Context-Based Intelligent Assistant Systems: A discussion based on the Analysis of Two Projects

Patrick Brézillon
LIP6, Case 169, University Paris 6
Patrick.Brezillon@lip6.fr

Abstract

After the success and the rejection of the first expert systems, we are now on the road of the design and development of real intelligent assistant systems, i.e. intelligent systems that use context explicitly. Such systems are called context-based intelligent assistant systems. With the accumulated experience gained from large projects (in our cases two 6-year applications), it is possible to learn from our past failures and the solutions then adopted to fill the gaps. First, we developed a successful expert system for the French power company. However, its use in different locations was too strongly constrained by contextual cues. For solving these problems, we concluded that an intelligent system must have a configuration module and a simulation module in addition to the module developed for the main task (e.g. diagnosis or incident management). Then, from a second application for the subway company in Paris (France), we first validate the initial hypothesis about the need of configuration and simulation modules, and, second, point out the absolute need to make context explicit. This constituted the input for the context-based intelligent assistant system paradigm and its related representation as contextual graphs. In this paper we present the lessons learned from our two applications as well as the main characteristics of context-based intelligent assistant systems with an emphasis on the need to make context explicit. Particularly, we point out the role of incremental knowledge acquisition, learning and explanation in such context-based systems.

1. Introduction

In Artificial Intelligence, the lack of explicit representation of context is one of the reasons of the failures of many Knowledge-Based Systems (KBSs) [5]. Some of the major shortcomings associated with traditional KBSs include:

(1) The exclusion of the user from the problem solving. KBSs were given the role of oracles and users that of novices. However, most often KBSs cannot solve unexpected problems, as users, with their practical experience, are not given the opportunity to solve them initially. What is required is cooperative solving by the user and the system, and careful consideration of the context of the cooperation.

(2) KBSs do not correctly use their knowledge. Knowledge acquired from human experts has a highly contextual texture that generally is not acquired with the knowledge because knowledge engineers asked what the experts' solution is, not how they reach it. A solution is only valuable with respect to the context in which the problem must be solved.

(3) KBSs cannot initially have all the needed knowledge. Any KBS has limited resources for problem solving and limited influence: One can never anticipate or "design away" all the misunderstandings and problems that might arise during the use of such systems. This implies that knowledge must be acquired incrementally when needed, i.e., including its context of use.

(4) KBSs cannot generate relevant explanations for their users because they do not know the context of the user's question. The generation of relevant explanations supposes a joint building of contextual explanation by the KBS and the user.

With the expression "Joint Cognitive System" (JCS), Woods *et al.* [16] stressed the fact that neither the system nor the users are able to solve the problem at hand alone. The approach puts the emphasis on the complementarity of users and the systems' competence, with this complementarity leading to a symbiosis. An important word in the acronym JCS is the word "cognitive." It implies that the system and the user are able to understand each other when solving a problem. This may entail some very complex requirements for the design of the system.

A first requirement concerns the sharing of cognitive representations, something that would be fulfilled if the users' cognitive representations are disclosed and implemented in the system. However, in many human tasks a comprehensive view of the mental representations at work in the subject's brain is not readily possible.

Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)
0-7695-1874-5/03 $17.00 © 2002 IEEE

A second requirement concerns the agreement about the context of a task. It is a necessary component for the achievement of any task. The problem is the representation of the context in machine. What does the system know? In what context is it safe to use the system or not?

A third requirement concerns the ability of both machine and human to comment and explain their respective reasoning. More generally the question at stake is the problem of cooperation between the user and the system.

A last requirement is related to the question of controlling the interaction in an interactive system. From a practical viewpoint, this implies interruptability, reorientation, resuming facilities at any point, and the possibility to obtain an information on the status of the search (Where are we? What can be done from this state? Etc.).

Other approaches such as cooperative systems and situated cognition (e.g. [9]) have also explored types of relationships between the user and the machine, but they are not fully satisfying because the contextual dimension is not the main focus. As the pendulum, we began with the machine as an oracle and the human as a novice, going to the machine as a partner of the human. However one forgets that a user and a machine interact in order to achieve something (a task, an activity, etc.) in a given context (e.g. under time pressure).

