In: J. W. Schmidt and C. Thanos (eds.): Foundations of Knowledge Base Management:
Contributions from Logic, Databases, and Artificial Intelligence. Berlin: Springer, pp. 391-195.

KBMS Requirements of Knowledge-Based Systems

Matthias Jarke[1], Bernd Neumann[2], Yannis Vassiliou[3], and Wolfgang Wahlster[4]

ABSTRACT *This overview paper provides a customer perspective of the requirements for knowledge base management systems. The customer is taken to be the developer of knowledge-based application systems, such as rule-based expert systems, natural language interfaces, vision systems, and design support environments. We conclude that there are area-specific knowledge base management functions that, if provided by a KBMS, could substantially simplify the development and maintenance of knowledge-based applications. However, the range of requirements appears too wide to permit the development of a completely generalized KBMS, in the same sense as existing generalized DBMS.*

1. A Customer View of KBMS

One way to introduce a Knowledge Base Management System (KBMS) is to impose its existence and use on the users: "Here is a general-purpose system - a mixture of the best ideas from artificial intelligence and databases - that will solve most problems inherent in important application domains." This deliberately exaggerated view has been followed by many of us who are attempting to derive KBMS concepts deductively by extending existing database or knowledge-directed technologies.

This paper and the subsequent ones in this part of the book take a different view. We ask: "Who are the potential customers of KBMSs and what are the services they would like to buy?" Following [MYLO86], we assume that the direct customers for KBMSs will be the developers of knowledge-based applications, rather than the end users of those applications.

Generally speaking, we expect developers of knowledge-based applications to need the following:

- *knowledge representation* languages that express the structure of *the* given application effectively (i.e., formally, concisely, naturally, etc.);
- *knowledge organization* tools that allow for the safe and efficient handling of large amounts of complex knowledge structures;
- *methodologies and environments* through which the customer can create, maintain and query knowledge bases efficiently and effectively.

Additionally, end users often need to have multiple knowledge-based applications and other applications interact with each other efficiently. This leads to a fourth general customer need:

- support for the re-use of *existing* hardware and software facilities.

In general, knowledge bases can be characterized by the degree to which they offer the above facilities. For example, Brachman and Levesque [BL86] define (the content of) a knowledge base as the set of non-ruled-out possibilities. If the knowledge base is empty, everything is possible. An operation called TELL informs the knowledge base about certain

[1] Universität Passau, FRG
[2] Universität Hamburg, FRG
[3] Foundation of Research and Technology in Hellas, Greece
[4] Universität des Saarlandes, FRG

restrictions on the set of possibilities. The operation ASK allows the user to query the knowledge base about its knowledge. It is shown that databases are special knowledge bases in which only very simple TELL operations are possible. In the relational database model, for example, only function and variable free n-ary predicates of predefined structure can be asserted. Because of this restricted form of the TELL operation. the ASK operation of DBMS can be efficiently implemented by direct retrieval.

As more complex TELL operations (e.g., including rules, incomplete information, complex objects, etc.) are permitted, efficiency (and, with further expressiveness, even decidability) is reduced. The only known way out of this dilemma is *not* to start with an empty knowledge base but to embed as much domain-specific knowledge as possible into a KBMS, in order to reduce the search space and to control combinatorial explosion. In the past, most AI researchers have therefore built their own tailored knowledge base management facilities. The only products available on the market, commercial databases and, more recently, expert systems shells, typically do not satisfy their KBMS needs. These general tools appear insufficient for serious knowledge-based applications; however, rebuilding KBMS for each application severely restricts the economic feasibility of AI. Therefore, the question arises:

> *Are there classes of (knowledge-based) applications large enough to make the construction of special KBMS economically feasible, yet focused enough to provide effective and efficient support for the developer?*

This overview chapter, based on four position papers delivered at the Xania workshop, attempts an affirmative answer to this question by sketching knowledge base management requirements for four important classes of AI applications:

- rule-based expert systems,
- natural language understanding and generation systems,
- vision systems, and
- design environments.

