowledge around the contextualized knowledge at one step of the incident solving. We thus have a set of discrete contexts attached to the different steps of the incident solving. The

incident-solving context itself evolves continuously during the incident solving. The next step is to model a context-based representation of the reasoning. Such reasoning will be context-based and incident-based.

SART is built on a multi-agent architecture. We implement now three agents, namely a line configurator, a traffic simulator and an incident manager, and we plan the design and development of another agent, the communication agent. The multi-agent approach presents several advantages. Firstly, we have an evolutive architecture in which agents will be added progressively according to the need of the application. Secondly, the same architecture can be reused for other tasks in the same domain. Now, the task at hand is the subway control. However, tasks as line exploitation and train maintenance can be implemented rapidly from this architecture. Thirdly, SART is not limited to the subway of Paris, but can be used for most of the subways in the world.

The agents in SART have all the same architecture with three knowledge bases, a personal knowledge base, a static knowledge base and a dynamic knowledge base. Such a division of the knowledge permits to make context explicit in the knowledge representation. Knowledge in the system is for the user as well as the system itself. In the case of the incident manager, the incident base will growth with the use of SART. Incremental knowledge acquisition is the best way to adapt a system such as SART to a changing environment.

The open architecture of SART is interesting for a long use of the system because most of the failures in previous knowledge-based systems come from the lack of consideration for the users (Brézillon & Pomerol, 1996, 1997). Taking into account users in their workplace requires making context explicit. The SART project shows that making context explicit opens a new insight of knowledge-based systems that can become real intelligent assistant systems.

Acknowledgments

The SART project is part of a cooperation agreement between two universities, the University Paris 6 (France) and the Federal University of Rio de Janeiro (UFRJ, Brazil), a contract between the University Paris 6 the subway company in Paris (RATP), and another contract between the UFRJ and Metrô that manages the subway in Rio de Janeiro. Grants are provided by COFECUB in France and CAPES in Brazil. We also thank A. Chevrier and J.-M. Sieur at RATP, and C. Gentile, L. Pasquier and M. Secron, Ph.D. students working in the SART project.

References

(1) Aamodt, A. & Nygard, M. (1995). Different roles and mutual dependencies of data, information, and knowledge – An AI perspective on their integration. *Data and Knowledge Engineering* (North-Holland Elsevier), **16**, 191-222.

(2) Bainbridge, L. (1997). The change in concepts needed to account for human behavior in complex dynamic tasks. *IEEE Trans. on Systems, Man, and Cybernetics – Part A: Systems and Humans*, **27**(3), 351-359.

(3) Branki, N.E. & Bridges, A. (1993). An architecture for cooperative agents and application in design. European Conference on Artificial Intelligence (EUropIA-93), Elsevier Science Publisher.

(4) Brézillon, P. (1996). Context in human-machine problem solving: A survey. Technical Report 96/29, LAFORIA, University Paris 6, October, 37 pages. (ftp://ftp.ibp.fr/ibp/reports/laforia.96/laforia.96.29.ps).

(5) Brézillon, P. & Cases, E. (1995). Cooperating for assisting intelligently operators. Proceedings of COOP-95. INRIA Ed., 370-384.

(6) Brézillon, P. & Pomerol, J.-Ch. (1996). Misuse and nonuse of knowledge-based systems: The past experiences revisited. **In:** *Implementing Systems for Supporting Management Decisions* [edited by P. Humphreys. *et al.*], Chapman and Hall, ISBN 0-412-75540-8, 44-60.

(7) Brézillon, P. & Pomerol, J.-Ch. (1997). User acceptance of interactive systems: Lessons from Knowledge-Based and Decision Support Systems. *Failures & Lessons Learned in Information Technology Management*, June, **1**(1), 67-75.

(8) Brézillon, P. & Pomerol, J-Ch (1999). Contextual knowledge sharing and cooperation in intelligent assistant systems. Le Travail Humain, **62**(3), 223-246.

(9) Brezillon, P.; Gentile, C.; Saker, I. & Secron, M. (1997). SART: A system for supporting operators with contextual knowledge. First International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-97), Rio de Janeiro, Brasil, Federal University of Rio de Janeiro (Ed.), 209-222.

(10) Brézillon, P.; Pasquier, L. & Pomerol, J.-Ch. (2000). Reasoning with contextual graphs. *European Journal of Operational Research* (to appear).

(11) Debenham, J. (1997). Strategic workflow management: An experiment. Proceedings of the International Workshop "Distributed Artificial Intelligence and Multi-Agent Systems", DAIMAS'97.

(12) Degani, A. & Wiener, E.L. (1997). Procedures in complex systems: The airline cockpit. *IEEE Trans. on Systems, Man, and Cybernetics – Part A: Systems and Humans*, **27**(3), 302-312.

(13) Drogoul, A. (1993). De la simulation multi-agents à la résolution collective de problèmes. Ph. D. Thesis of the University Paris 6.

(14) Edmondson, W.H. & Meech, J.F. (1993). A model of context for human-computer interaction, Proceedings of the IJCAI-93 Workshop on Using Knowledge in its Context, Technical Report 93/13, LAFORIA, University Paris 6, 31-38.

(15) Ekdahl, B.; Astor, E. & Davidsson, P. (1995). Toward anticipatory agents. **In:** *Intelligent Agents Agents* [edited by M. Wooldridge & N. Jennings], Lecture Notes in AI, 890, Springer Verlag, Berlin, 191-202.

(16) Fischer, G. (1990). Communication requirements for cooperative problem solving systems. *Information Systems*, **15**(1), 21-36.

