International Journal Failure and Lessons Learned in Information Technology Management, 1997, 1(2), Special Issue on Successes and Pitfalls of Knowledge-Based Systems in Real-World Applications, pp. 89-98.

Lessons learned on successes and failures of KBSs

Brézillon Patrick and Pomerol Jean-Charles

LAFORIA-IBP, Case 169, University Paris 6
4, place Jussieu, 75252 Paris Cedex 05, France
Tel: +33 1 44 27 47 21 - Fax: +33 1 44 27 70 00
{brezil, pomerol}@laforia.ibp.fr

Abstract

Many papers describe the first steps of the implementation of Knowledge-Based Systems (KBSs). Some papers report the successful use of KBSs in organizations. For example, there is a special issue of the Artificial Intelligence Journal (Volume 59, 1993) that collects reports on KBSs in use during ten years or more. On the other hand, papers reporting failures or nonuse of KBSs are rare. The International Conference on "Successes and Failures of Knowledge-Based Systems in Real-World Applications" (Bangkok, Thailand) is the first attempt to collect and analyze such experiences [BKK96, 1996]. This paper provides a summary of the results of the conference, it also relies on our own experience in KBS development in organization, and on the literature. Among the main results, it appears that the success of a knowledge-based system depends on dimensions often neglected as the types of knowledge involved in the systems, the organizational framework and the architecture of the system. This permits to discover what the new challenges for KBS development are.

1. INTRODUCTION

A large number of industrial Knowledge-Based Systems (KBSs) has been developed since implementation of first expert systems. However, we do not know how many KBSs are really operational within companies or administrations. Durkin (1993) lists about 2500 systems, but their exact operational status is often not precisely stated. Majchrzak and Gasser (1991) point out that more than 50% of the systems, which are installed in companies, are not used. The main evoked reasons are that KBSs solve 80% of problems when users mainly need a support for the other 20%, and that integration of KBSs in an organization implies interaction of KBSs with classical software and changes in the organization itself. A special issue of the AI Journal (Volume 59, 1993) presents the most successful systems. Mizoguchi and Motoda (1995) say that there are from 20 to 30 operating KBSs in the top five Japanese companies. Thus, some clues about system use exist.

Studies on the use of KBSs in real-world applications permit to point out four main failures:

(1) The user cannot intervene during the problem solving: KBSs are supposed to be oracles and users as novices. However, unexpected situations are rather the norm than the exception. KBSs cannot solve such situations when users do it.

(2) KBSs do not always use correctly their knowledge because knowledge, being acquired from human experts, has a high contextual component that is generally not acquired with the knowledge.

(3) KBSs cannot initially have all the necessary knowledge. Thus, any KBS has limited resources for problem solving and limited influence. One can never anticipate all the misunderstandings and problems.

International Journal Failure and Lessons Learned in Information Technology Management, 1997, 1(2), Special Issue on Successes and Pitfalls of Knowledge-Based Systems in Real-World Applications, pp. 89-98.

(4) KBSs cannot generate relevant explanations for users: KBSs do not know what the user's problem-solving context is. Thus, KBS and the user must jointly construct the explanation.

Other systems--successful and unsuccessful ones--represent considerable experience that may provide many useful lessons for the community. Successes and failures occur for a variety of reasons that, once shared, may permit to save time, work and money. The International Conference on Successes and Failures of Knowledge-Based Systems in Real-World Applications [BKK96, 1996] was an attempt to gather and share all this knowledge. Its goal was to focus on the experience acquired during design, development and use of KBSs.

This paper presents a survey of the literature on KBSs and Decision Support Systems and of our own previous works [Brézillon, 1995; Brézillon and Pomerol, 1996, 1997a, b and c; Pomerol and Brézillon, 1996]. This paper focuses on four topics on which discussions focused, namely domain knowledge (section 2), users (section 3), working organization (section 4) and architecture of KBSs (section 5). We conclude in section 6 by presenting the main orientations that must be considered for the development of successful systems.

2. THE DOMAIN KNOWLEDGE

2.1 Knowledge acquisition

Knowledge acquisition still remains one of the main problems. Knowledge generally is acquired according to any expertise recording method before any use of KBSs. The knowledge is tested only when the system is able to reason. If a problem with the knowledge is then detected, the developer must modify the knowledge, re-load (re-compile or re-interpret) and then test the reasoning again. The different reasons generally evoked are: one does not know in advance the type of knowledge that can be acquired, experts do not make explicit many necessary steps that they think of as being common sense, experts generally consider that it is tedious and time-consuming to write out all their experience and they have no time to do this. Moreover, new requirements emerge during development because they cannot be identified until parts of the system have been designed and implemented [Fischer, 1996]. For instance, Tian and Huang (1996) discuss the problem of knowledge elicitation from textbooks and the need to complement it with a knowledge acquisition from experts.

