Lecture Notes in Artificial Intelligence 622

Subseries of Lecture Notes in Computer Science
Edited by J. Siekmann

Lecture Notes in Computer Science
Edited by G. Goos and J. Hartmanis

UBR UBR UBR UBR UBR
069008343590

F. Schmalhofer G. Strube
Th. Wetter (Eds.)

Contemporary Knowledge Engineering and Cognition

First Joint Workshop
Kaiserslautern, Germany, February 21-22, 1991
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

11|15296

00/SS 4800 - 622

Series Editor

Jörg Siekmann
University of Saarland
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, W-6600 Saarbrücken 11, FRG

Volume Editors

Franz Schmalhofer
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Erwin-Schrödinger-Straße, Postfach 2080, W-6750 Kaiserslautern, FRG

Gerhard Strube
Universität Freiburg, Institut für Informatik und Gesellschaft
Friedrichstr. 50, W-7800 Freiburg, FRG

Thomas Wetter
IBM Deutschland GmbH, Wissenschaftliches Zentrum IWBS
Wilckensstr. 1a, W-6900 Heidelberg, FRG

Univ.-Bibliothek
Regensburg

6 570 010

CR Subject Classification (1991): I.2.0, I.2.4, I.2.6, I.2.8

ISBN 3-540-55711-3 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-55711-3 Springer-Verlag New York Berlin Heidelberg

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1992

Printed in Germany

Typesetting: Camera ready by author/editor
Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
45/3140-543210 - Printed on acid-free paper

Foreword

The roots of this book can be traced back to a conversation I had with Gerhard Strube at the German Workshop on Artificial Intelligence (GWAI) in September 1989. As spokespersons of the Special Interest Groups (SIG) Cognition and Knowledge Engineering of the German Society for Informatics (GI) Gerhard and myself were wondering whether any knowledge engineering tools could be applied or analyzed in cognition research and what insights and methods of cognitive science might be relevant for knowledge engineers. To answer these and related questions we decided to have a common workshop organized by the two SIGs. At the next SIG meeting on knowledge engineering in April 1990 at Berlin, I asked Franz Schmalhofer and Thomas Wetter to organize such a workshop together with Gerhard. This joint workshop was then held February 21–22 at Kaiserslautern.

At the workshop, the first thing I learned was that the relationship between our two disciplines is not a simple import/export business. For instance I was told that repertory grids, the best automated knowledge elicitation technique of all, are not very popular with scientifically oriented psychologists. And imagine, knowledge engineers imported it blue-eyed! On the other hand, I would never bore and consequently nerve an expert with a repertory grid technique, even if some psychologist told us that enraged experts tend to answer more to the point.

But how should knowledge engineers, being too busy to become a semi-expert for each new application, keep up-to-date with cognitive science as well? Nor could we require cognitive scientists to become knowledge engineers! Well, we have to keep ourselves mutually informed about the hot spots, will say, problems, approaches, trends, or shifts of paradigm in each discipline. This is exactly what we did at our workshop.

- For instance, the last few years have witnessed a shift of paradigm in knowledge engineering. It was recognized that expertise cannot be simply extracted from the human expert and his books and mapped onto the machine. Neither is an expert's head a container of thousands or millions of rules. Second-generation knowledge engineering, as we might call it, is viewed as a creative process that engages knowledge engineers and experts in (re-)constructing knowledge and problem solving methods so that they can be made to run on a computer, resulting in an expert support system rather than an expert replacement system. While first-generation knowledge engineers might have been able to simply import methods from other disciplines to extract the knowledge, cognitive science is now becoming more important in the new paradigm. This subject came up quite a number of times.

- A more specific issue concerned the generic problem solving methods which are being adopted by more and more knowledge engineers. Are experts actually in command of such generic procedures which they suitably instantiate for their respective tasks? Or don't they distinguish domain-specific and generic knowledge at all? Another question addressed to cognitive scientists inquired their opinion on multimedia representations.

- As a second type of cooperation it was suggested that cognitive scientists could take the knowledge engineer's methods, tools, or expert systems back into their laboratories in order to experimentally determine their cognitive adequacy, whatever is meant with this term.
- A subject where both disciplines were already cooperating is that of cases, both as they arise during knowledge acquisition and as they are used for case-based reasoning. Questions tackled were: How do humans proceed from cases to more general rule-like knowledge? When do they reason by cases or by analogies, when do they use heuristics or first principles? How does case-based reasoning work, and how is it related to learning?

The workshop benefitted from international contributions from Canada, England, France, Switzerland, and the USA, demonstrating how knowledge engineering and cognitive science are interwoven between those countries. But to be quite honest with you, the (potential) reader of this book, I was not the only attendant of the workshop who was surprised by the wide gap between our two disciplines.

Then why did we write this book? Because by now we understand much better which questions we should ask and which we should better forget. And although Franz, Gerhard, and Thomas put lots of work and pain into organizing the workshop and editing the book (and this foreword), it still does not answer all the questions we raised. Reading this book will consequently not give you any final answers, but hopefully provide you with intriguing stimulations for producing your own answers. Those of you who are only interested in a quick import/export affair, need not go on reading. Our book is intended for persons who are really interested in the cognitive science aspects of knowledge engineering. But be warned: the articles reflect their authors' different backgrounds. And they assume a certain familiarity with central notions. For instance, you should have heard about KADS or MOPS.

The book is structured into three parts: The first one contrasts work in knowledge engineering with approaches from the side of the "soft sciences". The second part deals with case-based approaches in expert systems. Cognition research and the cognitive adequacy of expert systems are discussed in the third part.

My personal route through this book, which I do not want to conceal from you, deviates from this structure and is more oriented towards the course of the workshop:

Franz Schmalhofer sets off to explain the paradigmatic shift leading to a second generation of knowledge engineering approaches. He argues that the import/export attitude which sometimes emerged during the workshop must be replaced by interdisciplinary approaches.

How he personally experienced the shift of paradigm in his knowledge acquistion project is reported by *Marc Linster.* He sees the new task of cognitive scientists in helping to find an adequate modelling terminology and later in evaluating the resulting expert systems.

Gerhard Strube picks up a panel discussion which, according to the opinion of many participants, was the highlight of the workshop. It centered around the fuzzy notion of cognitively adequate expert systems. Everybody claims to build such systems – just like everybody claims to follow a model-based approach – but Gerhard elaborates at least three different readings of that notion. He argues why we should strive at building "strong cognitively adequate" systems, and thus imposes certain requirements on knowledge engineering paired with concrete advice on the first steps to be taken.

Four articles present different methodological views on knowledge engineering. Although I would not call them completely incompatible, they demonstrate how far the field is still from having a consistent view of itself.

- In their very detailed survey on psychological literature, *Brian Woodward, Mildred Shaw, and Brian Gaines* stress the cognitive processes going on while knowledge engineering.
- *Beate Schlenker and Thomas Wetter* view knowledge acquisition as an iterative process of scientific theory formation driven by falsification. They try to reformulate a scientific paradigm in order to make it applicable for knowledge engineering.
- *Dieter Fensel* argues that knowledge acquisition and qualitative social science have common goals, and suggests how to adopt techniques developed by the latter for knowledge engineering.
- *Rolf Pfeifer, Thomas Rothenfluh, Markus Stolze, and Felix Steiner* present the most concrete approach. They suggest how to match think-aloud protocols with generic problem solving models. Thus they partially answer one of the questions I raised above.

The next three articles report on experiences with actually employed knowledge acquisition systems. The tools developed by the three groups are candidates to be taken back to the laboratories of cognitive scientists.

- Their work on knowledge acquisition front-ends that are to completely replace the knowledge engineer drives *Frank Puppe and Ute Gappa* to pose two urgent questions to cognitive scientists, namely the ones I already mentioned before: How cognitively adequate are "canned" problem solving methods, and what about graphical knowledge representations?
- *Nigel Shadbolt* presents problems that arose in designing an integrated knowledge acquisition workbench in the ACKnowledge project. He discusses different types of users whose different needs have to be taken into account.
- *Geoffroy Dallemagne, Georg Klinker, David Marques, John McDermott, and David Tung* describe Spark, Burn, Firefighter, a knowledge-based software engineering tool. It helps application programmers with workplace analysis, selecting pieces to be automated and configuring these programs from available mechanisms.

The last group of articles is about cases, as they arise during knowledge acquisition and in case-based reasoning.

- *Klaus-Dieter Althoff* establishes the terminology and gives a survey of case-based approaches as compared to machine learning. His article should help to classify the following ones.

- In a short survey, *Sonja Branskat* gives the flavour of a tool she developed to support the knowledge engineer in gradually transforming cases as they appear in the real world, laden with context, to the formal and often decontextualized representations used by case-based reasoners.

- *Peter Reimann and Thomas Schult* report on experiments they conducted to find out how humans learn from examples in physics text books. In particular, they deal with the basic mechanisms involved in learning from cases in complex domains. Their results should carry over to knowledge engineers who typically are confronted with such situations.

- *Franz Schmalhofer, Christoph Globig, and Jörg Thoben* describe how they built a system implementing the generic problem solving method of skeletal plan refinement. They elicited cases to acquire the skeletal plans employed by their system. Their system is situated in the sense that new skeletal plans can be acquired during normal operation. They relied on the expert's experience, perception, and selected attention which enable him to identify the right cases as a basis for refinement.

- *Ralph Bergmann* goes on to present the explanation-based learning method used to automatically abstract cases into skeletal plans. They are partially based on common sense knowledge.

- *Michel Manago and Noel Conruyt* describe their extension of the ID3 induction algorithms to a frame-based knowledge representation language. They show that mechanical learning techniques can be considerably enhanced when the knowledge engineer imposes a suitable structure on the representation of cases. Their paper includes a one-page comparison between learning and case-based reasoning.

- From their cognitive science perspective, *Dietmar Janetzko and Gerhard Strube* compare case-based reasoning approaches with those using generic problem solving methods, coming up with suggestions of how to integrate both. By transferring ideas from cognitive science into the knowledge engineering terminology of the KADS methodology, their article builds a bridge between the two disciplines.

In his concluding remarks, *Thomas Wetter* does a tremendous job in bringing together many controversial arguments we encountered at the workshop and presents, if not a final word, a comparative assessment.

Now you are asked! What is your opinion about this book, and more importantly, about the questions it raises, and the tentative answers it proposes? Please let us know, possibly using the forum of our two special interest groups in the GI. Hopefully, we thus get loaded with a lot of dynamite for a successor workshop.

St. Augustin, May 1992 Angi Voss

Table of Contents

Part 1: Knowledge Engineering and Cognition in Comparison

Part 2: Case-Based Approaches to the Development of Expert Systems

Part 3: Cognitive Adequacy of Expert Systems

Concluding Remarks

Part 1:

Knowledge Engineering and Cognition in Comparison

Relations between Knowledge Engineering and Cognitive Science: From Import/Export to a Truly Interdisciplinary Knowledge Acquisition Enterprise

Franz Schmalhofer

German Research Center for Artificial Intelligence
University Bldg 57
Erwin-Schroedinger Str.
W-6750 Kaiserslautern

email: schmalho@informatik.uni-kl.de

1. Introduction

Knowledge Engineering is generally known as the field that is responsible for the analysis and design of expert systems and is thus concerned with representing and implementing the expertise of a chosen application domain in a computer system. *Research on cognition* or *cognitive science*, on the other hand, is performed as a basic science, mostly within the disciplines of artificial intelligence, psychology and linguistics. It investigates the mental states and processes of humans by modelling them with a computer system and combining analytic and empirical viewpoints.

Early on, *knowledge acquisition* was known as the activity of making explicit the human knowledge that is relevant for performing a task, so that it can be represented and become operational in an expert system. Knowledge acquisition and the field of knowledge engineering are consequently closely related to human cognition, which is studied in cognitive science. The specific relationship between knowledge engineering and cognitive science has changed over the years and therefore needs to be reconsidered in future expert system developments.

Although knowledge acquisition activities are at most twenty years old, there is already a respectable history with noticeable successes and some initially disappointing failures to be looked back upon. Actually, more progress was made by the analysis of the failures than with the short term successes.

2. Early Knowledge Acquisition

Early knowledge acquisition was supported by knowledge acquisition systems such as TEIRESIAS (Davis, 1978) which were designed as front-ends for existing expert systems (i.e. MYCIN) and knowledge engineers viewed knowledge acquisition as the process of transferring knowledge from a human expert to a program.

After it was observed that humans can hardly express their knowledge in distinct chunks, so that each chunk can somehow be transformed into a rule (or some other syntactically defined structure), which would then do "the right thing" in combination with an already existing expert system shell (e.g. EMYCIN), knowledge acquisition became recognized as "a bottleneck in the construction of expert systems" (Buchanan et al., 1983, p.129): Not the development of the original expert system (shell), but the acquisition of the domain specific rules for that shell turned out to be the tough part in building a fully functional system.

Since some of an expert's relevant knowledge is tacit or implicit (Schachter, 1987), experts cannot directly verbalize all relevant rules. Knowledge engineers therefore concluded that some special psychological method would be necessary in order to acquire the desired knowledge in the form that was needed for rule-based (or other) systems.

For mining the expert's deeply hidden knowledge, various data collection and data analysis methods were subsequently imported from psychology into knowledge engineering (Hoffman, 1989) and respective computer tools were built (Boose & Gaines, 1989). Some of these tools were quite successful in constructing rules for relatively small application domains.

This early knowledge acquisition period was determined by the knowledge engineers who emphasized the full implementation of small scale knowledge acquisition tools over a common and cognitively founded design rationale for the expert system and its knowledge acquisition components.

Knowledge engineering and cognitive science followed two separate research agendas during this period and those slots of the research agenda which were difficult to fill from inside the field of knowledge engineering were assigned to the field of cognition (e.g. supplying the rules for some rule interpreter). The cooperation of the two disciplines thus consisted of quickly importing selected research items (vague ideas, theoretical frameworks or methods) from the other discipline in a relatively unreflected way. The use of repertory grids (Kelley, 1955) in knowledge acquisition is probably a good example of such a type of import/export relation between knowledge engineering and psychology which is one of the disciplines contributing to cognitive science.

While the problem of transferring human expertise into computer programs was (at least partially) solved, it was discovered that the knowledge acquisition problem had been incorrectly stated, right from the beginning. One piece of evidence for that is: Even after the successful construction of an operational rule-base, the meaning of the individual rules remained a mystery (Clancey, 1983; p. 241). The maintenance of larger expert systems was consequently impossible. Since such systems were found to have several other disturbing deficiencies (e.g. brittleness), the definition of knowledge acquisition needed to be changed.

3. Knowledge Acquisition as a Truly Interdisciplinary Task

One of the necessary changes in the definition of knowledge acquisition is already well established: Knowledge acquisition is now understood as a modelling activity where models of the conventional expertise in an application domain and models of the target expert system are to be developed (Breuker & Wielinga, 1989). Unfortunately, the cognitive science issues which have become important for successful knowledge engineering are hardly discussed in this context. The nature of different types of models and their relationship to one another needs to be determined: How should the models of existing or future artifacts (e.g. expert systems) be related to models of natural systems (e. g. human cognition)? Can they be structurally similar or even identical or do they need to be quite different? Since knowledge engineering deals with such artifacts and cognitive science with the modelling of human cognition, the two fields need to intensively cooperate to successfully address the question of the relation between the models. Newell's (1990) assertion of describing human intelligence as a symbol system is equally important for this discussion as Searle's (1981) views about intrinsic intentionality and human commitment.

Another question, where the expertise of cognitive scientists needs to be respected by knowledge engineers, is: What kind of mental models (Norman, 1983) do humans develop about expert systems? How are the mental models of a domain expert, of a knowledge engineer and of the future users of some target system related to one another? What kind of mental models are users capable of and willing to maintain and how can the mental models about different systems be related to one another? How can expert systems play the role of fancy representations, which allow the communication of knowledge between the domain expert and knowledge engineer on the one side and the users of the system on the other side?

Knowledge engineers must finally learn to appreciate that expert systems have to function in the real world in order to become a success in business. Unlike the microworlds, in which knowledge engineers liked to test their rapid prototypes, the real world refuses to be (correctly) represented once and for all time by some formal specification. The future application requirements can consequently only be partially predicted. This basic fact is often ignored. Expert systems must be developed so that new types of inputs can be processed at the time when the system is applied (Schmalhofer & Thoben, 1992). In other words, expert systems must allow for situated applications (Clancey, 1991) and that means that they must be end-user modifiable (Fischer & Girgensohn, 1990). These challenging demands can only be successfully met, when the engineering science and the cognitive and social sciences cooperate with the mutual respect for one another, which is required to make an interdisciplinary enterprise a success.

References

Boose, J.H. & Gaines, B.R. (1989). Knowledge Acquisition of Knowledge-Based Systems: Notes on the State-of-the-Art. Machine Learning, 4, 377-394.

Breuker, J. & Wielinga, B. (1989). Models of expertise in knowledge acquisition. In Guida, G. & Tasso, C. (Eds.), Topics in expert system design, methodologies and tools (pp. 265 - 295). Amsterdam, Netherlands: North Holland.

Buchanan, B.G., Barstow, D., Bechtal, R., Bennett, J., Clancey, W., Kulikowski, C., Mitchell, T. & Waterman, D.A. (1983). Constructing an Expert System. in: Hayes-Roth, F., Waterman, D. & Lenat, D.B. (eds.) Building Expert Systems. Reading Massachussetts: Addison-Wesley Publishing Company, Inc. pp.127-167.

Clancey, W.J. (1983). The Epistemology of a Rule-Based Expert System - a Framework for Explanation. Artificial Intelligence, 20, pp. 215-251.

Clancey, W.J. (1991). Situated Cognition: Stepping out of Representational Flatland. AI-Communications, 4, 2/3, 109-112.

Davis, R. (1978) Knowledge Acquisition in rule-based systems - knowledge about representations as a basis for system construction and maintenance. In: Waterman, D.A. & Hayes-Roth, F.(eds) Pattern-Directed Inference Systems. Academic Press, New York.

Fischer, G. & Girgensohn, A. (1990). End-User Modifiability in Design Environments, Human Factors in Computing Systems, CHI'90, Conference Proceedings (Seattle, WA) ACM, New York (April 1990), pp. 183-191.

Hoffman,R. A Survey of Methods for Eliciting Knowledge of Experts. In: *SIGART Newsletter 108*,19-27, 1989.

Kelley, G.A. (1955). The Psychology of Personal Constructs. New York: Norton.

Newell, A. (1990). Unified Theories of Cognition. Cambridge, Massachusets: Harvard University Press.

Searle, J.R. (1981). Minds, Brains and Programs. In Haugeland, J. (Ed) Mind Design, Cambridge Massachussets: London, 282-306.

Schachter, D.L. (1987) Implicit memory: history and current status. Journal of Experimental Psychology: Learning, Memory and Cognition, 13, 501-518.

Schmalhofer, F. & Thoben, J. (1992). The model-based construction of a case oriented expert system. AI-Communications, 5, 1,3-18.

Making Application Programming More Worthwhile

Geoffroy Dallemagne, Georg Klinker, David Marques,
John McDermott, David Tung

Digital Equipment Corporation
111 Locke Drive
Marlboro, MA. 01752

e-mail:dallemagne@airg.dec.com

Abstract. We are designing and implementing an integrated programming framework to assist application program developers with the automation of a broad range of tasks. Our framework encourages the following activities:

- analyzing the situation in which automation is going to be introduced,
- capturing the results of the analysis as a model,
- building a workflow application program to manage all of the activities,
- configuring small collections of reuseable mechanisms to perform or assist with some of the activities,
- customizing the configured mechanisms thus generating one or more application programs,
- refining the resulting application programs on the basis of user reactions to them.

1 Introduction

Our research problem is how to make application programming more worthwhile. Our initial explorations focused on making it easier -- ie, making it more worthwhile by allowing less experienced people to create programs more quickly [see Klinker 90 and McDermott 90]. A number of researchers have focused on the issue of how to make application programming easier [see Krueger 89]. The efforts most closely related to our own include [Bennett 85, Birmingham 88, Breuker 89, Chandra 83, Clancey 83, Davis 79, Eshelman 88, Klinker 88, Marcus 88, Musen 91 and Yost 89]. Each of these research efforts has identified one or more problem-solving methods and shown how the methods can be exploited in the development of application programs that use them.

Recently it has become clear to us that our goal of making application programming easier was under-constrained and needed to be married to the goal of making application programming more effective. There is substantial evidence that many application programs that are developed are not used anywhere nearly as extensively as their developers anticipated [see, for example, Leonard-Barton 87]. One significant factor in this under-utilization appears to be the mismatch that often exists between the functionality provided by the application program and the functionality that would actually be useful in its intended situation. Insight into why these mismatches are so pervasive and ideas for reducing their magnitude are provided by research in situated action [Suchman 87, Wenger 90].

This paper describes a framework for identifying homes for effective application programs and for making the construction of those programs easier. A high level overview of our framework is presented in section 2. Sections 3 through 7 provide more details. The framework includes a place for workplace analysis (section 3), using the results of the analysis to model the workplace (section 4), generating a workflow management application program (section 5), selecting and configuring reusable mechanisms that can assist with some of the work (section 6), customizing the mechanisms for the workplace thus generating one or more application programs (section 7).

2 An Application Programming Framework

Our framework (graphically depicted in Figure 1) assumes that some group wants computer assistance with one or more of the tasks they perform. The framework is structured around a small set of methods and tools -- a few of which we have developed and the rest borrowed from others. Our tools, which we call Spark, Burn, and FireFighter, help with the design, implementation, and refinement of application programs. They take leverage from a workplace analysis methodology and a workflow manager; for the purposes of this paper, we will use BAM-2 (Business Architecture Methodology) to show the role a workplace analysis methodology plays[1] and we will use EDGE (an event driven workflow controler) to show the role a workflow manager plays, but other workplace analysis methods and other workflow managers could serve just as well within our framework.

The framework is designed to help a group of people who want automated assistance with some task. We refer to a group with shared responsibility for a task as a community of practice. Our framework guides the community through a series of steps that will result in their more easily creating effective application programs. The first step in our framework is workplace analysis. A facilitator, proficient in some workplace analysis methodology (eg, BAM-2), helps the group identify the processes that are actually at play in their workplace, the activities that comprise those processes, the agents who perform the activities, the resources consumed by the activities, and the products produced by the activities (see 1 in Figure 1). The second step in our framework is capturing the results of the workplace analysis in a model. Burn is used by one or more of the people in the group to record these results. Burn provides a knowledge acquisition tool designed to capture knowledge of the sort needed by a workflow manager -- ie, knowledge of what work is done under what circumstances, of the processes that support that work, of the agents who perform the work, and of the pieces of work that flow among the workers (see 2 in Figure 1). Once these two tasks (the analysis task and the modeling task) have been performed, Burn uses the model to generate a workflow management appli-

[1] The creator of the BAM-2 methodology is Jane Roy; Digital Equipment Corporation sells BAMming as a service.

cation program (see 3 in Figure 1) which helps the workers manually perform their work by tracking who is to do what and when and by making resources available to the appropriate worker at the appropriate time (see 7 in Figure 1).

Another step in our framework supports is identifying opportunities for automation. Spark is used by one or more of the people in the group (the community of practice using our framework) to do this identification. Spark has a library of mechanisms (reusable software structures) that can be used in composing application programs; associated with each mechanism are descriptions of the kinds of situations in which that mechanism can provide assistance. The trick is to make contact between the activity descriptions that Spark uses to index into its mechanisms and the activity descriptions in the workplace model. Spark tries to help its users make contact between these two sets of descriptions (see 4 in Figure 1). If an automation possibility is identified and if Spark has an appropriate set of mechanisms in its library, it passes Burn a configuration of mechanisms that it believes, on the basis of its interactions with the users, will allow an effective application program to be generated (see 5 in Figure 1). Burn is used, as it was in the second task described above, to elicit relevant details about the workplace. In this case, it provides a knowledge acquisition tool for each of the mechanisms selected by Spark; each knowledge acquisition tool interacts with some of the people in the group who perform the activities to be automated and elicits from them the domain knowledge that its associated mechanism requires. The outcome of Burn's efforts is a program integrated into the workflow management application program generated previously that automates a previously manual activity (see 6 in Figure 1).

The remaining step in our framework is that of refining and maintaining the application programs that have been generated. As a consequence of using the framework described in this paper, the group of workers will have created a workflow management application program which helps them keep track of their work, and they will have created activity-specific application programs within the context defined by that workflow application. FireFighter's role is to assist with the refinement and maintenance of these application programs (including the workflow management application) to insure that the programs are and remain effective (ie, are modified to reflect any changes in the work the group does or in the processes supporting the work). Since both Spark and Burn deliberately use simple heuristics to configure and instantiate the mechanisms, it is to be expected that several versions of each application program will have to be generated before getting to one that satisfies the users. There are several reasons why a mismatch between an application program and a task is likely: (1) Burn may not elicit enough knowledge, (2) the mechanism configuration may not be appropriate for the activity, (3) the task analysis may have been incorrect. FireFighter assists the users in determining and resolving these problems. It then re-invokes Burn to acquire more knowledge or re-invokes Spark to modify the mechanism configuration (see 8 in Figure 1).

The following sections provide an example of the use of our framework in its current state. To understand the sections, you need the following information about the task that our examples focus on: A research group inside a large company is in charge of sponsoring research efforts in various universities; this sponsoring is done through grants and involves several activities including the selection of what researchers to sponsor and a fairly elaborate process, administered by Tiera Washington, for actually presenting the grants. The research group decided to see whether automation could smooth the process and thus free up time and the minds of its researchers.

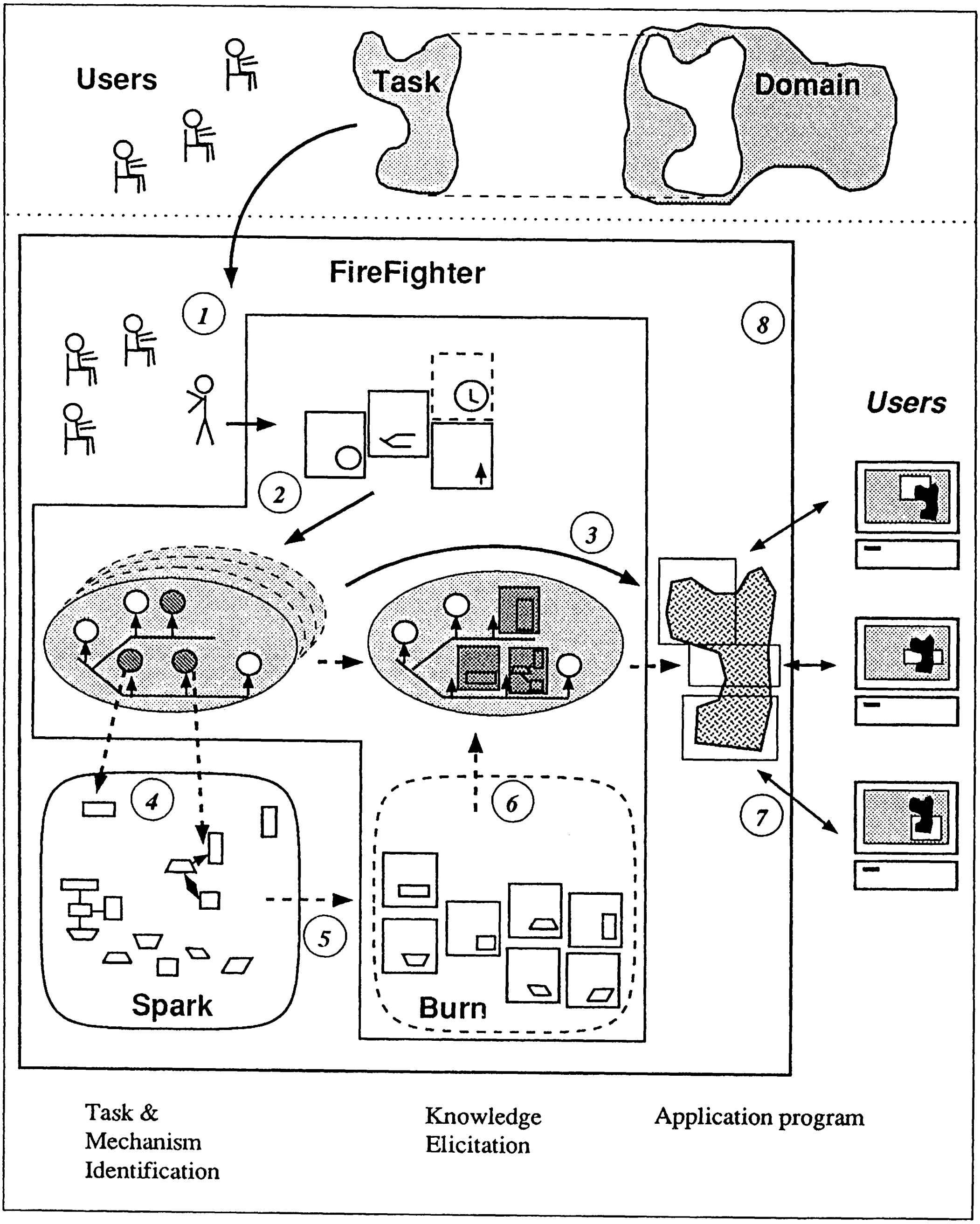


Figure 1 An Application Programming Framework

3 A Methodology That Allows Workers to Describe Their Work

The task of processing a grant is comprised of many activities. The problems with automating this task are first, understanding what work is actually done and what ought to be done, and second, since not all of the activities comprising the task would benefit from or are suitable for automation, determining what to automate and how to package the automation. As the work is done in the real world (with all its incoherence) within a real organization by real people trying to work together, understanding what activities comprise the work is at least a big task. But in addition to being big, it is slippery in the sense that no collection of characterizations ever quite do the activities or their interrelationships justice.

Our framework enjoins the use of some kind of workplace analysis methodology to initiate the application programming enterprise. The methodology that we use in our own application programming experiments is called BAM-2. The following characteristics make BAM particularly attractive as a practical methodology:

- its result is a description of work produced by the collection of people (a community of practice) actually performing the work;
- rather than being an essentially unconstrained fabrication, the description produced is constrained to take into account most, if not all, of what is consumed and produced while the task is being performed;
- a complete analysis of a task takes only a few days to do.

Figure 2 depicts the seven steps of the BAM methodology. The methodology assumes that the people involved in performing a task or set of tasks have gathered together in the same room with a BAM facilitator. In the first step (see Figure 2.1), the facilitator asks the workers to use verb-noun phrases to begin to identify the work they do. The participants are encouraged not to edit or filter what they say, but rather to be as spontaneous as possible; the idea is to break away from preconceptions and vague abstractions as much as possible. When out example task of sponsoring external research was BAMmed, the result of the first step was 66 verb-noun phrases.

The next two steps group the pieces of work identified in the first step; pieces of work that deal with the same kind of thing are grouped (see Figure 2.2 and Figure 2.3). The purpose of these steps is to provide the participants with a basis for describing activities in terms of what they consume and produce. In our example, the work involved in sponsoring external research was grouped into four subtasks: define a research program, issue a grant, finance the grant, and follow the relationship. Defining a research program deals with whom to support and how much support to provide; issuing a grant deals with creating a grant letter, getting it approved, and getting a check to send the researcher; financing the grant deals with handling the financial transactions required to get the check generated; and following the relationship deals with monitoring the research, making suggestions, and figuring out how to exploit the results. This grouping process forces participants to argue through each other's concepts and thus results in a common understanding of the task.

At this point BAM focuses on what each subtask consumes and produces. The facilitator asks the participants to identify all of the products produced by each subtask and then to identify all of the resources consumed (see Figure 2.4). This step draws attention to the objects manipulated within a subtask (eg, grant letter, check voucher) and also draws attention to the stages each of those objects goes through (eg, a grant letter template, an unsigned grant letter, a signed grant letter). Then the participants are asked to identify, for each subtask, the customer for each product and the producer of each resource. A subtask typically consumes some of its own products (eg, issue grant consumes an unsigned grant letter to produce a signed one), it typically produces products for and consumes products of other subtasks within the broader task (eg, issue grant produces a check voucher request which is consumed by the finance grant subtask), and it typically produces products for and consumes products of activities outside the task being BAMmed (see Figure 2.5). Part of the role of this step is to uncover inadequacies in the emerging task description. A product that goes nowhere is unneeded work; a resource that comes from nowhere is a bottleneck. The participants collectively decide what is wrong with their emerging picture.

The sixth step in the BAM process creates an event diagram for each subtask (see Figure 2.6). The facilitator helps the participants interleave the products and resources by asking them to identify the initial triggering event (the resource that starts it all) and the end product (the product that justifies the existence of the subtask). Then the participants work back from the end product, insuring that each product is preceded by the resources consumed in its creation, and forward from the triggering resource, insuring that each resource is succeeded by the products produced by its consumption. If not all of the products and resources of a subtask show up in the main stream of the event diagram, one or more additional interleavings are created. These secondary interleavings either correspond to a support function or to the discovery of an unnoticed subtask in which case the BAM facilitator returns to step three (see Figure 2.3) and helps the participants further refine their picture.

When one or more event diagrams have been created for each subtask, each is converted to a process diagram by noticing the product/resource boundaries in the event diagram. The idea is that in the workplace being BAMmed, these boundaries demark units of work that "make sense" in that workplace because each of these pieces of work (each function) has been defined in terms of the way the resources in that workplace present themselves (see Figure 2.7).

4 Capturing the Results of the Workplace Analysis

Given that a method like BAM can assist in creating a grounded understanding of some task in some workplace, our next issues are how to capture that understanding and then exploit it to provide useful automation. To capture the understanding, we provide in Burn a knowledge acquisition tool that allows all the information uncovered during the BAMming to be stored as a coherent collection of non-ephemeral declarative structures (ie, the different functions the work decomposes into, the community of practice and its agents, the resources consumed and the products produced are modeled). It is important to notice that, at this point, no information about how any piece of work is performed is available. Thus the knowledge acquisition tool is prepared to be told only about the "what" "who" "when" and "why" of the task, leaving the "how" for later. As Figure 3 shows, the tool presents three interfaces to the user: Function/Activity, Organization/Agent, and Data; (a fourth interface, one which will allow users to enter information about the duration of activities, is not yet implemented). We will illustrate the use of this tool for the issue grant subtask.