We conclude that the KBMS requirements of such applications are rather diverse. With present-day technology, we do not expect a generalized KBMS to be able to handle all of these applications, nor will a simple tool-kit approach suffice. Instead, we advocate a "revolutionary" approach [MYLO86] that develops specialized representation schemes for classes of knowledge-based applications like those sketched above. Two more detailed examples of such systems are presented in subsequent chapters by Neumann (geometrical scene descriptions) and by Borgida et al. (database software development environments).

2. KBMS Requirements of Rule-based Expert Systems

Expert Systems (ESs) are computer programs tackling problems where, even when no straightforward analytic methods exist, humans are able to achieve results.

The first expert systems were developed more than a decade ago for a number of well-bounded stand-alone applications: the assistance of organic chemists (e.g., DENDRAL), similar systems for diagnosis of pulmonary diseases, internal medicine, and infections (e.g., MYCIN, INTERNIST/CADUCEUS, PUFF). Since then, ESs were applied to several application domains including engineering, mathematics, and geology. More recently, attempts to solve business problems with Expert Systems in areas such as insurance and banking [PAU86, REIT84b] have caused great interest. Several ES surveys exist in the literature [HWL83, NAU83, SZOL86].

Abstracting from the development of ESs, three common functional characteristics can be identified - characteristics that have become de facie guidelines for building ESs. These are:

a. *Knowledge base. A* representation of heuristic and factual information, often in the form of facts, assertions and deduction rules.
b. *Inference engine. A* mechanism, playing the role of an interpreter, that applies the knowledge as represented in a suitable way to achieve results.

c. *Man-machine interface. A* mechanism that transfers queries from and answers to the user, sometimes seeking additional information for the Inference Engine. This includes explanation facilities for the user.

While the latter two components are of paramount importance for the efficient operation and usability of an ES, the Knowledge Base is generally considered the "heart and soul" of the system, and, consequently, the one that has attracted most research interest.

It is often mistakenly assumed that rules in the form of

IF <situation> THEN <action>,

are the basis for representing knowledge in all ESs. Even though many other ways to represent knowledge exist (e.g., frames), rules have indeed been the dominant representation, especially in commercial applications Logic-based systems can also be considered rule-based, where rules and the factual part of the knowledge cannot be distinguished. Our focus then in this section is on rule-based ESs.

The commercialization of rule-based ESs has been the source of more research interest, but new requirements are also surfacing. Informally commercialization of ESs implies the ability to develop them in a MIS department, and to have end-user departments as well as the MIS department use them in their daily work.

The first requirement for ESs that was identified is that they can be built and used by MIS personnel. ES development tools and appropriate MMI designs tackle this requirement ("ES shells").

The second major requirement, the importance of which was only recently recognized in industry and concerns this chapter, is the need to tie the ES with other MIS applications [VCJ83]. For example, the XCON system has currently about 4000 rules but works on an underlying database of 9000 VAX component descriptions with 20-125 attributes each [FM86]; commercial databases, used, e.g., in connection with ESs for planning applications, are of course much larger but often used with smaller rule bases.

Several examples of large organizations attempting to extend the scope of ESs from small specialized applications to mainstream applications have been reported. The backbone of such extensions is the ability to tie the ES with the organizations' large operational databases. As an indicative example, we mention the effort at a major charge card company to create a verification system with the help of an ES, that requires frequent reviews of the customers' credit history.

Trying to meet the above requirement, and parallel to research efforts, the industry is following three separate paths. The first path is a custom-based approach to couple specific ESs and commercial database systems (e.g., Inference Corp.). The second path is a general-purpose mechanism, within the ES software development system, that translates ES commands to database queries (e.g., Kee Connection tying directly into an SQL database). Finally, the third path is the integration and storage of ES rules with the database itself (e.g., Postgres by Relational Technology, Inc.). There is little indication that, in current commercial efforts, the management of very large sets of rules plays an important part; but this may come with increasing sophistication of the rule bases, for example, when very large specifications have to be managed.