The main point here is that knowledge acquisition must be integrated in the task at hand and be thought in an operational context of use. Thus, knowledge should be acquired incrementally along the problem solving. This leads to the notion of cooperative problem solving by the user and the system. Incremental knowledge acquisition plays an important role in two situations:

(1) Some knowledge is missing in the system: The user can add new knowledge through explanation. Here, explanations permit to make explicit the context of the knowledge use.
(2) The system does not use correctly its knowledge: A link with the current context is missing. This link could be added under the form of a refinement of the current context.

For giving the capability to the user to incrementally add knowledge, the system must be able to acquire the knowledge in a structured and general manner. For this reason, it is essential that the knowledge in the system be structured and tractable. It appears that the old paradigm of separating of knowledge representation from knowledge use leads to the

development of flat knowledge bases that became rapidly intractable due to the lack of structures in them.

2.2 Knowledge and its context

A second problem--related to knowledge acquisition--is the lack of consideration for the contextual dimension of knowledge. A human expert provides knowledge for a problem solving in a precise context and often points out that "In this situation, the solution of this problem is ...". Such contextual information, which delimits the area of appliance of the acquired knowledge, is missing. This leads to tentatively apply knowledge out of its right context of use. The main reason is that the knowledge engineer's objective at this step is different from the expert's ones. The knowledge engineer tries to generalize the acquired knowledge for a class of similar problems, whereas experts give solutions for specific cases in specific context. One possible solution for the system is to incrementally acquire knowledge when needed. Thus, the KBS knows the validity context of the acquired knowledge given by the step of the problem solving at which the knowledge is acquired and will use it after in a correct way.

The need to make context explicit is encountered in different domains. Brézillon (1996) presents a survey of the literature on context in several domains. There is not yet a consensus on a clear definition of context. One main reason is that there are different types of context according to the level of the knowledge representation, to the level of the reasoning mechanism applying to the knowledge, and to the interaction between the system and the user. For example, knowledge transfer must be accompanied by a transfer of contextual knowledge to face all the implicit needs in dialogue, and extend the possibilities of the common language between agents. The cooperation context contains items like the dialogue memory, the task at hand, the spatio-temporal situation (place, moment, elements, etc.), and the psycho-social situation (user model, roles and social status, etc.). Even if there is no consensus on the definition of context, it nevertheless appears that one cannot talk about context out of its use in applications.

2.3 Knowledge and explanation

The role of explanations is to persuade the other agent that a partial solution is correct, not just to provide a final justification of an already determined completed solution. Explanation aims at modifying the knowledge of the other agent (bringing new pieces of knowledge or revising existing ones), clarifying the context of the shared knowledge, conveying new information pieces, reach a common agreement, coordinating the agents' activities, forcing each agent to trace its own reasoning process and revealing alternative ways of thinking about a problem. Thus, explanations become an intrinsic part of the problem solving and may modify the reasoning line of the system [Brézillon, 1996].

Explanations contain contextual information missing in the other's context [Brézillon 94]. When two agents disagree at a step of the problem solving, each one may offer explanation to change the other's interpretation of the problem-solving step, so that an agreement may be achieved. This aims at adjusting both agents' contexts to reach compatible interpretations by an incremental acquisition of the missing knowledge. Spontaneous explanations enrich the context that is shared by both agents, leading to a mutual understanding and a mutual acceptance. Other advantages of spontaneous explanations are: making understanding easier, insuring that the interpretation of the agents' solutions was exactly the one they wanted to communicate, and decreasing the number of dialogue exchanges needed to achieve the interaction goal.

Up to now, explanation generation was only studied **from** the system **to** the user. The system having not all the needed knowledge, it was often unable to provide the user with

convincing arguments. With incremental knowledge acquisition, explanation generation is now also considered **from** the user **to** the system. However, the explanation generation by the user for the system implies that the system incrementally acquires knowledge, assimilates it in its knowledge base, and propagates the consequences of the new knowledge. Capturing and using knowledge in the context of use greatly simplifies knowledge acquisition and explanation.