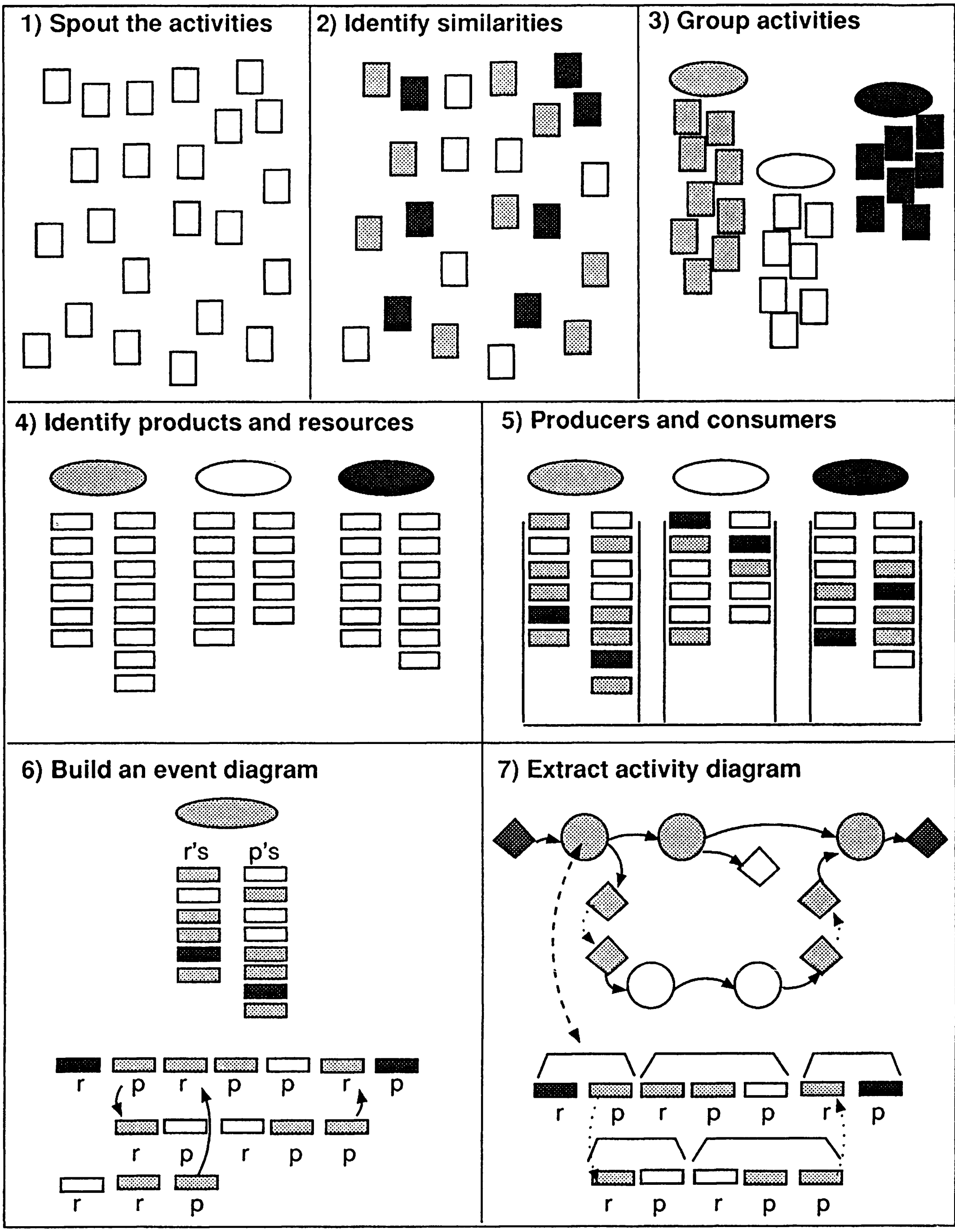

1) Spout the activities
2) Identify similarities
3) Group activities
4) Identify products and resources
5) Producers and consumers
6) Build an event diagram
r's
p's
r
p
r
p
p
r
p
r
p
r
p
p
r
r
p
7) Extract activity diagram
r
p
r
p
p
r
p
r
p
r
p
p

Figure 2 The BAM Methodology

With the Function/Activity interface, the user creates a function view for each subtask identified in step 3 of the BAMming. Each view is constructed with a simple graphical editor that presents basic building blocks on the left and a drawing area on the right. For the Function/Activity interface in Figure 3.1, the building blocks are: Activity (a circle), Flow (an arrow) and Subtask Port (a diamond); subtask ports allow the current view to connect, via resource links, to other subtasks. The user describes a subtask by selecting a building block and creating one or more instances of it in the drawing area; the user can describe the functions that comprise a subtask (identified in step 7), can identify other subtasks (identified in steps 5 and 6) that either produce products for or consume products of the subtask being described, and can indicate which functions are directly connected to one another or to functions in other subtasks (also identified in steps 5 and 6).

With the Organization/Agent interface, the user creates a view for each organization or community of practice; the building blocks for these views are Agent, Group, Solid Line, and Dotted Line (see Figure 3.2). Here the user inputs the knowledge about the players (identified in steps 5 and 7), the user can also identify the formal relationships among the players. To associate players with functions, the user selects an agent in the organization view and then selects, in the function view, whatever functions that player is involved with.

With the Data interface, the user creates a view that defines the products and resources produced and consumed by the subtask; the building blocks of this view are Object, Attribute, and Has-a (see Figure 3.3). The first thing the user does when creating a data view is to create one instance of each resource and product type (identified in step 4). The user can then indicate what objects are produced and consumed by what functions by selecting an object in the data view and then selecting, in the function view, whatever flows that object rides.

The model that is being created using this knowledge acquisition tool contains all of the information required to generate a workflow management application program that will run on EDGE. To automatically generate such a program, the user goes to the function view and indicates which functions are to be included; one (and perhaps the only) reason for excluding a function or a subtask from the purview of the workflow manager is if the people responsible for some of the functions do not have access to workstations connected to an EDGE server. Before generating the application program, the knowledge acquisition tool checks the model to insure that no essential information is missing (eg, to insure that an agent has been associated with each function). The knowledge acquisition tool also generates a workflow management window for the workstations of each of the people involved.

5 Workflow Management Functionality

Together, the three views discussed in the previous section allow a user to enter information that is required for any workflow management assistance. If someone enters information of the sort we described, then based on that information, Burn's workflow knowledge acquisition tool generates a program that provides workflow assistance from a function/activity perspective. In other words, the program keeps each person in the group apprised of what activities are currently in his or her work queue (see Figure 4 for an example of the kind of information each person is provided with). Each time one of the members of the community of practice selects a

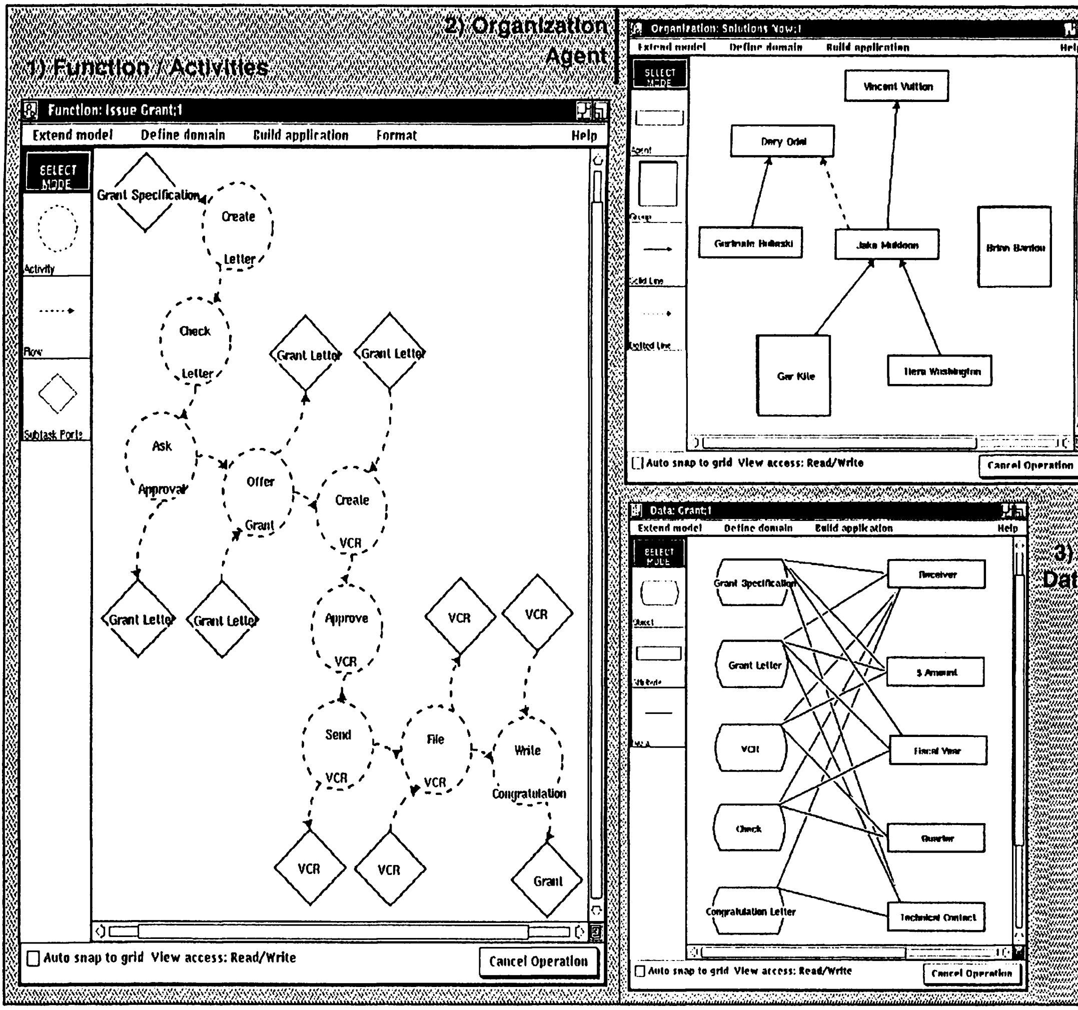


Figure 3 A Knowledge Acquisition Tool for the BAM Methodology

piece of work and tells the workflow manager that the work has been completed (by clicking on the "Work Done" button), a message is sent to the person responsible for the next activity. For example, if Tiera selects

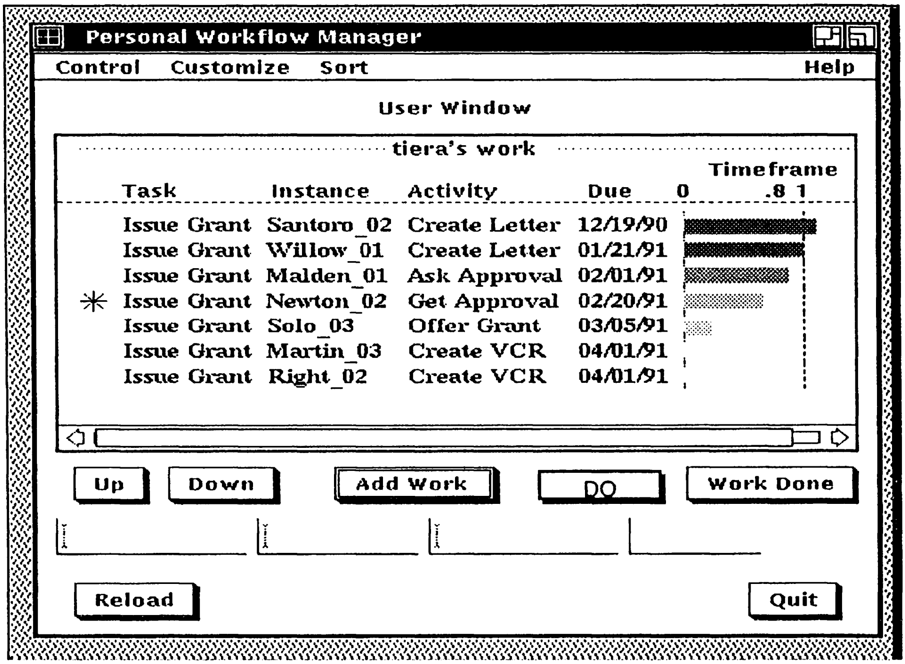
Issue Grant Martin_03 Create VCR 04/01/91

and then clicks on "Work Done", the following message would appear on Jake's screen (since the Approve VCR activity is always done after the Create VCR activity and Jake is responsible for doing Approve VCR).

Issue Grant Martin_03 Approve VCR 04/03/91

Assistance from a function/activity perspective is confined to reminders. That is, the people who will actually perform the work are informed when an activity instance is in their queue, but they are not given any assistance beyond that. However, substantial additional assistance can be provided if assistance from a data perspective is coupled with assistance from a function/activity perspective. If the workflow manager can deal with data instances as well as with activity instances, then not only can people be informed of what work is in their queue, but they can also be provided with pointers to the data that they will operate on. For example, with respect to the Approve VCR activity, Jake could be told where to find the VCR for Martin.

Finally, assistance from the organization/agent perspective can be coupled with the assistance from the other two perspectives. Agents are the repositories of the knowledge needed to perform the activities. If an agent's understanding of how to perform some activity is put into an application program, then that program becomes an instance of the agent. This allows the workflow manager to direct some of the work to programs instead of to people. For example, with respect to the Create VCR activity, Tiera, instead of having to display the specification for some grant and then invoking her favorite editor on the template she instantiates when preparing a VCR, could instead rely on the workflow manager to display the grant specification and invoke her editor for her on a blank VCR form. This is detailed in the following sections.



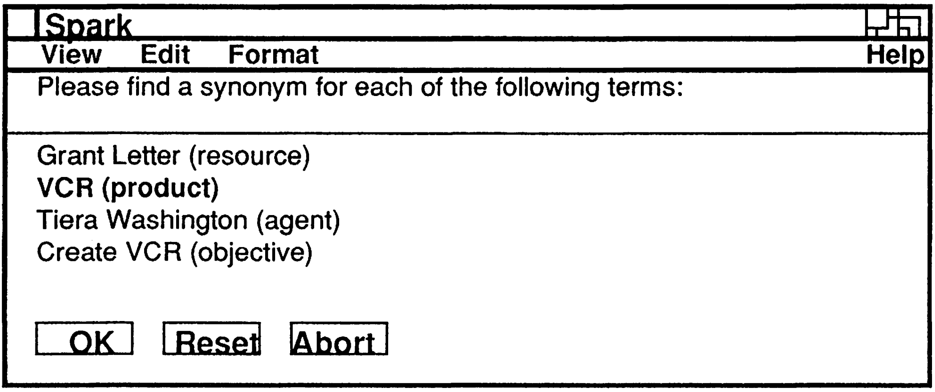
	Task	Instance	Activity	Due	Timeframe 0 .8 1
	Issue Grant	Santoro_02	Create Letter	12/19/90	
	Issue Grant	Willow_01	Create Letter	01/21/91	
	Issue Grant	Malden_01	Ask Approval	02/01/91	
✳	Issue Grant	Newton_02	Get Approval	02/20/91	
	Issue Grant	Solo_03	Offer Grant	03/05/91	
	Issue Grant	Martin_03	Create VCR	04/01/91	
	Issue Grant	Right_02	Create VCR	04/01/91	

Figure 4 User interface for each Agent

6 Identifying Automation Opportunities

As indicated in the previous section, the task activities described in the workflow diagram of Figure 3 represent possible opportunities for automation. If the user decides to explore the feasibility of automating a task activity, he or she mouses on that activity and selects the "automate" option from the popup menu. In our example, the user chooses to automate the "Create VCR" activity by clicking on that activity. This invokes Spark's browsing capability which assists in identifying whether a given task activity has been previously automated in the context of another task analysis. That is, Spark assists the user in identifying activities in its library that are similar to the current one. The idea is that to the extent two task activities are similar, the mechanisms implementing them are the same and can be reused for the task at hand.

Two activities are considered similar if they consume the same type of resources, produce the same type of products, accomplish the same objective, and are performed by people with similar competencies. Two resources, products, objectives or agents are similar if their labels are synonyms. Figure 5 identifies the resource, product, agent, and objective of the "Create VCR" activity. The user must now relate these new terms to the terms in Spark's vocabulary. That vocabulary was defined in the process of previously performed task analyses (whenever a new term is defined, it is added to Spark's vocabulary). In Figure 5 the user indicates that he wants to identify Spark's term for "VCR".



Please find a synonym for each of the following terms:

Grant Letter (resource)
VCR (product)
Tiera Washington (agent)
Create VCR (objective)

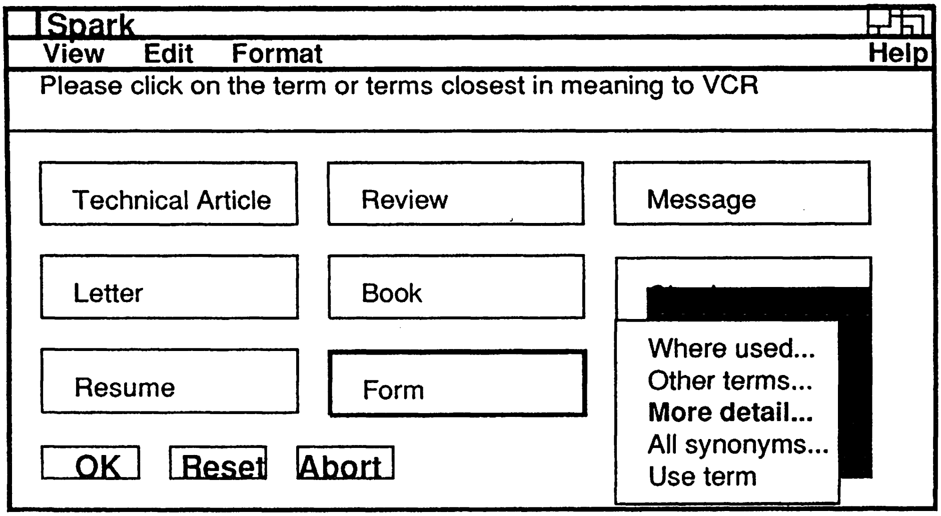
Figure 5 Terms defining the "Create VCR" activity

Spark uses simple heuristics to suggest a subset of its vocabulary to the user as candidate synonyms. First, Spark's vocabulary is divided into three classes: resource/product, agent, and objective. Only the terms from the relevant class are displayed. In our example, resource/product is the relevant class. In reducing the set of relevant terms further, Spark takes into account the terms that are already identified for the activity in question. For example, Spark's synonym for "Tiera Washington" is "Administrative Secretary". This reduces the set of candidate synonyms for "VCR" to the terms denoting a resource consumed by an activity performed by an "Administrative Secretary". A third heuristic for reducing the set of relevant terms is to consider the terms defined for the subtask. In our case, the "Issue Grant" subtask produces a "Congratulation Letter" defined as a "Letter". This heuristic reduces the set of candidate synonyms to the ones denoting the products of activities that -- among other things -- produce letters.

Figure 6 demonstrates the basic vocabulary that Spark considers relevant for the "VCR" resource after applying the above heuristics.

If the user does not see a synonym in the set of terms provided by Spark, Spark can help the user by

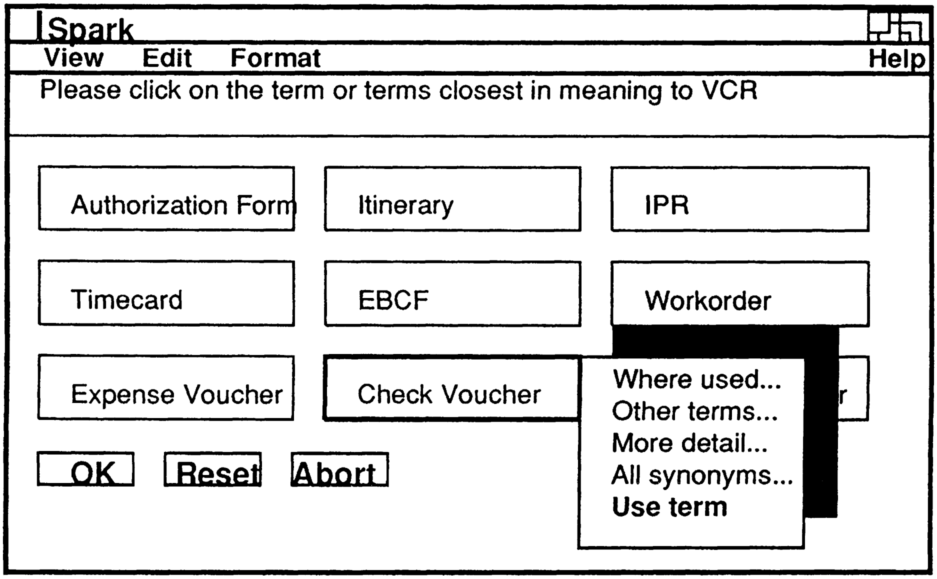
- demonstrating a candidate term's use within the context of other task analyses,
- displaying more general related terms,
- displaying more specific related terms,
- displaying all synoyms.



Please click on the term or terms closest in meaning to VCR

Figure 6 Spark's suggested synonyms for "VCR"

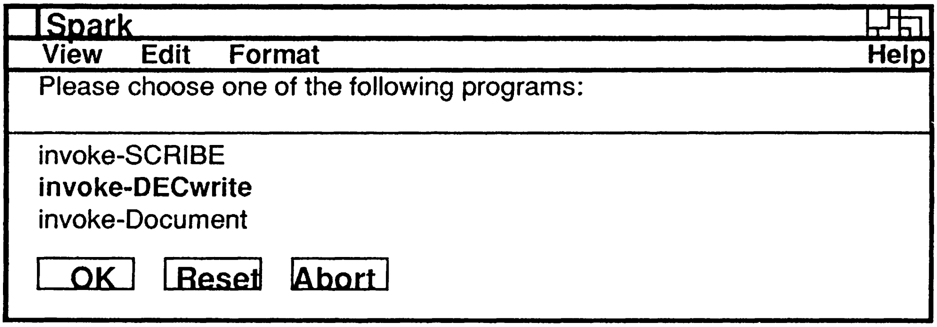
The user identifies "Check Voucher" as a synonym of "VCR". This is shown in Figure 7. He then identifies synonyms for the other terms in Figure 5.



Please click on the term or terms closest in meaning to VCR

Figure 7 Shared Resource Vocabulary detailing "Form"

Once all terms from the new task are identified, Spark checks whether a mechanism configuration has been associated with those terms (by the software engineer who created the mechanism). In our example, other activities similar to "Create VCR" have been automated before and their mechanisms can be reused. Three potentially relevant mechanism configurations are shown in Figure 8.



Please choose one of the following programs:

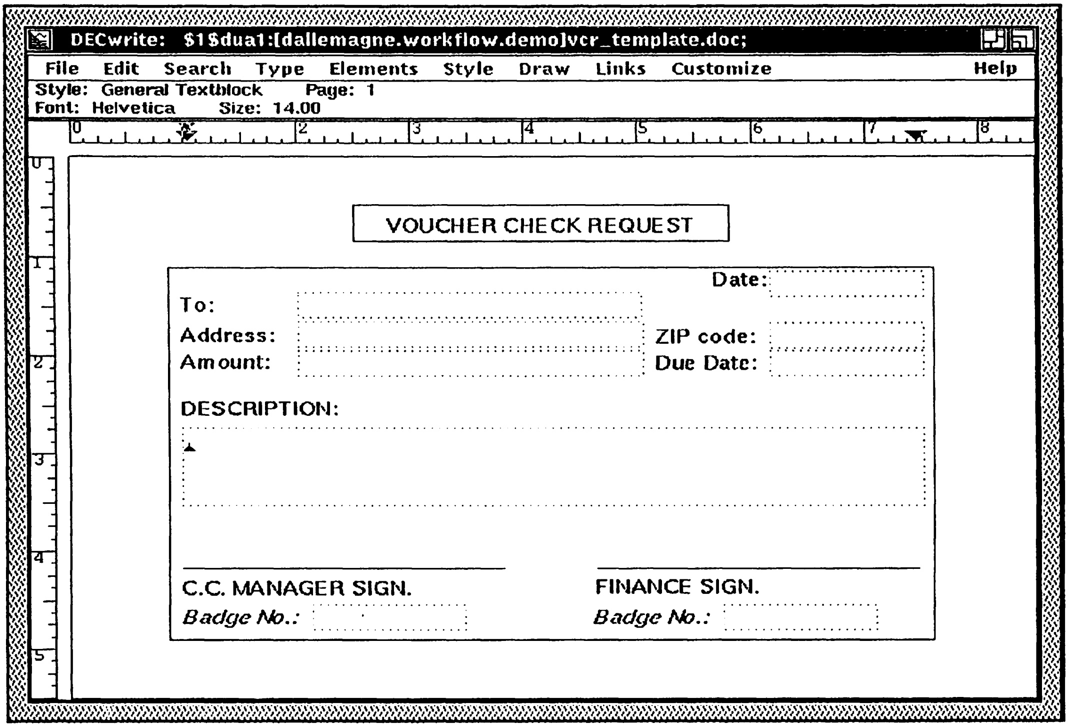
invoke-SCRIBE
invoke-DECwrite
invoke-Document

Figure 8 Relevant Mechanisms

Each of the configurations can assist an "Administrative Secretary" in accomplishing the "Create Form" objective. The resources are a "Form Specification" and a "Form Template" and the product is a filled out "Form". Spark now asks the user to select one of the three configurations. The user indicates the "invoke-DECwrite" configuration since DECwrite is Tiera's favorite editor. If Spark had not found a mechanism, that would have indicated that the automation opportunity might not be much of an opportunity; in order for the activity to be automated, a software engineer would need to design one or more new mechanisms from the specifications provided by the task model.

7 Tailoring Application Programs to the Workplace

When the user selects "invoke-DECwrite", Spark informs Burn that it is time for the user to interact with the knowledge acquisition tool associated with it. Burn activates the knowledge acquisition tool which asks the user to type a template for "VCR"; that is, it asks the user, in this case Tiera Washington, to type in the text that will be common to all VCRs (see Figure 9). For the "Create VCR" activity in our example situation, this is all the knowledge that needs to be supplied in advance of the task. Now when the workflow manager window informs Tiera that she should create a VCR, Tiera clicks the "Do" button, and two windows open before her eyes: one displays the specification for the grant she is preparing (ie, recipient, amount of grant, etc) and the other is a DECwrite window containing a copy of the template she had previously created. She then fills in that blank form and saves it; that instance of the VCR flows with the subtask instance from that point on.



DECwrite: $1$dua1:[dallemagne.workflow.demo]vcr_template.doc;

Style: General Textblock Page: 1
Font: Helvetica Size: 14.00

VOUCHER CHECK REQUEST

Date:

To:

Address:

ZIP code:

Amount:

Due Date:

DESCRIPTION:

C.C. MANAGER SIGN.

Badge No.:

FINANCE SIGN.

Badge No.:

Figure 9 VCR template

The mechanisms that we have used to date in the context of a workflow manager are few and simple. However, we are now focused on integrating work we have done on more interesting mechanisms and knowledge acquisition tools [see Marques 91] with our work on workflow managers.

8 Conclusion

To explore the value of our proposed application programming framework, we used the BAM-2 methodology to analyze a small, but real, task in our own workplace. We then used Spark and Burn, two tools we are developing, to introduce additional automation into that workplace. We interacted with one of Burn's knowledge acquisition tools to create a model of our task, given the information that came out of the BAMming. Burn then generated a workflow management application program for our task. We interacted with Spark to select a mechanism that could provide assistance with one of the activities in the task -- the activity of creating a VCR. And finally, we interacted with another of Burn's knowledge acquisition tools, the tool associated with the mechanism Spark selected, to provide the knowledge that mechanism would need to perform the activity. Burn then generated an application program to assist with the Create VCR activity and made that program available from within the workflow management application.

Thus the work reported in this paper marshals a small amount of data to support the following claims:

- The potential usefulness of application programs is significantly increased if they are viewed, not as isolated pieces of automation, but as agents that need to be carefully situated within a community of practice.
- A workplace analysis methodology that looks for structure in tasks by focusing on the resources the task consumes and products the task produces can make it straight-forward to construct an application program to manage workflow.
- A workflow management application program provides a context that allows other application programs to be integrated within a community of practice.

We are now in the process of gathering more data by using our approch for several other tasks. We are analyzing the BAM-2 methodology; we are exploring the use of other workflow managers -- based on different paradigms; and we are analyzing and extending our library of mechanisms and associated knowledge-acquisition tools.

Acknowledgments

We would be remiss if we did not mention co-workers in the project: Jane Roy is the architect of the BAM-2 methodology and Steve Kennedy is the architect of the EDGE workflow controller. Glee Cameron, Patrice Gautier, Thérèse Mersereau, Charlie Reed and Tina Whitney are members of our research group and made significant contributions.

References

[Bennett 85] Bennett, J. ROGET: A Knowledge-Based System for Acquiring the Conceptual Structure of a Diagnostic Expert System. *Journal of Automated Reasoning,* 1, 1, 1985.

[Birmingham 88] Birmingham, W. Automated Knowledge Acquisition for a Computer Hardware Synthesis System. *Proceedings of the 3rd Knowledge Acquisition for Knowledge-based Systems Workshop.* Banff, Canada, 1988.

[Breuker 89] Breuker, J., B. Wielinga, M. van Someren, R. de Hoog, G. Schreiber, P. de Graf, B. Bredeweg, J. Wielemaker, and J. P. Billault. Model-Driven Knowledge Acquisition: Interpretation Models. *Deliverable task A1, Esprit Project 1098, Memo 87, VF Project Knowledge Acquisition in Formal Domains,* Amsterdam 1989.

[Chandra 83] Chandrasekaran, B. Towards a Taxonomy of Problem Solving Types. *AI Magazine,* 4, 1, 1983.

[Clancey 83] Clancey, W.J. The Epistemology of a Rule-Based Expert System -- a Framework for Explanation. *Artificial Intelligence,* **20**, 3, 1983.

[Davis 79] Davis, R. Interactive Transfer of Expertise: Acquisition of New Inference Rules. *Artificial Intelligence,* **12**, 2, 1979.

[Eshelman 88] Eshelman, L. MOLE: A Knowledge-Acquisition Tool for Cover-and-Differentiate Sytems. In S. Marcus (ed), *Automating Knowledge Acquisition for Expert Systems.* Kluwer, 1988.

[Klinker 88] Klinker, G., C. Boyd, D. Dong, J. Maiman, J. McDermott, and R. Schnelbach. Building Expert Systems with KNACK. *Knowledge Acquisition,* **1**, 3, (299-320), 1989.

[Klinker 90] Klinker, G., C. Bhola, G. Dallemagne, D. Marques, and J. McDermott. Usable and Reusable Programming Constructs. *Proceedings of the fifth Knowledge-Acquisition for Knowledge-Based Systems Workshop,* Banff, Canada, November 1990.

[Krueger 89] Krueger, C. Models of Reuse in Software Engineering. Technical Report CMU-CS-89-188, Department of Computer Science, Carnegie Mellon University, 1989.

[Leonard-Barton 87] Leaonard-Barton, D. The Case for Integrative Innovation: An Expert System at Digital. *Sloan Management Review,* Fall 1987.

[Marcus 88] Marcus, S. SALT: A Knowledge-Acquisition Tool for Propose-and-Revise Systems. In S. Marcus (ed), *Automating Knowledge Acquisition for Expert Systems.* Kluwer, 1988

[Marques 91] Marques, D., G. Dallmagne, P. Gautier, G. Klinker, J. McDermott, D. Tung. Some Data on the Effectiveness of Software Reuse. Submitted for publication.

[McDermott 90] McDermott, J., G. Dallemagne, G. Klinker, D. Marques, and D. Tung. Explorations in How to Make Application Programming Easier. Japanese Knowledge Acquisition Workshop, Osaka, Japan, 1990.

[Musen 91] Musen, M., and S. Tu. A Model of Skeletal-Plan Refinement to Generate Task-Specific Knowledge-Acquisition Tools. Report KSL-91-05, Knowledge Systems Laboratory, Stanford University, 1991.

[Newell 81] Newell, A. The Knowledge Level. *AI Magazine,* 2, 1, 1981.

[Suchman 87] Suchman, L. *Plans and Situated Actions*. Cambridge University Press,1987.

[Wenger 90] Wenger, E. Toward a Theory of Cultural Transparency. PhD Dissertation, Department of Information and Computer Science, University of California, Irvine, 1990.

[Yost 89] Yost, G. A Problem-Space Approach to Expert-System Specification. *Proceedings of the Eleventh International Joint Conference on AI,* Detroit, Michigan, 1989.

Using information technology to solve real world problems

Michel MANAGO, Noël CONRUYT
ACKNOWLEDGE, 16 Passage Foubert, 75013, Paris, France.

Abstract. We present an induction algorithm, KATE, whose learning strategy is similar to the ID3 algorithm but which can handle examples described by several object, relations between objects, and use background domain knowledge to constrain the search space. The efficient numeric learning techniques used in ID3 have been combined with a rich symbolic knowledge representation language (frames) which allows using known induction techniques for a broader range of applications.

1 Induction

Since the early 1980's, induction tools have been used to generate knowledge based systems from databases. From a set of training cases, an induction system automatically builds a knowledge base in the form of a decision tree or a set of production rules. For instance, from a database of patients whose diseases are known, the induction engine learns diagnostic rules that are then used to identify the disease of new incoming patients.

The ID3 algorithm [Quinlan, 1983] is such an induction system. Its descendants have been used for building numerous real-life applications [Michie, 1989]. Nevertheless, all the potential applications cannot be tackled by ID3. Its knowledge representation capabilities are too limited to cope with the training data when it is made out of complex entities and it lacks the ability to handle objects and relations. Due to the increasing sophistication of Database Management Systems (relational and objet-oriented DBMS), there is a clear gap between what can be achieved with ID3 and the needs to fulfil. This was our motivation for developing an induction tool based on the ID3 algorithm but with more powerful representation language. In the next section, we analyze why ID3 cannot be used for an application in tomato plant diseases [INSTIL 88]. We show that the problem with ID3 does not arise from its induction algorithm but from its knowledge representation formalism which fails to capture the true meaning of the information contained in the data.

2 The Notion of Object

A diseased tomato plant can be affected by several different symptoms. Each symptom is a complex entity, or object, whose description depends on both its type and on its location. For example, a symptom of type spot is not described the same way as a symptom of type hole since a spot has a color, and a hole does not. Likewise, a symptom on a leaf is not described the same way as a symptom on a fruit since the position of the symptom with respect to the nerves of the leaf is relevant for describing a symptom on a leaf but not for describing a symptom on a fruit. For instance, the symptom *tache-ou-plage-sur-folioles* (spot-on-leaves) is described by the following 17 features:

> REPARTITION-SUR-PLANTE, REPARTITION-SUR-FEUILLE, NOMBRE, COULEUR, ZONATIONS, MESURE, LIMITES, LOCALISATION-SUR-FOLIOLE, PROPORTION-SUR-FOLIOLE, REPARTITION-SUR-FOLIOLE, JAUNISSEMENT-POURTOUR, MYCELIUM-FRUCTIFICATIONS, ASPECT-DU-MYCELIUM, TOUCHER, FORME, RELIEF.*

Another symptom, *chancre-extérieur-collet* (canker-on-the-outside-of-the-stem), is described by the 14 features:

> ZONATIONS, TOUCHER, MESURE, FORME, LIMITES, RELIEF, LIEN-SYMPTOME-INTERNE, DEGRE-D-ATTAQUE, MYCELIUM-FRUCTIFICATIONS, REPARTITION-SUR-PLANTE, LOCALISATION-SUR-EXTERIEUR-TIGE, REPARTITION-SUR-EXTERIEUR-TIGE, NOMBRE, COULEUR**

These two complex objects (symptoms) share some common features such as COULEUR (color) and FORME (shape). They also have some unshared features: JAUNISSEMENT-POURTOUR (yellowing-around-the-edges) for the spot on leaves, LOCALISATION-SUR-EXTERIEUR-TIGE, (location-on-the-outside-of-the-stem) for the canker on stem etc. In the tomato application, there are **147** different kinds of symptoms. On the average, each kind of symptom is described by 9 features. Furthermore, more than one symptom (out of these 147) can appear in the description of a training example (there are up to 6 symptoms for the same example). How can we represent such an example with a vector of attributes?

A possibility is to introduce an attribute for each type of symptom (ex: exists-spot-on-leaves) and an attribute for each one of its feature (ex: color-of-spot-on-leaves). Unfortunately, since there are 147 different symptoms with an average of 9 features per symptom, this leads to a total of 147 * 9 = 1323 attributes. Such a large number of attributes generates so much computations that ID3 fails to find a solution (combinatorial explosion). At the root node, ID3 makes 1323 information gain computations, then for each son 1322 computations and so on. In this application, the search space is thus very large because each example is very complex to describe and not because there are too many examples (there are less than 400). Unfortunately, ID3 is good at handling databases with a large number of examples, but the examples are usually described by less than 100 attributes. Since the mathematical complexity of ID3 is in O(n) with the number of examples, and in O(n^2) with the number of bits of information needed to represent an example [Manago, 1988], it is clear that having a very large number of attributes is a source of problems. In addition, with this choice of representation, it is impossible to have two or more symptoms of the same type in the same example. For instance, one cannot adequatly represent and example where there are two symptoms of type spots-on-leaves with different colors and shapes.

* distribution-on--plant, distribution-on-leaf, number, color, spherical-zones, measure, limits, localization-on-folioles, proportion-on-folioles, distribution-on-folioles, yellowing-of-edges, mycelium-fructifications, aspect-of-mycelium, feel, form, scraps

** spherical-zones, feel, measure, limits, scraps, link-to-internal-symptom, state-of-damages, mycelium-fructifications, distribution-on-plant, localization-on-the-outside-of-the-stem, distribution-on-the-outside-of-the-stem, number, color

There is a second solution which seems to be a little more economical in terms of the total number of attributes required to describe an example. In the first solution, when a feature is shared by two different symptoms, two attributes are generated anyway. For instance, canker-on-the-outside-of-the-stem and spot-on-leaves share the color feature but, with the above choice of representation, the two attributes color-of-spot-on-leaves and color-of-canker-on-stem are generated. We could thus take advantage of the fact that the symptoms have some features in common in order to reduce the number of attributes. A special attribute called "type-of-symptom", which has one of 147 values (the 147 kinds of symptoms), is introduced. We then take the union of all the features for all the attributes and associate an attribute to each one. This yields a total number of 80 attributes per symptom. Thus, since there is an average of 9 feature per symptom, 71 attributes are, on the average, marked as irrelevant. For example, color is not relevant for describing a hole-in-leaves but it must still appear in the description of this symptom since it is useful for describing some other symptom such as a spot-on-leaves. Since the same tomato plant (i.e. an example) can have up to 6 different symptoms, these figures are multiplied by 6 which yields a total number of 480 attributes for describing all the symptoms in an example, 429 of which are irrelevant.