In two real-world applications, our concern was the operator that is in charge of a process monitoring. The SEPT application concerned the diagnosis of equipment failures in extra high voltage substations of the French national power company (see [4] and at http://www-poleia.lip6.fr/~brezil/SEPT). The SART application concerns the monitoring of the traffic on a subway line and the incident solving for RATP, the subway company in Paris (see [7] and at http://www.lip6.fr/SART). In a control room, an operator has few minutes (often few seconds) when an incident occurs to analyze the situation, collect contextual information, identify the context in which the incident occurs, and make a decision. Moreover, in both applications (as in others), diagnosis and decision making are strongly intertwined with the task at hand (e.g. monitoring of the process). As a consequence, any verbose system or system asking continuously questions will be rejected.

We consider operators' activity through procedures and practices. General procedures provide operators with a secure reference for incident solving, but it is a decontextualized reference. In a real situation, each operator develops his own practice to solve a particular problem in a given context from a procedure. Thus, there are almost as many practices as operators for a given procedure because each operator tailors the procedure in order to take into account the current context of the incident at hand, which is particular and specific. Procedure modeling is easy but practice modeling is a difficult task because operators assemble in a structure a number of contextual elements for making a decision for solving complex incidents.

The importance of the notion of context in intelligent systems is now largely acknowledged [3]. Pomerol and Brézillon [15] distinguish between the part of the context which is relevant at a given step of a problem solving, and the part which is not relevant (see Figure 1). The latter part is called external knowledge. The former part is called contextual knowledge, and depends on the agent, the current situation and on the decision at hand. A part of the contextual knowledge is proceduralized to be used at a given step of the decision process. The proceduralized context is the part of the contextual knowledge, which is invoked, structured and situated according to a given focus of attention (i.e. the current step of the decision process). This is the practice developed by operators. The context is not distinguished from the other objects of the reasoning, objects being in the context or not according to particular circumstances.

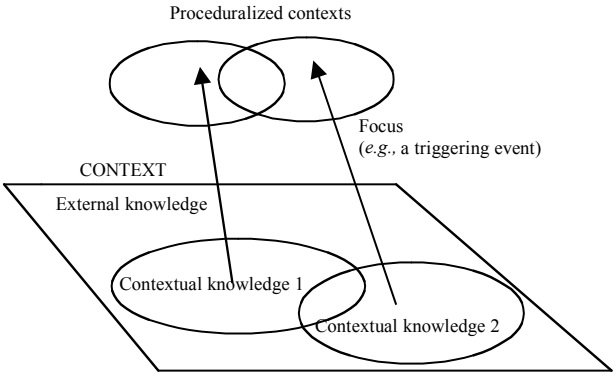

Figure 1. The three types of context

The contextual knowledge constitutes backstage knowledge whereas proceduralized context is immediately useful for the task at hand. The important issue is the passage from contextual knowledge to a proceduralized context, i.e. the dynamic dimension of the context. This contextualization results from the focus on a task at a given step.

Hereafter the paper is organized in the following way. The first two sections present each application, the SEPT application and the SART application. The following section presents the integration of the lessons learned in the framework of the context-based intelligent assistant systems. After, we discuss some points to consider for going a step further with context-based intelligent assistant systems and draw conclusions.

2. SEPT: Lessons learned from the experiment

2.1. Introduction

The French national power system is composed of controlled systems (Extra High Voltage substations, 400 000 volts) and control systems. Elements of the controlled system are lines and busbars that constitute a network. Elements of the control system are controllers. A controller links a busbar to a line or another busbar (the controlled system). A controller contains pieces of equipment like protective relays and a circuit breaker. The protective relays are triggered by a fault occurrence in the controlled system, and the fault elimination stops the functioning of equipment pieces. A protective relay detects the fault in the controlled system and, after a delay, automatically sends a trip order to the circuit breaker. Actions of the equipment pieces are associated with signals that are sent to a central printer from which the operator diagnoses the events that occur.

The diagnosis concerns the equipment pieces in controllers that present a dysfunction (e.g., the protective relay does not "see" the fault or the circuit breaker does not open). The diagnosis is threefold: determination of eventual dysfunction of equipment pieces; location of the fault in the controlled system; and detection of incoherence between fault characteristics and equipment functioning.

