Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research
Report
RR-94-31

Integrated Knowledge Utilization and Evolution for the Conservation of Corporate Know-How

Otto Kühn, Volker Becker,
Georg Lohnse, Philipp Neumann

August 1994

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Deutsches Forschungszentrum
für
Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, Sema Group, and Siemens. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct systems with technical knowledge and common sense which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

Intelligent Engineering Systems
Intelligent User Interfaces
Computer Linguistics
Programming Systems
Deduction and Multiagent Systems
Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Dr. Dr. D. Ruland

Director

Integrated Knowledge Utilization and Evolution for the Conservation of Corporate Know-How

Otto Kühn, Volker Becker,
Georg Lohnse, Philipp Neumann

DFKI-RR-94-31

This report is a slightly modified version of the paper to be published in the Proceedings of ISMICK'94, International Symposium on the Management of Industrial and Corporate Knowledge, Compiègne, October 26 - 27, 1994.

This work has been supported ba a grant from The Federal Ministry for Research and Technology (413-5839-ITWM-9304/3).

Integrated Knowledge Utilization and Evolution for the Conservation of Corporate Know-How

Otto Kühn, Volker Becker
German Research Center for AI (DFKI)
P. O. Box 2080
D-67608 Kaiserslautern, Germany
Phone: +49-631-205-3463
Fax: +49-631-205-3210
E-mail: kuehn@dfki.uni-kl.de

Georg Lohse, Philipp Neumann
Stuttgarter Strasse 80
D-71332 Waiblingen, Germany
Phone: +49-7151-26-0
Fax: +49-7151-26-1164

Abstract

Insufficient consideration of knowledge evolution is a frequent cause for the failure of knowledge-based systems (KBSs) in industrial practice. Corporate know-how about the design and manufacturing of a particular product is subject to rather rapid changes, and it is hard to specify in advance exactly what information will be requested by various users. Keeping a KBS for the conservation of corporate know-how up-to-date or even enhancing its utility, thus requires the continuous monitoring of its performance, noting deficiencies, and suggestions for improvements. In the current paper, we discuss different ways in which information collected during knowledge utilization can be exploited for system evolution. We present structure-based rule and concept editors which allow for an immediate integration and formalization of new information, even by rather unexperienced users. A prototypical knowledge conservation system for crankshaft design which was developed in cooperation between the DFKI and a German company is used to illustrate and evaluate our approach.

Keywords: conservation of corporate know-how, knowledge evolution, user feedback, KBSs as communication media

Contents

1 Introduction

The successful installation of a knowledge-based system (KBS) which captures a company's technical know-how in the design and manufacturing of a particular product requires a thorough consideration of knowledge evolution. The captured expertise is subject to rather rapid changes due to the perfection of new technologies and the activities of competitors on the marketplace which call for continuous adjustments of existing products and of established guidelines for design, manufacturing, and marketing. Keeping such a corporate KBS up-to-date or even enhancing its utility, thus requires continuous efforts which will only be invested, if the extra workload for the involved experts can be kept at a minimum and if the system satisfies well the actual needs of its users.

In order to meet these requirements, knowledge utilization and knowledge evolution have to be well coordinated. Knowledge utilization not only reveals the most serious deficiencies of a KBS, but it also shows where possible extensions would be most beneficial. Such user feedback is indispensable for an efficient and effective evolution of a KBS. Furthermore, since many users of a corporate KBS are also experts in some particular area, one should take advantage of their expertise even more directly for updating and improving the KB.

In the next section of the paper, we will first investigate the benefits and drawbacks of different ways of integrating knowledge utilization and knowledge evolution. In the subsequent section, we will present the KONUS knowledge conservation system which was developed in cooperation with a German company in order to support the capturing, distribution, and continuous evolution of the corporate know-how about crankshaft design. We will then show in more detail, how knowledge evolution is supported by a graphical structure editor which allows modifications of the KB to be made also by rather unexperienced users. Finally, we will discuss and evaluate the suggested approach based on some preliminary customer feedback on the KONUS prototype.

2 Integrating knowledge utilization and evolution

2.1 KBSs as communication media

According to the traditional AI view, a KBS should be seen as some sort of a substitute for a human expert (hence the term "expert system"). Ideally it should be endowed with expert-like capabilities for problem solving, explanation, learning, and communication. The adequacy of this view has been challenged both on theoretical grounds and on AI's *failure to deliver* [WF88]. All expert systems which have been developed up to now are far from showing expert-like qualities. According to currently established methodologies for knowledge engineering and system development such as KADS [BW89], they can best be understood as complex machines which have been carefully designed and tuned for the solution of a particular task.

Like the construction of any complex, special-purpose machine, the development of a task-specific KBS is rather time consuming and cost intensive. It can only be performed if the target task is well understood and the involved expertise is readily available. Furthermore, the resulting system will necessarily be brittle [Ste85], require specialists for maintenance, and the captured expertise will not easily lend itself for sharing and reuse [NFF+91]. All this imposes serious limitations for the successful application of KBSs in industrial practice. Whether these problems can indeed be overcome by endowing expert systems with extensive common sense knowledge, as is currently attempted in the CYC project [LGP+90], is still a rather controversial issue.

Recently, a different view on KBSs has been proposed which offers new insights into their development, utilization and evolution, and it also much better suits the most acute demands of industrial practice [Whi91]. Instead of regarding KBSs as human-like agents or as complex machines, they are conceived as useful tools [1] and as versatile communication media for the support of human problem solving [WO88]. A KBS can thus be compared to a book or manual in which relevant information about some problem domain has been composed in order to make it generally available. Of course, such a computerized knowledge depository is much more flexible than a printed medium, and it can also provide active support for the solution of various tasks [KH94].