Even though the philosophy, methodology, and time horizon for their effective use differ for these three approaches, the objective and direction are common: shifting the emphasis in an ES towards the type of system that we call a KBMS, while meeting the requirement to tie an ES with other applications. We can summarize the KBMS requirements of rule-based ESs as follows:

- *Knowledge representation:* Similar to databases, knowledge representation in rule-based ESs is relatively simple, consisting of a representation of underlying facts and of a representation of deduction rules. Often, these rules correspond to Horn clause form such that generalized reasoning mechanisms can be rather efficient.
- *Knowledge organization:* If fact bases become large or external fact bases must be accessed, DBMS mechanisms must be made available to the ES. Typically, this includes some metaknowledge about the database schemata, the integrity constraints, and the available interfaces through which facts can be stored and retrieved. Sophisticated deductive query optimization tools are required to

keep response times acceptable. These can be combined with other strategies that provide control to rule application.

- *Environments:* Basically, these are the tools available in the respective expert system shells. In one methodology (tight coupling), database access will be hidden from the user as far as possible; in the other (loose coupling), the user will load required external data explicitly before starting the consulting dialog [JV84].
- *Coupling:* Besides accessing the external databases mentioned above, expert systems also often need a connection to other information systems tools, including numerical computation [KOWA86] or graphics, as, e.g., in VM-Prolog by IBM France.

The chapters in part II of this book present some proposals that address the questions of integrated fact and rule management required to support KBMS for rule-based systems. However, the following sections will demonstrate that other knowledge-based applications may require more substantial extensions to DBMS.

3. KBMS Requirements of Natural Language Access Systems

There are several ways of viewing the relationships between natural language interfaces (NLI) and KBMS. First, one can try to build NLI KBMS. Second, one may need KBMS to implement NLI. Finally, a can build NLI (supported by KBMS) to knowledge-based systems (supported by KBMS), cf. Fig. 1. In this section, we concentrate on the sect* and third problems.

A piece of software is called a NLI if

(a) a subset of the input or the output of the system is coded in a natural language, and
(b) the processing of the input (the generation of the output) is based knowledge about syntactic, semantic, and/or pragmatic aspects a natural language.

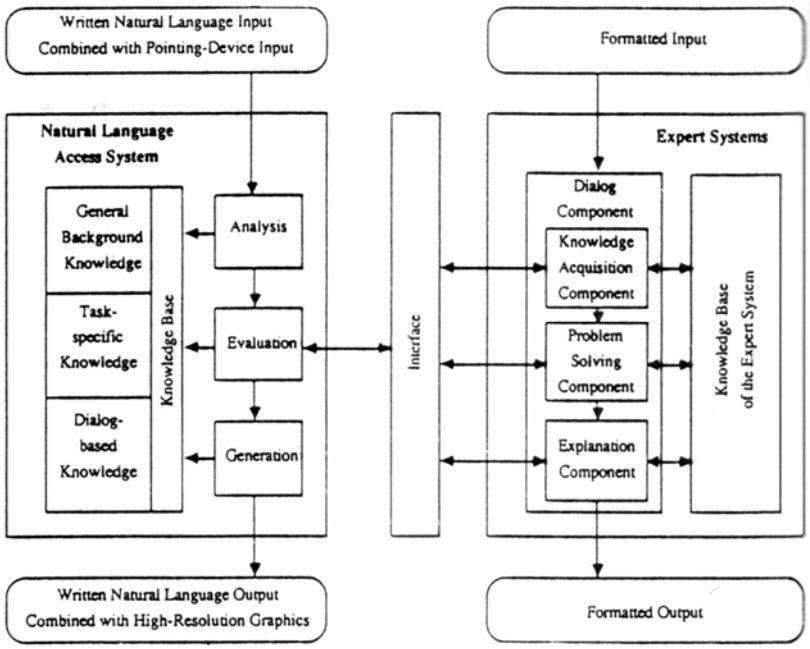

Fig. 1.

This definition implies on the one hand that NL systems are knowledge-based applications which could benefit from a KBMS. On the other hand, they are by definition not expert systems because every human knows at least part of one natural language without necessarily being a language expert.

The general problem of getting computers to understand natural language is far from being solved. Current NLI are a long way from broad-based, universal language capabilities comparable to human dialog partners. Most commercial NLI provide only database access but are insufficient for access to more complex knowledge bases; for a survey of such systems, see [WAHL86].