2.2. Knowledge organization

For dealing with several tasks (diagnosis and simulation), a system must manage different types of knowledge as domain knowledge and compiled knowledge as the know-how for each task. Thus, the same pieces of knowledge may exist as different forms of compiled knowledge. As a consequence, a multi-agent architecture at the "knowledge level" (a diagnosis agent and a simulation agent) must be organized in terms of knowledge at the implementation level. In the SEPT application, for example, the domain knowledge is organized as structure knowledge and functional knowledge [1].

2.3. Specificity of machine reasoning

Human know-how comes to a large extent from heuristics that speed up the reasoning towards a solution among a number of alternatives. Computer capacity comes from high-speed processor and large data storage capacity. For some tasks, the computer is better than a human, as in the case of certain games is. In the SEPT application, the machine made the diagnosis of all the equipment pieces, when, conversely, the operator tried to identify the problem directly. The main lesson here is that the system may hold its own reasoning without mimic a human reasoning. What is important for the system is to justify and explain its reasoning in terms of the human reasoning.

2.4. Use of the simulation

Simulation is a usual way to express the dynamic of complex events in a domain. It is a precious tool for justifying, and convincing oneself and others of what has happened, and for predicting what could happen at a given state. This often constitutes the only way to draw a coherent picture from a large and complex set of data in a contextualized representation of a system's behavior. Simulation can also be required by other tasks, such as training or incident analysis. For example, simulation allows replaying the same incident in different contexts to observe the consequences and to identify the best strategy (i.e. the better contextualization of a procedure) for solving the incident.

2.5. Making context explicit

Context must be discussed with respect to its use: the context of the interaction, the context of the task, etc. Accordingly, the context will be static (e.g. the context of use) or dynamic (e.g. the context of an activity). In the SEPT application, context is a way to organize dynamically pieces of knowledge at a given stage of a problem solving. The identification of a context (recognition of a particular circuit breaker of a given control system) leads the inference engine to select only the relevant rules in the unique context of that circuit breaker. By making the reasoning local, the context allows a system to deal with different tasks working on the same domain knowledge because all the working contexts of the tasks are made compatible. Context and explanation generation have strong links as well.

2.6. Lessons learned

First, there is a need to make context explicit in the representation of knowledge and reasoning. Second, it appears that any system should have a set of basic functions such as: A knowledge configuration module for managing the knowledge bases of the system, which will be used in an incremental way; A simulation module for developing the dynamic of a particular reasoning sequence on the domain knowledge (for validation purposes, for checking alternatives, for replaying a reasoning sequence in different contexts, for explanation purposes, etc.); and A communication module for managing interaction among the modules as well as between the modules and the users. Third, the architecture of the system at the "knowledge level" should be based on a multi-agent platform with three basic agents (knowledge configuration agent, simulation

agent and communication agent) and an agent for the task for which the user needs support (e.g. monitoring or diagnosis). Fourth, the same architecture at the implementation level can assume different representations. One reason is that the different agents use domain knowledge either directly or indirectly in a compiled form that is efficient for the task at hand. When the same knowledge is used in different tasks, a problem of coherence may occur. A solution would be to compile dynamically the domain knowledge when needed, but it is still a problem at this level.

3. SART: Lessons learned from the experiment

3.1. Introduction

The project SART (French acronym for Support system for traffic control) aims at the development of an intelligent decision support system capable of helping the operator make a decision intended to solve an incident that occurs on a subway line. When an incident occurs, the operator must make a prediction or take corrective action 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 reestablish a normal situation after the incident (at least partially). We developed a context-based adaptive decision support system for incident management on subway lines. With context playing a crucial role in the decision making process, we have developed a context-based representation of the domain knowledge and of the reasoning based on this knowledge. This formalism is called contextual graphs referring to the central role that context and it dynamics play in the structure.

3.2. Representing the domain knowledge by the onion metaphor

The onion metaphor provides some interesting insights into the knowledge representation as a step in an incident solving takes a meaning in a given context. Contextual knowledge does not intervene directly at this step of the problem solving but constrains the step; pieces of contextual knowledge may be partially ordered; a piece of contextual knowledge itself takes a meaning in a context; and pieces of contextual knowledge relate incidents together. With an association between a number of pieces of contextual knowledge, a relationship emerges between a given incident and others.