This new perspective on KBSs promises to greatly enhance their practical utility and cost efficiency: On the one hand, there is a high demand for an explicit documentation and better distribution of corporate know-how so that computerized support promises here a high pay-off. On the other hand, the development of a useful communication tool can be accomplished with much lower investments than the automatization of a complex task. In particular, as pointed out by [FMO+94], one does not have to "put in all the knowledge at the beginning", but one may start with an initial "seed" which may then grow and evolve during utilization based on feedback and additional information from the users. Before exemplifying how such a system can be built for the conservation of corporate know-how about a particular product, we will first analyze different means by which the users can be made to participate in the evolution of a KB.

2.2 User participation in KB evolution

Getting users to participate in the evolution of a KB requires a consideration of several factors. First of all, users are often overloaded with work and do not like the usual flow of work being disrupted by some extra activity which is not directly rewarded [SM94]. In order to overcome this problem, one may try to: 1) keep the extra effort at a minimum, 2) highlight the importance and utility of the task, and 3) make the task more rewarding.

A second factor which must be considered is the quality of the newly added information and its effect on the overall quality of the KBS. Whereas the deletion and modification of stored knowledge should obviously be restricted to KBS administrators, even the unconstrained addition of questionable and poorly structured information may impede

[1] In analogy to mechanical tools, which too, are much cheaper to build, easier to adapt, and in many situations much more useful than a complex, special-purepose machine.

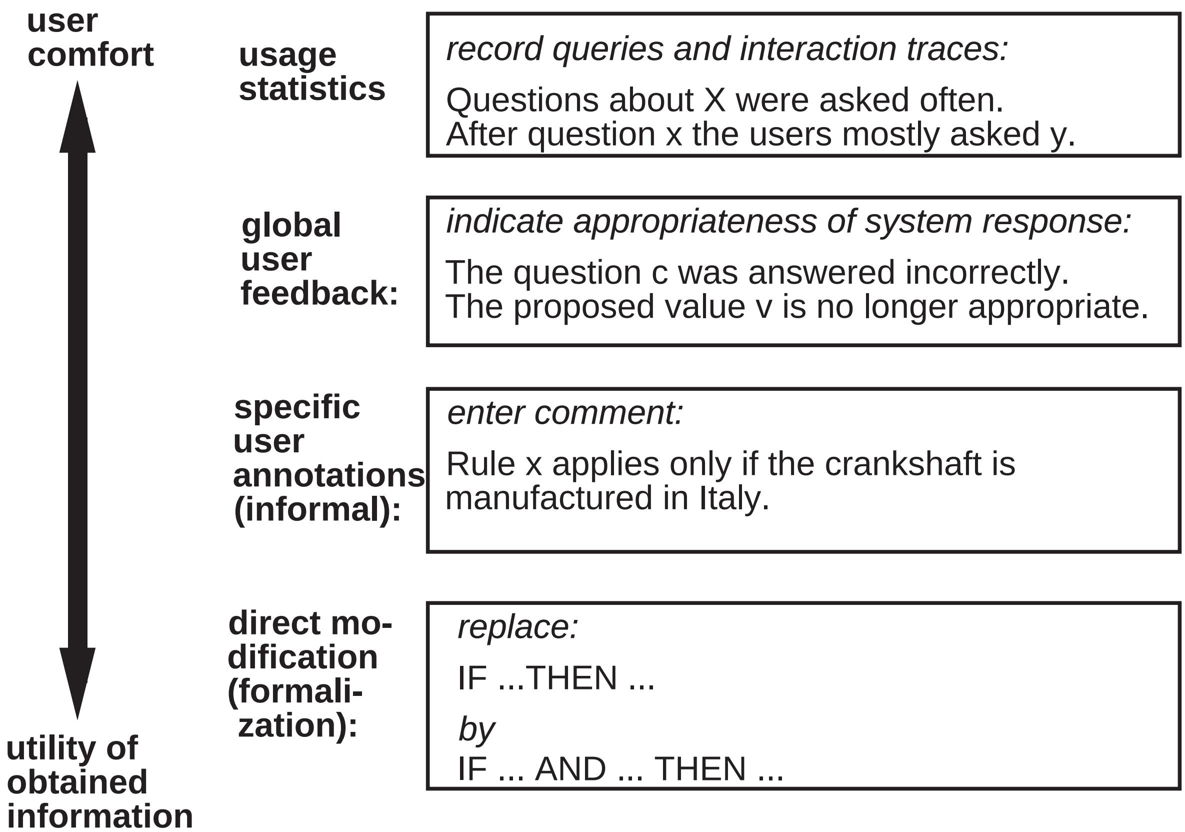

Figure 1: Ways of integrating knowledge utilization and knowledge evolution.

the present utility of a KBS as well as its further evolution.[2] Therefore, as suggested in [FMO+94], an occasional *reseeding* should be conducted in which software designers together with domain specialists re-organize the knowledge added during the use of the KBS.

Figure 1 shows different ways in which information for knowledge evolution can be collected during the utilization of a KBS. The four exemplary methods differ with respect to the amount of additional effort required and the associated disruption of normal workflow (labeled as "user comfort"), and the inversely related "utility of the obtained information".

The first method, which entails absolutely no workflow disruption, consists in the mere collection of **usage statistics** which may nevertheless provide useful information for knowledge evolution. The frequency of questions about various topics may indicate where a further extension of the KB would be most beneficial. Recurring question sequences may reveal that the prior questions were answered insufficiently. An analysis of usage statistics

[2] Many INTERNET newsgroups exemplify how an electronic communication medium is rendered useless for serious users, since the least knowledgeable users tend to make the largest number of contributions.

can thus provide useful hints for the further evolution of the KB without, however, telling how that evolution should be performed. The advantage of zero user interference is thus coupled with a restricted exploitability of the collected information.