2.4 The nature of the knowledge

Another problem concerns the nature of the knowledge itself. In real-world applications, knowledge engineers must face the domain complexity and the task at hand. This often implies to simultaneously tackle different types of knowledge (e.g., often data are inaccurate, incorrect or redundant) and thus different types of reasoning. This prevents to exploit a compiled version of these different types of reasoning. Indeed, first KBSs relied on fixed knowledge. The knowledge was initially acquired, coded in machine and finally used. The knowledge engineer's role ended here, after it was the user's role to deal with the knowledge base to adjust minor details.

However, knowledge is not something that is definitively fixed. Knowledge evolves, new pieces appearing or changing, others disappearing. Without maintenance and updating, a KBS becomes rapidly obsolete and is abandoned by users that were not qualified or did not wanted waste their time with this additional task of maintenance and updating. Fischer (1996) addresses this problem and proposes a knowledge management by the KBS to correct its reasoning according to change in the knowledge coming from the external world. Again, the solution is here an incremental knowledge acquisition.

2.5 Large knowledge bases

A problem that one also encountered is the size of knowledge bases in real-world applications. Often expert systems are endowed with too small knowledge bases. The reason is that the role of the knowledge engineer is to initiate the developing of a KBS on the basis of a very narrow problem. Again, this implies that the role of the end-users in the knowledge-base management is largely underestimated. This concerns mainly the problems of maintenance, updating, validation and verification of large knowledge bases on real-size applications. The problem is reinforced by the fact that such large knowledge bases are not structured in a way adapted to the expert system.

3. THE USERS

3.1 Confusion about end-user's abilities

Expert systems were built on the assumptions that the user has (say, in diagnosis): (i) a minimal knowledge about the device, (ii) a minimal knowledge about how to find faults, and (iii) a minimal information about the symptoms of the device being tested. The initial assumption was that KBSs were oracles and user only had to accept the solution proposed by systems [Karsenty and Brézillon, 1995].

However, end-users are not novices. As experts, they also have an experience, although this experience has a more practical nature than that of experts. Users' experience is not taken into account by knowledge engineers and thus not used by KBSs. As a consequence, end-users perceive KBSs as blackboxes with incomprehensible behaviors,

asking non-correlated questions, etc. Moreover, end-users prefer to consult a colleague rather than using either manuals or other types of help. For instance, when you are working under UNIX, you can use the command 'manual' to know the functions of a command that you are looking for. However, this supposes that you know beforehand the name of that command. Generally, one prefers to explain to a colleague what the problem is, and after try the various suggestions of the colleague. With an incremental knowledge acquisition, the KBS may acquire the users' knowledge with that of experts.

End-users do not constitute a homogeneous population. While the less experienced users could sometimes benefit from a straight suggestion of how to proceed, the more experienced ones wanted something more and something else. For example, they want the opportunity to try their own ideas, to "freeze" some parameter of the solution or simply be able to collect some basic information and subsequently take the whole decision themselves [Forslund, 1995].

3.2 Social and cultural effects

Specificity of end-users goes beyond the simple expert vs. novice consideration. Some user's specificity comes from the task itself. For example, Zhuge and Pan (1996) present a KBS for art pattern design. With artists as end-users, clearly it is essential that a KBS takes into account the reasoning and the creativity of such users. In this situation, the KBS must behave rather as an intelligent assistant system.

Schmidt (1995) gives an example of aspects that are common to situated cognition with automatic dispensers. His conclusion is that the interface is often a barrier to communication between end-users and machines as automatic dispensers. Such machines impose a forced interview, a prior determination of all the interaction, and an amount of instructions that is the same for all the end-users. However, the more instructions are displayed in an explicit manner by the machine, the less know-how is required from the user. In such an approach, there is no consideration for the user context and its evolution during the interaction with the system.

Other users' specificity comes from the social and cultural environments. Software developers rapidly faced this problem when trying to sell their products across the world. For example, a color has different meanings from one country to another. This is the same thing for the training of pilots of Airbus that implied to revisit the design of the interface of the simulator [Boy, 1991; Hoc, 1996]. This means that users' training and interaction with users must be tailored to these social and cultural constraints.

3.3 One or several users with one or several KBSs

Another serious limit was to consider that a KBS was developed for a standard user. Indeed, several different humans may interact with a given KBS.

Several experts may work together to achieve a task. Most real world problems require the integration of specialists in different domains. Each of these specialists contributes to a unique point of view. One aspect of expertise is the ability to integrate specialist knowledge in some real problem context [Woods, 1995]. This implies that the KBS must be able to switch between different types of reasoning and handle knowledge in different contexts.