3.3. Representing the reasoning by contextual graphs

Contextual graphs are a unified representation of the different manners in which an incident can be solved depending on the context (the practices). They constitute an open structure with incremental knowledge acquisition and learning mechanisms [13]. A contextual graph is associated with a task and makes dynamically available the different known methods to carry out the task. A first concrete result is the integration in one structure of an official procedure for solving an incident, and all the practices established by operators that take into account the context in which an incident must be solved. Second, the natural capabilities of learning, incremental knowledge acquisition and explanation generation offered by the formalism are intrinsic parts of the task at hand. A third result is that the dynamics of the proceduralization has a simple representation: the contextual elements attached to the different branches are proceduralized at a diverging node, stay in this state for different action sequences, and finally, are deproceduralized when the branches are merged. These are few central characteristics of a context-based intelligent assistant system.

Figure 2 give the classical example in the literature of the coffee preparation (e.g. see UML manual). In this figure, vertical bold lines represent parallel grouping expressing the fact that some actions (e.g. "Take coffee" and "Take filter") can be executed either in parallel or in any order. The contextual-graph formalism allows the representation of a number of variants (“To be in hurry”, “Choice of the type of water”, “Moment of the day”, etc.).

This example shows also the limits of the representation of some contextual knowledge at the level of the temporal branching. In some situations, the choice of the ordering of the actions “Take reservoir”, “Take filter”, “Take coffee” depends on a dense net of contextual nodes for finally six possibilities. For instance, the type of coffee machine will bring some constraints (e.g. the reservoir is fixed in the machine and filled with water with the wished number of cups). Thus, it is better to impose only a global constraint on the execution of the three actions before to continue the process.

