ELSEVIER

Knowledge-Based Systems 10 (1998) 253–257

Knowledge-Based SYSTEMS

Letter

Successes and failures of KBSs in real-world applications: Report on the International Conference

Patrick Brézillon

LIP6, Case 169, University Paris 6, 4. place Jussieu, 75252 Paris Cedex 05, France

Received 20 December 1996; revised 12 June 1997; accepted 28 July 1997

Abstract

The International Conference on Successes and Failures of Knowledge-Based Systems (KBSs) in Real-World Applications was held at the Asian Institute of Technology (Bangkok, Thailand) on 28–30 October 1996. This report provides a presentation of the papers and the discussions during the conference. The conference focused on some new issues that have rarely been addressed before. 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 work, the organizational framework and the architecture of the system. This permits us to discover what the new challenges for KBSs are. © 1998 Elsevier Science B.V.

Keywords: Knowledge-based systems; Architecture; Cooperation; Context

1. Introduction

A large number of industrial Knowledge-Based Systems (KBSs) has been developed since the implementation of the first expert systems. However, we do not know how many KBSs are really operational within companies or administrations. Durkin [1] lists about 2500 systems, but their exact operational status is often not precisely stated. Majchrzak and Gasser [2] point out that more than 50% of the systems, which are installed in companies, are not used. The main 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 AI Journal (Volume 59, 1993) presents the most successful systems. Mizoguchi and Motoda [3] say that there are from 20 to 30 operating KBSs in the top five Japanese companies. Thus, some clues about system use exist.

However, 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 lead to savings in time, work and money. The conference was an attempt to gather and share this knowledge. Its goal was to focus on the experience acquired during design, development and use of KBSs.

The Proceedings[1] consist of 30 papers that have been selected among 80 submitted papers. The papers are organized in five chapters, namely Development Environments for KBSs, Tools for building KBSs, KBS Techniques, Implementation of KBSs, KBSs in Applications. Forty persons attended the conference, and 30 students of the Institute followed the sessions, all of them participating actively in the discussions. Most of the papers describe experience in domains as different as power systems, law, knowledge-based front-ends, intelligent tutorial systems, design, computer equipment, software specification, electronic dictionary, blast furnace, finance and art.

The report is organized around the four points on which discussions focused, namely domain knowledge (Section 2), users (Section 3), working organization (Section 4) and architecture for KBSs (Section 5). We conclude in Section 6 by presenting the main orientations that must be considered for the development of successful systems. Papers referred to in the text which do not have corresponding entries in the References are included in the conference Proceedings.

[1] The Proceedings are published by the Asian Institute of Technology and can be obtained from Dr. D. Batanov, AIT, CSIM, PO Box 4, Klongluang 12120, Pathumthani, Bangkok, Thailand, or by e-mail to batanov@cs.ait.ac.th.

0950-7051/98/$15.00 © 1998 Elsevier Science B.V. All rights reserved
PII S0950-7051(97)00032-4

2. The domain knowledge

Knowledge acquisition still remains the main problem. The different reasons generally given are: one does not know in advance the type of knowledge that must be acquired, experts do not make explicit many necessary steps that they think of as common sense, experts generally consider it tedious and time-consuming to write out all their experience and they have no time to do this. The same applies to knowledge elicitation from texts. For instance, Tian and Huang (Northern Jiaotong University, China) discuss the problem of knowledge elicitation from textbooks and the need to complement this by acquisition from experts.

A second problem—related to knowledge acquisition—is a lack of consideration for the contextual dimension of the knowledge. A human expert provides knowledge for 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. The main reason is that the knowledge engineer's objective at this step is different from that of the expert. The knowledge engineer tries to generalize the acquired knowledge for a class of similar problems, when experts give solutions for specific cases. The proposed solution is to acquire knowledge incrementally when needed. Thus, the KBS knows the context of validity of the acquired knowledge and will use it in a correct way.