The use of a KBS by different specialists must be taken into account explicitly when designing the KBS. Vanwelkenhuysen and Mizoguchi (1995) describe work practice of two troubleshooters--testers and engineers--performing on the same devices (digital

processor boards in a telecommunications production plant), with access to the same information but adapted to different workplaces. They find that the two troubleshooters never came to an agreement because they solved their problems differently (e.g., oscilloscope versus logic state analyzer), each way being effective for routine problems in their workplace but inadequate for the other's. The main difference arises from a different viewpoint on priority on performance requirements (e.g., fidelity and precision versus efficiency). Here, the failure to explicitly represent the two different contexts is responsible of the misuse of the KBS.

Grusenmeyer (1995) underlines that functional communication is an essential mean for operators to cooperate, coordinate and adjust their knowledge and representations. This is particularly crucial at the changing of operators: the coming operator must adjust his understanding and operational representation of the current situation with those of the leaving operator. Leaving and coming operators need to share knowledge and build common references. Functional communication increases dramatically when the change occurs during a malfunctioning phase of the process that they have to monitor successively. The verbal exchanges during this phase are not reported on working documents because it is expensive and strongly reduces the possibilities of these exchanges. The main difficulty to design and develop a KBS in this situation is to take into account the context of the malfunctioning phase and the contexts of the two operators to facilitate the information transfer between these context.

A second situation occurs when several clones of the KBS are used by different users to accomplish the same task. This situation is encountered with a given class of users, e.g., in the use of a database from several workstations. Each user has his own working context, and the KBS must adapt its behavior to each specific context. This cannot be planned beforehand, and the KBS must have the means to tailor automatically its behavior to each user. As a consequence, the KBS would be able to identify the user's profile, his preferences, etc.

3.4 The user-with-a-KBS

From their survey of the literature, Pomerol and Brezillon (1996) point out several reasons of KBS rejection by users, e.g., a change in the working practices of the users (fear of an additional overload in their work) and a change in the relationships between the users and the other employees and their chiefs. For Inman and Burrell (1996), the factors that limit the acceptance and the value of the system are: (1) changes in the end-users with different problems which were not anticipated at the beginning of the project; (2) new end-users are more skillfull than first chosen end-users; (3) there are problems in testing the KBS on the real-world process (e.g., trigger fault on the national power net because faults are rare on the net). Frasson and Aïmeur (1996) describe similar problems in the framework of intelligent tutorial systems installed in companies and underline that industrial training suffers from several flaws: training sessions are very rare, quality of training is questionable, update of courses are long and expensive, and training does not exempt users to accomplish their normal work. Indeed, a general observation in computer science is that learning is partly due to an oral tradition well-developed.

Concerning more specifically the user, after the introduction of the system, the user's work change and he must share with the KBS what was beforehand his unique competence. This sharing may lead the user to have a negative advice on the KBS: the fear to face an intractable system, a danger to be replaced by the KBS (and thus lose his job), a danger to be spied by the KBS that may be controlled by his chief, etc.

All these problems can be avoided if the introduction of the KBS in the organization is foreseen and prepared. For instance, users' reactions may be positive if a prototype is provided as soon as possible. Thus, future users have the feeling to participate in the KBS development and are happy to see their suggestions taken into account. Indeed, introducing the user in the development loop raises some significant new questions that emerge from the trials and sometimes new possibilities have to be considered. For instance, the design of a KBS, as quoted by Frasson and Aïmeur (1996), can be problem-oriented (the main interest for users) rather than solution-oriented (the main interest for knowledge engineers).

3.5 Lessons learned

The lessons learned about users are the following. First, the user must participate at the design and development of the KBS. The user' suggestions concern mainly the use of the KBS. Taking them into account also would permit to acquire more easily the user's experience that often is complementary of the expertise.

However, this is the ideal situation, and often it is not possible to include the user in the design loop. The promising solution is then to equip the KBS with functions that permit it to tailor automatically its behavior to any particular user with no trouble for designers, developers and knowledge engineers. With such functions KBSs would become really intelligent assistant systems.

4. KBSs WITHIN THE WORKING ORGANIZATION

4.1 Introduction of a KBS in an organization

The introduction of a KBS in an organization needs to be prepared to account for its industrial component as soon as possible. Duribreux and Houriez (1996) underlines that KBS development within an industrial environment has to face various perturbations coming not only from the problem complexity, but also from the context in which the project evolves. Generally, KBSs are developed in Laboratories or in Research & Development Department of large companies without consideration for the future end-users in the company (generally in another department of the organization). One reason evoked is that people working in the R&D Department come from target departments and thus think to perfectly know users' needs. (With the human integration, there is also a software integration that is discussed in the next section.)