As we have seen, in both cases we must introduce a special value to mark that an attribute is not meaningful in a certain context. For example, if we choose the first representation, we must introduce a special value for the attributes of the symptoms which do not exist (i.e. what is the value of color-of-spot-on-leaves in an example where exists-spot-on-leaves = no?). If we choose the second representation, we must mark that a feature is irrelevant for describing a certain kind of symptom (i.e. what is the value of color when the symptom is hole-in-leaves?). This value, which we will call "irrelevant", is processed by the induction engine in a special way so that it does not disrupt the information gain computation.: the training examples with value "irrelevant" are propagated in each branch of the node. This increases the complexity of the induction algorithm which is no longer linear with the number of examples. Let us assume we have chosen the second solution for representing the data. Consider a symptom :

	TYPE-SYMPTOME	COULEUR	...	JAUNISSEMENT-POURTOUR	REPARTITION SUR-EXTERIEUR-TIGE
EX109	TACHE-OU-PLAGE-SUR-FOLIOLES	BRUN	...	OUI	**IRRELEVANT**
EX206	CHANCRE-EXTERIEUR-COLLET	GRIS-BEIGE	...	**IRRELEVANT**	GENERALISEE

Fig.1 An attribute-vector representation of an object of type "spot-on-leaves"

The set of attributes which describes the symptom is the union of all the features for all the possible symptoms (80 attributes). On the average, there are 71 attributes with value "irrelevant".Therefore, during induction, as soon as the type of the symptom is known, ID3 makes **71** useless computations of information gain **at each subsequent node** of the decision tree. Since there can be up to 6 symptoms per plant, there are 6 times as much attributes and ID3 make 6*71=429 useless information gain computations at each node of the tree.

Furthermore, the set of possible values for these features is also increased. For instance, the set of possible colors for a symptom of type TACHE-OU-PLAGE-SUR-FOLIOLES is different from the one of a symptom of type CHANCRE-EXTERIEUR-COLLET. Using ID3's attribute-value representation, the set of legal values for each feature is the union of the set of values possible for the feature when it is attached to any symptom. The legal values for the color attribute include the ones for a symptom of type TACHE-OU-PLAGE-SUR-FOLIOLES and for a symptom of type CHANCRE-EXTERIEUR-COLLET. In the tomato domain application, this yields an average number of 30 different possible values for each attribute. Since in its computation of information gain, ID3 evaluate the entropy at each branch (i.e. each possible value for the attribute), ID3 makes 429 * 30 = **12870 useless computations of entropy** for irrelevant features plus computations for irrelevant feature values for the remaining relevant features **at each node** of the decision tree.

As we have seen, due to the fact that ID3 lacks the ability to use common sense knowledge about the entities which describe the examples, it performs massive amounts of totally useless computations. While these computations turn out to be marginal for some applications, they cannot be afforded when the training data is complex since the induction algorithm is in O(n^2) with the number of attributes (combinatorial explosion).

3 Beyond propositional calculus

In the tomato application, there is more than merely constraining the search space during induction. The fact that there are several objects in the training examples may drive ID3 into learning incorrect rules. In order to represent that there are up to 6 symptoms per example with an attribute-value vector, we introduce 6 attributes called exists-symptom1, ..., exists-symptom6 in the description of the examples. There are other attributes which describe each symptom in more details (such as color-symptom1,..., color-symptom6, etc.). Note that the choice of label "symptom1"..."symptom6" for a given symptom on a tomato plant is purely random. For instance, there are no objective reasons to decide that a particular spot-on-folioles is to be called "symptom1" instead of "symptom2". There is no ordering of the symptoms that forces a type of symptom to always have the same label since any 6 out of the 147 symptoms may appear in an example. Unfortunately, the choice of label has a strong impact on what can be learned. It can even drive the induction engine into learning incorrect rules. Consider what would happen if in every examples of a disease "A" there is a spot-on-folioles called symptom1.

Ex1 (A): <exist-symptom1 = yes> **<symptom1 = spot-on-folioles>** <color-symptom1 = yellow> ... <symptom1 = canker-on-stem> <exist-symptom2 = yes> <color-symptom2 = white>...

Ex2 (A): <exist-symptom1 = yes>**<symptom1 = spot-on-folioles>** <color-symptom1 = brown> ... <exist-symptom2 = yes> <symptom2 = hole-in-folioles> <color-symptom2 = irrelevant>...

Ex3 (B): <exist-symptom1 = yes> <symptom1 = mold-on-fruit> <color-symptom1 = white> ... <exist-symptom2 = yes> **<symptom2 = spot-on-folioles>** <color-symptom2 = brown>

ID3 might possibly learn the rule "IF <symptom1 = spot-on-folioles> THEN disease A". However, if there is a "spot-on-folioles" that is called "symptom2" in an example of disease "B", the rule is syntactically consistent with the data but it is semantically incorrect. This is due to the fact that ID3 consider the two attributes exist-symptom1 and exist-symptom2 as two different unrelated entities. The fact that both these attributes give some information about the existence of a symptom is ignored. In fact, as far as ID3 is concerned, these two attributes could just as well have been called X and Y since it only cares about their numerical information gain and ignore the true symbolic meaning of the information that they convey. What is needed here is a way to express a test of the form "Is there any symptom of type spots-on-folioles?" instead of the tests "is symptom1 of type spots-on-folioles?", "is symptom2 of type spots-in-folioles?" etc. The induction engine will then compute information gain of this more generic test. In other word, this application requires a more powerful knowledge representation language and a more powerful pattern matching algorithm. The pattern matcher must handle variables and it must be based on some form of first order logic instead of propositional calculus.

There are other reasons why ID3 cannot be used for this application. For example, it fails to represent adequatly relations between objects. However, as we have shown, ID3 its induction mechanism is not inadequate but its knowledge representation formalism is insufficient. Having noticed this fact, the obvious idea is to build an induction tool with the same learning mechanism as ID3 (hill-climbing search strategy, heuristic preference criterion based on entropy) but with a more powerful knowledge representation language.

4 Constraining Search During Induction

4.1 Representing Background Knowledge with Frames

About twenty years ago, researchers in knowledge representation came up with the notion of frames [Minsky, 1975]. Rapidly, frame based languages invaded most of the Artficial Intelligence tools: expert system shells, blackboard systems, image recognition systems, natural language parsers, planing systems etc. The practical interest of these languages has been well demonstrated: they are powerful, efficient, the knowledge is made of modular reusable entities, the formalism is natural and it enables object-oriented programming.

A frame can be viewed as a data structure. It represents a set of object (a *class)* or of the objects themselves (the *instances*). Properties and relations are attached to the frame and are called *slots*. Facets are attached to the slots. Facets enable representing different kinds of information about the slots. For example, there are which state the actual value of a slot, the type (or legal range of values) of a slot , procedures which deduce features from other features in backward or forward chaining etc. Consider the following frame:

```
[TACHE-OU-PLAGE-SUR-FOLIOLES
own slots:
        <SUPERCLASSES {(VALUE (TACHE-OU-PLAGE  SYMPTOME-SUR-FOLIOLES))}>
member slots:
        <LOCALISATION-SUR-FOLIOLE
                {(TYPE (FACE-SUP  FACE-INF  NERVURES  LIMBE)
                   (CARDINAL 1 2)}>
        <MESURE
                {(TYPE  (REAL 0.2 60))}>     etc...]
```

Frames may be organized in a hierarchy of generality. The most general concepts appear toward the roots of the hierarchy. The value facet of slot "SUPERCLASSES" indicates that the concept of TACHE-OU-PLAGE-SUR-FOLIOLE (spot on leaves) inherits slots from its two superconcepts TACHE-OU-PLAGES (spots) and SYMPTOME-SUR-FOLIOLE (symptom on leaves). For instance, slot LOCALISATION-SUR-FOLIOLE has been inherited from SYMPTOME-SUR-FOLIOLE and MESURE from TACHE-OU-PLAGES. This hierarchy of concepts is entered manually and is part of the background knowledge. The hierarchy of symptom can be organized both according to the location of the symptom or according to the kind of symptoms (KATE handles multiple inheritance).

The frame representation is modular and economical: once the superconcepts have been declared, a new lower level concepts, such as FLETRISSEMENT-SUR-FOLIOLE, is declared by simply stating that it inherits from the two superconcept FLETRISSEMENTS and SYMPTOME-SUR-FOLIOLE which has already been defined for TACHE-OU-PLAGE-SUR-FOLIOLE. All the slot declarations attached to the two superconcepts are transmitted to FLETRISSEMENT-SUR-FOLIOLE. For complex problems such as the one we are describing, having such a flexible, modular and economical representation is very important.

4.2 Using frames to constrain candidate nodes during induction

The notion of "slot" bears some similarities with the notion of "attribute" in ID3. The difference lies in the fact that a slot is attached to an object. The slot LOCALISATION-SUR-FOLIOLE of the object TACHE-OU-PLAGE-SUR-FOLIOLES is a different attribute (i.e. it has a different range of allowed values and a different cardinal) than the slot LOCALISATION-SUR-FOLIOLE of the object FLETRISSEMENT-SUR-FOLIOLE. The induction engine will behave differently when it computes information gain for that slot depending on wether it is attached to one object or the other. For instance, it will generate different branches at that test node since the range of allowed slot values is different.

Since a list of all the features which are relevant to describe an object in a given context is found in the appropriate frame, KATE never considers irrelevant features for the information gain computation. For example, we may have a class frame called MEN with boolean slot BEARDED and a class frame called WOMEN with boolean slot PREGNANT. Since INSTIL and KATE test for information gain of a slot attached, they never test the state of pregnancy of an object who is known to be a man since PREGNANT is not a slot of the MEN frame. In other word, they compute information gain of PREGNANT(WOMEN) and BEARDED(MEN) instead of PREGNANT or BEARDED in general.

KATE uses common sense domain knowledge, contained in the frames, when building the decision tree in order to to constrain the set of features considered for the information gain computation. In the tomato domain, considering every features generates such a large number of computations that the induction algorithm fails to complete its task. If at the first root of the decision tree, the most informative feature is the SIZE slot of a symptom of type HOLE-IN-A-LEAF, at subsequent nodes in the tree only the slots of frame HOLE-IN-A-LEAF are considered. Features such as COLOR, TEXTURE etc. which are irrelevant for this concept are discarded. Thus, at different nodes of the decision tree, different features are considered.

4.3 Constraining the Set of Branches for a Decision Node

Information gain is computed for a slot that is attached to a certain object type.KATE computes information gain of COLOR(SYMPTOM-ON-LEAVES) and not information gain of the COLOR attribute by itself as ID3 does. Two different frames may have different "type" facets for the same slot when the range of legal slot values is different for two different objects. The induction engine then uses this knowledge to constrain the set of branches generated for the current candidate node in the decision tree.

Consider two kinds of MEN called PUNK-ROCKERS and BUSINESMEN (defined as subclasses of MEN). Let us assume that the frames have been defined the following way:

```
[PUNK-ROCKERS <HAIR-COLOR {(TYPE (PURPLE BLUE GREEN BLOND BROWN RED))}>]
[BUSINESMEN <HAIR-COLOR {(TYPE (BLOND BROWN RED WHITE))}>]
```

When the induction engine computes information gain of HAIR-COLOR(PUNK-ROCKERS) it does not consider the WHITE branch since there are no old PUNK-ROCKERS with white hairs. Likewise, when it computes information gain of HAIR-COLOR(BUSINESMEN), it does not compute the entropy of branches PURPLE BLUE and GREEN since there are no BUSINESMEN with hairs dyed in funny colors. Background domain knowledge is used to constrain the set of branches considered at a candidate node.

To summarize, the frame-based language is used to represent common sense domain knowledge about the entities which describe the training examples. This knowledge is used during induction to constrain the set of features that are relevant in a given context (i.e. reduce the number of candidate nodes considered for the information gain computation), and to constrain the set of values for these features (i.e. reduce the number of branches for the candidate nodes). Note that our frame based language is, from a theoretical perspective, a form of first order logic as shown in [Nilsson, 1980]. In addition, any database which can be represented using ID3's attribute-value vectors can also be trivially represented by frames: there is a single frame whose slots correspond to the ID3 attributes. The induction engine then behaves exactly like ID3.

4.4 Handling Examples Described by Different Objects

As we have shown, the tests considered for the information gain computation at different nodes in the decision tree are different. The procedure which dynamically generates candidate nodes for the information gain computation is described below.

- When all the examples at the current branch of the decision tree contain an object of the same type, information gain of the non relational slots of this object is computed (i.e. nominals, ordered nominals, integers and reals). This can either introduce a new object in the current path of the decision tree, or specialize an object that already appears above the current node. Numerical tests are binarized so as to maximize information gain.

- When some of the examples at the current test node do not contain an object of the same type, information gain of the special test "Existst(type-of-the-object)" is computed.
- Information gain of a relational slots is computed when the relation links two objects that already appear in the current path (i.e. above the current node) or when it introduce one new object. Relations that introduce two new objects are not considered in order to constrain the search space.

The boolean test "exists(object-type)" which appears in the decision tree presented below means "is there an object of that type in the example?". Each object that appears in the tree can be indexed further down. The object's identifier is a variable which is bound to the appropriate object in a training example. The induction engine can then index the slots of that object at subsequent decision nodes.

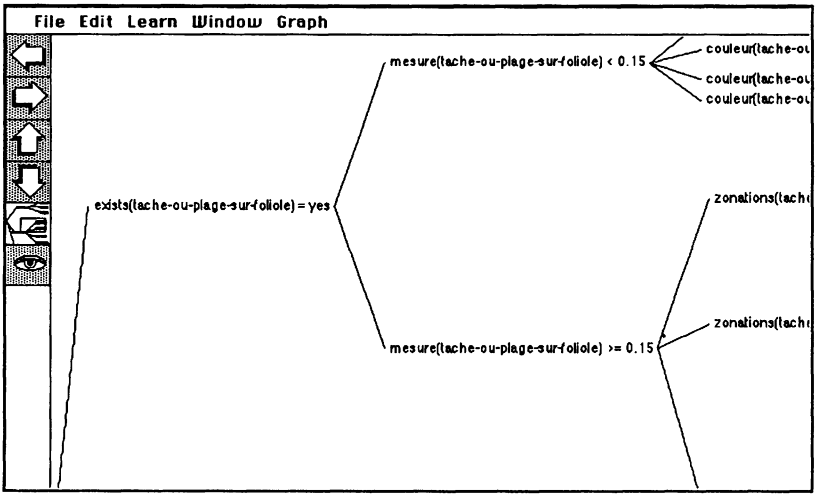

File Edit Learn Window Graph
mesure(tache-ou-plage-sur-foliole) < 0.15
couleur(tache-ou
couleur(tache-ou
couleur(tache-ou
exists(tache-ou-plage-sur-foliole) = yes
zonations(tach
zonations(tach
mesure(tache-ou-plage-sur-foliole) >= 0.15

Fig. 2 When the training examples contain different objects, the special test "Exists(object-type)" is considered when building the tree

This procedure for generating candidate nodes dynamically enables dealing with examples described by different objects. Once information gain has been computed for all candidate nodes, the one with highest information gain is retained as the current decision node. The search strategy (hill-climbing) and preference criterion (maximize information gain) remain the same as in ID3. The difference lies in the set of candidate nodes generated by the algorithm: ID3 computes information gain for all the attributes which do not already appear in the decision nodes above the current node, while KATE does more work to dynamically generate the set of candidate nodes which are considered.

5 Experimental Results

5.1 Diagnosis in Plant Pathology

In the tomato application, there is a small number of examples per concept (there are less than 400 examples for 60 diseases). There are even some diseases for which there is no examples since, according to the expert, it takes about 7 years to see all the diseases. It is therefore meaningless to use a training set and a test set to measure the accuracy of the learnt rules and extrapolate results which are not statistically significant. The rules have been evaluated empirically by the human expert. One of his conclusions was that the rules describing features of existing objects seemed to be meaningful. In fact, the first time the system ran on the data, it found the exact rule he had given for the disease of "Moëlle noire". However, he did not like negative rules such as the one shown below:

IF exists(tache-ou-plage-sur-foliole) = no & exists(anomalie-de-la-forme-ou-de-la-taille-sur-foliole) = no & exists(jaunissement-sur-foliole) = no & exists(autres-anomalies-sur-foliole) = no & exists(jaunissement/dessechement-sur-foliole) = no & exists(dessechement/tache-ou-plage-sur-foliole) = no & exists(fletrissement/jaunissement-sur-foliole) = no & exists(ravageurs-sur-foliole) = no & exists(tache-ou-plage/autres-anomalies-de-coloration-sur-foliole) = no & exists(jaunissement/tache-ou-plage-sur-foliole) = no & exists(fletrissement/tache-ou-plage-sur-foliole) = no & exists(fletrissement-sur-foliole) = no & exists(dessechement-sur-foliole) = no & exists(autres-anomalies-de-coloration-sur-foliole) = no **THEN** Oidium (0.20), Pvy (0.80)

This rule says "if there are none of these 16 symptoms, the disease is "Oidium" with a probability of 0.2 or "Pvy" with a probability of 0.8. In this application domain, considering that each year there are new diseases or variations in existing diseases (mutation, changes in the climate etc.), the expert simply refuses to view his application domain as a closed world. Although a purely negative rule is consistent with the training data, it is totally meaningless for him. The final conclusion is that, for this application, induction is not so useful for knowledge acquisition considering how difficult it is to obtain training examples. However, this project showed that information technology can be extremely interesting for the purpose of maintaining a diagnostic system. In this domain, it is not unusual that a new disease appears or that an existing disease mutate and presents some unseen features. Thus, maintaining rapidly the knowledge base from one crop season to the other is of vital importance. As a consequence, the decision tree that is generated by KATE is not used for interactive consultation as it is usually done. Instead, the on-line interactive questionnaire, that is normally used to collect the description of a training example, is used for entering the full description of the case. The user then chooses to record the description as a training example (the expert provides his diagnosis in a second stage) or to consult the decision tree through KATE's auto-consultation mode. The expert system is then maintained by regenerating the decision tree as new examples are integrated in the database.

5.2 Credit assesments

The french "Société d'Informatique et de Système (SIS)" has tested KATE on an application delivered to one of France's major bank. The expert system's task is to evaluate financial risks when the bank covers credit assessed by a supplier to one of its customer. SIS is a large french servicing company with 20 years of experience in business data processing and 6 years of experience in expert system technology. They have performed a full comparison of knowledge acquisition with KATE, knowledge elicitation by interviewing the human experts and knowledge acquisition using statistical analysis (scoring techniques) for this application.

The training set presented to KATE contains 700 examples described by one object with 31 slots: 12 slots with integer values, 2 slots with real value, 17 slots with nominal values (with an average of 5 values per slot). There are 7 concepts to identify. The problem is complicated due to the fact that, on the average, 80% of an example's description is unknown. Considering the massive amount of unknown values, one cannot associate to the value a probability that is computed on the rest of the training set as it is described in [Quinlan, 1989]. When building the tree, KATE propagates examples with unknown values in each branch of a decision node. Furthermore, the customer has to give a quick answer for a large number of cases and the information available is mostly qualitative. The decision taken has to be explained and justified to the customer. The rule that is extracted from the decision tree provides this explanation. The following table summarize the results of the comparison between the panel of experts, KATE, statistical analysis, and knowledge elicitation (interview of experts).

	board of experts	KATE	knowledge elicitation	statistical analysis
perfect (%)	91	70.3	73.2	48.6
good (%)	8.1	21.7	12.3	26.8
errors (%)	0.9	8	14.5	24.6
time required		14	60	25
men power		11	45	15
analyst		8	7	9
experts		1	8	2
specialist		2	30	4

Perfect means the exact answer that was expected, good means that the correct answer was suggested with a probability above 0.5, errors means that the system answered something incorrect, failed to answer or suggested the correct response with a probability that was too low, time means the total time it took to build the system, men power means the total men power that was required to build the application. The men power has been broken in between the specialist of the method (knowledge engineers for knowledge elicitation, specialists of the method for induction and statistical analysis), human experts who have been interviewed or who have evaluated the learnt knowledge base and computer analysts. The figures have been produced and published by the customer in a more detailed comparative evaluation of the techniques for the purpose of building financial applications [Perray, 1990].

14 men days were required to build the system from scratch using KATE (from an existing database). KATE was ran three times and the background knowledge was slightly modified in between each run. For instance, in between the first two runs, the expert added some useful ratios such as the total revenue of the company divided by the

number of employees. The resulting tree was tested using KATE's auto-consultation mode on a test set of 400 unseen examples. KATE made 70,3% perfect predictions (same response as the expert) and an additional 21,7% correct predictions (the right answer was suggested with the highest probability). 5% of the examples in the training set were classified with uncertainty through auto-consultation which demonstrates that the human expert has access to additional information or that there is some uncertainty in the domain.

Note that, for this application, the frame structure of KATE is not useful. The data is represented by a single object whose slots correspond to ID3's attributes. This proves our claim that any applications that can be tackled using ID3 can also be tackled using an induction tool with a more elaborated representation language such as KATE's.

5.3 Military Applications

KATE has also been used for a military decision support system (this last application is classified). Let us simply state that, like for the tomato application, the training data is represented by complex objects and that a flat attribute-based representation would not have adequatly captured the information contained in the training data for this last application. A second application for helping a human expert to identify objects in a 2-D pictures has been being built. The database consists of 134 examples which belong to 14 different classes. There are currently 12 types of objects and an average number of 58 slots (8 relations, 9 integers, 41 nominals). The description of an example varies depending on the type of the objects involved.

5.4 Failure analysis in a telecommunication network

We have also used the method for an application which analyzes failures in a telecommunication network. The database of examples contains 160 examples with 7 different faults described by a dozain features (mostly numerical). The fault analysis system was tested on 8 files with 200 unsen cases each. The 8th fault for which there was no examples in the training set, 75% saturated network, was in between two known faults (network is 50% saturated, and network is fully saturated). KATE outperformed an existing system by over 20%. Some faults, such as general overload, were identified with 99% accuracy.

6 From induction to case-based reasoning

Consider the following database of examples:

EXAMPLE	DISEASE	SYMPTOM	YELLOWING(SPOT)	SIYE(SPOT)	...
Ex1	BOTRYTIS	SPOT	YES	18	...
Ex2	OIDIUM	SPOT	NO	16	...
Ex3	ALTERNARIOSE	SPOT	YES	2	...
...	...	...	...	...	...

KATE works in two stages: it first learns the decision tree, such as the one shown below, from the examples and then uses the tree (and not the examples) to predict the disease of a new plant. Consider what happens if the user does not know whether the symptom is yellowing or not.

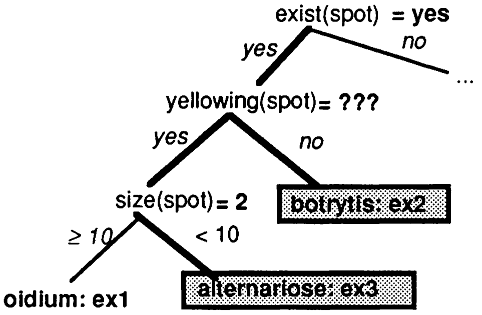


Figure 3: A consultation of the tree

When consulting the expert system, he is first asked wether there is a spot. He answers "yes". He is then asked wether the spot is yellowing. He answers "unknown". KATE proceeds by following both the "yes" and "no" branches and combines the answers at the leaf nodes. In the "no" branch, it reaches the botrytis leaf node. In the "yes" branch, we reach a test node and the user is queried for the value of the size of the spot. He answers "2". The system follows the "<10" branch and reach the "alternariose" leaf node. It then combines the leaf nodes and concludes that it is "botrytis" with a probability of 0.5 and "alternariose" with a probability of 0.5. However, if we consider the example at the "botrytis" leaf node (ex2), we note that its size is 16 (which is greater than 10). Therefore, the correct conclusion should have been alternariose with a probability of 1 since the current case is much closer to ex3 than to ex2. Unfortunately, the information about the size of ex2 was generalized away during the induction phase and it is lost.

The problem described above is not a consequence of a flaw in the induction algorithm nor is it consequence of a flaw in the decision tree formalism (we could have obtained the same conclusion if we had used production rules instead). It is a consequence of the fact that we are reasoning using some abstract knowledge (an abstraction of the information contained in the examples) instead of reasoning directly about the information contained in the training cases. The reasoning system uses general knowledge and not the actual examples. Note that this tree, or rules, could have been entered by hand instead of being derived from the examples by induction. It is therefore a flaw of the knowledge-based system approach to problem solving (reasoning about a problem using general knowledge). We argue that in order to provide a general solution to this problem, we need to use a case-based reasoning system instead of a knowledge based system.

One might object that if we had the same configuration of unknown values in the training cases (for instance, a fourth example with unknown size), the conclusion would have been correct. Unfortunately, this is not realistic for many practical applications. Consider an application where we try to assist a user in identifying an object on a photo. We are dealing with three-dimensional objects from which we can only see one part (ex: the right side). Furthermore, parts of the characteristics of the object may be hidden by other objects which are on the first plan. What would be the size of the database training example if we wanted to enter all the configurations of unknown values which could possibly occur on such a picture? We were faced to this problem in practice. In the application we had to deal with, an object was described by an average of 58 characteristics. There were 91 different classes of objects to identify. Each characteristic of the object could be hidden in any combinations. Therefore, there are:

$$C_2^{58} = 3306$$

combinations of unknown values per class (i.e. a characteristic is known versus it is unknown). In order to provide an exhaustive database of examples we must enter at least 3306 for each class of object to identify which comes to a total of 3306*91= 300846 examples. This is clearly not very practical and it is much more efficient, in this application, to enter only 91 prototypical cases (reconstructed from several photos) and to index them dynamically in order to retrieve the cases which best match the current picture.

As pointed out in [Bareiss, 1989] "case-based reasoning is the usual name given to problem solving methods which make use of specific past experiences rather than a corpus of general knowledge. The key distinction between case-based methods and other forms of automated reasoning is that in the former, a new problem is solved by recognizing its similarities to a specific known problem then transferring the solution of the known problem to the new one (...) In contrast, other methods of problem solving derive a solution either from a general characterization of a group of problems or by search through a still more general body of knowledge". We believe that this key distinction is essential to understand the fundamental differences between induction and case-based reasoning. The two technologies are often confused and some induction products appearing on the market are presented with the label "case-based reasoning tool" (for example, the REMIND™ product of Cognitive Systems inc.). In addition, in some implementations, the distinction is fuzzy: some researchers working on induction have developed systems which remember the training cases in order to be incremental such as ID5 (Utgoff 1988), and some researchers working on case-based reasoning have developed tools which build indexes into the case library in order to be more efficient (ex: Leibowicz's UNIMEM). Thus, the distinction is not technical but lies in how the technology is used.

In order to provide a solution to problems such as photo-interpretation, we have made some extensions to KATE which allows it to do some form of case-based reasoning. These extensions are part of the CAS-SYSTEM tool. The distinction between the two products is not the underlying technology (both use information metrics to find the most discriminant characteristic) but how it is used. KATE builds a static decision tree and uses the tree for consultation. CAS-SYSTEM reasons from the case library and dynamically builds a path which corresponds to the current case. At the root node, it computes information gain for all the features. It then ask the user for the value of the most discriminant one. If the user answers, it develop the corresponding branch. However, if the answer is unknown, then the next best test is tried until the user is able to answer.There are some links between the two technologies. CAS-SYSTEM (case-based reasoning) performs better than KATE when there are unknown values during consultation. It is incremental and new cases can be added without having to reconstruct a decision tree like in KATE. On the other hand, KATE is able to extract knowledge from the case library. It can detect inconsistencies in the case library (a leaf node covers two different classes with some probabilities attached to the classes), it can be used by the expert to notice that the case representation should be modified (add a feature to an object) and so on. In addition, KATE computes the tree once and for all and the consultation is much faster than with CAS-SYSTEM. Thus, KATE and CAS-SYSTEM are not redundant but complement each other. For some applications it is better to use induction, for some application it is better to use case-based reasoning.

7 Conclusions

KATE is an extension of known induction techniques which works on databases modelled by complex entities. The technical extensions that were brought into ID3 (dynamic generation of candidate nodes when building the tree, treating differently a ground property and a relation for the information gain computation, testing for the existence of an object etc.) are independent of the chosen knowledge representation language (frames). With predicate calculus, the same technical problems and more would have to be overcomed. The frame structure allows for efficient retrieval of all the relevant features of an object in a given context and the legal range of values for these features. This knowledge is used during induction to dynamically generate candidate decision nodes and their branches before computing information gain.

The induction system can handle large databases of examples, handle complex data described by several objects with relations, learn generalizations of all the concepts in a single learning cycle and is resilient to noise in the training data. It has been validated for building complex, large scale, real world applications in agronomy (diagnostic system), banking (risk evaluation), military (decision support systems) and telecommunications (failure analysis). However, the induction methology ("generalize and forget") was found to be too restrictive in some applications. We then developped a case-based reasoning tool, CAS-SYSTEM, which uses the same basic technology but reasons directly on the database of cases instead of reasoning on the decision tree which was induced from the database. Our case-based reasoning approach is currently being tested at the Museum of Natural History in a system which identifies marine sponges.

Acknowledgements

We would like to thank ACKNOWLEDGE for allowing us to publish this paper. Michel MANAGO would also like to thank the European Economic Community which supported its PhD research (ESPRIT contract P1063, the INSTIL project with the General Electric Company, CNRS and Cognitech).

ACKNOWLEDGE would like to thank all the organisms which supports its current research: ANVAR, MRT, ANRT and DRRT. KATE is a trademark of Michel MANAGO.

References

Bareis R. (1989). Exemplar-Based Knowledge Acquisition: A Unified Approach to Concept Representation, Classification, and Learning. Academic Press.

INSTIL (1989). Project Summary. ESPRIT Deliverable. Brussel, Belgium: Commission of Economic Communities.

Manago, M. (1986). Object-oriented Generalization: A Tool for Improving Knowledge Based Systems. *Proceedings of the International Meeting on Advances in Learning*, Les Arcs, France.

Manago, M. & Kodratoff, Y. (1987). Noise and Knowledge Acquisition, *Proceedings of the Tenth. International Joint Conference on Artificial Intelligence* (pp. 348-354). Milan, Italy:.Morgan Kaufmann.

Manago, M. (1988). Intégration de Techniques Numériques et Symboliques en Apprentissage Automatique. PhD dissertation, University of Orsay, France.

Manago, M. & Kodratoff, Y. (1990). Induction of Decision Trees from from Complex Structured Data. In G. Piateski-Shapiro & W. Frawley (Eds.), *Knowledge Discovery in Databases*. Detroit, Mi: AAAI Press.

Michie, D. (1989). New Commercial Opportunities Using Information Technology. *Proceedings of the third. International Gi-Kongress* (pp. 64-71). Munich, West Germany: Springer Verlag.

Minsky, M. (1975). "A framework for representing knowledge," in The Psychology of Computer Vision, Winston P. H. ed, Mc Graw-Hill, New York 1975.

Nilsson, N. (1980). Principles of Artificial Intelligence. San Matteo, CA: Morgan Kaufmann.

Perray, M. (1990). Etude Comparative Entre Trois Techniques d'Acquisition des Connaissances: Interview, Induction et Analyse Statistique pour Construire une Même Base de Connaissances. *Proceedings. of the Journées Informatique et Intelligence Artificielle*. Paris, France.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell & T. M. Mitchell (Eds), *Machine Learning: An Artificial Intelligence Approach* (Vol. 1). San Matteo, CA: Morgan Kaufmann.

Quinlan, J. R. (1986). Simplifying decision trees. *Proceedings. of the first AAAI Workshop on Knowledge Acquisition for Knowledge Based Systems* (pp. 36.0 - 36.15). Banff, Canada.

Quinlan, J. R., (1987). Generating Production Rules from Decision Trees, *Proceedings of the Tenth. International Joint Conference on Artificial Intelligence* (pp. 304-307). Milan, Italy:.Morgan Kaufmann.

Quinlan, J. R. (1989). Unknown Attribute Values in Induction. *Proceedings. of the Sixth International Workshop on Machine Learning* (pp. 164-168). Ithaca, NY: Morgan-Kaufmann.

Utgoff, P. (1988). ID$_5$: An Incremental ID$_3$. *In Proceedings of the fifth International Conference on Machine Learning*. Irvine, CA: Morgan Kaufmann.

Facts, fantasies and frameworks: the design of a knowledge acquisition workbench

Nigel Shadbolt
Artificial Intelligence Group
Department of Psychology
University of Nottingham
Nottingham NG7 2RD

Abstract This paper presents a number of core issues that are seen as fundamental to the success and well-being of the knowledge engineering enterprise. In particular, it examines the problems that dominate the current state of the art in knowledge acquisition (KA). These include: the development of KA methodologies, the construction of software support tools for KA, the integration of knowledge acquired from various sources, the problems of verification and validation of the acquired knowledge, and the elimination of bias from expert knowledge. Progress in our subject is likely to depend on our ability to advance our own knowledge and understanding in these key areas. The paper describes work which we have been conducting in the context of an ESPRIT project (P2576 ACKnowledge), and whose aim is to ameliorate some of the problems identified.

Section 1 Introduction

Our ability to build effective knowledge based systems is less than we would wish. Our understanding of many aspects of our subject is incomplete and doubtful. This does not prevent some people from claiming too much — distinguishing fact and fantasy in knowledge engineering can sometimes be a tricky business. The structure of this paper is based around a set assertions which I believe to be currently true of knowledge acquisition, and which I shall now enumerate.

1. Knowledge acquisition is difficult
2. We have only the beginnings of a KA methodology
3. There is little reusability of our knowledge in experts systems
4. Few integrated knowledge acquisition environments exist
5. There is little synergy between acquisition techniques
6. It is difficult to translate between the results of different KA tools
7. It is hard to integrate the results of different acquisition sessions
8. It is difficult to verify and validate the results of acquisition
9. The acquisition techniques are sometimes inappropriately applied
10. Experts and their cognitive processes are poorly understood

I do not pretend that these are the only pressing issues that are relevant to knowledge acquisition. However, I consider progress on these issues vital if we are to develop knowledge acquisition into a stable component of an engineering discipline.

The outline of this paper will therefore be as follows. In section 2 I will examine the state of the art with regard to methodologies in KA. I will also discuss the emergence of model based approaches to the KA process. In section 3 and throughout I will describe our own attempts to produce an integrated workbench to support the knowledge engineer in the process of KA.

Separate sections have been given over to the discussion of knowledge transformation between tools, section 4, and the problems of the integration of results from disparate sources, section 5. Section 6 addresses the problem of evaluating and validating the results of KA. Finally sections 7 and 8 consider the problems of acquisition inherent in our techniques and our subjects.

Throughout I will refer to our own experience in building knowledge acquisition workbenches as part of a large ESPRIT project ACKnowledge[1]. The philosophy of the ACKnowledge approach is discussed in Shadbolt and Wielinga (1990). In particular, discussion will focus around a system ProtoKEW[2], which we have developed at the University of Nottingham and which was built originally as one of a number of concept demonstrators for the ACKnowledge project.

Section 2 Knowledge Acquisition: Methodologies, models and frameworks

Many of the problems enumerated in section 1 can be traced back to the lack of robust, comprehensive methodologies for the KA process. Whilst there are a growing number of articles and books available on 'how to do knowledge elicitation', these often contain advice of the most general kind, and emphasise the pragmatic considerations of expert system development (cf for example, Welbank, 1983; Hoffman, 1987; Kidd, 1987 and Hart, 1986).

One of the most thorough attempts to provide a comprehensive KA framework is provided by KADS - Knowledge Acquisition and Domain Structuring (see Breuker and Wielinga, 1987 for an overview). KADS embodies various principles for the acquisition of knowledge and construction of expert systems. These include two central tenets:

- The analysis should be *model-driven* as early as possible.
- The content of the model should be expressed at the *epistemological level.*

The first of these requires that one should bring to bear a model of how the knowledge is structured early on in the KA process, and use it to interpret subsequent data. The notion of model we are using is some abstraction of expert competence or performance. Indeed the view now predominant is that KA is itself a modelling activity. KA is best regarded as a constructive process — data about the expert's (or experts') behaviour are used by the knowledge engineer, along with other information, to construct a model, or an increasingly complex series of models embodying the desired behaviour of the system. These models can later be reused to provide expectations regarding the general structure of problem solving systems.

The second insists that knowledge is formulated in terms of its role and function in problem solving. This formulation should be independent of particular implementation issues. Moreover, existing models of problem solving should be used, wherever possible, to organise the expertise.