More useful information may be obtained from **global user feedback** on the adequacy of the system's responses. Such feedback requires only one or two mouse clicks from the user and causes only a minor disruption of the usual workflow. Nevertheless, even such a global feedback supplies positive and negative examples of the desired system behavior which may be exploited for a revision of the knowledge base and an improvement of the system [HMS94].

Whereas the information obtained with these nonintrusive methods more or less only tells where deficiencies exist, the collection of **specific user annotations** directly tells what knowledge should be added and how inappropriate knowledge should be corrected. Since this method requires the users to enter self-explanatory text, it causes a major disruption of the usual workflow. On the other hand, the entered annotations can be made accessible to other users, so that the KBS now truly constitutes a communication tool between the various users. Furthermore, the annotations may be later formalized by the knowledge engineer, ideally without further queries to their author.

As a fourth method, the users of a KBS might be allowed and encouraged to make **direct modifications of the formal knowledge**. This would be optimal within the "KBS as communication tool" paradigm, since the newly entered knowledge can be automatically processed by the system and thus be made directly available to other users.[3] Fischer et al.[FMO+94] argue that a direct formalization of knowledge by users is impractical, since users do not know how to formalize, and since thinking about formalization constitutes too severe a disruption of the usual workflow. Furthermore, one might suspect that a direct modification of the KB by the users would do more harm than good. We will come back to these issues in the discussion of the paper, after having seen how the different methods of integrating knowledge utilization and knowledge evolution are conjointly employed in the KONUS system.

3 The KONUS knowledge conservation system for crankshaft design

3.1 Application domain and system requirements

The crankshaft is one of the core pieces of a combustion engine. When designing a new engine or modifying an existing one, the crankshaft will almost always be affected. As a consequence, there is a very high demand on the crankshaft specialist whose advice is urgently needed when a new engine is to be designed or when improvements of products

[3]Whereas informal annotations can also be shown to other users, they cannot directly modify the behavior of the KBS. The resulting communication is suboptimal, since the new information is only accessible via the old knowledge to which it is attached.

already on the market are to be performed.

In order to design a high-quality crankshaft which can be manufactured at low costs, the crankshaft specialist has to take into account many company-specific circumstances. He must consider: what machines are available for manufacturing, what quality has previously been achieved on this machinery with particular manufacturing procedures, what raw parts and components are offered at sufficient quality and at reasonable prices, what materials have the required mechanical properties and can be effectively processed with the available machines and tools, etc. All these factors may affect the geometry and technology of the crankshaft, and must thus be taken into account in the technical drawings of the crankshaft components at various stages of processing.

This company-specific expertise on crankshaft design is poorly documented and exists only in the heads of a small number of human experts. Often enough, particular questions cannot be answered if an individual specialist is on leave or otherwise not available. This information bottleneck causes severe delays in the development of new products and in improving the cost efficiency and quality of existing products [Loh93].

The knowledge conservation system KONUS[4] was developed in order to document the company-specific expertise on crankshaft design and make it effectively available to the various advice-seeking users. KONUS should not only answer questions about previously designed crankshafts, but it should also support the design of new crankshafts so that it can be done more quickly and possibly also by less experienced designers. Furthermore, KONUS must allow the captured knowledge to be continuously modified and extended. This is of utmost importance, since an essential portion of its knowledge consists of company specific experiences and guidelines which are based on the current engineering technology and market situation. For instance, the mere devaluation of a currency may affect the cost-efficiency of a particular crankshaft design by rendering a different supplier of raw components more competitive.

3.2 Overview of the KONUS system

The global architecture of the KONUS system is shown in figure 2. The various users, such as the crankshaft expert, other engineers in the product development team, and members of the quality assurance department interact with the KONUS system via a graphical user interface which integrates the different knowledge utilization and knowledge evolution services. These services can be grouped into three categories which were named according to [Gre92]:

The design aide provides direct support for the construction and modification of crankshafts, so that this activity can now also be performed by engineers who are not dedicated crankshaft specialists. Even though unable to design a crankshaft automatically, the design aide suggests a strategy, shows viable solution alternatives, and provides critique if design guidelines are violated.

[4]KONUS is a German acronym for conservation and construction support.