Taking into account users in an organization is not only a problem of interface. This also concerns the KBS development itself. One aspect of the problem concerns the change of the user of a KBS. The change may correspond to either a move of the user for another position into (or out of) the organization or a consequence of a change in the objectives of the managers. The change of user during the design and the development of a KBS has consequences on the KBS development and can even lead to the abandon of the KBS.

A solution would be to design a KBS according to **roles** that persons play in an organization, rather than the persons themselves. Roles in an organization are well defined and stable when persons move often from one role to another.

4.2 KBS objectives and organization objectives

Many reported failures are due to KBS use for something they were not intended for. Indeed, there is often a number of conditions, or secondary goals, that are not clearly pronounced. For example, the execution of the plan should not be too expensive, too time-

consuming or too complex. It would also be preferable if equipment already available can be used, as long as the result is satisfactory [Forslund, 1995].

The interest of KBSs in an organization relies heavily on the type of business of the organization. Some domains as diagnosis, monitoring, finance, insurance are a better target for KBSs than, say, places where the task can be entirely automated. Whatever the domain is, however the success of a KBS implementation relies on the fact that a project must meet a real business need and incorporate valuable knowledge. A reason is to avoid a change in the manager's objectives concerning the project. Ware (1996) gives some keys for a successful KBS integration in an organization accounting for business constraints. For instance, a project must support the overall business objectives, not only marginal needs. However, early projects must not be mission critical either for the client organization or for a manager's personal agenda. Pomerol and Brezillon (1996) showed that problems with KBSs have already been encountered with decision support systems, and that problems of users with must be replaced within the organizational context of the companies.

4.3 The actors in the KBS development

The knowledge and the expert are crucial human actors in the success of a KBS. However, there are other human actors that also play a crucial role as the project leader (decision maker or manager). The project leader's importance all along the project is often underestimated (and sometimes ignored). The project leader may be the same person during all the project realization but his objectives may change (e.g., management disillusionment with the technology and changing information technology priorities within the organization as quoted by Ware, 1996). The project leader may decide a change of target user or a change in the objective of the KBS development. This situation is described by Pomerol and Brezillon (1996) that have encountered often an alteration of the initial users' goal and target in different companies. This is also described by Inman and Burrell (1996) who report the change of end-users during the KBS development due to problems of response time of the system. Such aspects are well-known and described in earlier literature (e.g., see [Eason et al., 1995]).

At another level, the success of a project--especially in Artificial Intelligence--depends on the relationships between competing teams within an organization. The introduction of a KBS in an organization often implies a change of power when the KBS development concerns different teams. The best solution, underlined by Ware (1996), is to let all the responsibility and the leadership concerning the KBS at one team.

4.4 The user-with-a-KBS and his colleagues

Among the changes when introducing a KBS in an organization, there is a change in the relationships between the user-with-a-KBS and his colleagues. The status of the user-with-a-KBS may become isolate, being considered by his colleagues endowed with special privileges, and the other employees may be jealous of the user-with-a-KBS and his new power in the organization (especially when confidentiality of the data intervenes). This problem is similar to the problem of competition between teams.

More generally, when a KBS is devoted to a special role (and thus a given user), that user may possess a kind of power in the organization. This situation is also experimented by his chief that may be dependent on the user-with-a-KBS for having some information because only the user knows how to interact with the KBS, not the chief.

4.5 Lessons learned

Organizational problems can be solved by preparing the KBS implementation in the organization and the possible changes in the employees' work. Another positive argument to explore is that there is a neglected dimension of KBSs beyond its support to a user or a team, namely, its role of 'corporate memory'. Developing KBS along such a dimension may overcome the problems of users' movement and be related more closely to the notion of role discussed above. The problems between persons cannot be solved at a technical level except if all the employees receive a KBS to support them in their work.

5. THE ARCHITECTURE OF KBSs

5.1 The phases in KBS evolution

If the cognitive aspects in the design, development and use of a KBS are important, the core of the problems is the KBS development itself. For example, Vale and col. (1996) underline that the effort in developing a KBS is much greater that in traditional computer applications, at least because experts are very busy. In this section, we discuss successively the problems related to the design, the tools used for the development, the integration with other software, the maintenance and updating and the validation of the KBSs and put forward some elements of solution.