Though many database features (e.g., concurrency and security) appear less critical in NLI than in traditional database applications due to the single-user workstation approach usually taken by such systems, the availability of KBMS could significantly extend the practical scope for NLI. The most important issue is the integration of multiple diverse knowledge sources for which it seems infeasible to find a common knowledge representation. For example, KBMS for NLI have to integrate the following kinds of knowledge:

- *Linguistic knowledge* concerns a wide variety of aspects, ranging from the handling of ellipsis to the understanding of speech acts for cooperative question-answering [WAHL84, BJ86b].
- *Conceptual knowledge* implies the existence of a model of the slice of reality relevant to the dialog. One of the reasons why NLIs for database access have been relatively successful is that the conceptual knowledge domain underlying a database is rather clear and well-delimited.
- *Inferential knowledge* refers to a rule base similar to that of rule-based expert systems which works on the factual and conceptual knowledge to derive implicit facts or rules from stored ones.
- A *user model* describes the system's knowledge about stereotypes of the different kinds of users that could utilize the NLI, including aspects of background knowledge, experience, dialog history, or goals.

As evidenced by experimental systems such as HAM-ANS [HMM86], the knowledge base structure of NLI can become extremely complex. Figure 2 summarizes some of the important knowledge sources required for NLI systems. In terms of the aspects introduced in the introduction, the KBMS requirements can be summarized as follows:

- *Knowledge representation:* The required knowledge representation should be a hybrid of many individual representations, such as frames, database facts, and deduction rules. Often, knowledge units expressed in one of these languages must be aggregated to the hybrid knowledge representation of one knowledge source (such as linguistic, conceptual, etc.) which in turn must be further aggregated to a globally consistent knowledge base. It is important to have variety in expression while maintaining a high degree of consistency via the hybrid representation mechanism.

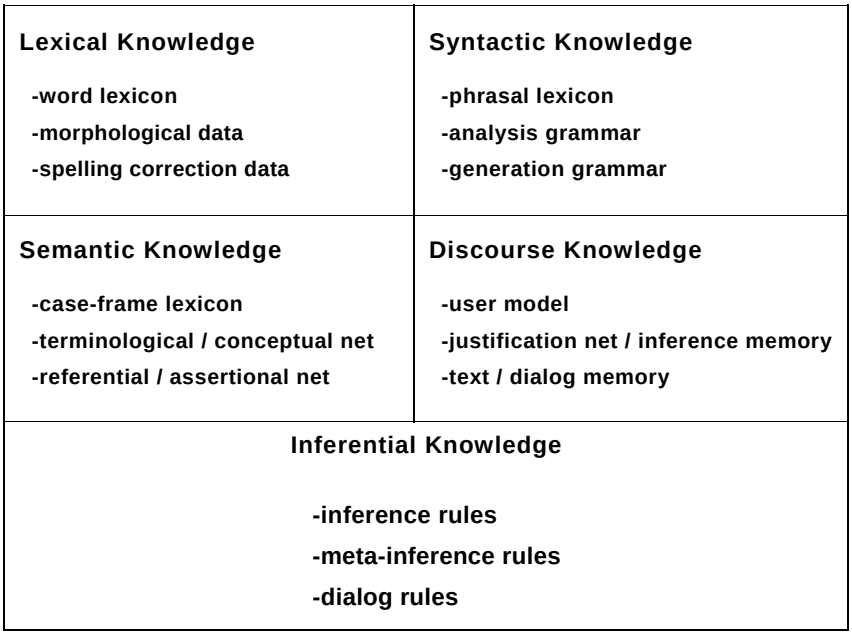

Fig. 2.

- *Knowledge organization:* There are two main knowledge bases involved if NLI for knowledge-based systems are concerned. The NL KB can be organized by the scope of validity of knowledge,

using a distinction of general background knowledge, task-specific knowledge, and dialog-specific knowledge (cf. Fig. 1). Additionally, the KBMS would also have to manage the background KB of the knowledge-based system to which the NLI provides access. Finally, there is a standarized dialog procedure consisting of the analysis of a user request, of the evaluation of this request with respect to the knowledge-based application system, and of the generation of a, hopefully cooperative, answer.