The KADS use of models relies on distinguishing four knowledge layers which should be discriminated in any expert system - human or otherwise. The first of these is domain knowledge and describes the domain concepts, elements and their relations to one another. A second type is task knowledge, which has to do with how goals and sub-goals, tasks and sub-tasks should be performed in any problem solving. A third sort of knowledge is referred to as strategic. This is used to monitor and control problem solving. Finally, inference layer knowledge is distinguished.

[1] The ACKnowledge consortium comprises: Cap Sesa Innovation, Marconi Command and Control Systems, GEC-Marconi Research Centre, Telefonica, Computas Expert Systems, Veritas Research, the University of Amsterdam, Sintef, and the University of Nottingham.

[2] Elsewhere we have described prototype versions of our knowledge engineering workbench (Reichgelt and Shadbolt, 1991, In Press). This work was carried out by the author in conjunction with Han Reichgelt and Nigel Major. Thanks are also due to GEC MRC, the University of Amsterdam and the University of Aberdeen who made available tools which served to inspire some of the components of ProtoKEW.

This has to do with how the components of problem solving and expertise are to be organised and used in the overall system. Within each KADS layer various structures or models can exist that generate expectations about the nature and form of the knowledge instantiating that layer.

The use of knowledge level models to inform and direct the construction of KBSs is central. In fact, a number of methodologies use models although the emphases are somewhat distinct (Steels, 1990; Karbach at al 1990). Nevertheless, common to all is the attempt to abstract aspects of expertise.

The KADS approach itself makes particular use of models of problem solving - or models at the inference layer (Breuker and Wielinga, 1989). An alternative approach is to build task layer representations (Bylander and Chandrasekaran, 1988). Here the aim is to use general, invariant features of a task to guide KA. A number of researchers exploit the fact that many KBS applications are built in similar, if not nearly identical domains (Eshelman 1989; Marcus, 1988). In this case one attempts to build reusable domain layer descriptions. Ultimately models and structures at all of these levels will be important in any informed and directed knowledge acquisition approach.

The use of mediating models to direct acquisition promises an additional advantage. It offers the prospect of reusability (Neches at al 1991). The aim would be to standardise on libraries of generic components — components derived in part from the models developed in KA. It is argued that in this way declarative knowledge, problem solving techniques and reasoning could all be shared among systems. As Neche et al point out there are currently a number of critical impediments to reuse: heterogeneous representations, dialects within language families, lack of communication facilities, and model mismatches at the knowledge level. Nevertheless, initiatives are underway to ameliorate these problems and produce effective reuse.

Section 3 Knowledge engineering workbenches

A variety of systems now provide computer support for KA. There are software products that implement specific acquisition techniques. An example is the implementation of the repertory grid technique (cf, for example, Boose et al, 1989) which we will discuss later in this paper. It is time consuming and labourious to apply manually, but the underlying algorithms are straightforward. Many other individual KA techniques have also been implemented. As implementations of individual techniques any one of these products is limited in scope to the elicitation of certain types of knowledge.

A second type of system, exploits the fact that common features can sometimes be found both within and between domains. OPAL (Musen et al, 1987) is an expectation driven acquisition system that elicits knowledge from experts in the domain of oncology treatment. The system comprises templates, or knowledge frames, which the user fills with appropriate knowledge. Such frames might contain, for example, the details of a particular drug treatment. These details can be expected to instantiate slots in frames which have a similar structure across treatment regimes. The MORE system (Kahn et al, 1986) is an abstraction from an original system MUD which diagnoses oil drilling fluid problems. As such it incorporates templates about expected domain components as well as the task structure and problems solving strategies that are general to all particular instances of such applications. In both OPAL and MORE an attempt has been made to reuse models of expert knowledge.

The disadvantage of these tools is the large amount of effort that needs to be spent building customised acquisition tools for generic application domains. However, it is an important point to discover whether associate domains really share the amounts of common structure required to make the customised tools approach viable.

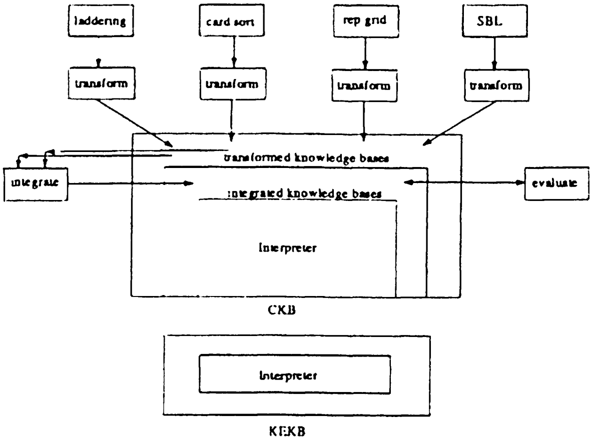
A further class of support tools provide a set of acquisition functions within a single system. Examples of such software include Shelley (Anjewierden et al, 1990) and KEATS (Motta, Rajan, Domingue and Eisenstadt, 1990). These combine a number of documentation and browsing aids to help knowledge engineers find their way around acquisition material. They also include a few particular elicitation methods. What they lack, is any significant degree of integration between the components, and any strong view as to the type of knowledge and knowledge level structures that might underlie an application.

We can see that each of these various types of support system is restricted in scope. The aim of the ACKnowledge project is to achieve integration between a wide range of acquisition techniques, and to combine the best features of current support tools into a knowledge engineering workbench (KEW). KEW will implement a range of elicitation techniques, incorporate machine learning methods, be applicable across domains, and embody a principled or knowledgable approach to the entire acquisition process. To build such a workbench requires that we construct, in effect, a knowledge-based system for knowledge acquisition (Shadbolt & Wielinga, 1990).

To help us design KEW we have built a number of concept demonstrators. One such concept demonstrator, ProtoKEW, comprises: a number of KE and ML acquisition techniques, methods of translating results from tool dependent formats into a common knowledge representation language, techniques to support the integration and evaluation of disparate knowledge, and model directed KA.

The best way to appreciate ProtoKEW is to consider its architecture as shown in Figure 1 below. We can see that a number of distinct KE tools are incorporated. These include: laddering, concept sorting, and repertory grids[3]. Elsewhere we describe these techniques in detail (Shadbolt and Burton, 1990) and their implementation in ProtoKEW is described in Reichgelt and Shadbolt (1991, In Press). In addition to KE methods a number of machine learning algorithms are supported: AQ11, ID and CN2. All these machine learning methods are similarity based induction techniques, and provide for the automatic construction of classifier rules.

Figure 1 The architecture of ProtoKEW



[3] More conventional KE techniques such as interviewing and protocol analysis have not been included in this demonstrator although they will be found in the final ACKnowledge workbench.

Individual KA tool results are transformed into distinct knowledge base "partitions" within what is termed the Common Knowledge Base (CKB). The representation within the CKB is full first order predicate calculus (FOPC). The CKB has an attendant theorem prover (the interpreter) for the logic. The various transformed knowledge bases can be integrated within our FOPC formalism to provide what we term "integrated knowledge bases". This process of integration can proceed as far as the knowledge engineer wishes. The various CKB partitions (integrated or not) can be viewed as logical theories capturing aspects of the domain of application. A theorem proving capability allows us to evaluate components of the emerging knowledge base. Of course, a FOPC inference engine is unlikely to be the final run-time implementation. However, this logic based formulation can be used to produce a high level specification of the ultimate system. As such it allows us to investigate the scope, coverage and consistency of the knowledge being acquired.

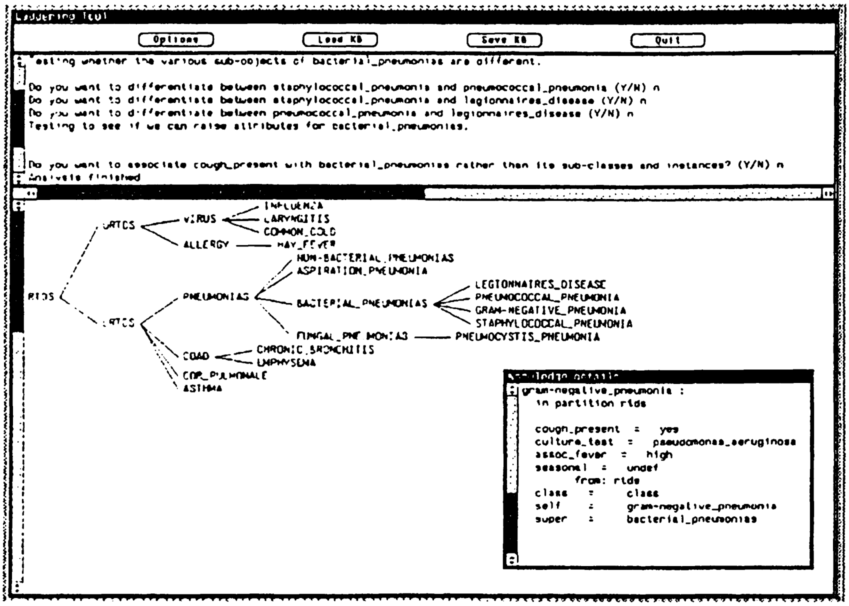
Finally, in Figure 1 the Knowledge Engineering Knowledge Base (KEKB) is distinguished. This embodies the knowledge about knowledge acquisition which is needed if our workbench is to be an expert system for acquisition. Any software system that is going to fulfill this goal will need to be informed by a variety of types of knowledge. A first type of knowledge has to do with knowledge about the knowledge acquisition process itself. This would be advice and guidance about what to do when. Knowledge about the knowledge acquisition tools themselves constitutes a second type of knowledge. Thus, tools make assumptions about data, how it is represented and analysed. The system and knowledge engineer should have explicit command and use of this knowledge. A further type of knowledge is an account of how to integrate results acquired from different tools into a consistent evolving application knowledge base. The emerging application KB needs to be evaluated. Our system needs to know something about how this might be achieved. And, of course, we should attempt to incorporate knowledge about the components of expertise.

In order to provide a context for the discussion of the general KA issues highlighted by ProtoKEW exemplars of a KE and ML method within ProtoKEW will be described — the laddered grid and CN2.

Our implementation of the laddered grid method (Major and Reichgelt, 1990) supports the expert and the knowledge engineer in the construction of a graphical representation of application knowledge in terms of the relations between knowledge elements. The result is a two-dimensional graph (or conceptual map) whose nodes are connected by labelled arcs. A number of settings on the tool allow for different interaction protocols which expand and explore the graph in ways that have been discovered to be natural and productive (Burton et al, 1987, 1988, Shadbolt and Burton 1989).

Figure 2 shows part of a laddering session from the domain of respiratory tract diseases. In ProtoKEW the tool uses a simple object-oriented language called CommonSloop (Reichgelt, Major and Jackson, 1990) as its tool-specific knowledge representation language. Each node in the laddered hierarchy is represented as an object in CommonSloop. An object can have user-defined slots which are properties of the item in the node hierarchy. CommonSloop supports multiple inheritance. Inheritance also follows a default principle; attributes associated with higher-level objects are inherited by their children but can be overridden lower in the hierarchy.

Figure 2 Laddering in the respiratory tract diseases domain



```
Testing whether the various sub-objects of bacterial_pneumonias are different.

Do you want to differentiate between staphylococcal_pneumonia and pneumococcal_pneumonia (Y/N) n
Do you want to differentiate between staphylococcal_pneumonia and legionnaires_disease (Y/N) n
Do you want to differentiate between pneumococcal_pneumonia and legionnaires_disease (Y/N) n
Testing to see if we can raise attributes for bacterial_pneumonias.


Do you want to associate cough_present with bacterial_pneumonias rather than its sub-classes and instances? (Y/N) n
Analysis finished
```

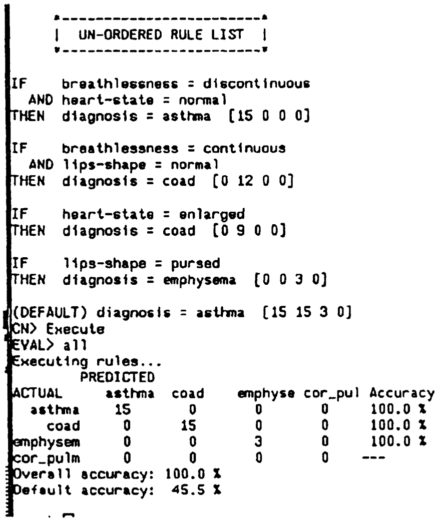
```
gram-negative_pneumonia :
  in partition rtds

  cough_present  =   yes
  culture_test  =   pseudomonas_aeruginosa
  assoc_fever  =   high
  seasonal  =   undef
        from: rtds
  class   =     class
  self    :     gram-negative_pneumonia
  super   :     bacterial_pneumonias
```

The most suitable internal representations for a laddering tool are structured object languages. In such representations we aim to bring together, under a simple indexing method, the associated properties of objects. The use of inheritance allows us to exploit taxonomic relations in an efficient way. Specialising objects by overwriting or adding attributes and their values provides for a means of discriminating between objects. These features of structured object representations clearly map onto the characteristics of the laddering technique itself. In this way different representations afford different possibilities for the various types of KA tool.

Our second exemplar technique, the CN algorithm (Clark and Niblett, 1989) was developed to address certain problems inherent in the widely known AQ algorithm (Michalski, 1969). Both algorithms operate by generating what is termed a *complex*. This is a conjunct of attribute tests — for example temperature = high & cough = present. The complex *covers*, or is satisfied by, a subset of the training examples. This complex forms the condition part of a production rule "if condition then predict class", where class is the most common class in the training examples which satisfy the condition.

In brief, the search method in CN proceeds by repeatedly specialising candidate examples until one is located which covers a large number of examples of single class, and a few other rules which cover a few of the other classes. CN is more robust than AQ since it can tolerate noise in the domain — this can arise where an example has been misclassified or perhaps an attribute has no known value or else an incorrect one.

Figure 3 A rule set induced by CN for respiratory diseases



```
     *-------------------------*
     |  UN-ORDERED RULE LIST   |
     *-------------------------*

IF     breathlessness = discontinuous
  AND heart-state = normal
THEN  diagnosis = asthma  [15 0 0 0]

IF     breathlessness = continuous
  AND lips-shape = normal
THEN  diagnosis = coad  [0 12 0 0]

IF     heart-state = enlarged
THEN  diagnosis = coad  [0 9 0 0]

IF     lips-shape = pursed
THEN  diagnosis = emphysema  [0 0 3 0]

(DEFAULT) diagnosis = asthma  [15 15 3 0]
CN> Execute
EVAL> all
Executing rules...
        PREDICTED
ACTUAL     asthma  coad    emphyse cor_pul Accuracy
  asthma    15      0       0       0      100.0 %
    coad     0     15       0       0      100.0 %
emphysem     0      0       3       0      100.0 %
cor_pulm     0      0       0       0     ---
Overall accuracy: 100.0 %
Default accuracy:  45.5 %
```

Figure 3 shows a run of CN on a set of cases from the respiratory diseases domain. The examples are described in terms of attribute value pairs with one attribute regarded as the assumed outcome class. In the situation above we have determined that we are interested in generating rules which discriminate cases in terms of their diagnostic category. The algorithm is run and a set of rules produced. Figure 3 also shows the performance of these rules on the training set.

As a stand alone tool CN has limited utility for the knowledge engineer engaged in extensive KA. However, by combining the technique with other simple KA methods we obtain synergy — combinations of methods provide greater benefits than the tools used in isolation. Illustrative is the example given in Shadbolt (1991) in which preprocessing of cases is carried out using a sorting method — the expert decides which attributes in a case description are relevant for a particular outcome class. Rapid *re-presentation* of rules back to an expert for critiquing has proved to be very useful. This evaluation can suggest ways in which the case description is deficient. This may leading, in turn, to the use of techniques such as repertory grids to obtain more discriminating case descriptions.

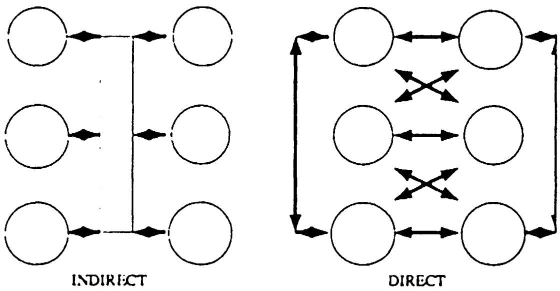
The interaction of KA tools in this way will provide the foundations for effective synergy between components of KA workbenches. However, to date relatively little work has focused on exploring the range of possible synergies between tools.

Section 4 Knowledge Interchange and Transformation

In order to construct expert systems out of the partial results of individual KA tools we need a means to integrate the results together. Before this can proceed, however, we must have some way of pooling the knowledge accrued within an acquisition workbench. There are two distinct approaches that could be taken to this problem. These are indirect and direct transformations.

Indirect transformation achieves communication through a common knowledge representation language (CKRL). Each KA tool, represented as a circle in Figure 4, has a translator into and out of the CKRL. The direct method allows for pairwise translation between all KA tools. Whilst direct translation is viable for small number of KA tools, beyond four or so the direct approach leads to an explosion of translation possibilities. For each additional tool added using the indirect method only two additional translations are required.

Figure 4 Knowledge transformation methods



In ProtoKEW we have adopted the indirect transformation approach. As stated our CKRL is the first order predicate calculus (FOPC). In addition to expressive power such a logic has a model-theoretic semantics which allows one to determine the correctness (soundness) of the interpreter. Moreover, in the present context, the use of logic as the CKRL has the added advantage that it makes the problem of transformation and integration semantically tractable, allowing us to address the issue of whether the knowledge remains valid under the effects of transformation.

The process of transformation itself we have come to see as an important additional stage in the acquisition and refinement of knowledge. The tools we build to support transformation are interactive in nature. Consider the knowledge base produced by the laddering tool in Figure 2. The tool specific knowledge base will contain a number of frames with different slots. Each frame either stands for a single object in the domain, or a class of such objects. What is the logical translation of such a frame structure? It is fairly clear that concepts such as COAD or PNEUMONIAS are classes and as such are translated into FOPC as one-place predicate. But are the leaf-nodes in such structures intended to be an individual object or a class of objects? The user is asked for each leaf node what the intended interpretation is. In the case that the intended interpretation is that the leaf nodes are individuals then they are translated into constants. Class leaf nodes are assumed to be non-empty, and we add axioms to the effect that there exists some object with the property denoted by the class-frame name.

Some parts of the translation can be automatic. The class superclass relation between two "predicate" frames p and q is translated as $((\forall x)(p(x) \rightarrow q(x)))$. Slots can normally be translated automatically: a slot corresponds to a two-place predicate. The exact translation of a slot and its filler depends on the type of filler. In most cases translation is straightforward[4]. A slot s with value v is translated as $(s(a,v))$ in the case of a "constant" frame, and as $((\forall x)(p(x) \rightarrow s(x,v)))$in the case of a predicate frame p.

In default inheritance the translation algorithm first determines whether any of the descendants of the frame have an incompatible slot value. If so, these are explicitly stored as exceptions in the antecedent of a rule. Thus the expression $((\forall x)((bacterial_{p}neumonia(x)\&\neg legionnaires_{d}isease(x)) \rightarrow gram_{s}tain(x,positive)))$

This is not the end of the translation process. Additional interaction with the user can serve to further elucidate the intended semantics of Figure 2. Consider the class VIRUS in Figure 2, is it intended that the three children of this concept form an exhaustive list? The

[4] Full details of this and other translations can be found in Reichgelt and Shadbolt (In Press).

answer, of course, very much depends on the context of acquisition, the scope of the elicitation sessions, and the coverage of the intended system. In any event if the answer is affirmative, our translation algorithm adds an axiom to the effect that any entity that is an instance of the parent frame must be an instance of at least one of the children — $((\forall x)(virus(x) \rightarrow (influenza(x) \vee laryngitis(x) \vee common - cold(x))))$.

During transformation implicit assumptions often inherent in the context of the local knowledge acquisition task have to be made explicit. Establishing the wider context in which the knowledge is to be used is an important stage in acquiring and validating knowledge.

The declarative translation of a structure such as that shown in Figure 2, along with all its slot and filler information is very substantial. Such a translation into FOPC affords certain advantages — not least of which is that the notion of semantic consistency can be examined. However, the original frame notation facilitates other sorts of computation. For example, it is very easy within the structured object representation to determine if objects can be discriminated in terms of their attributes and values, whether the attributes on an object have been specified to a sufficient level of abstraction in the hierarchy. The lesson is that a variety of representations enrich and extend the acquisition possibilities.

The issue of knowledge transformation and interchange within and between acquisition workbenches is, of course, a large and complex one. Ginsberg (1991) has argued that knowledge interchange between complex formalisms is fraught with dangers. One of which is the production of semantically opaque translation programs. His own proposals are that we effect interchange via a minimalist "first order predicate calculus, with specific notational conventions to handle variables, quantification, and so on". These ideas are close to our own. He also argues that there remains an open research question about whether substantial knowledge sharing is possible at all.

Section 5 Knowledge Integration

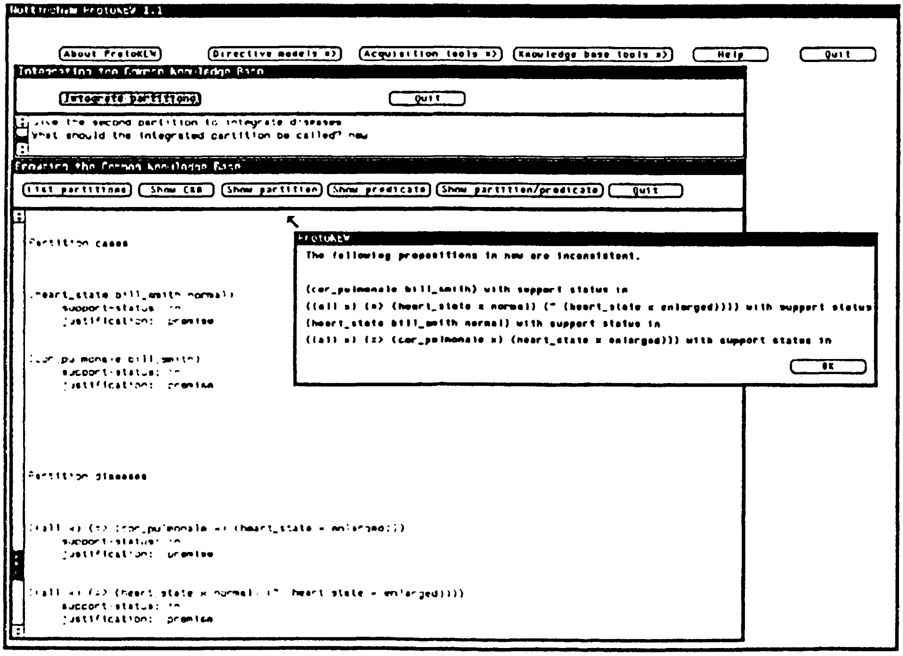
Once any KA tool has produced its results and they have been subject to either direct or indirect transformation there remains the issue of knowledge integration. How are we to assemble fragments of expertise into a consistent and evolving knowledge base?

Within ProtoKEW we have a very simple but appealing notion of integration — the merging of logical theories. In ProtoKEW, the CKB contains a partitioning mechanism which allows one to divide logical axioms into different partitions. In logical terms, each partition corresponds to a different theory. While the system attempts to ensure consistency within a single partition, it does *not* ensure consistency between partitions. The partitioning mechanism therefore allows users to explore different theories simultaneously.

Integration of two partitions is achieved by copying one partition into a new partition, and then incrementally trying to add the propositions from the other original partition. While adding these new propositions two tests can be performed. We can ask the system to determine if the new propositions are non-redundant and consistent. The redundancy test determines whether a proposition is already entailed by the existing propositions in the partition. If a proposition fails this test, then it is not added to the knowledge base. The consistency test involves determining whether the proposition is consistent with the other propositions stored in the partition. Can the system prove its negation from the information already in the knowledge base? If a proposition fails this test, then a reason maintenance system (McAllester, 1980) is called on to ensure consistency. If *both tests* are performed successfully, then the user is guaranteed that the set of propositions is minimal and consistent, at least within the limitations of the theorem prover[5].

In Figure 5 below we can see that two partitions in the respiratory diseases domain are being integrated. One partition called *cases* contains simple facts about a patient: the fact that he is currently diagnosed as suffering from cor pulmonale, and the fact that his heart state is normal. In a second partition *diseases* we see knowledge to the effect that all individuals who suffer from cor pulmonale would have an enlarged heart state, and that a heart state cannot both be normal and enlarged. Attempting to integrate these two partitions has resulted in the RMS being invoked with the inconsistent set of propositions.

Figure 5 Integration of knowledge in the respiratory diseases domain



```
Give the second partition to integrate diseases
What should the integrated partition be called? new
```

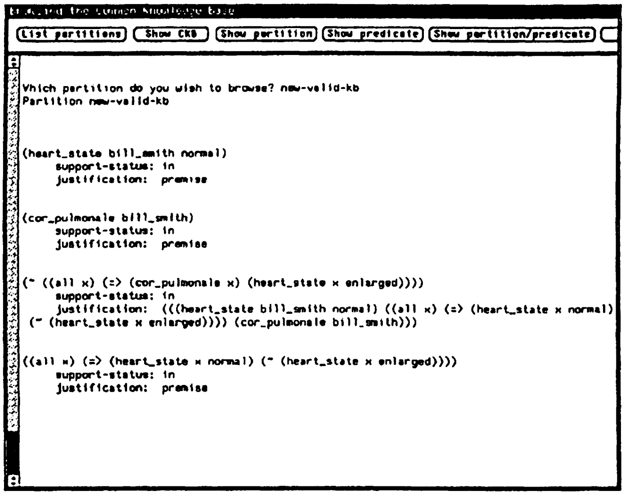
```
Partition cases

(heart_state bill_smith normal)
     support-status: in
     justification:  premise

(cor_pulmonale bill_smith)
     support-status: in
     justification:  premise

Partition diseases

((all x) (=> (cor_pulmonale x) (heart_state x enlarged)))
     support-status: in
     justification:  premise

((all x) (=> (heart_state x normal) (~ (heart_state x enlarged))))
     support-status: in
     justification:  premise
```

```
The following propositions in new are inconsistent.

(cor_pulmonale bill_smith) with support status in
((all x) (=> (heart_state x normal) (~ (heart_state x enlarged)))) with support status
(heart_state bill_smith normal) with support status in
((all x) (=> (cor_pulmonale x) (heart_state x enlarged))) with support status in
```

The expert and knowledge engineer decide what piece(s) of knowledge is suspect. In this case we see that in Figure 6 the general rule that any sufferer of cor pulmonale has an enlarged heart has been retracted. Notice, that in our implementation we have acquired new information in this process. We now believe that at least some patients with cor pulmonale will not have enlarged hearts. Moreover, we have a justification of why this piece of knowledge is now believed — namely that a counter example has been discovered.

[5] It is in general impossible to determine automatically whether a set of propositions in classical first-order predicate calculus is consistent. Any theorem prover needs to contain heuristics to halt the search for a proof. Because these heuristics may terminate the search for a proof too early we can never *guarantee* consistency for any set of propositions in KB.

Figure 6 Maintaining consistency in the respiratory diseases domain



```
Which partition do you wish to browse? new-valid-kb
Partition new-valid-kb



(heart_state bill_smith normal)
     support-status: in
     justification:  premise


(cor_pulmonale bill_smith)
     support-status: in
     justification:  premise


(~ ((all x) (=> (cor_pulmonale x) (heart_state x enlarged))))
     support-status: in
     justification:  (((heart_state bill_smith normal) ((all x) (=> (heart_state x normal)
 (~ (heart_state x enlarged)))) (cor_pulmonale bill_smith)))


((all x) (=> (heart_state x normal) (~ (heart_state x enlarged))))
     support-status: in
     justification:  premise
```

This type of integration ensures logical integrity. However, integration can mean a great deal more than this. In the type of integration described above we have no idea whether our integration provides better or more complete coverage of the domain application. There is no concept of knowledge integration at the *knowledge level* — the role and function of knowledge in problem solving. For a richer notion of integration we need to appeal to the idea of evaluating the consequences of integration with respect to a model of problem solving.

Section 6 Evaluating and validating the results of acquisition

One obvious way of evaluating a knowledge base is by presenting different propositions in the knowledge base to the expert. While this may in general allow one to detect certain false propositions, it cannot guarantee that we find all errors. Moreover, we are not guaranteed to find missing pieces of knowledge. An alternative is to check the knowledge base against a number of test cases, and to see whether it provides appropriate solutions for these cases. Adoption of this approach leads to two further issues. First, one has to generate a comprehensive set of test cases. Generating a satisfactory set of test cases is itself a non-trivial knowledge acquisition task. Second, given inappropriate performance on a test case, the knowledge base needs to be refined. If the system is able to derive a false conclusion, tools must be provided for retracting whatever propositions were responsible for the conclusion, and, conversely, if the system fails to derive a true conclusion, tools must be provided to add the relevant knowledge. We have seen that ProtoKEW provides limited support for refinement via retraction of propositions using RMS capabilities. Refinement by addition of missing knowledge is largely supported in our ACKnowledge workbenches by appeal to the notion of model based validation.

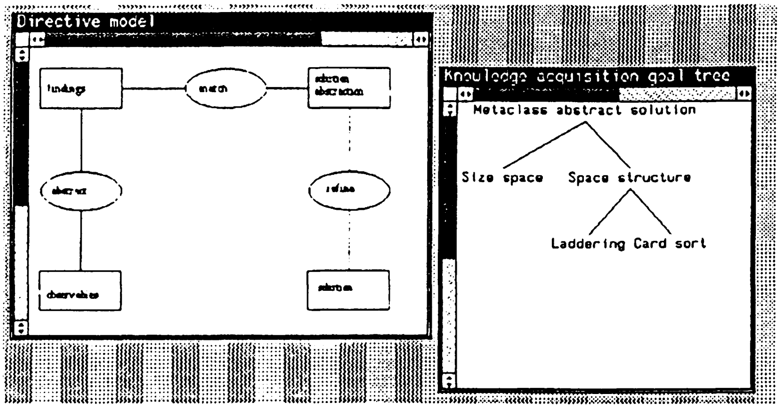
As discussed in section 3 we are aiming to build an active and directive knowledge acquisition workbench. The evolution of such a system is a gradual process. Initially we have used the knowledge engineer as the main controller of KA activity. Ultimately we hope to encapsulate more and more knowledge about knowledge acquisition into our knowledge engineering workbenches. Currently in ProtoKEW we have a mixed initiative system with control very much in the hands of the knowledge engineer. The directive knowledge encapsulated in

Univ.-Bibliothek
Regensburg

ProtoKEW is restricted to a number of compiled knowledge structures which can be used to direct acquisition activity.

The type of structure that is available is illustrated in Figure 7 .

Figure 7 Directive models in ProtoKEW



The *directive model* under consideration is the interpretation model for heuristic classification (Clancey, 1985). In the current version of ProtoKEW we assume that selection of this object has been made by the knowledge engineer from a library of such structures contained in the workbench. It serves as a candidate for the type of problem solving observed in the application domain[6]. The point to note about selection of this structure is that it sets the context for subsequent acquisition.

The directive model contains knowledge which can be used to inform acquisition. It shows us both the inputs and outputs (representing as rectangles) of various processes (representing as ellipses) that make up a generic type of problem solving. If we formulate our respiratory diseases example in terms of this model we would note four kinds of I/O classes: observables, findings, abstract solutions, and solutions. In our domain examples of knowledge that plays these roles in problem solving might be

observables — the patient has a respiratory rate of 25
observables — the patient has a blue hue to the tongue
findings — the patient has tachypnea
findings — the patient has central cyanosis
abstract solutions — the patient has COAD
abstract solutions — the patient has pneumonia
solutions — the patient has emphysema
solutions — the patient has staphylococcal pneumonia

[6] In Shadbolt and Wielinga (1990) we describe how this selection might ultimately be automated.

The processes that operate on these metaclasses are: abstract, match and refine. Each of these processes is in turn associated with a set of methods for effecting the change from input to output. The process abstract is defined as abstracting observables into findings; a method which can effect this is qualitative abstraction. Once again in our domain examples of knowledge that plays these roles might be:

abstract — if the patient has a respiratory rate > 20 the patient has tachypnea
abstract — haemoptysis is the presence of blood in the sputum
match — if the patient has creamy yellow sputum then the patient may have staphylococcoal pneumonia
match — if the patient has cyanosis then the patient may have emphysema
refine — emphysema is a kind of COAD
refine — staphylococcal pneumonia is a kind of pneumonia

Expectations about the *types* of knowledge that may be implicated in problem solving is precisely the sort of knowledge that can generate acquisition goals. In Figure 7 the *abstract solution* metaclass has been selected and this has activated knowledge in ProtoKEW. This in turn displays a knowledge acquisition goal tree in the window to the right of the directive model. This goal tree indicates how knowledge might be acquired about this part of the expertise space. The goal tree indicates that in this context we can use either laddering or a card sort to explore the structure of the solution space. It is this knowledge that might lead the knowledge engineer to select a particular acquisition tool for the task at hand. Knowledge acquired will be stored in partitions which reflect the structure shown in the directive model.

As well as directing acquisition the directive model can serve as a template for the adequacy of the knowledge accrued. For example, we might well discover that we have knowledge sufficient to generate a wide class of data abstractions:

FEV1/FVC ratio < 70% is obstructive
peak expiratory flow rate = 500L/min is normal
central cyanosis is a blue hue to the tongue

But then discover that many of the findings are not used in the matching process — this may well suggest incomplete coverage in this part of the problem solving model. In this sense models could be used constantly in the refinement of knowledge bases.

An obvious area of development in ProtoKEW is to increase the range and use of directive models. This has been done in the latest version of the KEW. Moreover, we have begun to operationalise the directive models. We are implementing a language which describes the structure of the directive models and serves as a means of invoking various KA tools to acquire certain types of knowledge, and organise partitions of knowledge to reflect the process of problem solving embodied in the model.

This has obvious implications for validation. It enables the knowledge engineer to determine for parts of the overall problem solving process whether there is knowledge sufficient to solve the role assigned to it. Validation and evaluation can take place in terms of local components of expertise and reasoning. It is hoped that the extra principles of structure which the models impose will be useful in dealing with the complexity of any knowledge base.

Section 7 Understanding acquisition techniques

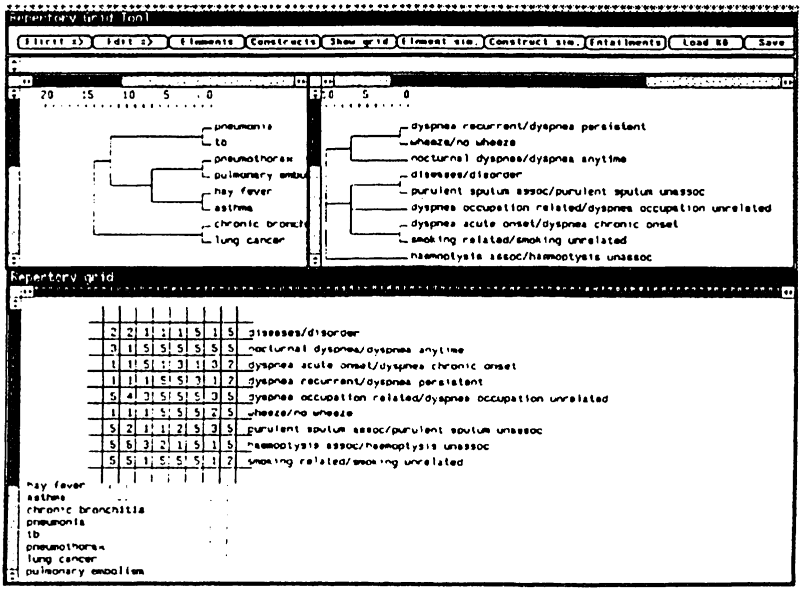
Although a reasonable number of acquisition techniques are now available there are a number of reasons to be worried about both the uptake and use of the techniques. To illustrate this point

I will examine the use of the repertory grid method in KA. The repertory grid is widely used and has formed the basis of some influential KA environments.

In this techniques an expert is given, or is asked to generate, a set of elements. They are then usually presented with three elements, and asked to describe some way in which two are similar or different from the third. Thus the expert might be presented with the disease elements, TB, lung cancer and asthma. The expert might judge that TB and lung cancer are similar in that they are often associated with haemoptysis (blood in the sputum). The dimension by which these elements have been separated is called the construct. The ends of the construct are called poles and the construct is taken to be measurable, that is we assume that elements can be rated or ranked along the construct.