The design and development phases of a KBS are very similar to the design and development of any software (except the knowledge-acquisition phase). The weakness here concerns the lack of consideration for the 'industrial' character of a project that would constrain the design and development of a KBS. Due to the rapid prototyping approach (and, eventually, the change during the project of the knowledge engineer), analysis and design stages are often inadequate, system documentation is sparse, project planning is rudimentary and scheduling rarely attempts and then often ineffective. Ware (1996) and Fischer (1996) discuss this point in real-world applications. This considerable amount of essential information, which is lost, implies substantial costs because this information must be reconstructed by the persons who maintain and update the system. Moreover, new requirements emerge during development because they cannot be identified until parts of the system have been designed and implemented or because there is a change in the objectives.

The type of lifecycle chosen (e.g., waterfall model or spiral model) affects seriously the way in which a KBS may be developed and its final quality. For Seta et col. (1996), the main shortcut comes from the lack of the conceptual level consideration. It is thus imperative to lead a knowledge-level analysis and ontological commitment in a systematic manner. Generally, ad hoc solutions are preferred. For instance, Inman and Burrell describe the practical experience of developing a KBS for diagnosing faults in an electricity supply system. The use of the hybrid "surface" and "deep" models as a reasoning strategy, and a blackboard for communication between modules, has been proved to be an effective mean of solving problems where real-time constraints have been imposed.

The transfer of the knowledge into the machine is a crucial problem. Often the means are chosen on the basis of the tools that are available in the company, not of the domain complexity. One generally chooses a shell to reduce duration of the projects [Vale et al., 1996]. There is a first danger: the abandon of the product by the software developer. This may put organizations developing KBS with that product in a difficult situation and, indeed, encourage them to develop their own products instead of re-write the KBS with the new version of the product. Frasson and Aimeur (1996) reach the same conclusion and

developed tools adapted to their problems when tools on the market were not relevant. It is crucial to be sure that the chosen tool fully meets the needs of the application to be developed (real-time constraints, integration with other software, independence with respect of the platforms, importance of a wide portability across hardware platforms, etc.).

Early evaluations must be undertaken so that lessons can be learned for full scale implementation. The main problem encountered by knowledge engineer in some real-world applications is the KBS validation. It is sometimes difficult--and even impossible--to conduct real-size experiments facing a high reliability in a real-world process, as the diagnosis of equipment in power or nuclear plants [Inman and Burrell, 1996; Pomerol and Brezillon, 1996]. Conversely, there are some domains--as fraud risks in credit card company (Killin and Curet, 1997)--where the number of instances of the problem is very high and the KBS cannot be validated in an exhaustive way. This again supposes that only the end-users can lead the validation, which then appears as a subjective exercise, on exceptional events when the KBS is developed to react to such events.

The last point to consider is related to gains coming from the use of a KBS. If it is relatively easy to evaluate the interest of a KBS at the user's level (optimization of the task solving in terms of time and quality of the work), it is difficult to evaluate at the level of the organization this interest in terms of money. Everybody knows the example of R1/XCON at Digital Equipment Corporation [McDermott, 1993]. Reasons for the success of the system are: R1 was continuously maintained and updated, the knowledge base has been structured explicitly, and they developed the methodology RIME that supports developers. However, it is a rare case.

Another reason is that it is difficult to lead parallel experiments with and without the KBS. For instance, validation of a KBS (a very similar problem to the benefit evaluation) is generally performed off-line and often on already solved problems. The main success of KBSs in such situations relies only on the maintainability and reusability of their modules.

The era of stand alone system is now over for all software pieces, especially nowadays at the multimedia era, and KBS is only a component in a final hybrid system including classical software. The main problem is a semantic mismatch between them. This semantic mismatch arises because each component is associated with a particular ontology that does not always fit with the ontologies of the other components. Ji and Jin (1996) propose to develop a meta-ontology containing the knowledge of coordination management as an upper level structure of integration, above ontologies associated with each piece of software.

5.2 Human interventions and KBSs

The knowledge engineer ensures the design and development of a KBS according to given specifications. We have already discussed that often these specifications are not provided by end-users, but the conversely KBS is imposed to end-users. The knowledge engineer ensures the training of end-users. After, the users become responsible of the "good health" of the KBS. This means that the end-user is supposed to be able to maintain and update the KBS. If end-users are knowledgeable with the knowledge in the KBS, they generally are not knowledgeable with the way in which the KBS was built.

Moreover, the knowledge engineer intervenes to modify and adapt the knowledge according to the chosen tools and to what s/he thinks is good for end-users. Lekova and Batanov (1996), and Myint and Tabucanon (1996) underline that the first basic point for knowledge engineers should be the nature of the problem domain of interest to define appropriate general knowledge representation scheme and knowledge base structure.