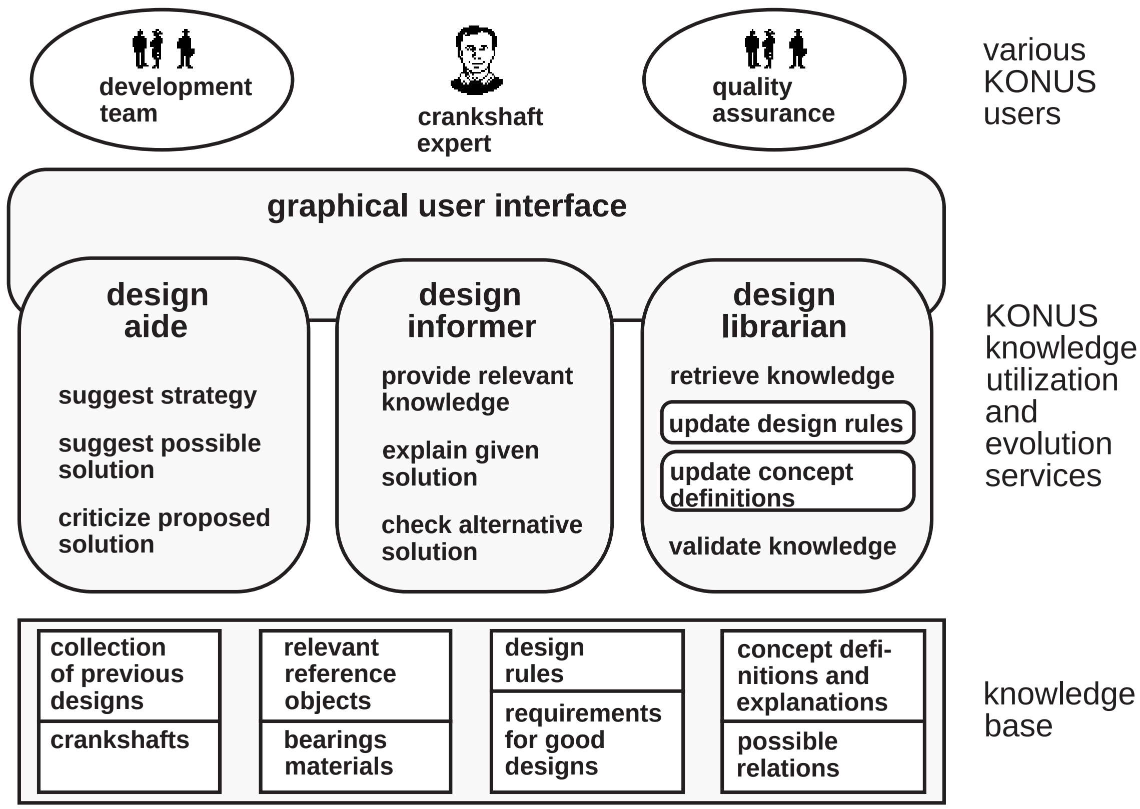

Figure 2: Overview of the KONUS knowledge conservation system.

The design informer provides answers to different types of questions about previous designs which frequently occur in everyday practice. Most important are "Why?" and "Why not?" questions which explain previously made design decisions and check the feasibility of design alternatives. For a more detailed description of the design informer (and the design aide) see [KH94].

The design librarian supports the management and evolution of the design knowledge by providing functionalities for knowledge retrieval, update and validation. For knowledge update, a rule and a concept editor are provided which will be described in detail in section 4. of this paper.

Even though shown in different boxes the knowledge utilization and evolution components are tightly integrated so that most services of each component can be accessed from the others. The captured design expertise is stored in a **knowledge base** in which four different types of knowledge are distinguished.

A collection of previous designs (case base) comprises descriptions of crankshafts and related crankdrive components (e.g. connecting rod). Each component (design object) is described by attribute value pairs. Contrary to the technical drawings, the attribute-value representation provides an abstract and qualitative description which lists those design features which play a central role in the experts' reasoning when solving a design problem.

Relevant reference objects such as materials and bearings are also described by attribute value pairs. They are distinguished from the design objects, since they are not designed but selected. Their properties must, however, be taken into account when determining the features of the design objects.

The concept definitions and explanations constitute a sort of ontology of all terms which may occur in the attribute value descriptions of the crankshaft designs and the relevant reference objects. Thereby they also define the vocabulary for the design rules. Besides a formal definition of concepts and relations, this ontology also comprises informal explanations as well as information about how the various object descriptions are to be displayed to the users. Examples of concept definitions and explanations will be given in section 4.2. together with the description of the concept editor.

The design rules contain the essence of the company's design expertise. They indicate what criteria (attribute value combinations) should be satisfied by good designs with respect to engineering, cost-efficiency, and manufacturing concerns. The deeper design rationale is not represented formally, but is given in natural-language explanations which are attached to the design rules.[5] Examples of design rules will be given in the subsequent sections.

3.3 Collecting information during knowledge utilization

Figure 3 exemplifies how knowledge utilization and knowledge evolution are integrated in the KONUS system. The left window in the figure displays the answer to a "Why not?"-question which was asked by a user of the design informer. The answer summarizes the pro and con arguments and refers to individual design rules which may be inspected by users wishing more detailed information. One such design rule is displayed in a separate window on the right side of the figure.

For the **collection of usage statistics**, the KONUS system automatically records the question asked (including a reference to the design case), the summary answer, and the additional information (design rules) requested by the user.

In order to give **global feedback** the user may click on the "feedback" button at the bottom of the answer window. A feedback window then pops up displaying a five point rating scale on which the user may judge the appropriateness of the answer by a simple mouse click.

[5] The informal explanations comprise a considerable portion of the design expertise and are of equal importance as the formal design rules. The latter could even be seen as mere navigation links to the Hypertext constituted of the rule explanations.