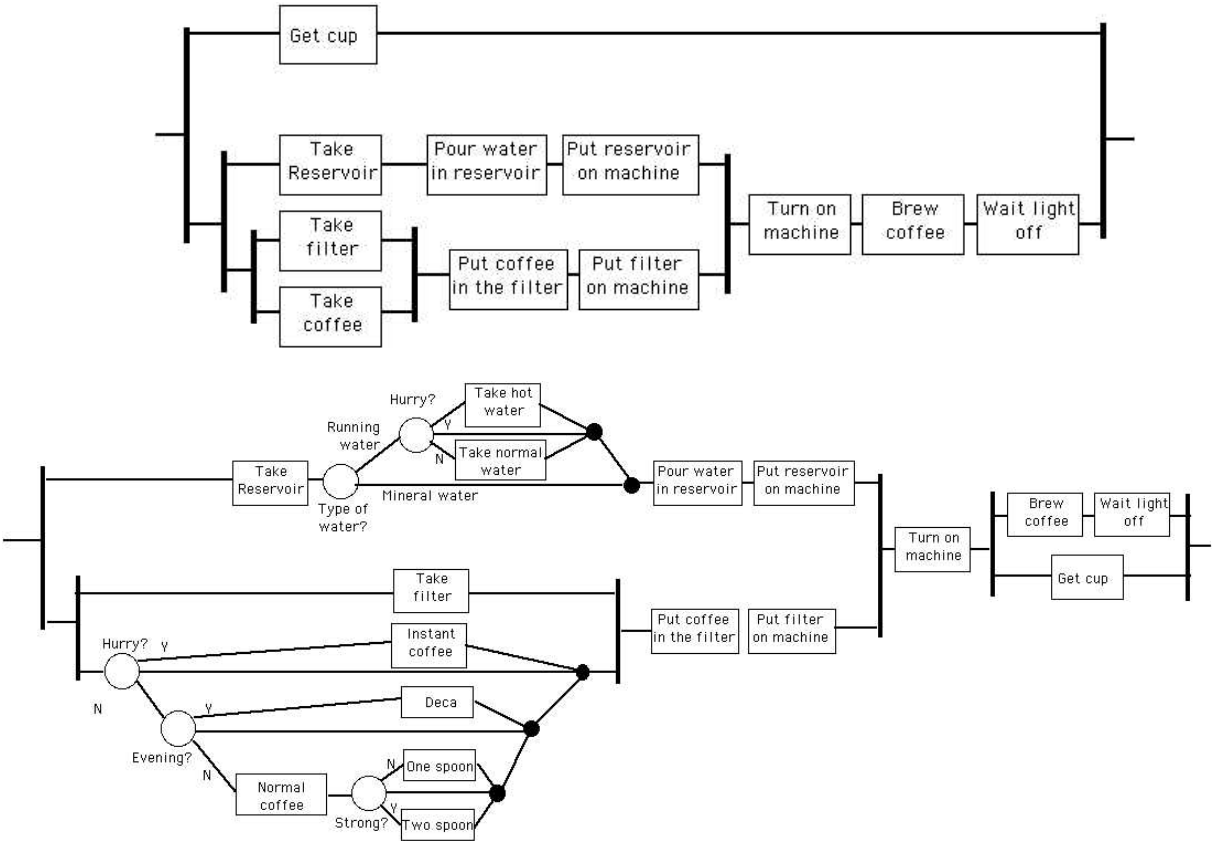

Figure 2. Example of coffee preparation as found in the literature (top) and after integration of some practices (bottom)

4. Context-based intelligent assistant systems

4.1. Introduction

We developed the Context-based Intelligent Assistant Systems (CIAS) framework on the basis of the above two applications and a survey of the literature [6]. An intelligent assistant 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. As the intelligent assistant of an operator, the system must be able to solve automatically minor problems and prepare elements for complex problems that the operator will have to solve. It must also benefit from its interaction with the operator to learn new practices from the operators and acquire new knowledge when it fails to solve a problem, thanks to a context-based formalism as contextual graphs. For example, when the user overrides a decision for various reasons, the assistant may not understand; there is an incompatibility with its internal representations of the problem solving. If that decision still is valid, it is because the problem must be solved in altered circumstances (i.e. another context). Thus, the assistant should know when it is out of its validation domain and ask the user for his new decision in the context at hand. As context plays an important role for such systems, we call them Context-based Intelligent Assistant Systems (CIASs).

4.2. Main characteristics of CIASs

At the knowledge level, a Context-based Intelligent Assistant System has two sides: the real-world process side and the human side. This implies that a CIAS 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 3.

Figure 3 represents the relationships between the CIAS, the operator and the real-world process, and the three types of knowledge that the system needs to cooperate.

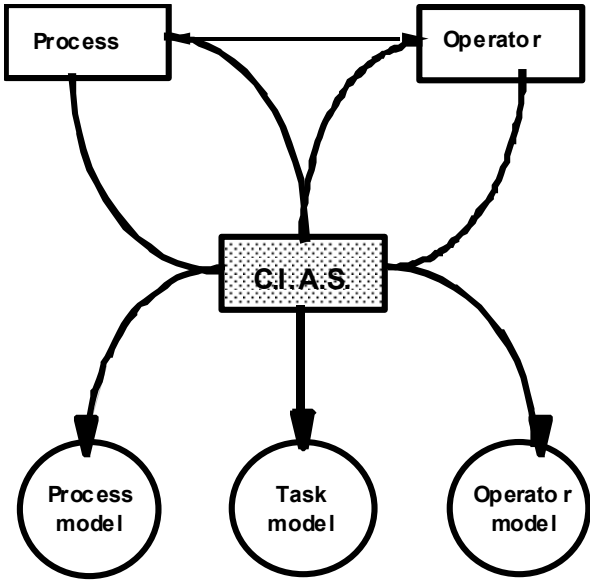

Figure 3. General architecture of a CIAS

4.2.1. The model of the real-world process. The process model is based on knowledge of the real-world process. The model permits the CIAS to compare the process behavior with the simulated behavior from any particular initial state. There are three goals: (1) To observe the evolution of the process; (2) To verify the coherence between varying process behavior and operator actions; and (3) To permit the operator to simulate alternative solutions before making a decision. Models of real-world processes generally exist beforehand because they are important simulation tools in process monitoring, training, etc.

4.2.2. The Task model. The task knowledge allows the CIAS to simulate operator activity. The goals are the identification of the operators' intentions from their action sequences, observation and explanation of the process behavior. The task analysis may be established in two steps. The first step corresponds to an elicitation of knowledge from operators and the use of reports, books and related matter. The second step starts with the model obtained in the first step and concerns the incremental acquisition 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 (e.g. acquisition of a new practice). Note that a main role of a CIAS is to be an observer and stay in an attentive waking state in order to avoid disturbing the operator at a crucial moment. Indeed, a CIAS must solve automatically simple problems and prepare for the operator the needed elements for solving complex problems.