For completing or modifying knowledge, users intervene superficially by adding, say, new rules to control the firing of existing rules. As a consequence, the KBS rapidly becomes intractable. Stathis and Sergot (1996) analyze the construction of two knowledge-based front-ends using logic programming tools and techniques. One of the main problem for one of the two systems was maintenance. Deleting one user's answer implies deleting all the following answers in the answer tree even if part of them stays valid. The authors present a successful approach based on the game metaphor, interactions made by participants being interpreted as moves made by the players of a game. This meets the general recommendation to include the user in the design loop to overcome this problem.

6. CONCLUSION

The idea behind many expert systems was to build a system that can arrive alone at the final solution. This implies to define the problem very strictly, simplifying it if necessary to ensure that the computer really will be able to find a correct solution [Forslund, 1995]. However, the knowledge and reasoning of a KBS cannot be a simple translation of those of the human experts. Beyond the problem of knowledge-engineer's interpretation of the expert's knowledge, some deep problems raise. One of them is that structures are generally imposed on knowledge bases and new concepts are introduced by the knowledge engineer. Such a macro-reasoning is useful for an incremental development of knowledge bases but does not correspond to human expertise [Hatchuel and Weil, 1992]. Another problem is the alive nature of knowledge that implies that a KBS must be able to incrementally acquire knowledge during problem solving. The incremental knowledge acquisition permits a KBS to acquire the knowledge and its context of use. For Fischer (1996), the "put-all-the-knowledge-in-at-the-beginning" approach failed because KBSs must be considered as an evolutionary process. KBSs also do not exist in an isolated context but should be embedded within dynamic human organizations. Thus, they cannot be completely designed before their use and must evolve in the hand of the users.

This is one of the main distinction between automatic system and interactive system [Pomerol and Brezillon, 1996]. Effective decision support system requires that computational technology aids the user in the process of reaching decision, and not simply make or recommend solutions [Woods, 1995]. The relationship between users and interactive systems should be one of controller and assistant and would not threaten the autonomy of the user. This implies that a KBS must:

- be able to learn (e.g., learning-by-doing) and incrementally acquire new knowledge to improve its support to users;
- be equipped with a number of functions as symbolic learning and incremental knowledge acquisition. For example, a KBS must be able to store its own experiences with users to be able to improve its support;
- be comprehensible to users as any computer system. This implies in turn that the KBS must be a transparent box, not a black box.

Several paths are now explored as intelligent interfaces, intelligent assistant systems, active environment, etc. Such paths try to bring a solution--at least a partial one--to the problems discussed in the paper as the organizational ones. This is the new challenge for the future.

7. REFERENCES

BKK96 (1996) Proceedings of the International Conference on Successes and Failures of Knowledge-Based Systems in Real-World Applications, Asian Institute of Technology ed., PO Box 4, Klongluang 12120, Pathumthani, Bangkok, Thailand.

Boy G. (1991) Intelligent Assistant Systems, Academic Press, Series Knowledge-Based Systems, Vol. 6.

Brézillon P. (1994) "Context needs in cooperative building of explanations", Proceedings of the First European Conference on Cognitive Science in Industry, Luxembourg, September 28-30, pp. 443-450.

Brézillon P. (1995) Intelligent cooperative systems: The generation after expert systems, International Conference on Retructuring Strategies for Information Industry, Bangkok, Thailand, April 26-28, 1995.

Brézillon P. (1996) Context in Human-Machine problem Solving: A Survey, Technical Report 96/29, LAFORIA, October, 37 pages.

Brézillon P. (1997) Successes and failures of KBSs in real-world applications: Report of the international conference, International Journal on Knowledge-Based Systems (submitted).

Brézillon P., **Pomerol J.-Ch.** (1996) Misuse and nonuse of knowledge-based systems: The past experiences revisited, In Implementing Systems for Supporting Management Decisions, Humphreys P., Bannon L., McCosh A., Migliarese P., Pomerol J.-Ch. (Eds.), Chapman and Hall, ISBN 0-412-75540-8, pp. 44-60.

Brézillon P. et J.-Ch. Pomerol (1997a) User acceptance of interactive systems: Lessons from Knowledge-Based and Decision Support Systems", Int. Journal of Failures & Lessons Learned in Information Technology Management, J. Liebowitz Editor-in-Chief (to appear).

Brézillon P. and Pomerol J.-Ch. (1997b) Organizational experiences with multicriteria decision support systems: Problems and issues, Proc. of the 30th Hawaï International Conference on System Sciences, Hawaï, USA, Vol. 3.