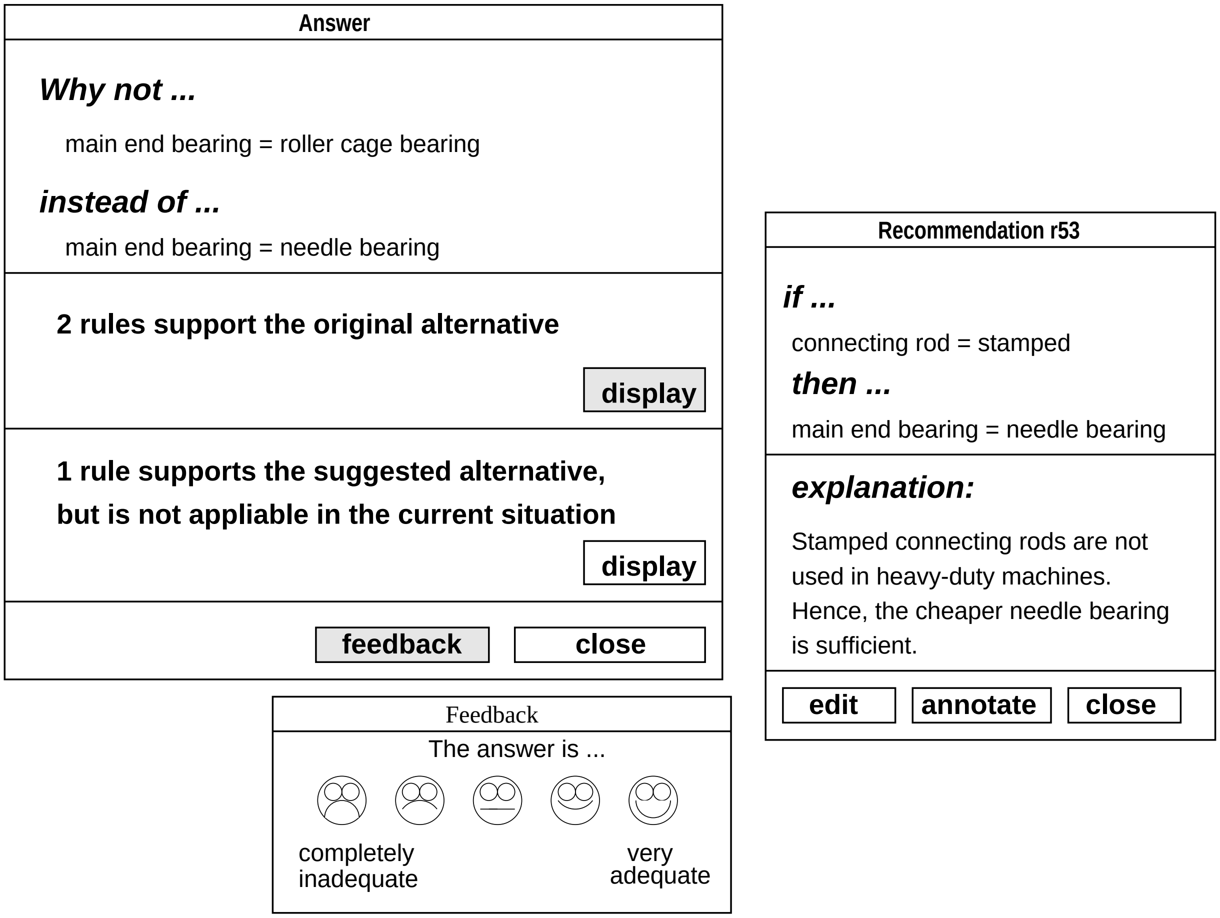

Figure 3: Integration of knowledge utilization and knowledge evolution in KONUS

Specific annotations may be made to individual design rules by clicking on the "annotate" button provided with each rule window.

Similarly, the user may **directly modify** the design rule by clicking on the "edit" button. This action will call the rule editor on the particular rule which allows to make modifications by menu selections and mouse clicks.

For each piece of information collected during knowledge utilization, KONUS records the date, the time, and the author. Making the latter public, will hopefully motivate users to publish their knowledge, since they will be credited for their individual contributions to the corporate knowledge depository.[6]

[6] Maybe the employer could additionally offer some monetary or other incentives.

4 Supporting knowledge evolution

In order to provide optimal computational support and to always deliver the latest relevant design knowledge, the information collected during knowledge utilization has to be formalized and properly integrated into the KBS. As already mentioned, the knowledge formalization may be either done directly by the users during use or in special "reseeding" phases by system developers and domain experts. In both cases, the formalization should be supported by computerized tools so that the users and knowledge engineers don't have to deal with the syntactical details of the knowledge representation formalism.

For an easy manipulation of formally structured information, structure-oriented editors have been proposed [Min92]. Such editors are based on the principle *error prevention is better than error correction*, and they guarantee at least the syntactical correctness of the entered information. Although up to now structure-based editors have rarely been used in industrial applications, this is rather due to problems of devising an adequate user interface than due to the structure editor concept itself [Min92].

In the KONUS knowledge conservation system, two structure-based editors (one for design rules and one for concept definitions) have been integrated into the graphical user interface. These knowledge editors were found to substantially facilitate knowledge evolution by system users, system experts, and knowledge engineers.

4.1 The rule editor

The rule editor allows to display, modify and delete existing design rules as well as to create new ones. Rule selection can be performed in various ways, e.g. by the attributes occurring in the rule conclusions. The selected rule is then displayed in the editor window as shown in figure 4.

The syntax for design rules is rather complex (see [KK94]), since we wanted to represent chunks of knowledge as stated by the expert. Dissecting meaningful chunks of knowledge would also greatly impede knowledge evolution. In particular, every design rule may have an arbitrary number of premises and conclusions. The rule editor allows to add or delete premises and conclusions by pressing the "+" or "−"-buttons displayed on the left.

Each premise or conclusion of a design rule consists of an attribute specification, a relation, and a value specification. An attribute specification is created (or modified) by selecting an attribute from a listbox which appears upon clicking on the attribute specification box. If the chosen attribute belongs to more than one object, the user is asked to select the desired reference object. By this method complex attribute specifications, such as *the heat-treatment of the material of the main-end bearing*, can be constructed.