A third problem concerns the nature of the knowledge itself. In real-world applications, knowledge engineers must face the domain complexity and the tasks at hand. This often implies tackling simultaneously different types of knowledge (e.g. often data are inaccurate, incorrect or redundant) and thus different types of reasoning, and not exploiting a compiled version of these different types of reasoning. Indeed, first KBSs were bases of fixed knowledge. The knowledge was initially acquired, coded into a machine and finally used. However, knowledge is not something that is definitively fixed. Knowledge evolves, new pieces appearing or changing, others disappearing. This is generally the problem of end-users, not of knowledge engineers. Without maintenance and updating, a KBS rapidly becomes obsolete. Fischer (University of Colorado, USA) addresses this problem and proposes knowledge management by the KBS to correct its reasoning according to change in the knowledge coming from the external world. Again, the solution is the incremental acquisition of knowledge.

A fourth problem is the size of knowledge bases in real-world applications. Often knowledge engineers use expert system shells that are developed on small knowledge bases. The reason is that the role of the knowledge engineer is to initiate the development of a KBS on the basis of a narrow problem. Again, this implies that the role of end-users in the knowledge-base management is largely underestimated. This mainly concerns the problems of maintenance, updating, validation and verification of large knowledge bases on large-size applications.

3. The users

Initially, an expert system was built on the assumptions that the user has (say, in diagnosis): (i) minimal knowledge about the device, (ii) minimal knowledge about how to find faults, and (iii) minimal information about the symptoms of the device being tested. The initial assumption was that the KBS was an oracle and the user only had to accept the solution proposed by the system. However, end-users are not novices. They have more practical experience than experts that is not taken into account by knowledge engineers. As a consequence, users preferred to consult other people 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. However, this supposes that you know beforehand the name of the command that you are looking for. Generally, one prefers to explain to a colleague what the problem is, and then to try the various suggestions of the colleague. With incremental knowledge acquisition, the KBS may acquire the user's knowledge together with that of the expert.

It is generally agreed that each user has his own specificities that go beyond a simple expert vs. novice consideration. For instance, users do not have a unique cultural and social origin. Software developers face this problem when trying to sell their products across the world. For example, the meaning of a color varies from one country to another. User's specificities also come from their work. For example, Zhuge and Pan (Zhejiang University, China) present a KBS for art pattern design. With artists as end-users, clearly the KBS must not constrain the reasoning and creativity of such users. In this situation, the KBS must behave rather as an intelligent assistant system.

Several humans may interact with a given KBS. This occurs when either several clones of the KBS exist or there is a sequential use of a given KBS. The former situation is encountered either with a unique class of users (e.g., use of a database from several workstations) or different classes of users (e.g., engineers and technicians working with a given KBS). Each user and class having their own working context, the KBS would adapt its behavior to each specific context. The latter situation appears, say, in the 24 h monitoring of a process. The KBS would ensure the continuity of the role and the transfer of information from one operator to the next.

From their survey of the literature, Pomerol and Brezillon (University Paris 6, France) point out several reasons of KBS rejection by users: a change in the working practices of the users, a change in the relationships between the users and other employees and their bosses, etc. Frasson and

Aïmeur (University of Montreal, Quebec, Canada) describe similar problems in the framework of ITSs installed in companies and underline that industrial training suffers from several flaws: training sessions are very rare, quality of training is questionable, course updating is long and expensive, and training implies that users do not accomplish their normal work.

All these problems can be avoided if the introduction of the KBS in the organization is foreseen. For instance, users' reactions may be positive if a prototype is provided as soon as possible. They then have the feeling that they are participating in the development of the KBS and are happy to see their suggestions taken into account. Indeed, introducing the user into 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, can be problem-oriented (the main interest for users) rather than solution-oriented (the main interest for knowledge engineers).

4. The working 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 (University of Valenciennes, France) 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 the research and development departments of large companies without consideration for the future end-users in the company (generally in another department). One reason given for this is that people working in the R&D department come from target departments and thus suppose to know perfectly users' needs. A correct introduction of a KBS in an organization supposes a twofold integration of the KBS, namely a software integration and a human integration. We discuss human aspects in this section, the software aspects being considered in the next section.