4.2.3. The operator's model. The knowledge in the operator model corresponds to the action sequences of operators facing a problem in the process (called practices above). From operator actions and knowledge in the process and task models, the CIAS may deduce operator's intentions and preferences from their choices during the problem solving. If the CIAS later identifies a similar situation, it may propose to any operator a similar solution in the spirit of the case-based reasoning. As for the process-modeling component, one may proceed in a short elicitation phase and a resultant incremental development of the knowledge base, again, thanks to a context-based formalism as contextual graphs.

4.3. Some properties

The advantage of the approach presented above is to rapidly provide a mockup that will be improved incrementally through interaction with operators. During the incremental knowledge acquisition process, knowledge is acquired when needed and, importantly, its context of use is made explicit. Only a kernel of knowledge has to be elicited directly from operators, mainly to select the right representation formalism for the knowledge. For managing knowledge, the system must be able to accomplish tasks such as acquisition, assimilation, learning, validation-verification, information retrieval, and indexing.

During its interaction with an operator, a CIAS can be in two states, namely, a wake state and a cooperative state. In the wake state, the CIAS must follow the operators' actions on the process, observe the corresponding effect on the process' behavior and compare it to a simulated behavior. The CIAS does not intervene if there is no discrepancy between the observed and simulated behaviors of the process. When a problem is detected, the CIAS first waits to allow time for the operator to react and, only after a while, alerts the operator of the problem. The main reason for the delay is to avoid disturbing the operator in a critical situation. If the problem is not seen by the operator, then the system enters a cooperative state. However, more frequently, the operator triggers the cooperative state.

5. Some basic tasks about a CIAS

During the accomplishment of a task, four interdependent aspects are particularly important for CIASs (taking into account all previous requirements):

- Explanation generation implies that the system and the user must provide and understand explanations to cooperate, and cooperate to co-build explanations.
- Incremental knowledge (and context of use) acquisition supposes that a system cannot have all the needed knowledge at the design time. Here, explanations enable the system to acquire the

knowledge in the right context (e.g. the proceduralized context introduced before).

- Learning from experiences with users during the work process implies that the system is able to acquire, represent and use local reasoning, i.e. a reasoning in its contexts of use. A CIAS "learns by interacting" and thus becomes increasingly familiar with users and their way to accomplish their activity.
- A communication module is needed for managing interaction, on the one hand, among the modules, and, on the other hand, between the modules and the users (as presented as an intelligent interface). Thus, the user has a unique interlocutor that will relieve him/her of a cognitive overload by avoiding him to use several languages to interact with the different modules and by managing complex questions in which several modules must intervene partially.

For representing these interdependent aspects in a CIAS, the architecture of a context-based intelligent assistant system should be based on a multi-agent platform with the three basic agents (a knowledge configuration agent, a simulation agent and a communication agent) and the agent for the task for which the user needs support (e.g. monitoring or diagnosis). Such an architecture is open in the sense that new agents (for new functions such as training, statistics, etc.) can be added at any time or the development of others stopped without putting into question the whole system.

The architecture of a CIAS as a multi-agent system is considered here at the "knowledge level." However, at the implementation level, the system can have a different expression. One reason is that the different agents use the same domain knowledge either directly or indirectly under a compiled form that is efficient for the task accomplish by the agent. In the latter case, the same pieces of knowledge may exist several times in the knowledge base. This could lead to problems of coherence, except if the system is able to compile dynamically the knowledge when needed.

5.1. Incremental knowledge acquisition

The main idea here is that experts provide their knowledge in a specific context and that knowledge can only be relied upon within this context. The knowledge provided by experts is essentially a justification of the expert's judgment in that context. Thus, knowledge must not be generalized when it is acquired, and it is crucial for the system to record the context in which the knowledge is acquired and can be used.

Incremental knowledge acquisition plays an important role in two situations. First, 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. Second, a chunk of knowledge may incorrectly be used by the system because a link to the current context is missing. Here, incremental knowledge acquisition must focus on the refinement of the proceduralized context, and explanation can support that refinement.

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. Henninger [11] noted that "you won't know what is really needed until you're in the design process." Moreover, the knowledge engineer may associate a context with the pair (problem, solution) that may be different from those in expert's mind.