This process is continued with different triads of elements until the expert can think of no further discriminating constructs. The result is a matrix relating elements and constructs. The matrix is analysed using a statistical technique called cluster analysis. This reveals clusters of elements and constructs which represent meaningful concepts. These concepts may not have been articulated in the original elicitation session. We are also able to subject the results of cluster analysis to the technique of *entailment analysis* (Gaines and Shaw, 1986) which generates implications between constructs. For example, we might find that the pole *smoking related* of the construct *smoking implicated* implies *dyspnea onset chronic* on the construct *onset of dyspnea.* Figure 8 shows our repertory grid tool.

Figure 8 The repertory grid tool in ProtoKEW


Repertory Grid Tool
First :>
Edit :>
Elements
Constructs
Show grid
Element sim.
Construct sim.
Entailments
Load KB
Save
20
15
10
5
0
pneumonia
tb
pneumothorax
pulmonary embo
hay fever
asthma
chronic bronch
lung cancer
10
5
0
dyspnea recurrent/dyspnea persistent
wheeze/no wheeze
nocturnal dyspnea/dyspnea anytime
diseases/disorder
purulent sputum assoc/purulent sputum unassoc
dyspnea occupation related/dyspnea occupation unrelated
dyspnea acute onset/dyspnea chronic onset
smoking related/smoking unrelated
haemnolysis assoc/hamooptysis unassoc
Repertory grid

2	2	1	1	1	5	1	5	diseases/disorder
3	1	5	5	5	5	5	5	nocturnal dysonea/dyspnea anytime
1	1	5	1	3	1	3	2	dyspnea acute onset/dyspnea chronic onset
1	1	1	5	5	3	1	2	dyspnea recurrent/dyspnea persistent
5	4	3	5	5	5	3	5	dyspnea occupation related/dyspnea occupation unrelated
1	1	1	5	5	5	2	5	wheeze/no wheeze
5	2	1	1	2	5	3	5	purulent sputum assoc/purulent sputum unassoc
5	6	3	2	1	5	1	5	haemoptysis assoc/haemoptysis unassoc
5	5	1	5	5	5	1	2	smoking related/smoking unrelated

hay fever
asthma
chronic bronchitis
pneumonia
tb
pneumothorax
lung cancer
pulmonary embolism

Whilst the method can reveal very interesting patterns and results there are a number of areas where bias and misinterpretation can occur. These include the selection of elements, constructs, and measures. Thus elements need to be of the same epistemological type. Ideally they should be representative for the purposes of elicitation. We have to take care lest the element set chosen be skewed or unrepresentative in some way. This is a difficult problem and considerable knowledge of the domain may itself be required to select a suitable set. Constructs elicitation are no less unproblematic. Yorke (1978) for example, has found differences depending on whether triadic (sets of three elements) or dyadic (sets of two elements) are presented to the user in order to elicit a construct. The labels that are ascribed to constructs and their poles are also problematic. The knowledge engineer should be aware that these labels are likely to be highly polysemous. That is they convey for the expert all sorts of assumptions, associations, implications, caveats and the like. It is important that these meanings are not lost or overlooked in the course of elicitation. There are also problems inherent in the notion of measuring semantic distance. Consider our example in Figure 8 above. Here we have a construct symptom *onset* — its poles are *acute* to *chronic*. In fact in the domain this can range in time from the symptom appearing over the past few hours to it having been present over months or years. What are we to do if an element has a wide range of values? Moreover, not all constructs are well suited to a continuous interpretation. Certain constructs have categorical values. These are more difficult to represent in continuous dimensions. Finally, when the grids are subject to cluster analysis the numerical techniques makes strong assumptions about the nature of semantic space.

A fuller description of the problems inherent in this technique can be found in Rugg and Shadbolt (1991). We are not decrying these techniques. However, it is important to be aware of the limits of any single method. This is one reason why we believe in combining techniques. It might also be appropriate to consider reevaluating KA tools now that they are becoming widely deployed. The aims of such research ought to be to produce an understanding sensitive of the strengths and weaknesses of KA tools, techniques and methods in a variety of application contexts.

Section 8 Understanding experts and expertise

We have discussed a number of important issues central to KA. What of the experts themselves? Work in the US by Arthur D Little, and in the UK at the University of Nottingham has indicated the importance of expert differences. Kindle et al (1989) distinguish between: "academics", "drones" and "samurai". Although in any particular situation one is likely to find a mix of these types in any one individual expert.

To see how these categories can be important we will consider an application context. Imagine we are building a system to assist in respiratory disease diagnosis. Our expertise will come from a local teaching hospital, in such contexts a hierarchy of "experts" is often evident. One can discern the "academic" who in this case might be the Professor of a Department of Respiratory Medicine. The "drone" might correspond to the "houseman" or doctor who is on a rotation around wards in a hospital. Both of these sorts of expert have qualified via traditional medical training - the Professor will have made the subject his speciality. Finally, it is a medical technician who will perform and initially analyse many of the routine tests — this is our "samurai". The training of the technician is not likely to be as long or as comprehensive as the other two types of expert. However, they spend much of their working day performing the same kind of task.

How are we to distinguish between these types of expert in terms of the expertise they might embody? Certainly one important difference is in the desired outcome of any piece of problem solving. The characteristic of the "academic" is the pursuit of the "true" solution - a conclusion that follows from the systematic application of domain principles. For the "drone" the concern is much more pragmatic - they require a solution that works given the resource constraints and context within which they have to work. For the "samurai" the issue of an outcome is simply performance - doing what they are expected to do, and to a level acceptable to the system within which they operate.

In our example the characterisation given to a test, for example, a chest x-ray, is fundamentally determined by the desired outcome. The medical technician may classify it as "abnormal" and forward the case for further investigation. The houseman may look at the "abnormal" case and decide that it is nevertheless not serious enough to warrant further testing because of the cost associated with the tests that would determine the diagnosis completely. The Professor looking at this same x-ray might decide that it represents an "interesting abnormality", it may present a number of features not normally seen in this type of case.

The nature of the outcome is just one of a number of dimensions which distinguish the expert types. Others include the problem solving environment in which they find themselves and the types of problem they are likely to encounter. The "academic" may have relatively large amounts of time to devote to a few pathological cases - seldom spending long on common or mundane problems. The "drone" spends a lot of their time in a problem solving environment where there is too much information, the cases they deal with tend to reflect the mainstream. The "samurai" may spend almost all of their time making judgements about a large number of cases. The judgements they are expected to make, however, are between a relatively small number of categories. They are not called upon to provide detailed explanations and justifications for the decisions made.

The nature of the training between these various types of expert is very different. The "academic" and "drone" are likely to have been taught by "academics" - the former going on to specialise exclusively in one problem solving domain, whilst the latter may have received additional training in a "journeyman" fashion. The "samurai" may have received some formal instruction but by and large they learn their problem solving from other "samurai".

All of the factors mentioned above obviously have an effect on the form in which the knowledge is represented - both internally and in the way in which the expert can externally communicate their knowledge. In the case of the "academic" the theory predominates. The "academic" is likely to be fluent and articulate, able to talk about their problem solving - though very often only in terms of the theory. For the "drone" heuristics dominate - the knowledge may be systematic although it is likely that understanding of the deep principles of the domain is patchy. Surface structure knowledge tends to predominate. For the "samurai" the knowledge may be highly "automated" or "compiled". Consequently competence is implicit and the expert may have great difficulty in articulating the basis of their expertise.

There is a crucial point in this for the knowledge engineer - depending on the type of expert we are dealing with then the kinds of KA techniques that are most effective are likely to vary. As will the form and content of the knowledge used by the expert, and in each case the context of problem solving is all important.

A further depressing feature of much expert systems development is the neglect shown by knowledge engineers of the principles of human cognition. Despite a number of researchers writing explicitly on this material (Slater, 1987; Hoffman, 1987; Chi et al, 1988; Meyer and Booker, 1991) little attention seems to be paid to the cognitive issues we shall discuss below. This is potentially disastrous. The problems indicated can be such as to vitiate any confidence we might have in the validity and robustness of the knowledge we elicit.

One of the striking features of expertise is the amount of information which experts process while developing their expertise, and the speed with which it can be recalled. A skilled radiologist, for example, may have seen literally hundreds of thousands of X rays (Lesgold et al, 1988); a chess master will probably have seen a similar number of chess positions in various games (Chase & Simon, 1973). The speed of recall is similarly impressive - in many domains it appears to be virtually instantaneous, and the accuracy of recall can be equally high.

The impressive aspects of expert memory should not, however, blind us to the considerable literature on limitations and bias in human memory. Memory is not a passive process like tape recording; it is an active process, with many facets of selectivity and large scope for unconscious bias.

Features such as primacy and recency effects, making the earliest and latest instances more easily memorable than the intermediate ones, have been accepted as basic concepts in psychology for decades. At a more sophisticated level, it has been known for over half a century that memory of events or stories is subject to considerable distortion, to the point of reversing sequences of events (cf Bartlett 1958).

The selectivity of memory, its reconstructive nature and so on, mean that the knowledge engineer should produce materials and approaches that can detect and correct these biases (cf for example Hoffman 1987, Meyer et al 1990)

Not only is human memory subject to error, but also the way in which the information is used. Humans are prone to systematic patterns of error across a number of apparently simple operations (Rips and Marcus 1977). For example, Modus Tollens states that *if A implies B* is true, and *not B* is true then *not A* must be true. However, people, whether expert in a domain or not, make errors on this rule. This is in part due to an inability to reason with contrapositive statements. Also in part it depends on what A and B actually represent. In other words, we are affected by the content.

There have been many explanations for this sort of data but what is interesting is that virtually all the research in this area shows that at a fundamental level the basic laws of logic and associated proof strategies are difficult for people to apprehend and follow (Wason 1961; Johnson-Laird 1983).

This means that one cannot rely on the veracity of expert reasoning. In assembling acquisition material these kinds of experiment indicate the need to keep chains of implication simple when asking the expert for knowledge or asking the expert to review or critique existing material.

There is also a considerable literature on human handling of uncertainty, most notably that by Kahneman and Tversky (e.g. Kahneman, Slovic & Tversky, 1982). This body of research has shown unequivocally that in many situations people are remarkably poor judges of probability, and that experienced experts may be demonstrably inferior in performance to even the crudest linear models. Problems arise when experts are asked to provide certainty values, judgements of probability and weights of evidence. People are known to undervalue prior probabilities, to use the ends and middle of the probability scale rather than the full range, and to anchor their responses around an initial guess (Kahneman et al, 1982). Cleaves (1987) and Hink and Woods(1987) both review these sorts of problems and make suggestions about how these biases might be ameliorated in the context of KA. The KA community must remain cogniscent of the important findings emanating from psychology.

Section 9 Concluding remarks

In this paper I have tried to detail a number of key issues that confront the knowledge engineering enterprise. I have concentrated, in particular, on the problems inherent in providing computational support for the KA process. I have explored these in the context of our own experience in building acquisition workbenches.

There is a large and difficult research agenda ahead of us. Nevertheless, in a number of areas we can see the emergence of ideas that will, I believe, assume a central role in the future. These include: the deployment of model based acquisition methods, reuse of knowledge structures within and between application domains, and the development of knowledgeable knowledge acquisition tools.

Finally, we should not forget that the formative influences on knowledge acquisition have been eclectic and interdisciplinary. In the attempt to secure a sound basis for knowledge engineering we must retain this broad based view of our subject.

Section 10 References

1. Anjewierden, A., Wielemaker, J. & Toussaint, C. (1990) Shelley — Computer Aided Knowledge Engineering. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber & M. van Someren, (ed), Current Trends in Knowledge Acquisition, pages 313–338. Amsterdam: IOS Press.
2. Boose, J.H., Shema, D.B and Bradsi, J.M. (1989) Recent progress in AQUINAS: a knowledge acquisition workbench, *Knowledge Acquisition*, volume 1(2), Academic Press, London.
3. Breuker. J. &. Wielinga, B. (1987) Use of models in the interpretation of verbal data. In A.L. Kidd, (ed), *Knowledge Acquisition for Expert Systems, a practical handbook*, New York: Plenum Press.
4. Breuker. J. &. Wielinga, B. (1989) Model driven knowledge acquisition. In P. Guida and G. Tasso, (eds), *Topics in the design of expert systems*, Amsterdam:. North Holland.
5. Burton, A., Shadbolt, N., Hedgecock, A. & Rugg, G. (1987) A formal evaluation of knowledge elicitation techniques for expert systems. In D Moralee, (ed), *Research and development in expert systems, IV*, 136–145.
6. Burton, A., Shadbolt, N., Rugg, G. & Hedgecock, A. (1988) Knowledge elicitation techniques in classification domains. *ECAI-88*, 85–90
7. Bylander, T. & Chandrasekaran, B. (1988) Generic tasks in knowledge-based reasoning: The 'right' level of abstraction for knowledge acquisition. In B. Gaines and J.Boose, (eds), *Knowledge Acquisition for Knowledge Based Systems*, volume 1, pages 65–77. Academic Press, London, 1988.
8. Chase, W.G. & Simon, H.A. (1973) Perception in Chess Cognitive Psychology, 4, 55-81
9. Chi, M., Claser, R. and Farr, M. (Eds) (1988) *The Nature of Expertise*. New Jersey: LEA.
10. Clancey, W. (1985) Heuristic classification. *Artificial Intelligence*, 27:289–350.
11. Clark, P. & Niblett, T. (1989) The CN2 induction algorithm. *Machine Learning Journal*, 3(4), pages 261–283.
12. Cleaves, D. A. (1987) Cognitive biases and corrective techniques: Proposals for improvingelicitation procedures for knowledge-based systems. International Journal of Man-Machine Studies, 27,155-166.
13. Eshelman, L. (1989) MOLE: A knowledge-acquisition tool for cover-and-differentiate systems. In S. Marcus, (ed), *Automating Knowledge Acquisition for Expert Systems*, pages 37–80. Dordrecht: Kluwer Academic Publishers.
14. Gaines, B. R. & Shaw, M. L. G. (1986) Induction of inference rules for expert systems. *Fuzzy Sets and Systems*, 8 (3), 315–328.
15. Ginsberg, M. L. (1991) Knowledge Interchange Format: The KIF of Death. AI Magazine, 12(3), pp. 57-63.
16. Hart, A. (1986) *Knowledge Acquisition for Expert Systems*. London: Kogan Page.
17. Hink, R. F. and Woods, D. L. (1987) How humans process uncertain knowledge. AI Magazine, 8, 41-53.

18. Hoffman, R. R. (1987) The Problem of Extracting the Knowledge of Experts from the Perspective of Experimental Psychology. AI Magazine, 8, pp. 53-66.
19. Johnson-Laird, P. N. (1983) Mental models. Cambridge. Cambridge University Press.
20. Kahn, G., Nowlan, S. & McDermott, J. (1986) Strategies for Knowledge Acquisition. PAMI Special Issue on Knowledge Representation.
21. Kahneman, D., Slovic, P. and Tversky, A. (Eds) (1982) Judgement under uncertainty: Heuristics and biases. New York: Cambridge University Press.
22. Karbach, W., Linster, M. & Voss, A. (1990) Model-Based Approaches : one label —one idea? In B. Wielinga, J. Boose, B. Gaines, G. Schreiber & M. van Someren, (ed), *Current Trends in Knowledge Acquisition*, pages 313–338. Amsterdam: IOS Press.
23. Kidd, A. L. (Ed.) (1987) Knowledge Acquisition for Expert Systems: A Practical Handbook. New York: Plenum Press.
24. Kindle, K. W., Cann, R. S., Craig, M. R. and Martin, T. J. (1989) PFPS: Personal Financial Planning System. In H. Schorr and A. Rappaport, (eds), *Innovative Applications of Artificial Intelligence* MIT Press: Cambridge, Mass.
25. Lesgold, A., Rubinson, H., Feltovich, P., Glaser, R., Klopfer, D. and Wang, Y. (1988) Expertise in a complex skill: diagnosing X-ray pictures. In Chi, M.T.H., Glaser, R. & Farr, M.J. (eds) *The Nature of Expertise* Lawrence Erlbaum; London.
26. McAllester, D. (1980) An outlook on truth maintenance. Technical report, MIT AI LAB.
27. Major, N. & Reichgelt, H. (1990) Alto: An automated laddering tool. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber & M. van Someren, (ed),Current Trends in Knowledge Acquisition, pages 222–236. Amsterdam: IOS Press.
28. Marcus, S. (1988) *Automatic knowledge acquisition for expert systems.* New York: Kluwer.
29. Meyer, M. A., Booker, J. M. and Bradshaw, J. M. (1990) A flexible six-step program for defining and handling bias in knowledge elicitation. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, M. van Someren, (eds), *Current trends in knowledge acquisition.* Amsterdam: IOS Press.
30. Meyer, M and Booker, J. (1991) Eliciting and analysing expert judgement: a proactical guide. *Knowledge Based Systems Vol 5*, Academic Press, London.
31. Michalski, R.S. (1969) On the quasi-minimal solution of htc general covering problem. *Proceedings of the 5th International Symposium on Information Processing (FCIP 69), Vol. A3 (Switching Circuits)*, Bled, Yugoslavia, pages 125–128.
32. Motta, E., Rajan, T., Domingue, J. & Eisenstadt, M. (1990) Methodological foundations of KEATS, the knowledge engineer's assistant. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber & M. van Someren, (ed),Current Trends in Knowledge Acquisition, pages 257–275. Amsterdam: IOS Press.
33. Musen, M. A., Fagin, L.M., Combs, D.M. & Shortliffe, E.H. (1987) Use of a domain model to drive an interactive knowledge-editing tool. *International Journal of Man-Machine Studies*, 26, pages 105–121.
34. Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T. and Swartout, W. (1991) Enabling Technology for Knowledge Sharing. AI Magazine, 12(3), pp. 36-56.
35. Reichgelt, H., Major, N. & Jackson, P. (1990) Commonsloop: The manual. Technical report, AI Group, Dept Psychology, University of Nottingham.
36. Reichgelt, H. and Shadbolt, N.R. (1991a). Knowledgeable knowledge acquisition. In L. Steels and B. Smith, Eds. *AISB91* Springer-Verlag
37. Reichgelt, H. and Shadbolt, N.R. (1991b). ProtoKEW: A knowledge-based system for knowledge acquisition. In D, Sleeman and N. Bernsen, Eds. *Research directions in cognitive science volume 5: Artificial Intelligence.* Lawrence Erlbaum.

38. Rips, L. J. and Marcus, S. L. (1977) Supposition and the analysis of conditional sentences. In Just, M. A. and Carpenter, P. A. (eds), *Cognitive processes in comprehension.* Hillsdale, NJ: Erlbaum.
39. Rugg, G & Shadbolt, N. (1991) On the limitations of the repertory grid technique. Technical report, AI Group, Dept Psychology, University of Nottingham.
40. Shadbolt, N. & Burton, M. (1989) The empirical study of knowledge elicitation techniques. *SIGART Newsletter*, 108, April 1989, ACM Press.
41. Shadbolt, N. & Burton, M. (1990) Knowledge elicitation. In J. Wilson and N. Corlett, (eds), *Evaluation of Human Work: A Practical Ergonomics Methodology*, pages 321–346. Taylor and Francis.
42. Shadbolt, N. & Wielinga, B. (1990) Knowledge based knowledge acquisition: the next generation of support tools. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber & M. van Someren, (ed), Current Trends in Knowledge Acquisition, pages 313–338. Amsterdam: IOS Press.
43. Slatter, P. E. (1987) Building expert systems: cognitive emulation. Chichester: Ellis Horwood.
44. Steels, L. (1990) Components of expertise. *The AI Magazine*, 11: 30–62.
45. Wason, P. C. (1961) Response to affirmative and negative binary statements. British Journal of Psychology, 52 ,273-81.
46. Welbank, M. A. (1983) A Review of Knowledge Acquisition Techniques for Expert Systems. British Telecom Research, Martlesham Heath.
47. Yorke, D.M. (1978) Repertory grids in educational research: some methodological considerations. *British Journal of Social and Clinical Psychology*, 9, 108–21.

Mapping Expert Behavior onto Task-Level Frameworks: The need for "Eco-Pragmatic" Approaches to Knowledge Engineering

*Rolf Pfeifer, Thomas Rothenfluh**, *Markus Stolze & Felix Steiner*
AI Lab, Computer Science Department
University of Zurich, Winterthurerstrasse 190
CH-8057 Zurich, Switzerland

Abstract

The main goal of this paper is to explore the possibilities of exploiting psychological methods for the purpose of knowledge engineering. Hypotheses are presented why both the pure "psychological" and the pure "engineering" positions are not viable for building expert systems. A "middle-out" strategy is proposed that preserves the best of both worlds while minimizing the problems of each. This "middle-out" strategy consists of the application of so-called "task-level frameworks". However, these frameworks do not sufficiently support one of the most crucial tasks in the knowledge engineering process, namely the mapping of the actual expert behavior onto conceptual models. In this paper, a new method which makes this process easier and more reliable is described and a standardized several-step procedure for mapping expertise-in-action protocols onto a task-level framework is illustrated with a case study. It is concluded (a) that protocol analysis is a good starting point for developing tools to support the knowledge engineering process—if appropriate methods are available, and (b) that methods are only appropriate if they are ecological on the one hand and pragmatic on the other.

Introduction

For more than a decade a sometimes heated debate has been going on over the problem of knowledge acquisition in building expert systems. Acquiring knowledge from a domain expert was recognized early on as a major task in developing an expert system. In fact it was considered the most time-consuming one and thus the term "knowledge acquisition bottleneck" was coined.

To cope with this bottleneck, a large variety of methods have been suggested—some involving principles from psychology, others mostly inspired by computer science and software engineering. The former typically assume that a human expert has a central role in the knowledge engineering process. The main idea underlying this paradigm is that a successful expert system will have to include as much expert knowledge as possible, and that this expert knowledge has to be extracted from a human expert. Proposals along these lines typically deal with (a) interviewing techniques, (b) various knowledge elicitation techniques—from observation to thinking-aloud methods, to video-recordings, to expert-guided novice problem

* Currently: CIS/LAIR, Ohio State University, USA

solving (e.g., Diaper, 1989), (c) methods for knowledge modeling, and (d) methods for protocol analysis.

Among the engineering-inspired approaches are the following: (a) Machine-learning methods, for example rule-induction from a set of examples, (b) model-based reasoning techniques (first principles approach, e.g., Davis, 1984, de Kleer, 1984), or (c) approaches which focus on the aspect of decision support (e.g., Sol, Takkenberg & DeVries, 1987), rather than on modeling expertise.

Proponents of the psychological perspective argue that human beings are the best problem solvers known and it is thus best to model a problem solving system after humans. Moreover, users of expert systems are—since they are human beings—most comfortable with a system behaving like a human problem solver. If the system behaves like a human expert the explanations will be more natural and thus more comprehensible to the user. Moreover, if the knowledge-base is to be tested, psychological methods can be applied, that is, the system's solution and solution path can be compared those of experts in natural ways. We will situate proponents of this approach in the *expert camp*.

Opponents of this "expert centered" view have put forward a number of arguments. First, so the argument, it is not sensible to model the problem solving behavior of human experts since different experts may differ widely in how they solve a particular problem. Second, the knowledge-base developed in this way will always be incomplete—it is hard to determine the boundaries of a system. Third, what the user of a system needs is not human expertise in the first place, but rather a system which supports his or her work. Fourth, it is very difficult for experts to express and verbalize their knowledge and thus it is also very difficult for the knowledge engineers to elicit this knowledge. And, last but not least, even experts may make errors or are unable to correctly recall all their behaviors and reasoning steps—in short, they are subject to the same psychological limitations as all other human beings. Proponents of this view advocate the use of systems and software engineering for the development of expert systems and are thus to be placed in the *engineering camp*.

While there are hard-liners in each camp there is—as is so often the case—a "middle-out" strategy. It consists in the application of so-called task-level frameworks. This strategy will be outlined below.

We will proceed as follows. First it is argued that neither a pure psychological nor a pure engineering approach will work, but that a "middle-out" strategy has to be applied which allows for both a more psychologically sound but still application-oriented knowledge engineering process. One example of such a task-level framework, the componential framework, is then introduced as a possible means to bridge this gap. Next, problems with task-level frameworks are listed and possible approaches to deal with them are outlined. This is followed by a discussion of a major problem which is not given adequate treatment in the literature, the one of mapping expertise onto a task-level framework. How this process can be supported is discussed in general. More specifically it is shown how protocol analysis can be exploited to develop appropriate methods and tools to support the knowledge engineer in this mapping activity. A concrete method for protocol analysis is then proposed and conclusions are drawn.

Three basic hypotheses concerning expert systems development

Earlier we stated that there are—in essence—two camps, the one who wants to include expert knowledge (the expert camp) and the one who does not (the engineering camp). Both claim that their approach to building expert systems is the best one. We claim that neither camp is correct and that a third strategy should be used. The following three statements of the hypotheses will be used as a starting point for our discussion. They are also intended to clarify the reasons why there is a large gap between them and include a proposal on how to get the best of both worlds.

> *Hypothesis 1:* Psychological methods, in particular the ones from cognitive psychology, typically yield results which cannot be directly exploited for expert system construction because they serve a different purpose, and are frequently too specific and low level.
>
> *Hypothesis 2:* The pure engineering approach will not work because of the fundamental epistemological problems one needs to cope with in the domains requiring intelligent behavior.
>
> *Hypothesis 3:* If appropriate psychological methods were available they would be used by knowledge engineers. What is needed are methods which are at the same time ecological and pragmatic, that is, we need an *eco-pragmatic* approach.

Hypothesis 1: As is well-known the methods of cognitive psychology have been criticized as being too narrow and thus producing only irrelevant results (e.g., Abelson, 1981; Neisser, 1982; Norman, 1988). Abelson and Neisser—although they are involved in very different sorts of research—argue both in a sense for a direction of psychology that is more ecological, that is, a kind of psychology which focuses on natural settings and avoids laboratory situations which are too artificial. Although this "ecological view" of psychology is getting to be more and more important there is still much skepticism if not scorn of this paradigm.

If we peruse the large body of literature of *psychological* studies of expert knowledge or problem solving in general, we can draw a number of conclusions (for reviews of the field of psychological expertise research, see Holyoak, 1990, or Waldmann & Weinert, 1990):

> (a) The studies mostly pursue the scientific goal of *understanding* (rather than engineering systems) and largely leave open what the results imply for the construction of problem solving systems. Examples are the investigations by Newell & Simon (1972), and Dörner (1976) on problem solving, or the studies on novice/expert comparisons (e.g., Chi, Feltovitch & Glaser, 1981, Elstein, Shulman & Sprafka, 1978, Patel & Groen, 1986), to mention but a few.
>
> (b) The studies are frequently not sufficiently ecological to be of relevance to expert system construction. For instance, they frequently focus on tasks which are too specific or are taken from domains which only remotely resemble the sorts of tasks that are of interest for expert system development, such as chess, memory, writing, motor behavior, algebra, high school physics, geometry, sports, etc., and which do not easily generalize to other domains. Moreover, many of them are still confined to controlled laboratory-like situations.
>
> (c) Although some studies have resulted in methods for actually building systems, they are at times motivated and used in a psychologically somewhat naive way (e.g., Repertory Grids or Laddering techniques).
>
> (d) There is a lot of research and there are many methods in psychology that try to capture real-world problem-solving behavior. Protocol analysis is used here as a pertinent example and will be discussed in more detail later on.

We would like to stress that from our discussion it does not follow that psychological studies of expertise are uninteresting, quite the contrary. In fact, the review of Holyoak (1990) is highly informative since—among other things—it dismantles a lot of folklore about expertise (e.g., that experts always solve problems with more ease than novices, that performance increases continuously with practice, etc.). We are only saying that psychological studies have not contributed much to improve knowledge engineering practice because of the methodology applied. We also want to mention that since psychological studies of expertise add to our understanding this might eventually lead to building better systems.

***Hypothesis** 2:* Because of the difficulties just outlined, a pure engineering approach to building expert systems has been proposed. "Engineering" in this context means that principles from systems and software engineering, and computer science in general are to be used. The resulting system consists of a collection of algorithms (mainly search, optimization, Bayesian classifiers) that are linked to data bases which are, perhaps, augmented by a number of production rules to support the interaction between user and data base. Another kind of "pure" engineering approach is model-based reasoning. Model-based reasoning tries to model the problem domain from "first principles" (Davis, 1984; de Kleer, 1984), i.e. without regard to the ways in which this model is to be used. Of course, since these principles are typically given by the laws of physics, for example of electronic circuits, and since the way a device works is described in the technical manuals, no human expert is needed. At the most, the human expert might be asked about technical details of a device, but his expertise of problem solving is not of interest.

The reason why these pure engineering approaches do not work well outside the research arena is that they do not give adequate consideration to the epistemological limitations that humans and computers have (Steels, 1990; see also Simon's concept of "bounded rationality"—Simon, 1969). Very briefly, these limitations concern time and space, observation, and theory formation. Time and space are always limited: there is only so much time that can be allocated to make decisions, and the amount of information which can be stored is restricted (resource limitation). Thus a search process which would have to explore too many alternatives, or computations which are too time consuming cannot be used. Observation must be done through interaction with the real world. Sensory organs are prone to error and—always, without exception—only partial information is available. Moreover, observations take time and effort and only a limited number can be performed. Since theories—again—have to be developed by interacting with the environment (physical, social) using limited resources, they will, by necessity, always be restricted.

A pure engineering approach will only work under highly unrealistic assumptions. First, the assumption must be made that all measurements and observations are correct and that all the measurements which are necessary for a particular type of model can be made within the time and cost limitations allocated to the problem solving process—an assumption which only holds in the most trivial cases. Second, the use of such models typically leads to combinatorial problems which are ignored in this approach. And third, it is assumed that indeed a good domain theory and thus precise models are available. This assumption is also only true in a few restricted, predominantly technical domains.

***Hypothesis** 3:* Given that neither the "cognitive psychology solution" nor the "pure engineering solution" seem to work as methods for building real life expert systems, an alternative is needed. The assumption is that if appropriate methods were available they would be used. The requirements for these methods have to be (a) *ecological* , that is they must be applicable with reasonable effort to realistic natural settings in which complex problem solving takes place, and they (b) must be directed toward building working expert systems, that is, they must be *pragmatic*. In other words, what we propose is an approach we will call *"eco-pragmatic"*.

If behavior is complex and subject to large variations, such as expert problem solving, it is difficult to devise tightly controlled experiments without severe interference with the process to be investigated. Reducing the problem to some highly constrained subtasks such as recognizing specific patterns (e.g., in sound perception, or patterns of pieces of LISP code) is likely not to provide the data needed to design a system for the complete problem solving process.

Alternatively, methods are used which are not so much directed at studying expert behavior but rather at eliciting certain parts—typically the declarative ones—of expert knowledge. Examples are card sorting and grid laddering techniques for knowledge acquisition. These techniques are a step in the right direction: First, they can be applied to real expert knowledge with reasonable effort (the ecological aspect), even though only a small part—static concepts and structures—of the expert knowledge is accessible in this way. Second, this knowledge can be used—more or less directly—to actually build a system (the pragmatic aspect). On the negative side, they ignore the vital aspects of problem solving knowledge, of control, and of strategies and will not be further studied in this paper.

Steps toward an "eco-pragmatic" approach: Task-level frameworks

Overview: We believe that knowledge engineering is paradigmatic for an interdisciplinary eco-pragmatic field: it is neither pure engineering (as the term engineering might suggest), nor is it pure psychology. There are psychological theories of problem solving (e.g., Newell & Simon, 1972, Dörner, 1976) but they are usually difficult to apply to a concrete problem with reasonable effort. However, out of the expert systems community, a number of proposals have been made for so-called task-level frameworks. They comprise computational theories or skeletons of theories of problem solving which are thus directly geared towards system development. Examples of such frameworks are the theory of *heuristic classification* (Clancey, 1985), *generic tasks* (Chandrasekaran, 1986), *problem solving methods* (McDermott, 1988), KADS (Breuker & Wielinga, 1989), and the *componential framework* (Steels, 1990). Although the main purpose of these frameworks is the development of systems, they are based partly on psychological evidence. But they are not meant to be *psychological* theories of problem solving. Rather they are theories of problem solving which are potentially applicable to humans or to machines. As such they provide appropriate means for performing knowledge acquisition. Let us briefly illustrate this point.

An example - the "componential framework": In the "componential framework" (Steels, 1990), so-called tasks are mapped onto problem solving methods and actions. Actions are methods which are executable without further decomposition. If an action can be found to perform a specific task, it is executed. A simple example of an action, i.e. an executable method, is "ask user" which performs data acquisition. This method can be applied whenever data are needed which cannot be deduced or whose deduction would be too costly. If no executable method is available the problem is recursively further decomposed into subtasks until executable methods are available. The task "diagnosis", for example, is decomposed into the subtasks "gather initial data" (inquire about the symptoms), "restrict context" (using general information about the patient's age, medical history etc. and the initial symptoms, narrow down context), "select" (select a particular diagnostic class, i.e. a disease). Now, for the selection step, many methods are potentially applicable. Examples are "linear search", "hierarchical classification", "weighted evidence combination", "Bayesian classification", "discrimination" and "association". Which one of these methods is selected by the expert depends on the pragmatic features of the task. Pragmatic features are concerned with the amount, the quality, the availability and the cost of the input data, and the way in which the desired output is defined (e.g., as concepts with necessary and sufficient conditions, or as prototypes). The idea is to ask the expert about the sorts of problem solving methods he knows about (e.g., Vanwelkenhuysen, 1989). At each decision point he typically has a choice of several methods one of which he has to select. He can be asked about the reasons for selecting one method over others. The important part here is that these reasons do not have to correspond to the "real" reasons for choosing a particular method. Even post-hoc rationalizations (e.g., Nisbett & Wilson, 1977) are fine as long as they are plausible and consistent. In other words, the reasons do not have to be psychologically valid. Plausible means that a potential user should be able to follow the rationale, and consistent means that there should be no contradictions between the selection criteria in different situations. In this way such a task-level framework can be used for knowledge acquisition purposes, that is, a problem solving program may be developed through the interaction with a domain expert without developing a psychological model of his problem solving behavior. Nevertheless, we are applying, in a sense, a psychological method.

In addition to providing a structure at the conceptual level some task-level frameworks include methods for mapping—at least parts of—the task-level analysis onto a computational framework (Vanwelkenhuysen & Rademakers, 1990). But this aspect is of less concern for the current arguments and will therefore not be further discussed.

Problems with task-level frameworks and what can be done about it

Problems

A number of problems with task-level frameworks are discussed by Steels (1990). They include the fact that some are at too high a level of abstraction and thus leave many important aspects unspecified (e.g., Clancey's inference structures leave the nature of the "heuristic match" entirely unspecified) and others make too many assumptions about the problem solving process (e.g., Chandrasekaran's "generic task approach"). Another problem—highly relevant to our argument—concerns the lack of consensus between the different frameworks. It is generally agreed that an abstraction level above the one of programming is needed, a proposal which was originally made by Newell (1982) in his "knowledge level" hypothesis. But there is only little agreement on the details (see below). A last problem we would like to mention—and we consider it to be a major one—is the difficulty of applying task-level frameworks to a specific problem at hand, or stated differently, mapping expert behavior onto the frameworks.

Experience with such frameworks and careful analysis of the literature shows that — although such frameworks are a major step forward—important parts are still left open. In trying to reconstruct the cognitive processes of the knowledge engineers there are major gaps to be filled in. For example, in the description of the MOLE system (Eshelman, 1988) it is not specified *how* a diagnostic problem is mapped onto the problem solving method "cover-and-differentiate". It seems that this is largely done by "magic". What may be obvious to a good knowledge engineer who has worked with a specific framework for a long time, remains obscure to an outsider.

Approaches

There are essentially two ways in which these problems can be approached, namely (a) to extend or refine task-level frameworks, or (b) to study the process of mapping expertise onto task-level frameworks and develop ways of supporting this activity.