The corresponding value specification can be constructed in similar way by clicking on the value specification box. The admissible values for the particular attribute are then displayed in a listbox (as shown on the right of the figure) from which one or more values may be selected. For numerical attributes, the user may enter a value and choose one of the relations $<$, $\leq$, $>$, $\geq$, $=$. Alternatively he may also specify that one of these relations holds

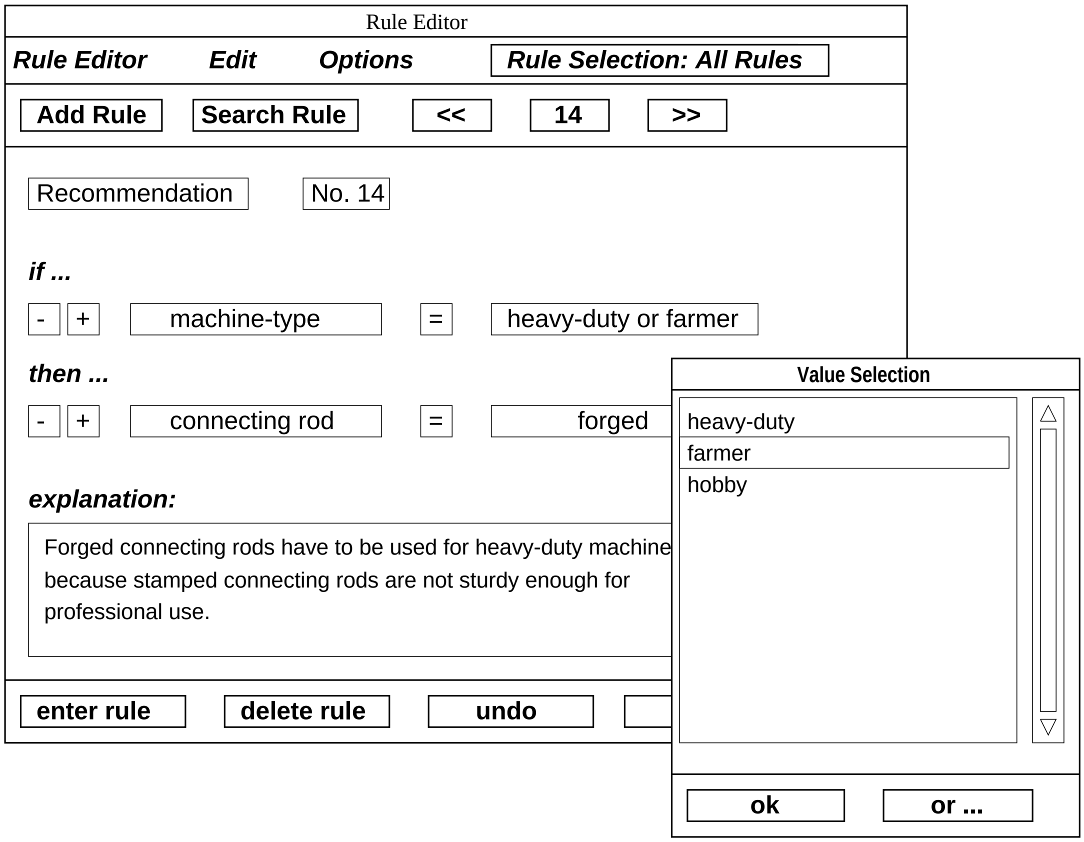

Figure 4: Using the rule editor.

between two attributes. This allows the construction of complex premises or conclusions such as: *the country of manufacturing is Italy or Spain*, *the width of the main-end bearing is greater than 1.8 cm*, or *the diameter of the left shaft is equal to the diameter of the right shaft*.

The described rule editor makes the generalization and refinement of rules particularly easy. In order to generalize *the country of manufacturing is Italy* to *the country of manufacturing is Italy or Spain*, one simply has to click on *Italy* and select *Spain* from the list of countries which pops up. Refining a rule by adding a premise (e.g. it applies only for heavy-duty machines), can be done by simply clicking on the "+"-button next to a given premise. An empty attribute and value specification box will then be displayed on a new line for which the appropriate contents, i.e. *machine-type*, and *heavy-duty* can be selected by mouse clicks.