5.2. Learning new practices

Contextual graphs give a unified representation of procedures and practices as discussed in previous sections (and see [7], [13]). They include a natural learning mechanism for integrating new practices by assimilation. This mechanism incorporates a new practice for the incident management in the contextual graph by contextualizing a procedure based on the current context. The system asks the operator to give the information about the incident that must be incorporated in the base of incidents, and its proceduralized context. During this operation, the system looks for the practice that the operator would follow (according to the current knowledge of the system). Then, the operator provides the system with the practice effectively followed, and the system compares it to the practice selected from its base of incidents. If there is a discrepancy, the system asks the operator the missing contextual information that differentiate the know practice from the practice just entered by the operator and, thus, acquires the new practice.

New practices are thus directly acquired from operators thanks to a highly interactive algorithm. This algorithm forces the user to explain the reasons underlying any choice made by the operator. Those explanations later are useful for the system to explain its viewpoint as a user needs its help.

5.3. Explanation generation

The organization in graphs/sub-graphs leads to a reuse of sub-graphs across contextual graphs (e.g. the evacuation of a damaged train in the subway is an activity found in several incident solving) in the spirit of the building blocks introduced several years ago (see [8]). This hierarchical organization allows the system to generate explanations at different levels of details. The explanation on a sub-graph is generally about its context of use, the pre-conditions to fill, the post-conditions and the results that are available.

Several years ago, explanations were generated to address *Why* and *How* questions. However, operators are

not interested, for example in how to evacuate a train, but they may need explanations on existing alternatives and their context of occurrence for accomplishing this activity of evacuation. The relevance of such an explanation is due to the incremental acquisition of new practices with their contexts of use. There is yet another context of interest: Contextualized explanations are the result of a process of interaction between the user and the system, and constitute a medium of communication between the user and the system during the problem solving [2].

Moreover, the incremental acquisition of new practices allows retrieval of the history of the uses of a given practice, its application in different contexts, and/or the use of alternative practices in a given context. An operator can thus replay the solving of a given incident and apply different practices to check the consequences and how the changes of context can affect the quality of the practices and, finally, retain for the next time the strategy that is the most relevant in the particular context.

5.4. The potential of CIASs

Explanation generation is intertwined with simulation for providing the system with anticipation capability (the *look-ahead* component described in [14]). This aspect of a CIAS is particularly important when, as in the SART application, the system is considered as a medium in the interaction between two operators (two experts or an expert and a novice). Simulation allows anticipation even if it is impossible to have a comprehensive a priori planning. For example, Hoc [12] thinks that 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: (1) uses knowledge about future states to decide what action to take at the moment; and (2) has a model of itself and of the relevant part of its environment and is able to use this model to anticipate the future (e.g. see [10]). The system then determines its behavior according to the predictions, i.e. it lets the future state affect its present states. This implies that an intelligent system must be equipped with a simulation component.

Context can also be a tool for managing an informational mass. For various reasons, we are now increasingly concerned with the management of data, information and knowledge in large and heterogeneous bases. The keystone here is clearly a dynamic organization of the data, information and knowledge according to a given focus of attention at a given moment. Making and using context explicitly would help such a dynamic organization of the knowledge in the memory (a kind of dynamic conceptual schema for a database with a context-dependent activation of the links between the tables) to: (1) address a request in a given context, (2) incrementally acquire a new piece of data/information/knowledge (especially useful for representing a procedure and the associated practices), and (3) manage retrieval/storage of items in the bases.

Other methods, such as Focus+Context views, have been developed for introducing deliberate distortion of the representation to show a large amount of contextual information in a given amount of screen. A description of these techniques starts from three premisses. First, the user needs both an overview (context, or global context) and detail information (focus, or local context) simultaneously. Second, information needed in the overview may be different from that needed in detail. Third, these two types of information can be combined within a single (dynamic) display, much as in human vision. Collecting the practices in contextual graphs is like to integrate several methods for problem solving, and choose dynamically according to the current context the right practice. Moreover, there are two mechanisms of expansion and aggregation (as in conceptual graphs) that allow focusing on part of a practice.