(17) Forslund, G. (1995a). Toward cooperative advice-giving systems. A case study in knowledge-based decision support. *IEEE Expert*, **10**(4), 56-62.

(18) Forslund, G. (1995b). Toward cooperative advice-giving systems. The expert systems experience, Ph. D. Thesis 518, Linkoping University, Sweden.

(19) Frontin, J.; Hadj Kacem, A. & Soubie, J.L. (1993). Acquérir des connaissances et structurer le système pour coopérer. 2ndes Journées Acquisition des Connaissances, Saint Raphaël.

(20) Hoc, J.-M. (1996). Supervision et Contrôle de Processus. La cognition en Situation Dynamique. Presses Universitaires de Grenoble, Sciences et Technologies de la Connaissance.

(21) Hollnagel, E. (1993). *Human Reliability Analysis Context and Control*. Computer and People Series. Academic Press, London, UK.

(22) Huang, J.; Jennings, N.R. & Fox, J. (1995). An agent architecture for distributed medical care. **In:** *Intelligent Agents Agents* [edited by M. Wooldridge & N. Jennings], Lecture Notes in AI, 890, Springer Verlag, Berlin, 219-232.

(23) Jansen, B. (1995). Context in context, http://mac145.syd.dit.csiro.au/Context/context.html (Working Draft V4).

(24) Jennings, N. (1994). *Cooperation in Industrial Multi-Agent Systems*. World Scientific, Singapore, 43.

(25) Jones, P.M. & Mitchell, C.M. (1994). Model-based communicative acts: human-computer collaboration in supervisory control. *Int. J. Human-Computer Studies*, **41**, 527-551.

(26) Jones, P.M. (1995). Cooperative work in mission operations: Analysis and implications for computer support. *Computer Supported Cooperative Work*, Kluwer Academic Publishers, The Netherlands, **3**, 103-145.

(27) Jurisica, I. (1994). Context -base similarity applied to retrieval of relevant cases. Proceedings of the AII Fall Symposium Series on Relevance, New Orleans, Louisiana, USA.

(28) Kolodner, J.L. (1993). *Case-Based Reasoning*. Morgan Kauffmann, San Mateo, CA.

(29) Konig, R.; Cord, S. & Moller, K. (1997). Development of an intelligent management clerk in a logistic domain, rkoenig@fzi.de (récupéré sur le web).

(30) Light, P. & Butterworth, G. (Editors, 1993). *Context and Cognition: Ways of Learning and Knowing*. L. Erlbaum Associates, Hillsdale, NJ.

(31) McCarthy, J. (1993). Notes on formalizing context, Proceedings of the 13th IJCAI, **1**, 555-560.

(32) Mao, J.-Y. & Benbasat, I. (1997). Contextualized access to knowledge in knowledge-based systems: A process-tracing case study. Proceedings of ECIS'97, Vol. 1.

(33) Mason, M. (1996). Réflexions sur les systèmes adaptatifs d'assistance aux opérateurs. *Le Travail Humain*, **59**(3), 277-298.

(34) Mentzas, G.; Linardopoulos, K. & Assimakopoulos, M. (1995). An architecture for intelligent assistance in forcasting process. Proc. of the 28th Annual Hawaii Int. Conf. on Systems Sciences, 167-176.

(35) Millot, P. (1995). La coopération homme-machine dans la supervision: les enjeux, les méthodologies, les problèmes. Actes du Séminaire "Supervision et coopération homme-machine, Paris.

(36) Müller, J.P. & Pischel, M. (1994). An architecture for dynamically interacting agents. CoopIS-94, 114-121.

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

(38) Reatgui, E.B.; Campbell, J.A. & Leao, B.F. (1997). A case-based model that integrates specific and general knowledge in reasoning. *Applied Intelligence*, **7**(1), 79-90.

(39) Reddy, R. (1996). The challenge of Artificial Intelligence. *Computer Journal*, **29**(10), 86-98.

(40) Stothert, A. & MacLeod, M. (1997). Distributed intelligent control system for a continuous-state plant. *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, **27**(3), 395-401.

(41) Suchman, L.A. (1987). *Plans and Situated Actions*. Cambridge, UK: Cambridge University Press.

(42) Tendjaoui, M.; Kolski, C. & Millot, P. (1991). An approach towards the design of intelligent man-machine interfaces used in process control. *International Journal of Industrial Ergonomics*, **8**, 345-361.

(43) Wavish, P. & Graham, M. (1995). Roles, skills and behavior: a situated action approach to organising systems of interacting agents. **In:** *Intelligent Agents* [edited by M. Wooldridge & N. Jennings], Lecture Notes in AI, 890, Springer Verlag, Berlin.

(44) Woods, D.D. (1995). Cognitive technologies: The design of joint cognitive human-machine cognitive systems. *AI Magazine*, 86-92.

(45) Wooldridge, M. & Jennings, N.R. (1995). Agent theories, architectures, and languages: A survey. **In:** *Intelligent Agents Agents* [edited by M. Wooldridge & N. Jennings], Lecture Notes in AI, 890, Springer Verlag, Berlin, 1-39.

(46) Worden, R.P.; Foote, M.H.; Knight, J.A. & Andersen, S.K. (1987). Co-operative expert systems. **In:** *Advances in Artificial Intelligence – II* [edited by B. Du Boulay, D. Hogg & L. Steels], Elsevier Science Publishers, Vol. II.

(47) Xiao, Y.; Milgram, P. & Doyle, D.J. (1997). Planning behavior and its functional role in interactions with complex systems. *IEEE Trans. on Systems, Man, and Cybernetics – Part A: Systems and Humans*, **27**(3), 313-324.