- *Environment and coupling:* First, one needs an environment through which the various knowledge bases can be filled. This is the aspect of knowledge acquisition or learning which can be partially done through the NLI itself (or another NLI), and which must be partially done by linguistic and domain experts in cooperation with computer scientists through a formal knowledge representation language [HAHN87]. The degree of coupling with other software is basically determined by the background system to which the NLI provides access. In particular, all of the systems described in the other sections of this chapter—with the possible exception of design KBMSs - have been provided with limited NLI: databases, expert systems, and geometrical scene descriptions.

4. KBMS Requirements for Visual Data

An increasing number of knowledge-based applications, including those in (robot) vision, natural language understanding and generation, trajectory planning, and spatial reasoning require a representation of time-varying scenes of spatial objects. Such a representation, called a *geometrical scene description* (GSD), calls for substantial KBMS support not offered by present-day database management systems.

GSDs are important for several reasons. They serve the objective of computer vision, "to know what is where by looking," and also satisfy various needs in robotics research. Furthermore, GSDs can serve as an intermediate-level, quantitative scene description between vision and natural language understanding. This representation is high-level enough to serve as a target knowledge representation for a natural language interface, yet precise enough to avoid loss of information with respect to information obtained by the visual channel.

GSDs are a quantitative description of visual properties of time-varying scenes, including 4D object shapes, 4D object locations, illumination aspects, and ties to a conceptual knowledge base. Natural language provides interesting space-time concepts ("events") which one may want to retrieve from a GSD. Natural language motion concepts lead to interesting primitives which express one of a limited set of interesting qualitative propositions about a limited set of observable scene properties; for instance, observables include position, orientation, distance, and velocity with respect to reference positions, whereas qualitative properties can be values such as "constant," "zero," "increasing," "decreasing," larger/smaller than reference," etc. Hence the requirements for event retrieval may be defined largely without reference to a particular application. Details are presented in the chapter by Neumann in this volume.

A final observation is that event primitives are "durative" propositions about time intervals. As a composite event may consist of several interrelated primitives, the corresponding time interval may be constrained *m* a non-trivial way. Hence event recognition involves *temporal constraint satisfaction*. The constraint satisfaction paradigm is as fundamental to event recognition as the instantiation paradigm.

In terms of the four aspects mentioned in the introduction, the KBMS requirements of GSDs can be summarized as follows:

- *Knowledge representation:* There is a need for a specialized knowledge representation of four-dimensional scenes. Additionally, higher-level concepts such as collision, occlusion, motion types, and expectations must be available. Reasoning capabilities cannot be just based on deduction rules as in rule-based expert systems but frequently involve the concept of constraint satisfaction, e.g., finding any answer that satisfies ill the temporal constraints imposed by the scene description and the query
- *Knowledge organization:* Visual data require extremely large, shared databases and very efficient storage and retrieval operations since there is interest in operating them in a real-time environment. The need quantitative and qualitative descriptions imposes different levels of abstraction, among which KBMS should serve to mediate.
- *Environment and coupling:* The GSD KBMS is often coupled with physical vision system (e.g.,

video camera) on the low end, and *with* natural language query and command interface on the high end.

In summary, GSDs demonstrate that there are basic requirements that

- are of general importance for a wide class of applications,
- are not met by current database or knowledge representation technology
- will not be met by KBMSs if a uniform approach to all applications taken.

5. KBMS Requirements of Design Environments

A growing class of knowledge-based applications is no longer concerned with just describing and classifying the world but with helping changing it. The activity of creating and maintaining artifacts is *called design* [SIMO81]. Computer-aided design (CAD) spans a wide range of applications, from shipbuilding, to VLSI and software design, to computer aided planning models in business. From early on, AI researchers have been interested in this area; more recently, database researchers have also given it a great deal of attention.

More than in other knowledge-based applications, the term "knowledge" appears questionable in design support. The design "knowledge base is a collection of evolving partial stored beliefs about a "good" system and its environment. These beliefs are represented as functional specifications, designs, or (in the case of VLSI or software design systems) implementations which may be contributed by different people with possible inconsistent views of the problem to be solved. Knowledge base consistency and dynamic belief acquisition are therefore of critical importance in design support environments.