The value of KBSs in an organization relies heavily on the type of task for which the KBS is developed. Some tasks such as diagnosis, monitoring, finance, and insurance are a better target for KBSs than, say, tasks that can be entirely automated. The success of a KBS implementation relies on the project meeting a real business need and incorporating valuable knowledge. A reason for implementing a KBS is to avoid a change in the manager's objectives concerning the project. Ware (Monash University, Australia) outlines some key factors for 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 show that problems with KBSs have already been encountered with DSSs, and that problems with users must be replaced in the organizational context of the companies.

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 a 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. A change of user during the design and development of a KBS has consequences on the KBS development and can even lead to it being abandoned. A solution would be to design a KBS according to the role that persons play in an organization, rather than the persons themselves because roles in an organization are well defined and stable when persons often move from one role to another.

There are other actors that play a crucial role, namely, the expert, the knowledge engineer and the project leader (decision maker or manager). The importance of the expert and knowledge engineer for the success of the KBS development is well-known. The importance of the leader throughout the project is often underestimated. 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). The project leader may decide on a change of target user or a change in the objective of KBS development. This situation is described by Pomerol and Brezillon, who have often encountered an alteration in the initial users' goal and target in different companies. This is also described by Inman and Burrell (South Bank University, UK) where a change of end-users during KBS development is due to problems of response time of the system. Such aspects are well-known and are also described in the literature (e.g., see [4]).

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 into an organization often implies a change in the sharing of power when the KBS development concerns different teams. The best solution, underlined by Ware, is to allow the team which is to have the leadership to take responsibility for the KBS.

Another problem is the changes in the relationships between the employee-with-a-KBS and his colleagues. First, the employee's work changes and he must share with the KBS what was previously his unique competence. This sharing may lead the employee to have a negative attitude towards the KBS, owing to fear of facing an intractable system, of being replaced by the KBS (thus losing his job), and of being spied on by the KBS, which may be controlled by his chief, etc. Second, the status of the employee-with-a-KBS among other employees may imply changes in their relationships: the employee may become

isolated, being considered endowed with special privileges, and the other employees may be jealous of the employee-with-a-KBS and his new power in the organization (especially when confidentiality of the data intervenes). This problem is very close to the problem of competition within a team.

Such problems must be solved by preparing the KBS implementation in the organization and the possible changes in 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 such a dimension of KBSs may overcome the problems of users' movement and be related more closely to the notion of role discussed above.

5. Architecture for KBSs

If the cognitive aspects in the design, development and use of a KBS are important, the kernel of the problems lies in the KBS development itself. Vale and colleagues (University of Porto, Portugal) underline that the effort required to develop a KBS is much greater than in traditional computer applications, at least because experts are very busy. In this section we discuss problems related to the design, tools used for the development, integration with other software, maintenance and updating and validation of the KBSs and present some elements of solution.

The design phase of a KBS is very similar to that of any software. The 'industrial' character of a project constrains the design and development of a KBS. A problem frequently mentioned is the lack of specification and implementation reports due to the rapid prototyping approach, and, eventually, the change during the project of the knowledge engineer. For Fischer, substantial costs arise because a considerable amount of essential information (such as design rationale) is lost during development and must be reconstructed by the persons who maintain and update the system. New requirements emerge during development because they cannot be identified until portions of the system have been designed and implemented.

Indeed, this is a general observation in computer science where learning is partly due to a well-developed oral tradition. One also encounters the type of life cycle chosen (e.g., waterfall model or spiral model) that has an impact on the way in which a KBS may be developed and its final quality. For Seta et al. (Osaka University, Japan), the main shortcut comes from the lack of the conceptual level execution. 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 proved an effective means of solving problems where real-time constraints have been imposed.

The transfer of 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 shells to reduce duration of the projects (Vale et al.). The first danger is the abandonment of the product by the software maker. This may place organizations developing KBS with that product in a difficult situation and indeed encourage them to develop their own products instead of re-writing the KBS with the new version of the product.

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 (University of Budapest, Bulgaria), and Myint and Tabucanon (Asian Institute of Technology, Thailand) note 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. Frasson and Aïmeur 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.).