(a) Extending and refining task-level frameworks: It is clear that the final word on task-level frameworks is not out and that extensions and refinements in several ways are still possible, and to some extent needed. However, it is an interesting observation that, even though the debate about the details of the various frameworks has been going on now for more than half a decade, there is still a striking lack of consensus, as pointed out before. The main reason seems to be that the knowledge level, which corresponds directly to Dennett's "intentional stance" (Dennett, 1971), is highly underconstrained and no agreement can be achieved on the precise nature of the constraints. There is an intrinsic arbitrariness at the knowledge level which will not go away, no matter how long one discusses the problems. So, one is lead to the conclusion that, perhaps, the extending and refining the frameworks and to search for the "right" one to eventually reach consensus, may not be the most appropriate goal. Experience has shown that most task level frameworks can lead to usable practical systems. The main difficulty which remains in all frameworks as a matter of principle (not merely because they are insufficiently developed) is mapping the human expertise onto the framework. If the framework is highly abstract the mapping process is severely underconstrained and thus the application to a particular problem is very difficult. In order to provide a high degree of detail, a great many models will have to be developed and some of them will always be missing or incomplete. Even if many such detailed models were available, the *selection* of the appropriate structures would become another major problem for the knowledge engineer. But irrespective of the level of detail, the mapping problem remains. And it seems to hold for all frameworks that if you are clever, you can successfully apply the framework, if not, you fail. As far as we know this mapping problem has not or only marginally been addressed in the literature. And this leads us to the second line along which improvements are possible, which in our view, is a promising one.

(b) Study and support of the mapping process: Rather than trying to refine the framework by adding many types of models, we propose the following strategy. Start with a framework from the literature: which one is not very essential. Study the ways in which experienced knowledge engineers go about applying it to a particular problem. It is exactly in the analysis phase where psychological methods will be highly useful. Since the complexity of this task is so overwhelming, the construction of an automated system is out of the question. It should be pointed out that the goal in this case is not to develop a computational model of knowledge engineering expertise but to better understand the process in order to support it by appropriate computerized tools. Since task-level frameworks have been developed for modeling problem solving behavior, we will apply one of them, the componential framework, to this analysis. This is a kind of "recursive" analysis of a task-level framework, similar to what has been suggested by Bouchet, Brunet & Anjewierden (1989). So far nothing has been specified about the particular method by which the experts, in our case the knowledge engineers, should be studied. A more or less unbiased registration of the problem solving behavior of experts can be obtained with expertise-in-action protocols, that is, think-and-talk-aloud protocols of experts

produced during their problem solving. We will use such protocols as empirical material for mapping expert behavior onto the structures required by the componential framework.

A note on protocol analysis

Protocol analysis does not have a high reputation for being useful in the knowledge engineering process. So we need to argue why we suggest to use it all the same. First we will point out why it is not popular in expert systems circles and second we will show why we still think it should be used. Perhaps it is worth mentioning that there are different ways to perform protocol analysis. Two main approaches are content analytic methods and theory-based ones. The former essentially perform statistical or interpretative analyses on the text (e.g., Holsti, 1969; Mayring, 1990), while the latter provide theoretical frameworks (such as our task-level frameworks) onto which the protocol needs to be mapped. The remarks in the sequel only refer to the latter type. Moreover, the kinds of protocols we are interested in most are "expertise-in-action" ones.

Why protocol analysis is unpopular: There are a number of reasons why protocol analysis is not popular among knowledge engineers. First, to most knowledge engineers the benefit from protocol analysis does not justify the effort required (transcription, a lot of detailed work, see e.g., Ericsson & Simon, 1984). A factor contributing to the low cost-benefit ratio is the simple experience that protocol analysis is hard. The main reasons why protocol analysis is hard are: (a) There are many omissions in the text. They have to be completed somehow to arrive at a coherent "causal story" (see our example below) even though there is often insufficient support for a particular assumption. (b) Assignments of instances to abstract variables (concepts, categories) of the framework is initially to a large extent arbitrary, but gets successively more constrained. Information appearing later in a protocol may invalidate previous assumptions which may entail significant revisions of the initial assignments (truth maintenance). (c) Many things have to be kept track of in parallel. This means that there is a considerable "cognitive load" on the protocol analyst. In particular revisions are cognitively costly and tend to be avoided.

A second reason why protocol analysis is unpopular has to do with the large variations in expert behavior. In other words, since protocols from different experts on the same task tend to be very different, the respective analyses will also differ widely and it is not clear what should be included in a system. A third reason is the incompleteness of the knowledge gained from protocols. A protocol will only yield a very small part of the knowledge needed to build a system (e.g., Shadbolt & Burton, 1989). A fourth reason concerns methodological problems related to the nature of expertise, that is, it is questionable whether the method of protocol analysis can be used to assess real expertise since it always starts from verbal accounts (Nisbett & Wilson, 1977). A fifth one is that, given a particular protocol and a task-level framework, different knowledge engineers would come up with different interpretations.

Why protocol analysis can be beneficially applied: In spite of these difficulties we argue that protocol analysis can applied with great benefit. But we don't argue that it should be used exclusively. It is useful for certain purposes like getting an initial idea of expert behavior in a domain, in particular where control knowledge (when to do what), and basic difficulties are concerned. It can also be helpful in trying to determine where the expert or the potential user could be most effectively supported by tools. However, it can, for the reasons given before, by no means replace the use of other methods.

Let us now briefly go over the caveats against protocol analysis and suggest remedies where possible. The reasons given for why protocol analysis is hard can mostly be traced back to insufficient methodology and insufficient tools. For example, there are currently no guidelines as to how omissions are to be dealt with. We will suggest one way to deal with this problem. Second there are only limited tools available to manage revisions, but this is a practical problem rather than a fundamental one and no major difficulties seem to prevent the development of appropriate tools. The fact that many things have to be kept in mind in parallel could perhaps be supported by graphical means, but more investigation is needed on this point. Some tools for supporting protocol analysis already exist (e.g.. the "KOD Station™"—Vogel, 1990; tools within KADS or Shelley—Breuker & Wielinga, 1989, Bouchet, Brunet, & Anjewierden, 1989). They provide powerful sets of tools but little guidance on how to map protocol elements to predefined conceptual elements. Our own view is expert centered rather than tool centered, that is, in our view tool support should start from an analysis of the user (in this case the knowledge engineer) rather than from a catalog of things compiled by a knowledge engineer that he or she thinks might be useful to have.

The problem of variability in expert behavior cannot be readily solved. In any case this is a problem for knowledge acquisition in general and not only for protocol analysis, although in expertise-in-action protocols individual differences tend to show up most strongly. There are methods for integrating knowledge from different experts (e.g., Dym & Mittal, 1985) but they are not without problems (e.g., Shaw & Gaines, 1989). The completeness of the knowledge base can be assured by using additional methods. In any case, the goal of protocol analysis is not to cover all the knowledge needed to build a system. The methodological point, that it is questionable whether protocols do indeed capture expertise, is a fundamental one and will never completely disappear, but it seems to apply least (compared to other forms of protocols) to expertise-in-action protocols. The last point, the disagreement between different protocol interpreters, can be taken into account by an appropriate methodology (see below).

In conclusion, for most problems with protocol analysis it appears that there are practical solutions. But there is one highly significant reason why protocol analysis will—in spite of difficulties—remain an important method: that is its highly ecological nature. Once experts have been trained in thinking-aloud, the protocols can be taken in real-life settings; this is one of the major reasons for using them. Moreover, they provide very rich sources of information and experience has shown that the careful analysis of protocols can yield highly valuable insights (Ericsson &Simon, 1984).

Protocol analysis for knowledge engineering support

Goals and preliminary remarks

The knowledge engineer can be supported in several ways in the task of mapping expert behavior onto abstract models. The most extreme case would be an automatic system in which the knowledge engineer would only have to provide the input (i.e., in our case the expertise-in-action-protocols) to the system which would then do all the work, but this is clearly beyond what is doable today (it is also highly questionable whether this would constitute a sensible goal in the first place). A second kind of support is in the form of a number of computerized tools including guidelines on how to use them. A third form is a set of verbal instructions. Our approach is to start with an operational set of verbal instructions and use them—after careful evaluation and validation—as starting points to build computerized tools.

The proposal is based on the following assumptions:

(a) The goal of protocol analysis is to produce a knowledge-level representation of the problem solving process in terms of a task-level framework.

(b) A structured procedure based on the theoretical framework can be defined to ensure more reliable encoding.

(c) If systematic studies (e.g., comparisons among different knowledge engineers, among different frameworks) are to be performed, a method is needed to achieve continuous consensus.

The results reported in this paper have been derived from a number of empirical studies. However, they are still of a preliminary nature and a more systematic investigation is needed. The main purpose is to stimulate the discussion and get feedback for a next "iteration cycle". The theoretical points are illustrated with results of an empirical study

An example

In order to determine the precise nature of the difficulties with protocol analysis that were described theoretically above, a number of experiments were conducted. In particular we had experienced knowledge engineers analyze expertise-in-action protocols. We then analyzed the protocols of these sessions in terms of the componential framework to gain a better understanding of the process by which knowledge engineers analyze protocols and derived a methodology that tries to overcome the observed difficulties.

The example that will serve as an illustration has been taken from an interview with an expert repairing old-fashioned record players. Figure 1 gives an idea of the sort of device we are discussing about. A segment out of the protocol which the knowledge engineers had to analyze is given in Figure 2. Figure 3 shows an excerpt from a thinking-aloud protocol while performing this protocol analysis task. A number of difficulties can immediately be identified.

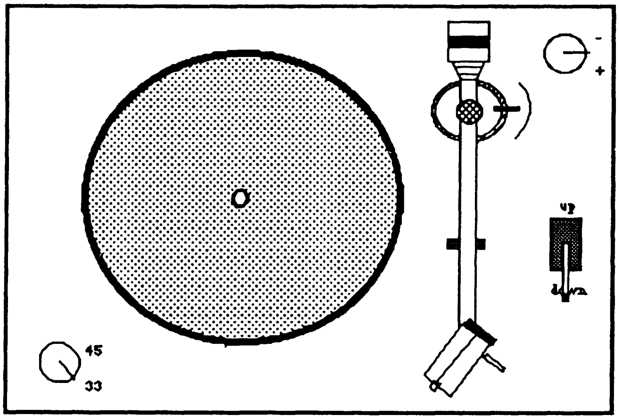


Figure 1: Example of a record player

Thinking aloud protocol: Record player diagnosis ("Expertise-in-action")
(E = Expert; I = Interviewer/Customer)

E: You said "it sometimes happens", "it jumps". Now does it jump towards the periphery or towards the center?

I: I think it exclusively jumps towards the periphery, no, towards the center, excuse me.

E: Towards the center. Well before it was jumping the other way—the same part of a piece was played over and over.

I: Yes but then it jumps ...

E: It is playing from the periphery towards the center, so when it jumps toward the periphery it repeats.

I: You are right, of course, it jumps toward the periphery.

E: Now there has been the suspicion that it would not stand horizontally, but it does not have that much weight, it would have to be really tilted. And now there is a first conjecture perhaps, that the bearing is somehow sticky and does not turn appropriately. But as far as I can tell this does not ...

I: ... seem to be the case?

E: Well, it is difficult to say because the forces are really small and it does not seem really sticky.

I: What is your reason for saying this?

E: If it lies on this ring for holding the arm then it is stuck and if you move it you can see its small uneven movements. This is because it does not rest on the bearing but on this little bolt.
And therefore I lift it slightly and then I move back and forth sideways and I feel that there is no real lateral resistance.

Figure 2: Transcript of a protocol (segmented)

The statement in line 1 of Figure 3 is typical for the non-monotonicity of the task. Statement 2 relates to the fact that more and more information has to be added into the task structure as the analysis proceeds. It is a clear indicator of missing editing facilities. Statement 3 refers to the way the knowledge engineer proceeds. He is using a list of generic problem solving methods (memorized or on a sheet of paper) which he can potentially use in the task structure. So he not only uses the information in the protocol, but consults the available methods in a sort of interactive process. Statement 4 represents an apparently underconstrained situation. There are several plausible possibilities and the knowledge engineer seems to postpone the decision (well, we'll see). It is not clear whether at this point he actually writes something down or just makes a mental "note" (which he might be likely to forget). In 5 the expert is trying to identify the domain model but he is uncertain. Statement 6 refers to his confusion and to the experimental situation itself (the "others" are the other participants in the experiments).

Thinking aloud protocol: Protocol analysis ("Expertise-in-action")

1. Well I might as well put it here, but perhaps I will have to change it again, later.
2. Well, this paper is going to be too small and I am not really sure what's generic or domain specific. Well, we'll see as we go along.

...

3. Now, "get-symptoms" decomposes into "ask-user" and something like "generate-new-observations".
4. Is this a subtask of "get-symptoms" or perhaps of something like "test-hypothesis" or doesn't he have a hypothesis yet? Well, we'll see.

...

5. Well, but that thing with the weight, that's more like causal reasoning, so I should put here two alternatives, well ...
6. space is getting tight, I am loosing track of things, I hope that the others are no better ...

Figure 3: Transcript of a protocol of a knowledge engineer analyzing a protocol (the lines are numbered for reference purposes only).

From this informal analysis there are a number of conclusions which can be drawn. But first let us analyze the "nature" of the task of protocol analysis itself by applying the componential framework to it. Protocol analysis is a task with inputs (the protocol) and outputs (the task structure) and thus lends itself to such an analysis. In order to determine the kind of support that can be provided for this type of task, we must look at its pragmatic features. Let us look at the protocol, that is, the input of the knowledge engineer's task: there is much irrelevant data (text which is not relevant to the task), the quality is limited due to the abstraction inherent in linguistic descriptions and due to practical problems (quality of recording and transcript), a lot of potentially relevant information is missing (due to the method of thinking-aloud), and additional information may be hard or impossible to get. The output, the task structure, is highly underconstrained and many interpretations are possible. Problem solving methods are very unstructured and a lot of common sense knowledge is involved. From this description it follows immediately that the protocol analysis task cannot be automated in the near future, since for most of the identified problems entailed by the pragmatic features there are no implementable methods available. What is needed is a way to filter and to complete the original data (in our case the raw text) in such a way that the mapping is more straightforward.

An obvious conclusion from the example shown in Figure 3 is that there is too much "cognitive load" on the knowledge engineer: there are things that have to be traced in parallel, there is the interpretation of the plain text on the one hand and the search of appropriate elements from the framework and the task-taxonomy on the other. This problem can be alleviated by computerized tools. This could also solve to some extent the non-monotonicity issue.

Such tools must allow for:

- easy modification (including consistency tracing) of the task structure
- ways of graphically represent the results to provide the knowledge engineer with an overview of the task structure.
- ways of browsing through the taxonomies (tasks and problem solving methods) to support top-down processing (thus enabling the knowledge engineer to do interactive processing, i.e. bottom-up (from the protocol text) and top down (from the concepts in the framework).
- ways of manipulating the protocol itself (marking, extracting, connecting, editing, etc.).

Some of this is already available in some tools or tool-kits such as KADS, Shelley or the KOD-station. While tools are certainly an essential part of any system for protocol analysis, more is needed. It seems that even with the availability of tools that essentially remove the large cognitive load, the basic problem of the mapping of the raw text to the framework largely remains.

A method of protocol analysis

Our proposal for a method of protocol analysis is based on a number of preliminary studies. The method is designed for protocols in which there is relatively little reflection on the part of the domain expert, as is the case in most expertise-in-action protocols. This implies that most of the expert's actual behavior can be reconstructed from the protocol.

The basic idea of the method follows a three-step procedure: Phase I performs a [K]nowledge-[G]uided [B]ehavior analysis in the form of a "re-narration" (KGB-phase); Phase II then tries to achieve [C]onsensus [I]n the [A]nalysis among different knowledge engineers (CIA-phase); Phase III [B]uilds [N]ew [D]escriptions in terms of the problem solving language of the task-level framework (BND-phase).

The "re-narration" phase produces from the raw text an "edited" text. A basic problem of behavior data is that on the one hand there is too much, and on the other there is not the "right" data available. These are characteristic pragmatic features of the input to the analysis task, as discussed above. Thus not only do we have the problem of data *re*-duction but also of data *pro*-duction, or data completion. The output of the first phase, that is the "edited" text, is a causally coherent story of the behavior of the expert, of what he does and thinks. The instructions which encode our method are as follows:

Instructions Phase I: "Re-narration" (rephrasing)

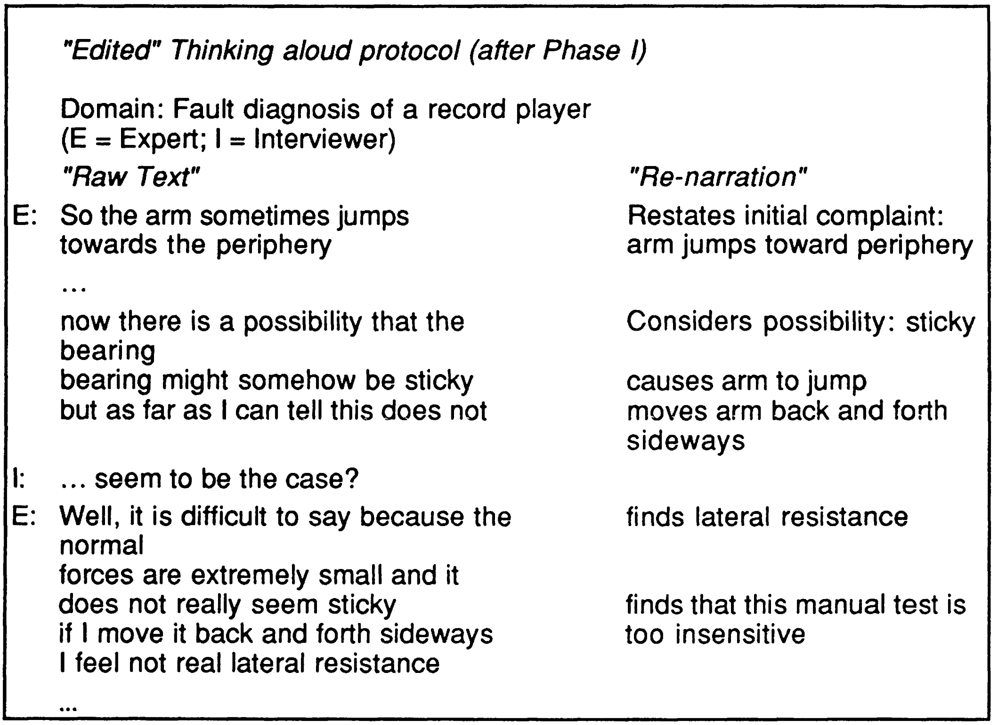
Try to "re-narrate" what is happening in the protocol using the following instructions. The purpose is to arrive at a coherent description of what the expert does, says and thinks that contains as little interpretation of his actions and thoughts as possible.

1. Eliminate "filler" sentences and obviously extraneous text.
2. Complete the text. In order to get a plausible story, actions may have to be inferred. Using common sense reasoning turn the text into a coherent causal story. It is a good strategy to visualize the situation.

3. If possible, use only a vocabulary which is close to behavior and to the domain terminology. Choose whenever possible terms from the vocabulary given (list of terms to be supplied). Avoid terminology from the taxonomy of the task-level framework. Unless it is explicitly mentioned in the protocol avoid terms like “hypothesis” or “goal”.
4. The segmentation should naturally be determined by the behavioral level of this description.

Figure 4 shows a sample output of phase I. A comparison of "edited" protocols (after phase I) of the same protocol segment by different knowledge engineers showed that there are a number of differences. This is undesirable if the protocols are to be compared and if conclusions are to be drawn about expert behavior. So whenever the protocols are analyzed by several knowledge engineers there must be a so-called “consensus session” after the first phase. In other words the knowledge engineers sit together, discuss their solutions and try to find a consensus on one solution (phase II).

It is surprising how quickly—at least in our experiments—agreement could be achieved. And agreement was virtually always unanimous, not by majority. Why this is the case, we do not know. It just seems that communication is conducive to discovering plausible solutions. Thus, preferably there should be at least two people doing the analysis. After agreement has been achieved among the raters, phase III can be tackled according to the following instructions:



"Edited" Thinking aloud protocol (after Phase I)

Domain: Fault diagnosis of a record player
(E = Expert; I = Interviewer)

	"Raw Text"	*"Re-narration"*
E:	So the arm sometimes jumps towards the periphery	Restates initial complaint: arm jumps toward periphery
	...	
	now there is a possibility that the bearing	Considers possibility: sticky
	bearing might somehow be sticky	causes arm to jump
	but as far as I can tell this does not	moves arm back and forth sideways
I:	... seem to be the case?	
E:	Well, it is difficult to say because the normal	finds lateral resistance
	forces are extremely small and it	
	does not really seem sticky	finds that this manual test is
	if I move it back and forth sideways	too insensitive
	I feel not real lateral resistance	
	...	

Figure 4: Output of Phase I: Re-narration

Instructions Phase III: Problem solving analysis phase (mapping the "edited" text onto the framework)

Try to map the "edited" protocol onto the task-level framework. There will be no need to go back to the original text at any point (except to revise the whole analysis). The goal of Phase III is a task-structure which describes the concrete task at hand.

0. It is a good strategy to do two tasks in parallel: (a) inserting the problem solving terminology into the protocol, and (b) developing the graphical task structure[1].
1. Try to identify the top-level task and decompose it into subtask as indicated by the "components" framework.
2. Add the generic labels to the subtasks and to the problem solving methods. Whenever possible, use labels from the taxonomy that is provided with the framework.
3. Whenever there is no appropriate label in the taxonomy, invent a new meaningful name.
4. Organize each problem solving method with a control regime.
5. For each problem solving method, identify the domain models, that is, the knowledge needed to perform the task. Again, in labeling use the predefined terms whenever possible.
6. For each task work out the epistemological problems and its associated pragmatic features and try to integrate them into the selection criteria of the identified problem solving methods.
7. Try to account for as many statements in the “edited” protocol as possible.

The output of phase III (see Figures 5 & 6) thus achieved is subject to less variation according to our preliminary experiments if this method of protocol analysis is applied. But more systematic tests are needed. The next step will be to follow the second phase up with a second consensus session. It would be of interest to see if consensus can also be achieved as easily as after the first phase. Moreover, the whole procedure would have to be done with knowledge engineers from different theoretical backgrounds (KADS, Componential Framework, Generic Tasks, Problem Solving Methods etc.). If it turns out that the inter-group reliability is high, this method would provide a good vehicle to be used in expert system construction.

Since methods of protocol analysis vary with the nature of the protocol (e.g., expertise-in-action, structured or a focussed interview, expert-guided problem solving, unstructured conversation), the protocol needs to be classified first. Our method will only work for expertise-in-action or behavior observation protocols. The construction of a causal story largely relies on this fact. It also restricts the applicability of the method. However, expertise-in-action is among the most valuable data for gaining more insights into the ways humans deal with the pragmatic features of a task.

[1] Some of the steps and products mentioned in the instructions are of course not required for the use of task-level frameworks, but serve purposes of documentation and enable discussion with other knoweldge engineers and experts.

Output of Phase III: adding problem solving terminology (protocol)

Domain: Fault diagnosis of a record player

Restates initial complaint: arm jumps toward periphery	*get-initial-symptoms*
Has suspicion: sticky bearing causes arm to jump	*cover-with-hypothesis:* sticky bearing
moves arm back and forth bearing sideways	*test-hypothesis:* sticky
finds lateral resistance normal negative,	*evaluate-test-result:*
	but test insufficient
finds that this manual test is too insensitive	hypothesis not rejected but given less weight

Figure 5: Output of Phase III: Problem solving analysis phase

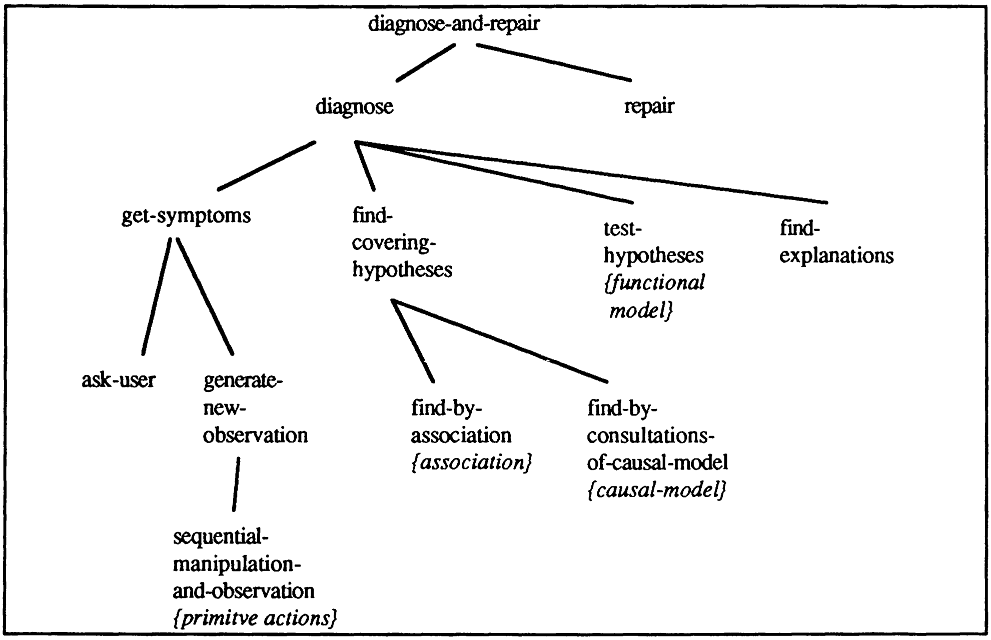

diagnose-and-repair
diagnose
repair
get-symptoms
find-covering-hypotheses
test-hypotheses {functional model}
find-explanations
ask-user
generate-new-observation
find-by-association {association}
find-by-consultations-of-causal-model {causal-model}
sequential-manipulation-and-observation {primitve actions}

Figure 6: Task structure (including the problem solving process) of the fault diagnosis task

Validation

It is clear that we have not said much about validity yet. In our case validity means to what extent the analysis reflects the actual thinking processes of the expert. Validity could be assessed by showing the final analysis back to the expert and getting his reactions. The expert could also contribute to the consensus sessions. However, it has to be kept in mind that it is *not* our goal to adequately describe the expert's reasoning, but that we aim at building expert systems, which is a very different story. For the present purposes we are therefore not worried too much about validity. What is more important is that the method is *ecological* (i.e. applicable to real-life problem solving) and *pragmatic* (i.e. leads to systems).

Summary and conclusions

We first introduced a distinction between two extreme positions, the "engineering" and the "expert" one. We argued that both suffer from significant problems and suggested that the use of task-level frameworks, if applied appropriately, provides a means for getting the best of both worlds. We identified a largely neglected problem, namely that of mapping expertise onto task-level frameworks. The goal of our current efforts is to support precisely this process. As a starting point we studied how expertise-in-action protocols can be mapped onto task-level frameworks. We developed a methodology to make protocol analysis more effective and more suitable for supporting this mapping activity. Given the high complexity of this activity, support will consist of a set of guidelines, mainly in the form of verbal instructions, plus a set of computerized tools. There will be little in terms of automated systems.

It is concluded (a) that protocol analysis is a good starting point for developing tools to support the entire knowledge engineering process — *if* appropriate methods are available, and (b) that methods are only appropriate if they are both *ecological and pragmatic.*

The methods developed on the basis of the approach outlined in this paper (using protocol analysis) should not be viewed as final or complete. Rather they serve as a starting point for further development: knowledge engineers must be studied carefully over extended periods of time in how they use the tools. What parts do they use most, what parts not at all, and—very importantly—how does the nature of their task change as they are using the tool. So, protocol analysis is only a starting point, but because of its ecological nature we expect that the tools developed on this basis can be naturally extended.

Acknowledgement

This work was partly supported by the University of Zurich, Tecan AG Hombrechtikon, and the "Swift AI Chair" of the Free University of Brussels.

References

Abelson, R.P. (1981). Whatever became of consistency theory. *Proceedings of the 3rd International Conference of the Cognitive Science Society.*

Bouchet, C., Brunet, E., & Anjewierden, A. (1989). Shelley: An integrated workbench for KBS development. *Proceedings of the 9th International Workshop on Expert Systems and Their Applications,* Avignon, 303-315.

Breuker, J.A., & Wielinga, B.J. (1989). Models of expertise in knowledge acquisition. In G. Guida & C. Tasso (Eds.). *Topics in expert system design*, 265-295. Amsterdam: Elsevier.

Chandrasekaran, B. (1986). Generic tasks in knowledge based reasoning: High level building blocks for expert system design. *IEEE Expert,* Fall, 1986, 23-30.

Chi, M.T.H., Feltovich, P.J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science, 5,* 121-152.

Clancey, W. (1985). Heuristic classification. *Artificial Intelligence, 27,* 298-350.

Davis, R. (1984). Diagnostic reasoning based on structure and behavior. *Artificial Intelligence, 24* 347-411.

de Kleer, J. (1984). How circuits work. *Artificial Intelligence, 24,* 205-281.

Dennett, D. (1971). Intentional systems. *The Journal of Philosophy*, 68, 87-106. (reprinted in J. Haugeland (Ed.) (1981). *Mind design* (pp.220-242). Montgomery, VT: Bradford Books.)

Diaper, D. (Ed.) (1989). *Knowledge elicitation.* Chichester: Ellis Horwood.

Dörner, D. (1976). *Problemlösen als Informationsverarbeitung (Problem solving as information processing).* Stuttgart: Kohlhammer.

Dym, C.L. & Mittal, S. (1985). Knowledge acquisition from multiple experts. *AI Magazine, 6* (2).

Elstein, A.S., Shulman, L.S., & Sprafka, S.A. (1978). *Medical problem solving: An analysis of clinical reasoning.* Cambridge, MA: Harvard University Press.

Ericsson, A., & Simon, H.A. (1984). *Protocol analysis: Verbal reports as data.* Cambridge, MA: MIT Press.

Holsti, O.R. (1969). *Content analysis for the social sciences and humanities.* Menlo Park, CA: Addison-Wesley.

Holyoak, K. (1990). *Symbolic connectionism: toward third-generation theories of expertise.* Techreport UCLA-CSRO-90-14. (to appear in I.A. Ericsson & J. Smith (Eds.). *Toward a general theory of expertise: prospects and limits.* Cambridge, MA: Cambridge University Press.)

Mayring, P. (1990). *Qualitative Inhaltsanalyse: Grundlagen und Techniken* (2., durchges. Auflage). *(Qualitative content analysis: Foundations and techniques).* Weinheim: Deutscher Studien Verlag.

McDermott, J. (1988). Preliminary steps toward a taxonomy of problem-solving methods. In S. Marcus (Ed.). *Automating knowledge acquisition for expert systems* (pp. 225-256). Boston, MA: Kluwer.

Neisser, U. (1982). *Memory observed.* San Francisco, CA: Freeman.

Newell, A. (1982). The knowledge level. *Artificial Intelligence, 18,* 87-127.

Newell, A., & Simon, H.A. (1972). *Human problem solving.* Englewood Cliffs, NJ: Prentice Hall.

Nisbett, R.E., & Wilson, T.D. (1977). Telling more than we can know: Verbal reports on mental data. *Psychological Review, 84.* 231-259.

Norman, D.A. (1988). *The psychology of everyday things.* New York: Basic Books.

Patel, V.F., & Groen, G.J. (1986). Knowledge-based solution strategies in medical reasoning. *Cognitive Science, 10,* 91-116.

Shadbolt, N., & Burton, A.M. (1989). The empirical study of knowledge elicitation techniques. *SIGART Newsletter, 108,* 15-18.

Shaw, M.L.G., & Gaines, B. (1989). Knowledge acquisition: Some foundations, manual methods, and future trends. *Proceedings of EKAW-89, Paris, France.* 3-19.

Simon, H.A. (1969). *The sciences of the artificial.* Cambridge, MA: MIT Press.

Sol, H.G., Takkenberg, C.A., & DeVries Robbé, P.F. (Eds.) (1987). *Expert systems and artificial intelligence in decision support systems.* Dordrecht: Reidel.

Steels, L. (1990). Components of expertise. *AI Magazine, 11*(2), 28-49.

Vanwelkenhuysen, J. (1989). *TroTelc: An expert system troubleshooting printed ciruit boards.* VUB AI Memo 89-17. Free University of Brussels, Belgium.

Vanwelkenhuysen, J., & Rademakers, P. (1990). Mapping a knowledge level analysis onto a computational framework. *Proceedings ECAI* (pp. 661-666). Stockholm, Sweden.

Vogel, C. (1990). KOD: A method for knowledge acquisition and modeling. Tutorial at the *Tenth International Workshop on Expert Systems and Their Applications.* Avignon, France.

Waldmann, M.R., & Weinert, F.E. (1990). *Intelligenz und Denken (Intelligence and thinking).* Göttingen: Hogrefe.

Knowledge Acquisition and the Interpretative Paradigm

Dieter Fensel
Institut für Angewandte Informatik und Formale Beschreibungsverfahren
University of Karlsruhe, P.O. Box 6980, 7500 Karlsruhe, Germany
e-mail: Fensel@aifb.uni-karlsruhe.de

Abstract. The problems arising during the early steps of knowledge acquisition are similar to problems in social research based on the interpretative paradigm. Therefore the article shows the transfer of principles, methods, and techniques from social science to knowledge acquisition. This transfer offers a methodological foundation for the gathering and interpretation of knowledge and a framework which guides the application of the different techniques for the different tasks of incremental and model-based knowledge acquisition.

1. Introduction

> "Knowledge acquisition, however, is not only the >>transfer of expertise<<. Knowledge acquisition is a creative process in which the systems builder constructs qualitative models of human behaviours." [Musen]

Knowledge acquisition is the process which gathers and models knowledge for an expert system. The problems [Ber87] arising in its course are well known:

- There is an insufficient understanding of experts´ abilities to solve problems in their field.
- Expertise is based on skill and implicit knowledge. Therefore it cannot directly be asked for.
- Experts tend to justify their behavior like any other person. They try to construct good reasons for their actions even if they do not know them.
- Experts are experts in solving problems, not in explaining their solutions.

The literature about knowledge acquisition offers some techniques to overcome these problems but there is still no clarity "***when*** and ***where*** to use these techniques" [NPB91]. This lack of methodological foundation and framework which connects the different techniques with the different tasks of the whole knowledge acquisition process is called the "mismatch problem" [NPB91].

The difficulties described above relate knowledge acquisition to sciences such as psychology or sociology. Both disciplines have a long tradition of questioning and observing people. As a consequence they are familiar with problems similar to those described above. This article attempts to show how knowledge acquisition can benefit from the principles, methods, and techniques of these fields. Social research based on qualitative methods only will be subject to discussion here as two major assumptions of social science working with quantitative methods, and oriented towards the normative paradigm, do not fit the described problems.

- Social research based on quantitative methods checks ***given hypotheses*** by conducting surveys and observations.
- It works with a ***representative sample*** and tries to get general statements by applying statistical methods.

Knowledge Acquisition is not mainly a process of checking already given hypotheses, and normally, there is no representative sample of data.

The following section presents the main ideas of social science working with qualitative methods. An introduction to model-based and incremental knowledge acquisition is given in the third section. The last section presents the application of principles, methods and techniques of social science in the early phase of knowledge acquisition.

2. The main ideas of social science working with qualitative methods[1]

> If mathematical theorems are related to reality they are not sure and if they are sure they are not related to reality. [Einstein]

The main idea of the ***interpretative paradigm*** [Blu66, Wil71, Blu73, May90a] is that human behaviour is not specified by fixed social or cultural rules and norms as is asserted by the normative paradigm. Every social activity or reaction of a person is based on a specific interpretative process. People give situations a special meaning, and their activities are based on this meaning. Therefore, in order to understand their activities, the researcher must understand their interpretation. This paradigm has some consequences on the goals of social research as described below.

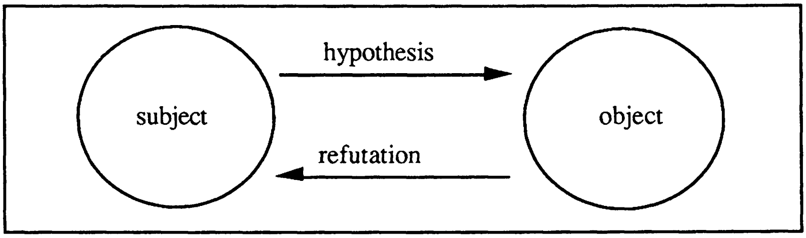


Figure 1. Separation of subject and object.

Overcoming the separation of subject and object: The main principle of the normative paradigm is the separation of the researcher (***subject***) and the research ***object***. This strict separation seems to be necessary to get impartial results. Every subjective influence of the

[1] See [Att74, Bot75, Bru66, Den70, Den89, Dou73, Dou85, Ras79, and Wil71] as introductions to principles and methods of qualitatively oriented social science written in English.

researcher is regarded as a disturbance which must be minimized by standardization of the elicitation process, for example. Based on this separation the coherence of the research process is achieved by hypotheses which are the premisses of the process. The presumed hypotheses are the means of connecting the two disjunct parts of the research process. The research activity attempts to refute them. No answer is given where the hypotheses are coming from.