6. Conclusion

For a long time, we have talked about "intelligent systems" without seeing real-world applications of them. On the basis of two real-world applications, the SEPT and SART applications, we show that the design and development a Context-based Intelligent Assistant System (CIAS) is possible if one follows some recommendations. First, CIASs aim to join together advantages of previous knowledge-based systems and decision support systems. Second, as shown in their names, the goal of CIASs is not to replace the user but to be an efficient assistant. Third, CIASs rely heavily on an effective use of the notion of context in the representation of their knowledge and the reasoning held on this knowledge. The contextual graphs are such an example (see a more developed presentation in [7]). Fourth, CIASs must be able to accomplish some basic tasks as explanation generation, incremental knowledge acquisition, and learning regardless of the task at hand. For this, a CIAS must have three modules (agents) for (1) knowledge configuration, (2) simulation and (3) explanation generation. Fifth, the architecture of CIASs is open, that is, the development of an agent can be stopped or an agent added without putting in question the whole architecture.

7. References

[1] Bau, D.Y. and Brézillon, P., "Model-based diagnosis of power station control systems: the SEPT experiment". *IEEE Expert*, 1992, pp. 36-44.

[2] Brézillon, P., "Context needs in cooperative building of explanations". First European Conference on Cognitive Science in Industry. Luxembourg, 1994, pp. 443-450.
[3] Brézillon, P., "Context in problem solving: A survey", *The Knowledge Engineering Review*, 1999, 14(1), 1-34.
[4] Brézillon, P., "L'application SEPT: Bilan d'une expérience".. Rapport de Recherche du LIP6, Université Paris 6, France < http://www.lip6.fr/reports/ lip6.2002.008.html >, 2002.
[5] Brézillon, P. and Pomerol, J.-Ch., "User acceptance of interactive systems: Lessons from Knowledge-Based and Decision Support Systems". International Journal on Failures & Lessons Learned in Information Technology Management, June, 1997, 1(1): 67-75.
[6] Brézillon, P., Cavalcanti, M., Naveiro, R. and Pomerol, J.-Ch., « SART: An intelligent assistant for subway control ». Pesquisa Operacional, Brazilian Operations Research Society, 2000, 20(2): 247-268.
[7] Brézillon, P., Pasquier, L. and Pomerol, J.-Ch., "Reasoning with contextual graphs". European Journal of Operational Research, 2002, 136(2): 290-298.
[8] Chandrasekaran, B., Johnson, T.R., and Smith, J.W., "Task-structure analysis for knowledge modeling". *Communications of the ACM*, 1992, 35(9): 124-137.
[9] Clancey, W.J., "The conceptual nature of knowledge situations, and activity". In: P.J. Feltovich, K.M.Ford & R.R. Hoffman (Eds.) Expertise in Context, AAAI Press / The MIT Press, 1997, pp. 247-291.
[10] Ekdahl, B., Astor, E. and Davidsson, P., "Toward anticipatory agents". In: Intelligent Agents, M. Wooldridge & Jennings N. (Eds.), Lecture Notes in AI, 890, Springer Verlag, Berlin, 1995, pp. 191-202.
[11] Henninger, S., "The knowledge acquisition trap". Proceedings of the IEEE Workshop on Applying Artificial Intelligence to Software Problems: Assessing Promises and Pitfalls (CAIA-92), 1992.
[12] Hoc, J.-M., *Supervision et Contrôle de Processus. La cognition en Situation Dynamique*. Grenoble: P.U.G, 1996.
[13] Pasquier, L., Brézillon, P. and Pomerol, J.-Ch., "Learning and explanation in a context-sensitive adaptive support system". In: C. Faucher, L. Jain & N. Ichalkaranje (Eds.) Innovative Knowledge Engineering. Springer-Verlag, 2002 (to appear).
[14] Pomerol, J.-Ch., "Artificial Intelligence and Human Decision Making". *European Journal of Operational Research*, 1997, 99, pp. 3-25.
[15] Pomerol, J.-Ch. and Brézillon, P., "About some relationships between knowledge and context. Modeling and Using Context (CONTEXT-01)". In: P. Bouquet, L. Serafini, P. Brézillon, M. Benerecetti, F. Castellani (Eds.) Lecture Notes in Computer Science, Springer Verlag, N° 1688, 2001, pp. 461-464. (Full paper at http://www-poleia.lip6.fr/~brezil/Pages2/Publications/ CXT01/index.html).
[16] Woods, D.D., Roth, E.M. & Benett, K, "Explorations in joint human-machine cognitive systems". In S. Robertson, W. Zachary & J.B. Black Eds. Cognition, Computing and Cooperation. Ablex, 1990, pp. 123-158.