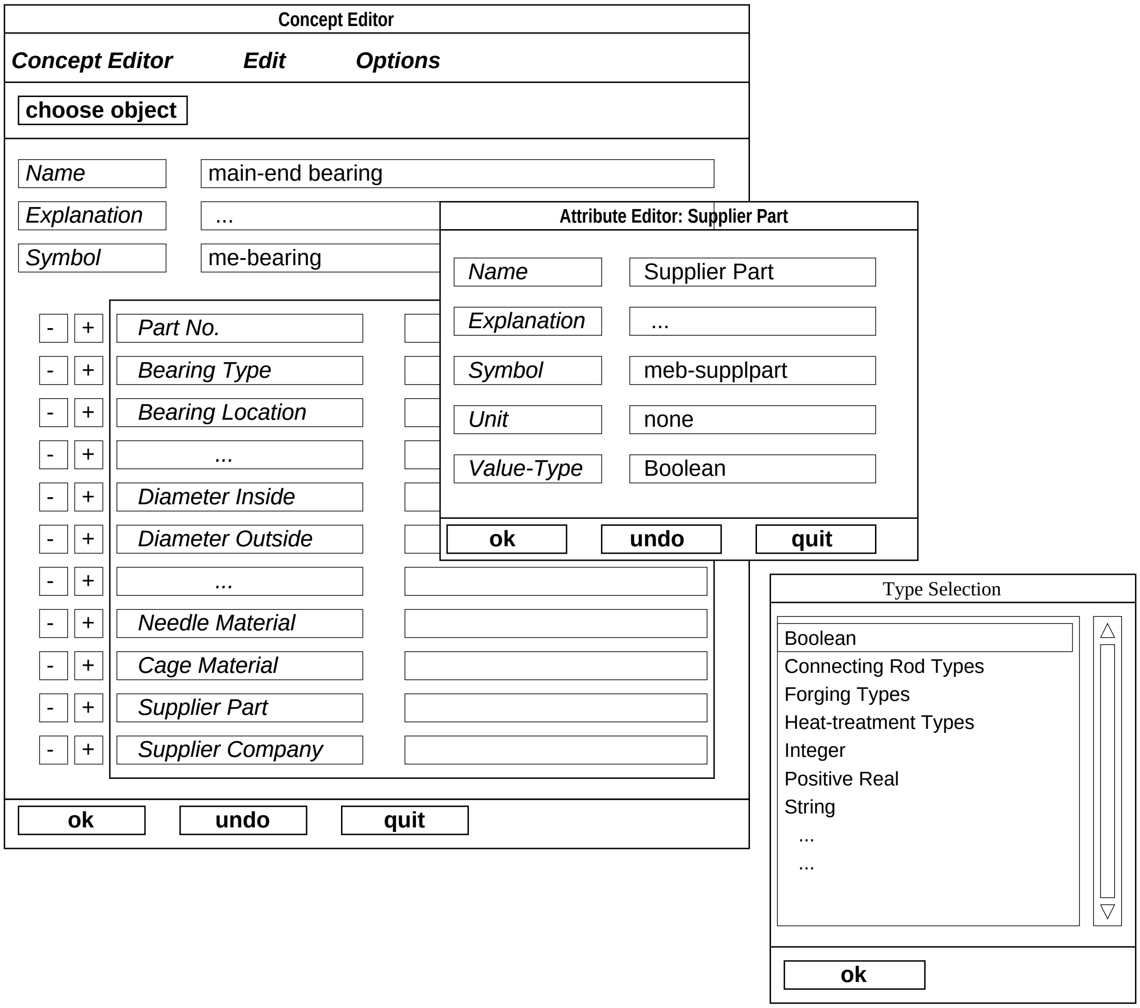

Figure 5: Using of the concept editor.

4.2 The concept editor

The previously described design rule editor allows only to specify design rules which can be expressed with the concept definitions (objects, attributes, value types, and values) given in the ontology. These definitions will sooner or later also require revisions, since it is impossible to specify all potentially relevant features (attributes) of the to be designed objects in advance. Therefore a concept editor is provided which, similarly to the rule editor, is based on the principles of selection and direct manipulation as far as possible.

Figure 5 exemplifies how the concept editor is employed to add the attribute *supplier part* to the object *main-end bearing*.

The leftmost window, the concept editor main window, shows a description of the properties of the object "main-end bearing". The general properties which contain information for display to the user, such as "object name" and "explanation", are shown on the top of the window in directly editable fields. The attributes of objects are displayed below, in the same sequence as they are displayed during knowledge utilization. Again, the buttons labeled "+" and "−" on the left can be used to insert or delete attributes. The row of buttons at the bottom of the main window invokes commands to add the modifications to the knowledge base, to undo changes and to abort the current task.

When clicking on an attribute name, the properties of this attribute (e.g. attribute-name, explanation, value-type) are displayed in a separate window which pops up next to the attribute. Whereas the attribute name and the explanation can be directly typed in, the assignment of a value-type can be performed by either choosing from existing types (enumeration and standard non-enumeration types such as string, real, integer ...) or creating a new type in a separate type-editor window. In the given example, the user can choose the predefined type *boolean* as the value-type for the attribute *supplier part* from a type selection box which is seen on the right side of the figure.

Even though it is not allowed to delete existing concepts from the knowledge base, it is possible to modify existing properties, e.g. to add a new value to a given enumeration type or to remove attributes from the attribute-list of a specific object. These removed attributes nevertheless remain stored in the knowledge base for later re-use and easier consistency checking.

4.3 Knowledge validation

Whereas the described rule editor guarantees the syntactical correctness of newly entered design rules and their consistency with the design ontology, it does not prevent the entry of rules which are in conflict with existing rules. One might argue that the validation of newly entered knowledge should be performed during subsequent knowledge utilization, since this will identify exactly those deficiencies which impede the utility of the KBS.

It is true that the strict fulfillment of formal criteria such as consistency and completeness is not essential for the knowledge conservation system to be useful. Furthermore a complete formal verification of the design rules seems hardly feasible due to the complexity of the rule syntax and the severe runtime constraints given in our evolution scenario. Nevertheless, some validation should be performed whenever a design rule has been added or modified. This helps to avoid a careless deterioration of the knowledge-base which could later only be repaired with much greater effort.

For instance, if two rules with the same premises have contradictory conclusions (i.e. different values are suggested for the same attribute) the user should be averted of this fact. He may then decide whether one rule with a value disjunction in the conclusion is adequate. Alternatively he might opt to delete one of the rules since it is no longer valid, or to specialize one of the rules by adding another premise.

As described in [KK94], some severe conflicts and redundancies can be detected by constructing partial case models for rule pairs and triples. Users are informed of the detected problems and the involved rules are displayed. Rules recognized to be incorrect may be immediately modified with the rule editor. Alternatively, the users may simply proceed, and the detected deficiency is recorded for a later correction by the knowledge engineer.

5 Evaluation and Discussion

A prototype of the described KONUS system was recently delivered to our industrial partner. Their general impression was that a system which offers the functionalities of KONUS might be very useful in overcoming the information bottleneck which often delays the development and enhancement of products. It was noted in particular, that KONUS does indeed provide adequate and comprehensive answers to typical questions which arise in everyday practice. This is mainly due to the combination of natural language explanations together with formal knowledge items which can be automatically processed.