Brézillon P. and Pomerol J.-Ch. (1997c) Contextual issues in the framework of multicriteria decision making, Proceedings of the First International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-97), Rio de Janeiro, Brasil, February 4-6 (to appear)

Duribreux M. and Houriez B. (1996) Problems and solutions encountered during an industrial expert system development, In: [BKK96], pp. 30-38.

Durkin J. (1993) Expert Systems. Catalog of Applications, Intelligent Computer Systems Inc., PO Box 4117, Akron, Ohio 44321-117, USA.

Eason K.D., Harker S.D.P., Raven R.F., Brailsford and Cross A.D. (1995) Expert or assistant: Supporting power engineers in the management of electricity distribution, AI & Society, 1995, 9, pp. 91-104.

Fischer G. (1996) Domain-oriented design environments: Knowledge-based systems for the real world, In: [BKK96], pp. 59-68.

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

Frasson C. and Aïmeur E. (1996) SAFARI: a university-industry cooperative project, In: [BKK96], pp. 225-234.

Grusenmeyer C. (1995) Les dialogues coopératifs en phase de relève de poste: Rôle dans la sureté des systèmes de production, Proceedings of the XXX° Congrès de la SELF "L'homme dans les nouvelles organisations", Biarritz, France.

Hatchuel A. and Weil B. (1992) L'Expert et le Système, *Economica*, Paris, France.

Hoc J.M. (1996) Supervision et Contrôle de Processus. La Cognition en Situation Dynamique, Presses Universitaires de Grenoble, Série Science et Technologies de la Connaissance.

Inman D. and Burrell P. (1996) An electricity supply fault analysis expert system: Problems and achievements, In: [BKK96], pp. 208-214.

Ji G. and Jin W (1996) Integration based on knowledge soup, [BKK96], pp. 98-107.

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

Killin J. and Curet O. (1997) A comparative evaluation of four commercial knowledge-based systems, Special Issue on Successes and Pitfalls of KBSs in Real-World Applications of the International Journal on Failures & Lessons Learned in Information Technology Management, this volume.

Lekova A.K. and Batanov D. (1996) Self-learning fuzzy-expert system for technological processes control, [BKK96], pp. 117-123.

McDermott J. (1993) R1 ("XCON") at age 12: lessons from an elementary school achiever, Artificial Intelligence Journal, 59, pp. 241-247.

Majchrzak and Gasser L. (1991) On using Artificial Intelligence to integrate the design of organizational and process change in US manufacturing, *AI and Society Journal*, **5**, 321-338.

Mizoguchi R. and Motoda H. (1995) Expert systems research in Japan, *IEEE Expert Magazine*, pp. 14-23.

Myint S. and Tabucanon M.T. (1996) The framework for the expert system to generate alternative products in concurrent engineering design, In: [BKK96], pp. 124-133.

Pomerol J.-Ch. and Brézillon P (1996) Are interactivity problems in KBSs similar to DSSs ones?, [BKK96], pp. 1-10.

Schmidt C.T. (1995) Information processing, context creation, setting minds in public arenas: investigative techniques for client/automaton dialogue design, Proceedings of the IJCAI-95 Workshop on Modelling Context in Knowledge Representation and Reasoning, Technical Report 95/11, LAFORIA, University Paris 6, France, pp. 121-131.

Seta K., Ikeda M., Kakusho O. and Mizoguchi R. (1996) Design of a conceptual level programming environment based on task ontology, In: [BKK96], pp. 11-20.

Stathis K. and Sergot M. (1996) A fast glimpse of knowledge-based front-ends, [BKK96], pp. 81-90.

Tian S. and Huang H. (1996) Knowledge elicitation from textbooks and experts, In: [BKK96], pp. 159-162.

Vale Z.A., Fernandes M.F., Rosado C., Marques A., Ramos C. & Faria L. (1996) Better KBS for rel-time applications in power system control centers: What can be learned by experienced?, [BKK96], pp. 215-224.

Vanwelkenhuysen J. & Mizoguchi R. (1995) Adaptation of reusable knowledge for workplace integration, Proceedings of the IJCAI-95 Workshop on Modelling Context in Knowledge Representation and Reasoning, Technical Repport 95/11, LAFORIA, University Paris 6, France, March, pp. 167-177.

Ware B. (1996) Knowledge-based systems experience in the Victorian Electricity Commission's production group, In: [BKK96], pp. 47-58.

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

Zhuge Y. and Pan Y. (1996) An art pattern layout knowledge-based system, In: [BKK96], pp. 163-168.