There are several different views of what a KBMS for design support should offer. *Database researchers* [KL84] tend to stress the aspect of complex objects which evolve over time under the cooperative influence of multiple designers. Several KBMS requirements can be derived from this view:

- It must be possible to compose complex *configurations* from simpler objects in a flexible manner which cannot be easily foreseen in a rigid database schema.
- Multiple equivalent or at least overlapping *viewpoints* and transformations among them must be supported.
- *Version management* must allow the temporal development of partial to full specifications, the concurrent exploration of alternative designs, and the maintenance of designs, i.e., error corrections, adaptation to changing environments, and enhancements.
- *Nested transaction* concepts are required to coordinate the cooperative and concurrent development of the design objects. These concepts define ranges of visibility for the versions of a design object, for instance: public release, project, work group, individual designer. They also form the units of recovery, in order to avoid unnecessary repetition of design work in case of errors.

AI researchers view design as a search process in a very large space of (partially specified) alternatives. To control this search process effectively, a KBMS for design support should not only manage knowledge about the *design states*, as described by the multi-version complex objects mentioned above. In addition, a *process perspective* becomes necessary which involves knowledge about [MOST85]:

- the *goal structure* of the design process. Knowledge about the desired functionality, performance, and critical factors reduces the search space for "good designs." More importantly, explicit knowledge about the design goals allows automatic choice among design alternatives, especially in the case of maintenance where certain design decisions have to be retracted and subsequently "replayed."
- the *design decisions* and their rationales. Knowledge about the reasoning behind design decisions facilitates the communication between designers and maintenance personnel, as well as the checking of consistency between existing and new design decisions. This knowledge reveals the

assumptions and commitments made by a designer; it also determines proof obligations for implementations with respect to the corresponding specifications.
- the *control* mechanisms that determine how the knowledge about design objects, design goals, and the documentation of design decisions are actually brought to bear in designing well with limited search effort. This may include knowledge about a specific design methodology which enforces "procedural rationality" [SIM081] in the design process.

These concepts also imply a need for working with *incomplete specifications*, possibly with an exception-handling mechanism to deal with overabstraction. If possible, design rules and structures should be organized in generalization hierarchies which facilitate automatic rule acquisition in the design process (learning by observation [SMWB85]). This is because design tasks are so diverse that not all rules applicable to a specific case can be defined beforehand. Additionally, design KBMS are an expert support tool where the users are often the best experts.

Finally, researchers with a communications perspective require design KBMS to mediate as a *communications medium* among different designers, or similarly, to serve as a memory aid for an individual designer who has to look at different aspects of a design in a consistent manner [WF86]. This perspective stresses the importance of developing mutually understandable representations of all the issues mentioned before; in other words, the KBMS is viewed as a *knowledge sharing system* [JARK86b].

As the discussion demonstrates, the perspectives on design support differ widely. Nevertheless, a number of common requirements for KBMS can be identified. In general, a design KBMS can be viewed as a documentation knowledge base attached to a problem-solving environment (Figure 3). It must be able to manage consistently large sets of design objects, most of them incomplete and developed from particular viewpoints. Furthermore, transformation and backtracking operations must be controlled in this complex object environment, not only in an initial design but often over extended maintenance periods. The KBMS must serve as a tool to enhance documentation, understandability of the resulting system for designers and end users (covering issues such as requirements validation, explanation, and verification of transformation correctness), and debugging and modification.