Concerning the integration of knowledge utilization and knowledge evolution, our industrial partner was highly skeptical of allowing users to make direct modifications of the KB. Therefore we opted to treat formal knowledge entered by the users in the same way as informal annotations. It is shown to other users only upon request, and it is allowed to take effect in the system behavior only after it has been approved by an expert who is responsible for maintaining the respective segment of the KB.

The provided knowledge editor was assessed as being highly promising for knowledge evolution by the company's design experts who are no computer specialists. Especially the easy handling and the offered user guidance were highly appreciated. One point of criticism concerned the selection sequence for attributes and values in the rule editor. Always having to choose the attribute first was considered to be inappropriate, since the attribute values are generally more intuitive than the rather artificial attribute names. Future versions of the knowledge editor will allow a more flexible construction of design rules. We are also considering to automatically suggest attributes and values based on keywords given by the user, similar to [SM94].

Although tailored to a specific application, the described KONUS system owes much to other KBSs which have been described in the literature. The closest relatives of KONUS are probably the knowledge based systems developed in the working group of Gerhard Fischer at the University of Colorado, the best known of which is perhaps the JANUS kitchen configuration system [FMM89]. This system integrates Hypertext with a knowledge-based design environment for the support of cooperative problem solving. It lets the users determine the course of action, providing critique [FLMM91], and making suggestions. The conjoint utilization and evolution of a knowledge base has also been suggested by this working group as well as by several authors within the Hypertext community (e.g. [CW89]). The overall architecture of the KONUS system and its organization of the knowledge base which includes a domain ontology together with explanations of the employed terms was inspired by recent suggestions for knowledge sharing and reuse [NFF+91].

Acknowledgement

This research was partly supported by grant 413-5839-ITW9304/3 from the German Federal Ministry for Research and Technology. The authors wish to thank the industrial partners who showed much patience during knowledge acquisition and helped us understand the real requirements for a practically useful knowledge conservation system. Concerning the implementation of the system, we owe special thanks to Björn Höfling and Thomas Malik.

References

[BW89] Joost Breuker and Bob Wielinga. Models of expertise in knowledge acquisition. In Giovanni Guida and Carlo Tasso, editors, *Topics in expert system design, methodologies and tools*, pages 265 – 295. North Holland, Amsterdam, Netherlands, 1989.

[CW89] P. Cooke and I. Williams. Design issues in large Hypertext systems for technical documentation. In R. McAleese, editor, *Hypertext theory into practice*, pages 93–104. Ablex, Norwood, NJ, 1989.

[FLMM91] G. Fischer, A. C. Lemke, T. Mastaglio, and A. I. Morch. The role of critiquing in cooperative problem solving. *ACM Transaction on Information Systems*, 1991.

[FMM89] G. Fischer, R. McCall, and A. Morch. Janus: Integrating hypertext with a knowledge-based design environment. In *Proceedings of Hypertext '89*, pages 105–117, New York, 1989. ACM.

[FMO+94] Gerhard Fischer, Ray McCall, Jonathan Ostwald, Brent Reeves, and Frank Shipman. Seeding, evolutionary growth and reseeding: Supporting the incremental development of design environments. Proceedings of CHI'94, 1994.

[Gre92] Marc Green. Conceptions and misconceptions of knowledge aided design. In Marc Green, editor, *Knowledge Aided design*, volume 10 of *Knowledge-Based Systems*, chapter 1. Academic Press, 1992.

[HMS94] Knut Hinkelmann, Manfred Meyer, and Franz Schmalhofer. Knowledge-Base Evolution for Product and Production Planning. *AI Communications*, 7(2):98–113, June 1994.

[KH94] Otto Kühn and Björn Höfling. Conserving Corporate Knowledge for Crankshaft Design. In *Seventh International Conference on Industrial & Engineering Applications of Artifical Intelligence & Expert Systems (IEA/AIE'94), Austin, Texas*, pages 475–484. Gordon and Breach Science Publishers, May 1994. Also available as DFKI Research Report RR-94-08.

[KK94] Otto Kühn and Dirk Krechel. Validating DATALOG Design Rules Including Finite Domains and Intervals. Paper to be presented at KI'94 workshop: Entwicklung, Test und Wartung deklarativer KI-Programme (Verfahren und Werkzeuge), September 1994.

[LGP+90] Douglas B. Lenat, Ramanathan V. Guha, Karen Pittman, Dexter Pratt, and Mary Sheperd. Cyc: Toward programs with common sense. *Communications of the ACM*, 33(8):30–49, August 1990.

[Loh93] Georg Lohse. Methodical product development - experience of a practical study. In N.F.M. Roozenburg, editor, *Proceedings of the 9th International Conference on Engineering Design*, 1993.

[Min92] Sten Minör. Interacting with structure-oriented editors. *International Journal on Man-Machine Studies*, 37:399–418, 1992.

[NFF+91] Robert Neches, Richard Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator, and William R. Swartout. Enabling Technology for Knowledge Sharing. *AI MAGAZINE*, pages 36–56, Fall 1991.

[SM94] Frank M. Shipman and Raymond McCall. Supporting Knowledge-Base Evolution with Incremental Formalization. In *Proceedings of CHI'94*, 1994.

[Ste85] L. Steels. Second generation expert systems. *Fifth Generation Computing Systems*, pages 213–221, 1985.

[WF88] Terry Winograd and Fernando Flores. *Understanding computers and cognition.* Addison Wesley, Reading, MA, 1988.

[Whi91] Edgar A. Whitley. Two approaches to developing expert systems: A consideration of formal and semi-formal domains. *AI and Society*, 5:110–127, 1991.

[WO88] R. Whitaker and O. Östberg. Channeling knowledge: Expert systems as communications media. *AI and Society*, 2:197–208, 1988.