However, this principle of the classical natural sciences is not applicable to social sciences for two reasons:

1. The examined object is a subject itself. The reduction of a human to a stimulus-response-object ignores the fact that human reactions are conveyed by meaning. In other words, the behavior is not direct, and humans do not only react. The behavior is a result of an interpretation process. In addition, humans are active and also create meaning.

2. The researcher`s subjectivity is an important tool for the research process. As opposed to natural science, the object of the investigation is not an extraneous object, but also a subject which is a member of the same or a similar social community as the researcher is. This common basis is <u>the</u> premise for ***understanding*** the object of the research. A substantial basis of the research process will be lost by the elimination of the researcher's subjectivity.

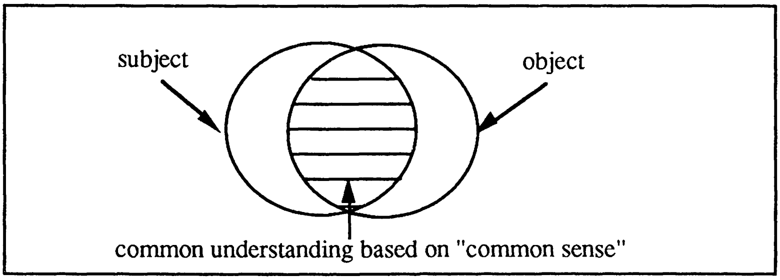


Figure 2. Overcoming the separation of subject and object

"Understanding" as the goal of research*:** The task of the classical natural science is to ***explain natural phenoma by causal reasoning. Asking for the meaning does not make sense. This situation changes when humans become the objects of research. They interpret reality and produce meaning by their activity. Examining a person means: try to understand him or her. This may create some new problems for the research process:

- The researcher and his object do not speak the same language;
- They speak the same language but interpret it differently;
- Often there is a difference between the articulated meaning and the real meaning, e.g. somebody is lying;
- Often there is a difference between conscious and tacit meaning.

The researcher consequently needs a "***second socialization***" that teaches him to think like the person he wants to examine.

Theory production vs. theory revision: An important principle of qualitatively oriented social science is the ***openness*** of the examination. I want to show this principle by contrasting it with quantitatively oriented social science. The task of elicitation and interpretation of data is to refute presupposed hypotheses. The researcher can only perceive features of the object of research which are within the range of his hypotheses, their operationalizations, and measurements. The only influence the subject of research can have is on refuting some hypotheses. In qualitatively oriented research, however no special theory or hypothesis is used as premise of the research process. Based on some knowledge about the object of research (prejudices), which must be kept open and flexible, the researcher must try to become familiar with the new field. Hypotheses can be created during or after the examination. This has two implications:

- The goal is not to verify given hypotheses but to create new ones.
- It is possible to gain new points of view about the object. More than only those features may be found which become prominent due to the presumed hypothesis of the researcher and his community.

Research as communication and context sensitivity of its results: Quantitatively oriented social science postulates a qualitative difference between common sense understanding and scientific explanation. It therefore works with standardized interviews, for example, in order to reduce the subjective influence of the researcher as much as possible. Qualitatively oriented social science claims on the contrary that this interaction is the main tool for understanding the object of research and is not an error to be eliminated. No standardized interview but an intensive and open talk enables the researcher to learn to think in the same way as his research object does. Therefore scientific understanding is not a qualitatively different way of understanding but an extented and methodologically well-defined common sense understanding (cf. [Lam88]).

A similar methodological problem of the stimulus-response approach is that it ignores the ***context sensitivity*** of its results. For example, thorough examinations of surveys show the context sensitivity of peoples´ opinions. They depend on the communication partner and on the situation the interaction takes place in. This is not a problem of opportunism, but it shows human complexity and ability to learn. A research method which produces an abnormal context creates abnormal results. Reducing the influence of the researcher and the context of the examination to an error to be minimized produces an uncontrolled influence on the object of research. Therefore qualitatively oriented social science assumes that data do not exist independently of their creation but as a result of a specific interaction between the researcher and the object of his research. They are no ***literals*** but ***indexical expressions*** [Wil71]. The influence of the researcher cannot be reduced but must be documented as an integrated part of the result.

The research activity is an iterative process: Understanding is an iterative, approximative and faulty process. A first judgement is the foundation for the first perception and interpretation of the object of research. This interpretation creates a new understanding. This procedure must be repeated several times. If the process works well it leads to an increasing understanding. Because of the different types of socializations this process is infinite. The process can be illustrated graphically by a spiral called ***hermeneutic circle***.

The single-case approach: The research methods of the normatively oriented social science are based on a representative sample of the examined domain. It establishes the general features of this domain with the application of statistical procedures. Therefore it needs a large and representative sample to reduce the statistical error of its results. This requires a standardization of the elicitation process and data reduction as an interpretation technique, for example multiple-choice tests. The collecting and interpreting of representative samples is not suitable for understanding because it is a very time-consuming process. Qualitatively oriented social science uses the ***single-case approach.*** The researcher does not collect a representative sample but tries to find some typical or interesting members of the examined group. Not reduction to standardized attributes but the explication of the whole case with all its relevant features guides the research process.

This shows the similarity of qualitative social science and the expert system building process: "A single proband is not regarded as an insignificant and exchangeable member of a sample and as a set of attributes defined by the researcher ... but the individuum is regarded as ***an expert of understanding and interpreting everyday life.***" [Lam89, p. 6]

3. Incremental and model-based knowledge acquisition

The knowledge engineering literature shows the contrast of rapid prototyping and life-cycle based approaches similar to earlier discussions in the field of software engineering. Rapid prototyping has the advantage that it quickly leads to a running system which can be used to evaluate the gathered knowledge. But it also leads to unstructured solutions, thus impeding understandability and maintenance of larger systems. It also needs to simultaneously consider knowledge aspects and their implementation in the used knowledge representation formalism. Therefore the view on the knowledge is determined by the chosen implementation language. Contrary to rapid prototyping, the application of the life-cycle paradigm leads to a well structured development process and result. But this result is often not the one wished by the client. Only the last phase produces a running system which can be used to evaluate the requirements and their realization. Subsequently, both approaches are discussed, and a method of development which combines both approaches is described (cf. [AFS90, ALF91, FAL91]); this method tries to combine their advantages by avoiding their disanvantages.

Model-based knowledge acquisition has two characteristical features. First, it distinguishes between the analysis and design/implementation phases. The expertise is analysed at the ***knowledge level*** independently of any details dealing with the subsequent technical realisation. The efficient realisation of the expertise at the ***symbol level*** is the task of a separate phase. The result of the analysis phase - the knowledge acquisition - is a ***model*** of the expertise. Secondly, knowledge acquisition is further subdivided into two different steps. The task of the first phase is to get the so-called ***process knowledge*** [Mus89a, Mus89b], which corresponds to the problem-solving method used by the expert. The result of this phase is a model of the problem-solving methods and their interactions. This is a very critical situation because the knowledge engineer must become familiar with a new field and task, and, besides, a great deal of the problem-solving capacity of the expert is based on tacit knowledge or skill. This model serves as guidance for the acquisition of field-specific ***content knowledge*** [Mus89a, Mus89b] in a second step. The domain knowledge is gathered as needed for the problem-solving process. The result of the analysis phase is a model of the problem-solving method expressed in terms which are specific of the field. This model is called a ***conceptual model*** [Mus89a, Mus89b].

[Mor87] describes knowledge acquisition as an ***iterative***, ***approximative*** and ***faulty*** process and therefore calls it ***sloppy modeling***. There is no complete and perfect model of the expertise in the expert`s head, but the expertise is based on experience, vague heuristics and tacit knowledge. Therefore a model is created as a ***result*** of the knowledge acquisition process. Knowledge acquisition shares these features with any other modelling process. Methods and tools for knowledge acquisition must therefore meet the following requirements:

1. They must allow iterative modelling.
2. They must be suitable for detecting errors.
3. They must support error corrections.

The incremental development of a prototype, its evaluation and modification meet the above given features. The disadvantages of rapid prototyping can be avoided if the prototype is not meant to become the real final system, but is only used for analysing the expertise in the way suggested by ***explorative prototyping*** (cf. [Flo84]).[2]

In order to integrate the model-based and the prototyping point of view, an ***model-based and incremental knowledge engineering*** process consisting of three phases is suggested. Like in a life-cycle oriented approach three different phases are distinguished: analysis, design, and implementation. Specifying the result of the analysis phase, the conceptual model, in an operational language allows to integrate explorative and throw-away prototyping during the analysis phase.

In the following the first phase, the analysis phase, is discussed in more detail. It is called ***model-based and incremental knowledge acquisition***. Before discussing the details of it, the research paradigm to be followed by the knowledge acquisition process must be determined. This is connected to the question: is knowledge acquisition related to natural sciences or to the arts? The normative paradigm and its separation of subject and object as practiced in natural science is unsuitable for the understanding process as it is conceived in human sciences. The interpretative paradigm which deals with understanding is unsuitable for causal explanations. [Bro89] proposes a complementary application of both paradigms due to the ambiguity of the research topic, the human expertise. The content of the expertise is "knowledge" of natural phenomena, like knowledge of mechanical faults or illnesses. This knowledge exists as a human quality and hence as an object of cognitive sciences. The general justification for the qualitative methods is the fact that, beside the model of a natural field, a model of the problem-solving process in the human being is needed.

This general statement must be restricted by the differentiation of knowledge acquisition in several steps. There are the three steps: ***theory production***, ***theory instantiation***, and ***theory evaluation***. Only for the first step an analogy to social sciences (oriented on the interpretative paradigm) can be drawn. The other two steps violate the central principles of

[2] She distinguishes three main kinds of prototyping: throw-away prototyping during the analysis phase, throw-away prototyping during the design/implementation phase and the stepwise development of a prototype to the target system. She calls the different kinds ***explorative***, ***experimental***, and ***evolutionary*** prototyping [Flo84].

openness and ***theory production.*** In the following section the first step, which is called ***knowledge elicitation*** according to [Add87][3], will be discussed in detail.

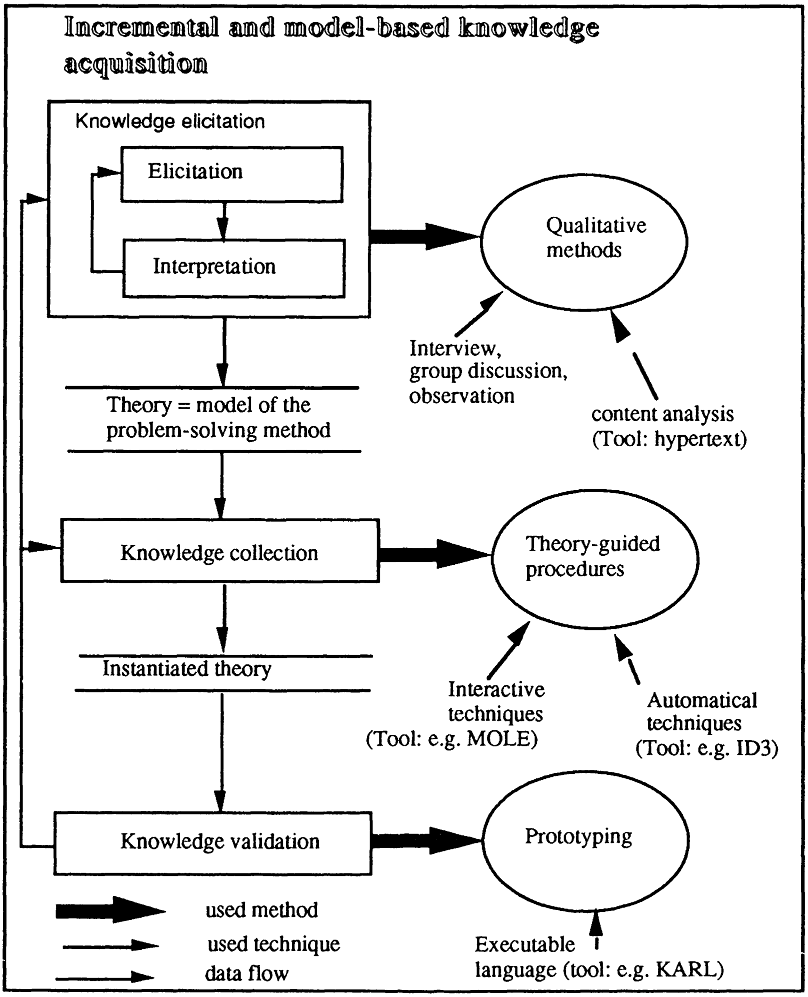

Incremental and model-based knowledge acquisition
Knowledge elicitation
Elicitation
Interpretation
Qualitative methods
Interview, group discussion, observation
content analysis (Tool: hypertext)
Theory = model of the problem-solving method
Knowledge collection
Theory-guided procedures
Instantiated theory
Interactive techniques (Tool: e.g. MOLE)
Automatical techniques (Tool: e.g. ID3)
Knowledge validation
Prototyping
Executable language (tool: e.g. KARL)
used method
used technique
data flow

Figure 3. Incremental and model-based knowledge acquisition

The second step, the collection of ***content knowledge***, is a theory-guided instantiation of a formal theory with domain-specific terms. The model of the problem-solving method is reformulated with structures, concepts, relations, and constraints of the specific domain (cf.

[3] "The terms elicitation and acquisition tend to be used interchangeably. However, there is a clear distinction between elicitation and acquisition in the creation of a model. Elicitation is the process of developing a model by interviewing the experts and observing their environment; it is theory formation or model design. Acquisition is the process of collecting the detailed information (facts) which fit into the framework defined by the model." [Add87]

[BWS87]). This violates the principle of openness and therefore prohibits the application of qualitative methods or techniques. This is no disadvantage. The guidance by a theory can be exploited for building powerful acquisition tools. For example, the model of the problem-solving method guides the collection of domain knowledge by interactive tools like e.g. MOLE, MORE, SALT [Mar88], and PROTEGE [Mus89a, Mus89b]. Automatical procedures like conceptual clustering (cf. [MiS84]) can generalize examples to taxonomies of concepts and relations.

The domain-specific theory is checked for correctness and completeness during the third step. It tries to ***refute*** a given theory and therefore does not fit the interpretative paradigm either. There are less operational criteria which can be applied for this process. This is comparable to traditional software engineering where no effective and efficient procedures exists to show the correctness of a program. A pragmatical answer to this problem is ***prototyping***. An implemented program is used to check whether the specification satisfies the intentions of the clients. Therefore, an operational language for the conceptual model is necessary to allow explorative prototyping. Currently some research is done for a language (cf. [AFL91, AHS90, FAL91, Lin90, KVS91, Wet90]) which enables the formulation of an operational conceptual model based on the different knowledge types proposed by KADS (cf. [BRW89]).

4. Knowledge Elicitation

The goal of knowledge elicitation is a model of the problem-solving method.[4] The human expertise which is partially based on skill, experience, and tacit knowledge must be transformed into a formal and well understood theory. In the following the application of principles, methods, and techniques of qualitatively oriented social science will be discussed. A similar idea and examples for its succesful application can be found in [BeH89]. Bell and Hardiman, however, do not separate the different phases of knowledge acquisition because they do not use the model-based approach. An analogous suggestion is made by [Kor87], which refers to the distinction of "formal" and "narrative thinking".

4.1. Principles for Knowledge Elicitation

Knowledge elicitation is not only concerned with "***what***" the expert is doing, but "***why***" he is doing something, and what meaning it has for him. The goal is to find general patterns of his behavior, the so-called problem-solving methods. They are to be elicited, systematized, and generalized. The elicitation of the so-called process knowledge also shows the main difference to traditional software engineering. Not only "***what***" the system should do but also "***how***" it is done is a topic in knowledge elicitation. The "how" cannot be solved simply by an algorithmic solution which is independent of a task and a domain. One needs specific knowledge, like heuristics, on the part of the expert in order to build an efficient solution (cf. [Par86, AFS90]). The principles of knowledge elicitation are:

Understanding: the task of the knowledge engineer can be described as a "second socialization". He must learn new words and new meanings for old words. Based on his

[4] This is similar to the idea of a "grounded formal theory" in social science (c.f. [Lam88]).

common understanding and experience he must overcome his own way of thinking to learn to think like the expert.

Openness and theory production: ***Process knowledge*** is knowledge implicitly encoded in human problem-solving abbility. It is based on skill and implicit knowledge. Therefore, a formalized model of the problem-solving process can only be achieved as a result of an open eliciting and modeling process. Any pre-judgement must be handed very carefully, because a wrongly chosen problem-solving method will strongly decrease the effectivity of the subsequent build system. In addition, this problem-solving method leads the further collecting of the content knowledge.

Searching for implicit meaning: An important part of the expert`s problem-solving capability is based on implicit knowledge. Therefore, not only his intentions have to be considered, but also his unconscious motives. Understanding the expert means in particular understanding unconscious parts of his expertise.

Understanding by communication: The influence of the knowledge engineer on the expert is not a distortion which must be reduced. Explaining, structuring, and generalizing the expertise during the knowledge elicitation process extends its reliability, validity, and applicability. The learning process of the expert can improve the elicited knowledge. Sometimes he learns so much about his own skills that an expert system is not needed anymore.

Process orientation: The modelling of expertise is an iterative, approximative, and faulty process. Therefore the activities of knowledge elicitation, knowledge collection, and knowledge evaluation have to be repeated several times. As modelling a problem-solving method is an infinite process, knowledge elicitation by itself is an iteration of the two activities ***elicitation*** and ***interpretation***.

Single case approach: Knowledge elicitation is usually based on the investigation of a single expert, and not on that of a representative sample of experts. For this reason, the information has to be interpreted and not reduced.

4.2. Methods and Techniques

This section does not give a complete survey of all methods and techniques of knowledge acquisition (cf. [BRW84, DaH88, COR89, KaL90, Kid87, McH89, OlR87]) but shows potential applications of methods developed by social scientists. Especially content analysis techniques - "Qualitative Inhaltsanalyse" and "Objektive Hermeneutik" - are new possible means of improving the interpretation step during knowledge acquisition. Because of the space limitation the article only touches on the methods of group discussion and observation (cf. [Gir84, Lam89, Man73]).

4.2.1. Interview

The ***standardized interview*** is an unnatural way of communication [Sch62], it is a means of trying to get context-free and objective information about the examined object. Like in an experiment, the central requirement is to obtain the same result under the same conditions.

Consequently the questions are determined before the interview and the interviewer provides the stimulus the subject has to react to. Often possible answers are supplied, and the interviewee has to choose among them. The ideal is a procedure which is free from all inter-subjective influences. This kind of interview violates central principles of qualitatively oriented social research and is therefore not one of its methods.

The ***narrative interview*** [Sch76] is the opposite type of interview. The person being interviewed is invited to tell a story about a given topic. He has the active part during the interview. The idea is that this active role of the person being questioned allows the researcher to detect the implications which the story has for this person. The interviewed person decides on which part of the topic he wants to discuss and stress. Everybody who tells a story wants to make the meaning which the story has for him plausible. The selection of the parts and their emphasis can help to comprehend the narrator`s understanding. In addition, one can postulate general patterns for the structure of every story. Every deviation of this scheme indicates hidden meanings which can be recognized. The general patterns of a story produce some pressure on the narrator - e.g. he has to choose between figures, to decide which details should be told, what should be shortened - therefore the interviewer can restrict his interaction to stimulation.

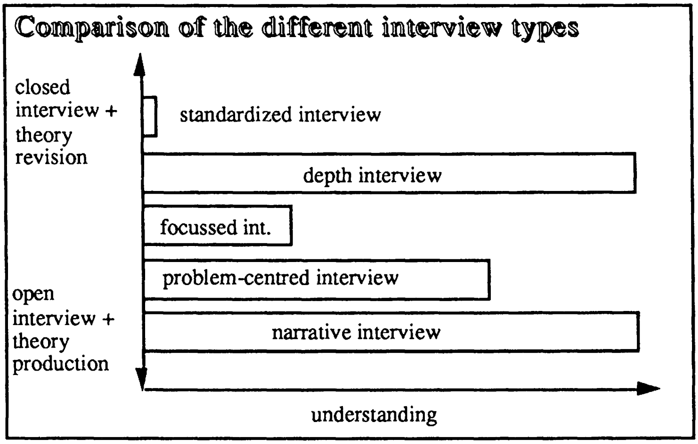

Comparison of the different interview types
closed interview + theory revision
standardized interview
depth interview
focussed int.
problem-centred interview
open interview + theory production
narrative interview
understanding

Figure 4. Comparision of the different interview types

The main purpose of the ***problem-centred interview*** [Wit82, Wit89] also is the production of hypotheses. On the other hand a suggested hypothesis is a premise which becomes modified during the interview. The suggested hypothesis is a guideline for the interview, providing it with structure and direction. The main difference to the narrative interview lies in the more active role of the interviewer. He tries to guide the flow of the narration by using examples of stories and other techniques. The individual steps of the problem-centred interview are [Spö89]:

1. Starting the discussion.
2. General survey.
3. Specific survey:
   - ***Reflection***: the researcher formulates his own understanding and accepts possible corrections of it.

- ***Questions***: he asks questions about contradictory and incomplete parts of the interview.
- ***Confrontation***: the interviewee is confronted with his contradictions.

4. Ad hoc questions: Asking about points of the interview guide which have not been covered yet.

Further interview types are the ***focussed interview*** [Lam89, Spö89], which contains qualitative and quantitative elements, the ***deep*** or ***intensive interview*** [Lam89], the ***receptive interview*** [Lam89], the ***intensive interview with two researchers*** [Lam89] and the ***ethnographic interview*** [Spö89].

Application of the different interview types

The degree to which an interview type conforms to the principle of qualitative social science or knowledge elicitation can be used to construct a framework which guides its application during different activities of knowledge elicitation. For example, the effort and closedness of the techniques leads to the following order of their application. This order also shows how focussed the research topic must be to allow the application of a interview type: narrative interview, problem-centred interview and focussed interview. This contributes to overcoming the mismatch problem as complained by [NPB91].

4.2.2. Group discussion and observation

There are mainly three arguments for applying ***group discussion*** as an elicitation technique. First, it is possible to get a great amount of data in short time. Secondly, the members of a group discussion stimulate each other, thus hidden meanings are more easily articulated than during an interview. Thirdly, the creation of meaning and the ability of understanding it is the result of social processes, they are results of interactions of many people. Group discussion offers a better simulation for these social processes than, for example, an interview. Interviews always produce an artificial context. There exists a lot of different kinds of discussion techniques. [Lam90, p. 142] proposes the following features to catalogue them:

1. Criteria applied to select the group members:
   - homogeneous vs. inhomogeneous groups
   - artificial vs. natural groups
   - groups with related and with unrelated members, e.g. randomly chosen samples vs. a family.
2. Discussion style:
   - thematical structuring vs. openness
   - formal structuring vs. openness
   - neutral or involved discussion leader
   - directly or indirectly guided discussion

On one hand ***observation*** is certainly the best technique to get familiar with a domain. On the other hand it is the most time-consuming technique. In the literature about knowledge acquisition this technique is mainly discussed as protocol analysis. The normatively oriented observation technique, which tries to create a context from an experiment in natural sciences is discussed in [Kön73]. The ***unstructured*** and ***structured participative observation*** oriented towards the interpretative paradigm is discussed in [Lam90, Gir84].

4.2.3. Content analysis

The topic of the ***content analysis*** is the analysis of past communications like texts, tapes, or videos. Research guided by the normative paradigm looks for directly visible features of interaction whereas research oriented towards the interpretative paradigm searches for the meaning intended in the document. Therefore, in the first case, the content analysis helps to get data, whereas in the second case, it is applied to interpret data which are gathered by interviews, observation, or group discussion.

Normatively oriented content analysis is a research technique for the objective, systematical, and quantitative description of the directly visible contents of interactions [May90b, Sil62]. ***Frequency analysis*** counts the relative portion of square centimetres in a newspaper dealing with the topic of research, for example. The ***valence*** and ***intensity techniques*** extend analysis by measuring the point of view of the communication act, e.g. whether it is pro or contra death penalty. ***Contingency analysis*** measures correlations of different communication acts, e.g. the attitude towards abortion and foreigners.

The ***"Qualitative Inhaltsanalyse" of Mayring*** is a systematical analysis of past communication acts based on theory and rules [May89]. It is used for interpreting results of open interviews. The purpose is to recognise the intentions of the interviewee, that is the meanings which he or she is conscious of. Figure 5 shows the life cycle of a complete interpretation. The definition of the analysis units - phase 7 - determines the ***code unit***, the ***context unit*** and the ***interpretation unit.*** The code unit defines the minimal size of the parts of a document which can be a member of a category, e.g. sentences, paragraphes, etc. The context unit determines the maximal size of the document´s parts which can be member of the category, and the interpretation unit determines the temporal succession for interpreting defined parts of the documents [May90b]. Step eight is the vital phase of text analysis. [May90b] proposes four techniques for its execution.

- ***Summary:*** The goal is to reduce the material while exposing its main structure and topics. Mayring discusses the macro operators ***generalization***, ***construction***, ***integration***, and ***selection***.
- ***Explication:*** The goal is to extend the given parts of the documents by related parts which are necessary to understand the intention.
- ***Structuring:*** The ***formally oriented structuring*** classifies the material formally, e.g. it defines the introduction of the story, the different chapters, and the conclusion. The ***content-oriented structuring*** classifies the material into different topics or aspects which are discussed in the different parts. The ***type-oriented structuring*** tries to organize the material by vital parts of it. It determines extreme, freuquent or theoretically interesting parts of it. The ***scale-oriented structuring*** maps parts of the documents as to values of variables like in valence or intensity techniques. All these four techniques need a definition of the category used for the structuring and an operationalisation by rules and examples.

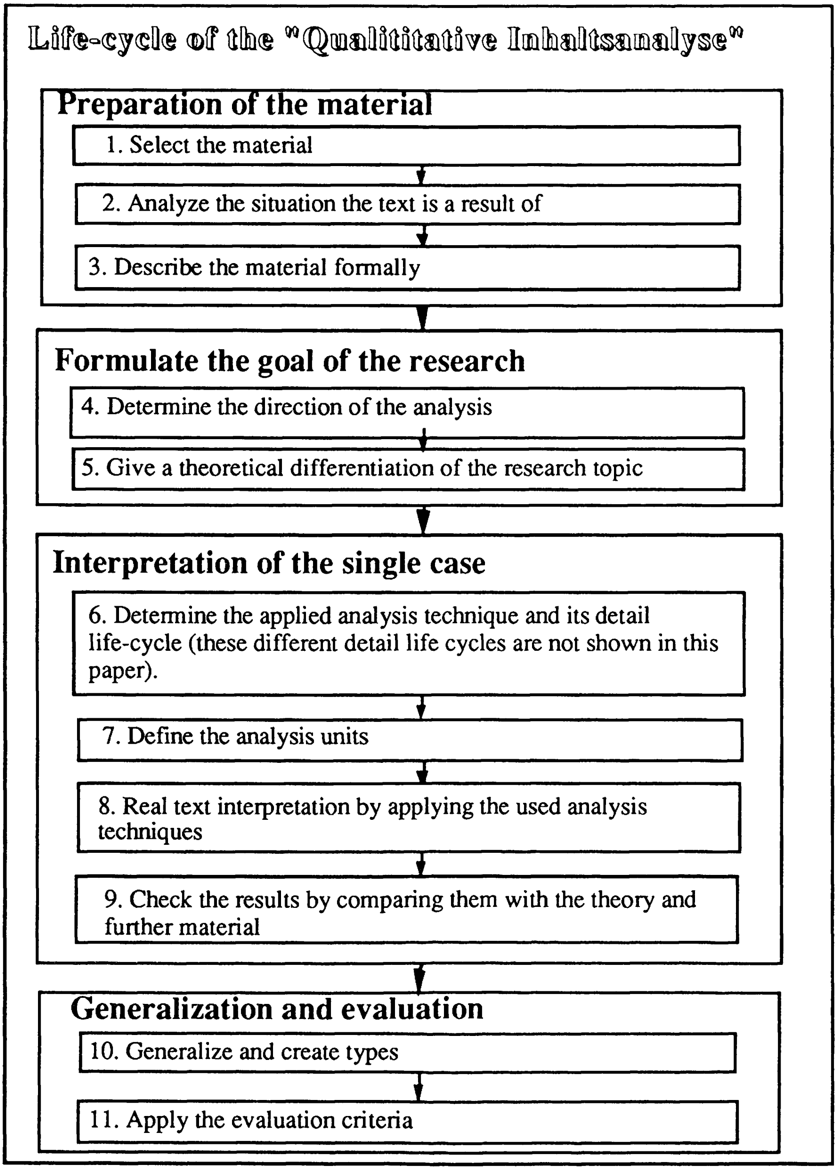

Life-cycle of the "Qualititative Inhaltsanalyse"
Preparation of the material
1. Select the material
2. Analyze the situation the text is a result of
3. Describe the material formally
Formulate the goal of the research
4. Determine the direction of the analysis
5. Give a theoretical differentiation of the research topic
Interpretation of the single case
6. Determine the applied analysis technique and its detail life-cycle (these different detail life cycles are not shown in this paper).
7. Define the analysis units
8. Real text interpretation by applying the used analysis techniques
9. Check the results by comparing them with the theory and further material
Generalization and evaluation
10. Generalize and create types
11. Apply the evaluation criteria

Figure 5. A life-cycle of the "Qualitative Inhaltsanalyse" of Mayring [May90b, p. 50].

The goal of the ***"Objektive Hermeneutik" by Oevermann*** is to discuss the implicit meanings of a communication act which are called objective meanings [Lam88, Lam89, OAK83, Sch89]. Understanding implicit meanings is a common sense-based ability, the "Objektive Hermeneutik" tries to systematize this ability. It proposes a life-cycle and rules for such an approach. It tries to carry out content analyses in a way like an engineering discipline. A second difference to common sense understanding is the intensity of the approach. In everyday life, some economic techniques are used which allow a rapid understanding to enable quick reactions. These techniques are given by socialization. The "Objektive Hermeneutik" does not use such economic techniques. It consists of four phases. Its third phase, the ***sequential detail analysis*** , is presented in figure 6 in detail.

Sequential detail analysis of the "Objektive Hermeneutik"

0. Explicate the context of the interaction: Show the alternatives which the person thinks he has.
1. Paraphrase the impartial content based on "common sense".
2. Explicate the intention of the person. Look only for those meanings the person would agree with you as being relevant. It is assumed that the person is not lying.
3. Explicate the impartial meaning and its impartial consequences. Use the context of the text, the context of the subject, and theoretical knowledge.
4. Explicate the function of the interaction by considering the different interaction roles of its members.
5. Characterize the linguistic and grammatical features of the interaction.
6. Compare the interpretation with the interpretation of other text parts. Look for general patterns.
7. Explicate general relations, create types and patterns.
8. Compare the results with the theory and the results of other parts.

[Lam90, AOK83]

Figure 6. The sequential detail analysis of the "Objektive Hermeneutik"

Further approaches: The ***"Hermeneutik"***, the science of understanding and interpreting texts has a long tradition in arts and in law (cf. [Lam88, Lam89, May90b]). The approach of ***Danner*** tries to differentiate several steps involved in such an analysis (cf. [May90b]). The ***"Strukturelle Beschreibung"*** by Hermanns is a technique for interpreting the results of narrative interviews [Lam89]. See [Lam88] for the procedure of ***Barton und Larsfeld***, [Lam88, May90a] for the procedure of building the so-called ***grounded formal theories*** of Glaser and Strauss, and [Hei87] for the ***"Sozialwissenschaftlich-hermeneutische Paraphrase***" by Heinze und Klusemann. For more procedures also see in [Wit82].

Application of the different interpretation types

The "Qualitative Inhaltsanalyse" of Mayring can be used to look for meanings. The "Objektive Hermeneutik" can be used to get implicit meanings. Because this is a very time-consuming procedure it can be used only for some very important sections. The "Hermeneutik" by Danner and the "structural description" by Hermanns can be used as a general framework for the interpretation of texts.

4.3. Tools for knowledge elicitation - Is it possible to shift knowledge elicitation to a theory-guided procedure?

Experience in software engineering shows that there is a need for combining bottom-up and top-down procedures. This article also suggests this for the knowledge acquisition process. It offers a mainly bottom-up approach for knowledge elicitation which has to transform the ***narrative gestalt oriented thinking*** [Kor87] of the expert into a sound formal model and a mainly top-down approach for the collection of domain knowledge, see also [MRD91]. The main facility for bottom-up modeling is ***hypertext*** (cf. [MRE90] or ACQUIST [MRD91]). Especially the context connections done by the text analysis require a non-linear organization of the texts. Theory-guided tools for knowledge collection are e.g. MOLE, MORE, SALT [Mar88], and PROTEGE [Mus89a, Mus89b].

As an unguided bottom-up development is a very time-consuming process it is necessary to think about integrating top-down elements not only into the knowledge collection but also into the knowledge elicitation process. The KADS group proposes a ***library of so-called interpretation models (i-models)*** which are generic problem-solving methods (cf. [BWS87, HJK89]). This library can contribute to elicitating the expert`s process knowledge. But until now there have been some problems with this approach. First, there is no clear list of criteria for selecting the optimal i-model. Second, the i-models are only vaguely described and still lack a clear semantics. Third, there are no good i-models for synthetical problem solving. Fourth, real-life applications often require combining different i-models. Finally, if the knowledge engineer wants to select a well suited i-model which can be used for building a conceptual model, he must be familiar with the problem. Therefore he needs some open techniques, even if he can reuse an existing i-model or a shell with a fixed problem-solving method. These open techniques, especially the narrative interview or the "Objektive Hermeneutik", are very time-consuming procedures. But the problem of knowledge elicitation is also a very ill-structured, and therefore complex, problem.

5. Conclusion

Knowledge elicitation, which is one step in a model-based and incremental knowledge acquisition, and qualitatively oriented social science show great similarities. Therefore principles, methods, and techniques of qualitatively oriented social science can be applied to knowledge elicitation. These similar features are created by the required openness in eliciting knowledge which is involved in the problem-solving ability of humans. As shown, this can be used for formulating principles for knowledge elicitation and obtaining categories to construct a framework which helps to improve the application of the various elicitation and interpretation techniques. The literature about knowledge acquisition offers a variety of techniques, but the question of when and for what purpose a technique should be chosen still remains to be dealt with in a more exhaustive manner. In addition, the article introduces interpretation techniques from the field of qualitatively oriented social science which can also be used for knowledge elicitation. The purpose is to contribute to shifting knowledge elicitation from an art to an engineering discipline.

Acknowledgement

I thank Jürgen Angele, Dieter Landes, Andrea Schneider, Rudi Studer, and especially Angi Voß for helpful comments, Christiane Rest and Gabi Rudnich for their support in correcting my manuscript.

References

[Add87] Addis, T.R.: A framework for knowledge acquisition. In *Proceedings of the first European Workshop on Knowledge Acquisition for Knowledge-based Systems (EKAW'87)*, Reading University, September 2-3, 1987.

[AFL90] Angele, J.; Fensel, D.; Landes, D.; Neubert, S.; and Studer, R: Knowledge Engineering in the Context of Related Fields of Research. In O. Herzog et.al. (eds.), *Text Understanding in LILOG*, Springer-Verlag, Lecture-Notes in Artificial Intelligence no. 546, Berlin, 1991, pp. 490-500.

[AFL91] Angele, J.; Fensel, D.; Landes, D.; and Studer, R: KARL: An executable language for the conceptual model. In *Proceedings of the 6th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, vol. I*, Banff, October 6-11, 1991.

[AFS90] Angele, J.; Fensel, D.; and Studer, R.: Applying software engineering methods and techniques to knowledge engineering. In D. Ehrenberg, D. et.al. (eds.), *Wissensbasierte Systeme in der Betriebswirtschaft, Reihe betriebliche Informations- und Kommunikationssysteme, no. 15*, Erich Schmidt Verlag, Berlin, 1990.

[AHS90] Ackermann, H.; van Harmelen, F.; Schreiber, G.; and Wielinga, B.: A formalisation of knowledge-level models for knowledge acquisition. In *International Journal of Intelligent Systems*, 1990, Forthcoming.

[Att74] Attewell, P.: Ethnomethodology since Garfinkel. In *Theory and Society*, 1, 1974, p. 179-210.

[BeH89] Bell, J.; and Hardiman, R.J.: The third role - the naturalistic knowledge engineer. In D. Diapper (ed.), *Knowledge Elicitation - principles, techniques and applications*, Ellis Horwood Series in Expert Systems, Chichester, 1989.