In terms of the four areas defined in the introduction, the requirements for a design KBMS can be summarized as follows:

- *Knowledge representation:* Design KBMS need a representation not only for complex design objects but also for knowledge about the design process and its underlying rationales (e.g., in the form of rules and justifications). Knowledge representation must be visualized at least externally such that the design knowledge base serves as a mutually understandable documentation system within and beyond the development group.
- *Knowledge organization:* Typical structures include configurations, views, and versions controlled by a generalization hierarchy of types which can also serve as a starting point for learning. Usually, there are only a few basic objects (e.g., one large software system, or one aircraft wing to be developed) which, however, have a very complex organization. Due to the complex relationships between (sub-) objects, design knowledge should also be organized in a teleological way, that is, objects should be justified in terms of the design goal structure and constraints. A very useful knowledge organization structure is that of a belief maintenance system.
- *Environments:* Querying facilities must include information about the design history as well as about different views (abstractions, representations) of the database. Consistency checking must be possible across different representations, and must include the possibility that design portions made by different designers can be temporarily inconsistent. This inconsistency must be gradually resolved using nested transaction concepts which organize formal group cooperation.
- *Coupling:* Ideally, it should be possible to couple a design KBMS with the usage environment. In this case, the end user can query the design knowledge base to get an understanding of what the designed system does and why it does it like that. However, this is currently only done with expert systems shells (the design objects being expert systems) where the shell remains as a usage environment as well as a development environment.

The chapter by Borgida et al. provides an example of this kind of KBMS.

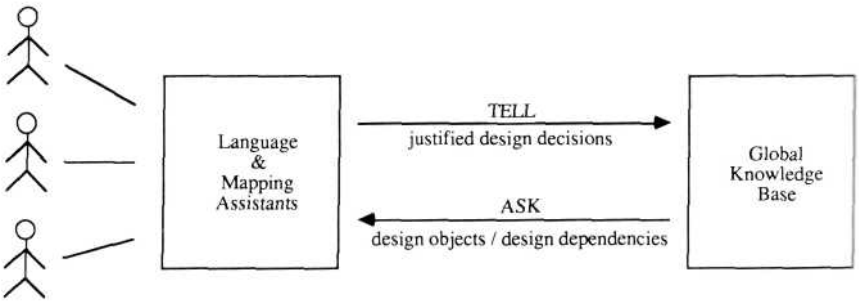

Fig.3.

6. Conclusion

The purpose of this chapter was to provide a customer view of KBMS: what KBMS tools do knowledge-based applications really need?

Our review of four important KBMS application areas has shown some commonalities but also significant differences. *Common* to all applications is the need for handling large amounts of factual data as provided by databases. In some cases, such as GSD and design KBMS, the structure of the data appears so complex that specialized DBMS with capabilities for handling complex objects are needed. In the case of design KBMS, transaction concepts must also be extended.

A second common requirement appears to be that metaknowledge about the available information is needed as a dictionary to integrate knowledge from different sources. This feature was particularly visible in the case of KBMS for NLI but also in design KBMS.

The *differences* lie in the kinds and specific representations of knowledge dictated by the need of efficiency in different domains. As can also be seen in earlier sections of this book, many capabilities of simple rule-based expert systems can be embedded into extended DBMS which provide limited deductive capabilities in an efficient, set-oriented manner. On the other hand, the more advanced knowledge-based applications require either very dedicated data structures not provided by DBMS (geo metrical scene descriptions are a good example), or the special purpose integration of a large number of different sources of knowledge in different special-purpose representations.

One feature of growing importance in this context is the representation of *time*. In GSD, we describe time-varying spatial objects and relationships. In design KBMS, we describe a history of our knowledge and conceptual-work on an artifact which in itself may contain a temporal model. Time also plays an important role in natural language. Thus, the different uses of time, each requiring specialized knowledge representation and (expensive) reasoning approaches such as constraint satisfaction, dependent: tracing, fuzzy reasoning, etc. may be one of the main factors that prevent a more general KBMS approach.

In summary, generalized KBMS for advanced knowledge-based applications appear presently infeasible. Either a KBMS has to limit itself to supporting efficiently very simple knowledge representations (at most deductive databases), or it must embed application-area-specific know I edge in its architecture to provide efficient reasoning with limited resources.

7. References

[BJ86b] Bolc, L., and M. Jarke (eds.), *Cooperative Interfaces to Information Systems*, Springer-Verlag, Berlin, Heidelberg, 1986

[BL86] Brachman, R.J., and H.J. Levesque, “What Makes a Knowledge Base Knowledgeable? A View of Databases from the Knowledge Level”, in KERS86, 1986, pp. 69-78.