The main problem encountered by the knowledge engineer in some real-world applications is the KBS validation. It is sometimes difficult—and even impossible—to conduct real-size costs facing a high reliability in a real-world process, as the diagnosis of equipment in power or nuclear plants (Inman and Burrel, Pomerol and Brezillon). Conversely, there are some domains—such as fraud risk in a credit card company—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 at such events.

The knowledge engineer ensures the design and development of a KBS according to given specifications. We have already mentioned that often these specifications are not provided by end-users, but that conversely the KBS is imposed on end-users. The knowledge engineer ensures the training of end-users. Thereafter, the users become responsible for the 'good health' of the KBS. This means that the end-user is supposed to be able to maintain and update the KBS. End-users may be familiar with the knowledge in the KBS, but generally are not so with the way in which the KBS was built.

To complete or modify 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 (Imperial College, UK)

analyze the construction of two knowledge-based front-ends using logic programming tools and techniques. One of the main problems for one of the systems was maintenance. Deleting one user's answer implies deleting all the following answers in the answer tree even if some remain valid. The authors present a successful approach based on the game metaphore, 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 bypass this problem.

The era of the stand-alone system is now over for all software pieces, especially nowadays in the multimedia era, and KBS is only one 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 (Zhejian University, China) propose the development of a meta-ontology containing the knowledge of coordination management as an upper level structure of integration, above ontologies associated with pieces of software.

The last point to consider is the benefits gained from use of a KBS. Although it is relatively easy to evaluate the value of a KBS at user level (optimization of task solving in terms of time and quality of the work), it is difficult to evaluate it at the level of the organization, and in monetary terms. The example of R1/XCON at Digital Equipment Corporation [5] is well known. However, this is a rare case. The main reason for this problem is that it is difficult to lead parallel experiments with and without a KBS. For instance, validation of a KBS (a very similar problem to that of benefit evaluation) generally leads off-line and often on already solved problems. Thus, the main success of KBSs relies on the maintainability and reusability of their modules.

6. Conclusion

The knowledge and reasoning of a KBS are not a simple translation of those of the human experts. Beyond the problem of the knowledge engineer's translation of an expert's knowledge, there are serious problems. One problem is that structures generally are 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 [6]. Another problem is the 'alive' nature of knowledge that implies that a KBS must be able to acquire knowledge incrementally during problem solving. Incremental knowledge acquisition permits a KBS to acquire knowledge and the context of its use. For Fischer, the 'put-all-the-knowledge-in-at-the-beginning' approach failed because KBSs must be considered as an evolutionary process. KBSs do not exist in a technological context alone but are 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 distinctions between automatic and interactive systems (Pomerol and Brezillon). The relationship between user and interactive system should be one of controller and assistant, respectively, and should not threaten the autonomy of the user that has to make the final decision. This implies that a KBS must:

- rely on some concepts of biological systems as self-organization and self-adaptation, without trying to mimic biological systems but transforming biological concepts into computer concepts;
- be able to learn (e.g., learning-by-doing) and incrementally acquire new knowledge in order to improve its support to users;
- be equipped with a number of functionalities 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 avenues of research, such as intelligent interfaces, intelligent assistant systems, active environment, etc. have not been explored. This is the new challenge for the future.

References

[1] J. Durkin, Expert Systems, Catalog of Applications, Intelligent Computer Systems Inc., PO Box 4117, Akron, OH 44321-117, USA, 1993.
[2] A. Majchrzak, L. Gasser, AI and Society Journal 5 (1991) 321–338.
[3] R. Mizoguchi and H. Motoda, IEEE Expert Magazine, (1995) 14–23.
[4] K.D. Eason, S.D.P. Harker, R.F. Raven, J. Brailsford, A.D. Cross, AI and Society 9 (1995) 91–104.
[5] J. McDermott, Artificial Intelligence Journal 59 (1993) 241–247.
[6] A. Hatchuel and B. Weil, L'Expert et le Systeème. Economica, Paris, France. 1992.