[BM86a] Brodie, M.L., and J. Mylopoulos (eds.), *On Knowledge Base Management Systems, Integrating Artificial Intelligence and Database Technologies*, Springer-Verlag, Berlin, Heidelberg, May 1986.

[FM86] Fox, M.S., and J. McDermott, "The Role of Databases in Knowledge Based Systems", in [BM86a], 1986, pp. 407-430.

[HAHN87] Hahn, U., "Modelling Text Understanding: The Methodological Aspects of Automatic Acquisition of Knowledge Through Text Analysis", *Proc. 1st international Symposium on Artificial Intelligence and Expert Systems*, Berlin, 1987

[HMM86] Hoeppner, W., K. Morik, and H. Marburger, "Talking It Over: The Natural Language Dialog System HAM-ANS", in [BJ86b], 1986, pp. 189-258.

[HWL83] Hayes-Roth, F., D.A. Waterman, and D.B. Lenat (eds.), *Building Expert Systems*, Addison-Wesley, Reading, MA, 1983

[JARK86b] Jark, M., "Knowledge Sharing and Negotiation Support in Multiperson Decision Support Systems", *Decision Support Systems*, Vol. 2, No. 1, 1986, pp 93-102.

[JV84] Jarke, M., and Y. Vassiliou, "Coupling Expert Systems with Database Management Systems", in [REIT84b], 1984, pp. 65-85.

[KERS86] Kerschberg, L. (ed.), *Expert Database Systems*, see [KERS84], 1st International Workshop on Expert Database Systems, Benjamin/Cummings Publisher Company, Inc., Menlo Park, CA, 1986

[KERS84] Kerschberg, L., (ed.), Proc. 1st International Workshop on Expert Database Systems, Kiawah Island, South Carolina, October 1984.

[KL84] Katz, R.H., and T.J. Lehman, "Database Support for Versions and Alternatives of Large Design Files", *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 2, pp. 191-200.

[KOWA86] Kowalik, J.S. (ed.), *Coupling Symbolic and Numerical Computing in Expert Systems*, North-Holland, Amsterdam, 1986

[MOST85] Mostow, J., "Towards Better Models of the Design Process", *Artificial Intelligence Magazine*, Vol. 6, No. 1, 1985, pp. 44-57

[MYLO86] Mylopoulos, J., "On Knowledge Base Management Systems", in [BM86a], pp. 3-8.

[NAU83] Nau, D.S., "Expert Computer Systems", *IEEE Computer*, Vol. 16, No. 2, 1983, pp. 63-85.

[PAU86] Pau, L.F. (ed.), *Artificial Intelligence in Economics and Management*, North-Holland, Amsterdam, 1986

[REIT84b] Reitman, W. (ed.), *Artificial Intelligence Applications for Business*, Ablex, Norwood, NJ, 1984

[SIMO81] Simon, H.A., *The Science of the Artificial*, 2nd ed., MIT Press, Cambridge, MA, 1981

[SMWB85] Smith, R.G., T.M. Mitchell, P.H. Winston, and B.G. Buchanan, "Representation of Use of Explicit Justifications for Knowledge Base Refinement", *Proc. 9th International Joint Conference on Artificial Intelligence*, Los Angeles, CA, 1985, pp. 673-680.

[SZOL86] Szolovits, P., "Knowledge Based Systems: A Survey", in [BM86a], 1986, pp.339-352.

[VCJ83] Vassilou, Y., J. Clifford, and M. Jarke, "How Does an Expert System Get its Data?", *Proc. 9th Conference on Very Large Databases*, Florence, Italy, 1983, pp. 70-72.

[WAHL84] Wahlster, W., "Cooperative Access Systems", *Future Generation Computing Systems*, Vol. 1, No. 2, 1984, pp. 103-111.

[WAHL86]Wahlster, W., "The Role of Natural Language in Advanced Knowledge-Based Systems", Report No. 6, Sonderforschungsbereich 314, Univ. des Saarlandes, Saarbrücken, FRG, 1986.

[WF86] Winograd, T., and F. Flores, *Understanding Computers and Cognition: A New Foundation of Design*, Ablex, Norwood, NJ, 1986.