[Ber87] Berry, D.C.: The problem of implicit knowledge. In *Expert Systems*, vol. 4, no. 3, August 1987.

[Blu73] Blumer, H.: Der methodologische Standpunkt des symbolischen Interaktionismus. In Arbeitsgruppe Bielefelder Soziologen (eds.), *Alltagswissen, Interaktion und gesellschaftliche Wirklichkeit, Bd. 1: Symbolischer Interaktionismus und Ethnomethodologie*, Rowohlt, Reinbeck bei Hamburg, 1973, pp. 80-188.

[BoT75] Bogdan, R.; and Taylor, S. J.: *Introduction to qualitative research methods*, John Wiley & Sons, New York, 1975.

[Bro89] Bromme, R.: Aufgaben- und Problemanalyse bei der Untersuchung problemlösenden Denkens. In G. Juttemann (ed.), *Qualitative Forschung in der Psychologie*, Roland Asanger Verlag, Heidelberg, 1989.

[Bru66] Bruyn, S. T.: *The Human Perspective in Sociology*, Prentice-Hall, New Jersey, 1966.

[BrW84] Breuker, J.A.; and Wielanga, B.J.: Techniques for Knowledge Elicitation and Analysis. Report 1.5 Esprit Project 12, University of Amsterdam, Department of Social Science Informatics) and Laboratory for Experimental Psychology, July 1984.

[BrW89] Breuker, J.; and Wielinga, B.: Models of Expertise in Knowledge Acquisition. In G. Guida et.al. (eds.), *Topics in Expert Systems Design*, Elsevier Science Publisher B. V., North-Holland, 1989.

[BWS87] Breuker, J.; Wielinga, B.; Someren, M.v.; de Hoog, R.; Schreiber, G.; de Greef, P.; Bredeweg, B.; Wielemaker, J.; and Billault, J.-P.: Model-Driven Knowledge Acquisition: Interpretation Models. Esprit Project P1098, University of Amsterdam, 1987.

[COR89] Knowledge elicitation techniques for knowledge-based systems. In D. Diapper (ed.), *Knowledge Elicitation - principles, techniques and applications*, Ellis Horwood Series in Expert Systems, Chichester, 1989.

[DaH88] Davis, M.; and Hakiel, S.: Knowledge harvesting: A practical guide to interviewing. In *Expert Systems*, vo. 5, no. 1, February 1988.

[Den70] Denzig, N. K.: *The Research Act,* Aldine Publishing Company, Chicago, 1970.

[Den89] Denzin, N. K.: *Interpretative Interactionism*, Sage Publications, Newbury Park, CA, 1989.

[Dou73] Douglas, J.D. (ed.): *Introduction to sociology*, The Free Press, New York, Collier Macmillan Publisher, London, 1973.

[FAL91] Fensel, D.; Angele, J.; and Landes, D.: KARL: A Knowledge Acquisition and Representation Language. In *Proceedings of expert systems and their applications, 11th International Workshop, Conference "Tools, Techniques & Methods"*, 27-31 Mai, Avignon, 1991.

[Dou85] Douglas, J.D.: *Creative Interviewing*, Sage Publications, Beverly Hills, CA, 1985

[Flo84] Floyd, C.: A systematic look at prototyping. In R. Budee et. al. (eds.), *Approaches to Prototyping*, Springer-Verlag, Berlin, 1984.

[Gir84] Girtler, R.: *Methoden der qualitativen Sozialforschung*, Hermann Bohlaus Nachf. Gesellschaft mbH, Graz, 1984.

[Hei87] Heinze, T.: *Qualitative Sozialforschung*, Westdeutscher Verlag, Opladen, 1987.

[HJK89] Hickman, F.R.; Killin, J.L.; Land, L.; Mulhall, T.; Porter, D.; and Taylor, R.M., *Analysis for Knowledge-Based Systems: a practical guide to the KADS methodology*, Ellis Horwood Limited, Chichester, GB, 1989.

[Hof89] Hoff, E.-H.: Datenerhebung als Kommunikation: Intensivbefragung mit zwei Interviewern. In G. Juttemann (ed.), *Qualitative Forschung in der Psychologie*, Roland Asanger Verlag, Heidelberg, 1989.

[KaL90] Karbach, W.; and Linster, M.: *Wissensakquisition für Expertensysteme. Techniken, Modelle und Softwarewerkzeuge*, Carl Hanser Verlag, München, 1990.

[Kid87] Kidd, A.L. (ed.): *Knowledge Acquisition for Expert Systems. A Pratical Handbook*, Plenum Press, New York, 1987.

[Kön73] König, R.: Die Beobachtung. In R. König (ed.), *Handbuch der empirischen Sozialforschung, vol. 2, Grundlegende Methoden und Techniken, part 1,* Deutscher Taschenbuchverlag, 3. Aufl., Stuttgart, 1973.

[Kor87] Kornell, J.: Formal thought and narrative thought in knowledge acquisition. In *Int. J. Man-Machine Studies*, vol. 26, no. 2, 1987, pp. 203-212.

[KVS91] Karbach, W.; Voß, A.; Schuckey, R.; and Drouven, U.: MODEL-K: prototyping at the knowledge level. In *Proceedings of expert systems and their applications, 11th International Workshop, Conference "Tools, Techniques & Methods"*, 27-31 Mai, Avignon, 1991.

[Lam88] Lamnek, S.: *Qualitative Sozialforschung, Band 1, Methodologie*, Psychologie Verlags Union, München, 1988.

[Lam89] Lamnek, S.: *Qualitative Sozialforschung, Band 2, Methoden und Techniken*, Psychologie Verlags Union, München, 1989.

[Lin90] Linster, M.: Declarative problem-solving procedures as a basis for knowledge-acquisition: a first proposal. In Arbeitspapiere der Gesellschaft für Mathematik und Datenverarbeitung (GMD), no. 448, June 1990.

[Mar88] Marcus, S (ed.): *Automating Knowledge Acquisition for Experts Systems*, Kluwer Academic Publisher, Boston, 1988.

[Man73] Mangold, W.: Gruppendiskussion. In R. König, *Handbuch der empirischen Sozialforschung, vol. 2, Grundlegende Methoden und Techniken, part 1*, Deutscher Taschenbuch Verlag, 3. Aufl., Stuttgart, 1973.

[May89] Mayring, P.: Qualitative Inhaltsanalyse. In *Qualitative Forschung in der Psychologie*, G. Juttemann (ed.), Roland Asanger Verlag, Heidelberg, 1989.

[May90a] Mayring, P.: *Einführung in die qualitative Sozialforschung*, Psychologie Verlags Union, München, 1990.

[May90b] Mayring, P.: *Qualitative Inhaltsanalyse*, Deutscher Studien Verlag, Weinheim, 2. Aufl., 1990.

[McH89] McGraw, K.L; and Harbison-Briggs, K.: *Knowledge Acquisition. Principles and Guidelines*, Prentice-Hall International, Inc., Englewood Cliffs, NJ, 1989.

[MiS84] Michalski, R.S.; and Stepp, R.E.: Learning from Observation: Conceptual Clustering. In R.S. Michalski et.al. (eds.), *Machine Learning. An Artificial Intelligence Approach*, Springer Verlag, Berlin, 1984, pp. 331-364.

[Mor87] Morik, K.: Sloppy modeling. In K. Morik (ed.), *Knowledge Representation and Organisation in Machine Learning*, Springer-Verlag, Berlin, 1987.

[MRD91] Motta, E., Rajan, T., Domingue, J., and Eisenstadt, M: Methodological foundations of KEATS, the Knowledge Engineer's Assistant. In *Knowledge Acquisition*, vol. 3, no. 1, March 1991, pp. 21-47.

[MRE90] Motta, E.; Rajan, T.; and Eisenstadt, M.: Knowledge acquisition as a process of model refinement. In *Knowledge Acquisition*, vol. 2, no. 1, March 1990, pp. 21-49.

[Mus89a] Musen, M.: Conceptual models of interactive knowledge acquisition tools. In *Knowledge Acquisition*, vol. 1, nr. 1, March 1989, pp. 73-98.

[Mus89b] Musen, M.: *Automated Generation of Model-Based Knowledge-Acquisition Tools*, Morgan Kaufmann Publisher, San Mateo, CA, 1989.

[NPB91] Nwana, H. S.; Paton, R. C.; Bench-Capon, T. J. M.; and Shave, J. R.: Facilitating the Development of Knowledge Based Systems. In *AI Communications*, vol. 4, no. 2/3, June/Sept. 1991, pp. 60-73.

[OAK83] Oevermann, U.; Allert, T.; Knau, E.; and Krambeck, J.: Die Methode einer "objektiven Hermeneutik". In P. Zedler et.al. (eds.), *Aspekte qualitativer Sozialforschung*, Leske Verlag + Budrich GmbH, Opladen, 1983.

[OlR87] Olson, J.R.; and Rueter, H.H.: Extracting expertise form experts: methods for knowledge acquisition. In *Expert Systems*, vo. 4, no. 3, 1987.

[Par86] Partridge, D.: *Aritificial Intelligence. Application In The Future Of Software Engineering*, Ellis Horwood Limited, Great Brtain, 1986.

[RaS79] Rabinow, P.; and Sulliva, W.M. (eds.): *Interpretative Social Science. A Reader*, University of California Press, Berkeley, 1979.

[Sch62] Scheuch, E., K.: Das Interview in der Sozialforschung. In R. König (ed.), *Handbuch der empirischen Sozialforschung, I. Band,* Ferdinand Enke Verlag , Stuttgart, 1962.

[Sch76] Schutze, F.: Zur Hervorlockung und Analyse von Erzählungen thematisch relevanter Geschichten im Rahmen soziologischer Feldforschung - dargestellt an einem Projekt zur Erforschung von kommunalen Machtstrukturen. In Arbeitsgruppe Bielefelder Soziologen (eds.), *Alltagswissen, Interaktion und gesellschaftliche Wirklichkeit, Bd. 2: Ethnotheorie und Ethnographie des Sprechens*, Rowohlt, Reinbeck bei Hamburg, 1976.

[Sch89] Schneider, G.: Strukturkonzept und Interpretationspraxis der objektiven Hermeneutik. In G. Juttemann (ed.), *Qualitative Forschung in der Psychologie*, Roland Asanger Verlag, Heidelberg, 2. Aufl., 1989.

[Sil62] Silbermann, A.: Systematische Inhaltsanalyse. In R. Konig (ed.), *Handbuch der empirischen Sozialforschung, Bd. 4: Komplexe Forschungsansatze,* Deutscher Taschenbuch Verlag, Stuttgart, 1962.

[Spö89] Spöring, W: *Qualitative Sozialforschung,* B.G. Teubner-Studienskripten, Stuttgart, 1989.

[Wet90] Wetter, T.: First Order Logic foundation of the KADS Conceptual Model. In *Current Trends in Knowledge Acquisition*, B. Wielinga et. al. (eds.), IOS Press, Amsterdam, 1990.

[Wil71] Wilson, T.P.: Normative and interpretative paradigms in sociology. In J.D. Douglas (ed.), *Understanding everyday life,* Routledge & Kegan Paul, London, 1971.

[Wit82] Witzel, A.: *Verfahren der qualitativen Sozialforschung*, Campus Verlag, Frankfurt a. M., 1982.

[Wit89] Witzel, A.: Das problemzentrierte Interview. In G. Juttemann (ed.), *Qualitative Forschung in der Psychologie*, Roland Asanger Verlag, Heidelberg, 1989.

Part 2:

Case-based Approaches to the Development of Expert Systems

Case-based Reasoning and Model-based Knowledge-Acquisition

Dietmar Janetzko & Gerhard Strube
University of Freiburg
Dept. of Cognitive Science

This chapter outlines two different yet complementary approaches to enhance cognitive adequacy in the process of knowledge engineering: model-based knowledge acquisition and case-based reasoning. Although both differ with respect to methods, goals and scientific background, arguments are advanced that a linkage of both approaches will result in a significant contribution to the methodology of knowledge acquisition for expert systems. To combine case-based reasoning techniques with conventional rule-based approaches poses the problem of when to use which technique. A conceptual framework for turn taking in problem solving is outlined that involves both heuristics of turn taking and architectural options for knowledge-based systems that impose constraints on turn taking.

Introduction

Concerning knowledge-acquisition the development of the first artificial intelligence (AI) systems started with a misunderstanding. Neither has *elictation* of knowledge taken place in early examples of expert systems, as the domain expert and the system programmer usually were the same person, nor has there been an *encoding* of knowledge, as the knowledge was just programmed in (Riesbeck, 1988). Thus, the problems linked to the crucial steps of what has later been coined knowledge acquisition have not been realized in their full extent.

When AI systems left the laboratories and set out to conquer real world domains the difficulties of knowledge acquisition became obvious. As a first reaction, primary attention has been given to increase the efficiency of knowledge elicitation. Methods to speed up knowledge elicitation (Hoffman, 1987), or tools that enable the domain expert to state his knowledge without help from a knowledge engineer (e.g. Boose & Bradshaw, 1987) are among the most prominent research objectives that are undertaken in that spirit. The attention given to knowledge elicitation contrasts sharply with the fact that the model of problem solving underlying most expert systems remained untouched. This line of research has been referred to as the *mining view* of knowledge engineering (Breuker & Wielinga, 1989). Breuker & Wielinga advocate an alternative approach, which they call the *modeling view*. Building a knowledge-based system is no longer understood as filling a shell with knowledge. According to the modeling view, a specification of expertise is built that defines a mapping between the real world of expertise and the artificial world of computer systems.

There is general agreement that the modeling view is currently one of the most promising approaches to knowledge engineering (Schmalhofer & Bergmann, 1990; Ueberreiter & Voß, 1991). Nevertheless, there is a substantial body of work directed at various aspects of knowledge acquisition not yet covered by studies inspired by the modeling view. Some of these take up the controversial issues whether techniques for knowledge acquisition might eliminate the knowledge engineer. There is a debate on whether knowledge engineering is just a variety of software engineering, or an approach in its own right based on its own methodology and equipped with methods that give it a clear-cut profile (Becker, 1991). Finally, the question how cognitive science can contribute to the development of theories and tools of knowledge engineering has been discussed at length.

This article attempts to outline relationships between cognitive science research and approaches to knowledge engineering based on the modeling view. Our focus lies in the area of case-based reasoning, which is rooted both in cognitive science and computer science. Special attention is given to turn taking, i.e. change of the mode of problem solving (case-based, rule-based etc.). The paper is organized as follows: The first part gives a brief account of case-based reasoning and sketches basic notions of KADS, the most advanced example of the modeling view. Then, case-based reasoning is specified anew in terms of the modeling view. Since turn taking is of special importance for linking case-based reasoning with model-based approaches to knowledge engineering, we review results of cognitive science that refer to this issue, and discuss the conditions (heuristics and architectural options) under which case-based reasoning or rule-based reasoning is (or should be) used to solve a given problem.

The Modeling View of Knowledge Engineering

The acquisition of knowledge and the maintenance of knowledge-based systems are the two basic tasks of knowledge engineering. Knowledge acquisition, in turn, is usually subdivided in knowledge elicitation and knowledge encoding (*cf.* Christaller, Güsgen, Hertzberg, Linster, Voß & Voß, 1988). The elicitation and encoding of knowledge results in a knowledge-base which is at the heart of knowledge-based systems. The two-phase model of knowledge acquisition gives the impression of being straightforward and simple. However, there is a number of difficulties related to this approach that have instigated research efforts directed at a methodology for knowledge acquisition. Among the problems that necessitate further research is the *task-model-mismatch* (Kurbel, 1989), which occurs when a knowledge-based system cannot solve the problems it is intended to solve because the model of expertise realized in the knowledge-base of the system is fundamentally inadequate. Often, this happens if elicitation and encoding of knowledge has been fitted to some particular expert-system shell. In this case, even slight changes of the knowledge base or the inference rules of the system can only be carried out at the price of time-consuming efforts. Partly in response to this problem, Breuker & Wielinga (1987) proposed a distinction between an analysis and design phase. The analysis phase leads to a description of expertise at the 'knowledge level' (Newell, 1980), called the conceptual model. The conceptual model is a functional specification of all kinds of knowledge (concepts, inferences, tasks, goals) and their relations being used for building a model of expertise. The conceptual model is a means to assess the completeness and consistency of the model of expertise that is realized in the knowledge-based system. Only after a conceptual model, at least a preliminary one, has been specified, work may advance into the design phase. Here, the conceptual model is translated into a design model, which provides the base for implementation.

To design a detailed specification first, is in accord with the conventions of software engineering (Pressman, 1987). At the same time, the specification of a conceptual model means to step away from common knowledge-engineering approaches like rapid prototyping that usually lack a conceptual level (Christaller et al., 1988, Kurbel, 1989). According to Karbach, Linster & Voss (1989), the use of a conceptual level entails a number of advantages with regard to knowledge acquisition and the maintenance of knowledge-based systems:

- Precise specification of the type of problem-solving
- Support of systematic elicitation of the required knowledge
- Opportunity of specification-guided implementation
- Better documentation of the implemented system

The Four-Layer Model of Expertise

Among the first to advocate and develop a detailed methodology of a model-based approach to knowledge engineering were Joost Breuker and Bob Wielinga (e.g. Wielinga & Breuker, 1984). Their approach, called KADS *(Knowledge Acquisition and Documentation Structuring)*, is intended to provide epistemological primitives for the description of expertise at the knowledge level. Thus, KADS allows for the development of conceptual models of expertise independent of particular implementational constraints. Though partially inspired by cognitive science (e.g. Norman, 1983), KADS models do not strive for genuine cognitive modeling of expertise. The reconstructive description of expertise used in KADS comprises four layers of knowledge *(four layer model of expertise)*:

Domain Layer: The knowledge at the domain layer covers concepts, relations and structures. In a domain like toxicology, for example, alcohol, vomiting, and gastric lavage are concepts. There are various classes of relations (subsumption relations, causal relations, empirical associations), e.g., causal relations between toxins and symptoms. Concepts and relations can be represented in an inheritance hierarchy (Voss, Karbach, Drouven, Lorek, & Schukey, 1990). Relations build up structures, i.e., networks of relations. Domain layer knowledge is static and task-independent, because procedural knowledge about a particular task is not represented at the domain layer.

Inference Layer: At the inference layer, knowledge is classified according to its functions in the model of expertise. The two basic objects on the inference layer are meta-classes and knowledge sources. Meta-classes constrain the range of potential roles concepts of the domain layer can take during problem solving. In the domain of toxicology, for example, toxins, symptoms and therapies are meta-classes. It is possible to build a hierachy of meta-classes; concepts may belong to more than a single meta-class. Knowledge sources are functional descriptions of primitive inference steps. Each knowledge source refers to a relation defined at the domain layer. Meta-classes are used to provide a specification of the knowledge which is expected at the input and produced at the output of a knowledge source. Examples of knowledge source applications are the operations match, select, or classify.

Task Layer: At the task layer, knowledge encoded at the inference layer is coordinated in order to achieve a defined goal. The three basic objects on the task layer are goals, tasks and control elements. Goals are states the system strives to reach. The representation of goals takes the form of concepts with specific attributes. Procedures used to achieve a goal are called tasks. According to the systematic ordering and decomposition of goals and subgoals, which result in a goal-tree, tasks are classified by building task-structures. Control elements are data structures representing information that dynamically changes during the process of problem-solving.

Strategic Layer: The pursuit of goal achievement by carrying out, or rearranging a sequence of tasks is done at the strategic layer. Knowledge at the strategic layer is used to model features of expertise like early recognition of problems and dead ends for a particual strategy chosen, change of the strategy in problem solving, and other aspects that pertain to the control of problem solving. Goal-trees and task-structures are used to represent the strategies required to model these properties of expertise.

In KADS, the notion of a conceptual model refers to the task-oriented description of expertise at all four layers. KADS-models may be used in two directions: Following the *constructive direction*, a conceptual model is an intermediary between the elicitation and interpretation of knowledge in the analysis phase and the design model and implementation of the design phase. If the domain layer is separated from a conceptual model, the resulting knowledge structure is called interpretation model. Thus an interpretation model is a domain-independent abstraction of a domain-dependent conceptual model. Interpretation models are representations of generic types of problem solving and can be collected in libraries. Following the *selective direction*, the addition of concepts and relations from a new domain to some suitable interpretation model chosen from a library results in a new conceptual model. In a word, knowledge engineering according to the modeling view provides the feasibility of taking either a bottom-up or a top-down approach. Apart from coarse-grained methods (e.g., some expert-system shells), which proceed in a top-down fashion, knowledge engineering according to the mining view is mostly dedicated to bottom-up approaches. Methodologies and methods developed with the modeling view in mind are doubtlessly more sophisticated and less prone to problems. However, important features of human expertise, which give flexibility and efficiency to human problem solving, are currently out of the scope of research within the modeling view. In the next section, we shall portray some of these features within the framework of case-based reasoning.

Experience and Episodic Knowledge

The ability to accumulate experience available for subsequent use in problem solving is still one of the most striking differences of reasoning in man and machine. The endeavor to endow computers with that faculty is an ongoing research enterprise, with a growing influence on knowledge engineering. Case-based reasoning is a research paradigm that addresses the issue of experience and its impact on reasoning (Slade, 1991). In particular, case-based reasoning aims at pursuing two goals for research, the description and explanation of the development and use of experience in humans, and the design of computer programms that are equipped with the same talents.

Remindings of previous cases can be used to solve current problems (e.g., Kolodner, 1991). This is not surprising as we use this kind of reasoning every day. Recent years have witnessed a growing interest in this type of problem solving in cognitive science and artificial intelligence (Caplan & Schooler, 1990, Strube & Janetzko, 1990). The knowledge used in case-based reasoning can usually be traced back to specific episodes, i.e., to the temporal and local conditions under which the relevant facts occured (e.g., 'I came across a similar problem in Detroit last summer, when I was...') This kind of knowledge has been termed *episodic knowledge* (Strube, 1989). Alternatively, but with more emphasis placed on the use in problem solving, this knowledge is referred to as *special purpose knowledge* (Shavlik, DeJong, & Ross, 1987).

In contrast, the type of knowledge typically used in knowledge-based systems is or pretends to be universally valid (e.g., 'A robin is a bird'). Common ways to represent semantic knowledge are production rules, frames, semantic networks, and constraints. When used in problem solving the advantages and disadvantages of episodic and semantic knowledge are quite different. Semantic knowledge provides the advantage of universal application and the disadvantage of costly adjustments to solve a specific problem. That is, reasoning from first principles when coping with a simple or well-known problem may become an expensive expedition into the problem space. Episodic knowledge, on the other side, has the disadavantage of being useful only within a narrow range of problems and the advantage of shortcutting processes of reasoning. This relationship has been termed the *operationalitygenerality trade-off* (Shavlik, DeJong, & Ross, 1987). A lesson lerned from this comparison is that the use or non-use of either general or episodic knowledge is not in itself an advantage or disadvantage for the process of problem solving. High performance with regard to the speed, accuracy and reliability of results calls for the retrieval and use of the appropriate type of knowledge. In the next section, we will pursue the question in which way models of case-based reasoning make use of episodic knowledge. In subsequent sections we shall return to the issue of when to use episodic and semantic knowledge in order to combine the advantages of both.

The Basic Model of Case-Based Reasoning

There is general agreement among researchers in case-based reasoning that the feasibility of this kind of knowledge-based problem solving is tied to a number of computational steps, which outline the current topics of research in case-based reasoning:

- How are cases represented, i.e., how to find adequate representations of episodic knowledge, and interfaces with general knowledge?
- How are cases retrieved from memory (e.g., from a case library)?
- What measure of similarity is appropriate to select a suitable case in memory (= source case) given a current case (= target case)?
- How is (partial) carry-over of the solution of a source case done to the target case?
- How may (parts of) the solution of a source case have to be adapted to the constraints of the target case?
- How can general knowledge be automatically acquired from episodic knowledge?
- How should a case library be organized (and re-organized)?

Most of these issues have already become the subject of research in other areas of artificial intelligence and cognitive science. Therefore, case-based reasoning provides a framework of research to link those areas and their results. This is especially true for KADS, which places emphasis on different aspect of knowledge engineering in comparison to case-based reasoning (Bartsch-Spörl, 1991). KADS belongs essentially to the province of software engineering, but aims at a specification of a set of epistemological primitives in order to improve knowledge engineering. The long-term perspective of KADS is to become the accepted industrial standard for the development of knowledge-based systems. Rooted in cognitive science, case-based reasoning is a unifying approach of theories of knowledge representation, learning and problem solving. A possible way of uniting both approaches through re-specifying case-based reasoning in KADS is presented in Figure 1. This results in an inference structure of case-based reasoning, which integrates metaclasses (e.g., *problem statement*) and knowledge sources (e.g., *reorganize*). Strategies of turn taking are both failure-driven and achievement-driven. A similar inference structure for KADS has been suggested by Bartsch-Spörl (1991).

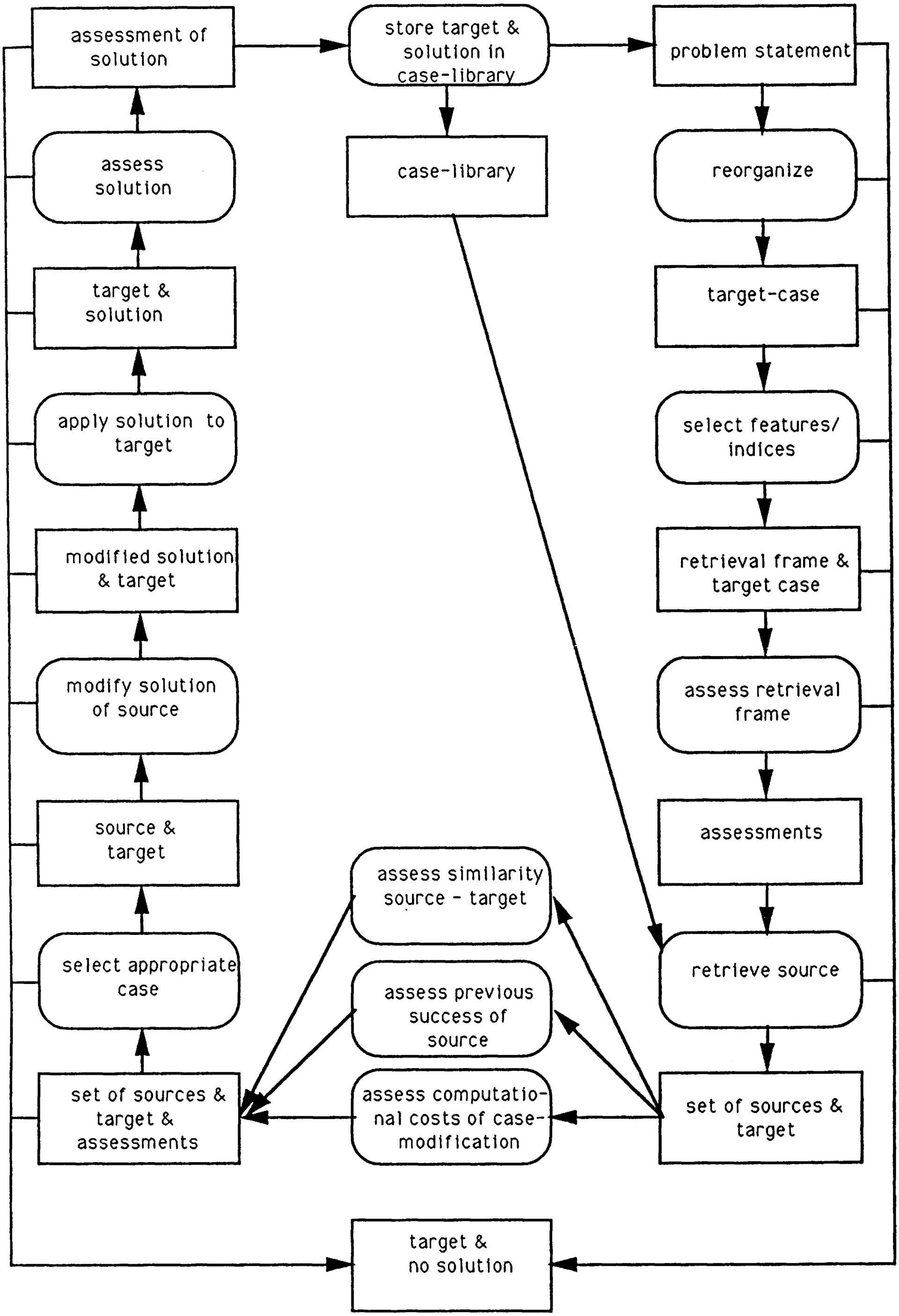


Figure 1: Case-Based Reasoning Inference Structure

From Case-Based Reasoning towards a Theory of Problem Solving

Our framework for case-based reasoning has the power to encompass a number of different models of tasks (retrieval, representation, similarity judgements, etc.) and types of problem solving (case-based, rule-based, constraint-based etc.). The combination of a case-based reasoner with a rule-based or from-scratch reasoner (Carbonell, 1983, Hinrichs, 1988, Koton, 1988, Rissland & Skalag, 1989) has drawn attention to the fact that high performance in problem solving depends on selecting the appropriate strategy of reasoning. Hence, the potential flexibility of hybrid expert systems is available only if they are equipped with facilities for demand-driven selection of the problem solving strategy. Which conditions and criteria make a particular one the strategy of choice depends on the theoretical perpective taken. We shall discuss two perspectives on turn taking, the cognitive scientist's perspective and the knowledge engineer's perspective. Both these perspectives yield answers that differ more in emphasis than in principle. In fact, results of cognitive science may serve to design a conceptual framework for turn taking, and the same conceptual framework may stimulate further empirical research in cognitive science. We return to that issue later.

Turn Taking in Problem Solving: The Cognitive Scientist's Perspective

From the perspective of the cognitive scientist, the question when to use a certain strategy of problem solving is answered by searching for conditions which systematically correspond with the preferred use of that strategy in natural problem solving. We now discuss some of the conditions that have been under study.

Conditions of Learning. In a series of experiments, Caplan & Schooler (1990) looked for the conditions that make a human problem solver use either episodic knowledge via episode-based processing, or semantic knowledge via rule-based processing. Their subjects first learned to use a microcomputer drawing package ('Fullpaint'), and then had to cope with a series of drawing tasks. Caplan & Schooler concentrated on the influence of different conditions of training schedules and complexity of training examples. Training instructions were provided in a random or an organized order and with or without an analogical model. The analogical model introduced the various functions of Fullpaint with the help of icons. Practise trials differed in visual and logical complexity both of which varied high and low. Performance was measured by using paper-and - pencil tests and recording actual use of Fullpaint. In paper-and-pencil and problem-solving tests theanalogical model condition yielded better performance than the no-model condition when practise trials were logically complex. When practise trials were logically simple the no-model condition yields better performances than the model condition both in paper-and-pencil- and problem-solving-tests. Caplan & Schooler dicussed their results in light of the *encoding specifity hypothesis* (Tulving & Thompson, 1973), which states that performance is best when the kind of processing used at encoding and retrieval are identical. If conditions of learning encouraged case-based reasoning, performance was best when problem solving in the tests was also of a case-base type (e.g., working with examples or cases). The reverse was true for rule-based learning.

Level of Expertise. Do experts prefer one way of problem-solving over another? Experimental findings indicate that the relationship between the level of expertise and the preferred mode of problem-solving depends on a number of different conditions. In fact, this relationship is by far more intricate than the question seems to presuppose. First of all, we may assume that experts have a great number of episodes or cases at their disposal. Additionally, experts have been shown to actually use episodic knowledge, i.e. cases, in problem solving (Voss, Greene, Post & Penner, 1983). But growing expertise is also accompanied by the acquisition of rules (Glaser, 1986). However, this is far from being necessary (Brehmer, 1980). Formation of rules becomes more likely if the expert has to give formal accounts of his activities regularly. Having acquired a body of rules, experts predominantly engage in rule-based processing, unless problems come up that are extraordinarily hard or exceptional. In contrast, several investigators have demonstrated that novices make extensive use of examples (e.g., Anderson, Farrell & Sauers, 1984, Reed, Dempster & Ettinger, 1985). Ross (1984) suggests that preference of case-based over rule-based reasoning results from the extent of similarity between the current and a previously solved problem.

Peculiarities of the domain. Rule-based problem solving has met with remarkable success in a number of domains, especially technical ones. Hence, rule-based reasoning is usually regarded as a universal way of problem solving, some domains like history are ill suited for this method. It is therefore not possible to grasp the essentials of each and every potential domain of application and transform it into a body of detailed and efficient rules. In some domains instable causal relationships, or a vast number of variables make rule-based simulations and predictions nearly impossible. Case-based reasoning is therefore of special importance for both problem solving and decision making in the domains of law, history, or politics (Neustadt & May, 1986). A recent example has been analyzed by Enzensberger (1991), who points out how Saddam Hussein was compared (via case-based reasoning) to Hitler in the Gulf War, and that the comparison suggests that measures that would have been the right answer to Hitler probably were the right reaction against Saddam. At the same time, domains like history and politics are rich in hypothetical reasoning. This is an important feature of human reasoning which only tentatively has been subject to modeling in knowledge-based systems (e.g. Rissland, 1986):

Previous Case:	"We know Israel's reaction in similar situations in the past."
Hypothetical Case:	"But what about Israel's reaction nowadays, if ..."

Summing up. Which kind of strategy is adopted for the solution of a given problem, depends not on a single factor, but upon a variety of influences. Obviously, expertise does not call for a particular problem solving strategy. Case-based reasoning, for instance, may become a last resort to the novice who has no rules available, or it may be the hallmark of expertise when tackling exceptional and highly complex problems. Experts and novices alike, however, need to have access to both case-based and rule-based knowledge, and their use of a strategy depends (at least partly) on the effort required to access that knowledge.

Another relevant factor in deciding which problem solving strategy to use is the prognostic evaluation of the strategies at hand. To our knowledge, no experimental evidence exists how human experts proceed, i.e., do they access both rules and cases, and evaluate both these approaches afterwards, or do they access only the knowledge that promises a solution readily? The majority of existing case-based reasoning systems attempt a two-step procedure. First, they select in a combined step of selection/evaluation a set of cases from a case-library that are similar to the problem at hand. In a second step, the cases in the selection set are evaluated with respect to their computational costs, etc. (Bareiss & King, 1989). Retrieval of information from long-term memory has been framed in a similar way (e.g., Raaijmakers & Shiffrin, 1981).

Turn Taking in Problem Solving: The Knowledge Engineer's Perspective

Decision making with respect to selection of a problem-solving strategy, i.e., turn taking, seems to be a function of the whole system of problem solving and not tied to a single facility or "turn-taking-decision-maker." The complexity of the issue calls for a careful treatment of results from cognitive science. Basically, we have two options: A *direct* way of using results from cognitive science is to look for results that can be translated in a straight-forward manner into *heuristics*. An *indirect* way of using results from cognitive science is to think about *architectural options* for knowledge-based systems that allow for flexible turn taking behavior as observed in humans. Thus, knowledge engineering might be inspired by cognitive science as far as heuristics and architectural options is concerned without necessarily striving for cognitive modeling of turn taking in natural problem solving.

Turn Taking in Problem Solving: Heuristics

Clear-cut criteria, or at least heuristics that motivate the preference of one type of problem-solver over an alternative one, are highly desirable for designers of knowledge-based systems. Turn-taking heuristics might be understood as conflict resolution strategies analogous to those used in rule-based systems (Jackson, 1986). Typically, conflict resolution strategies are used *within* the framework of a rule-based problem solver; their task is to decide which rule, i.e., step of reasoning, will be given priority when two or more rules could be applied. In a hybrid system, however, conflict resolution strategies are required as well *between* different problem solvers to organize turn taking, e.g., between case-based and rule-based reasoners. The following sample of heuristics should be judged with care, however, because only the interplay between the heuristics and the architecture of the system will suffice to motivate turn taking.

Specifity. If a case (source case) has initially been retrieved from the case library that is equivalent to the problem (target case) to a large extent, the likelihood for case-based reasoning rises. Weighting of features may be introduced to account for the fact that both corresponding and non-corresponding features of source and target case differ in their importance for problem solving and in the computational costs spent when adapting the source case to the target case.

Efficiency. The solutions of the cases that have been retrieved must be evaluated. If the source cases have not met with success, case-based reasoning should only be used if the prospect of using rule-based reasoning is even worse. Of course, measures of efficiency may be extended to aspects like processing time, neglected conditions, financial consequences, etc.

Domain. If the target case belongs to a domain or subdomain where case-based reasoning has formerly been applied successfully the probability to use case-based reasoning rises.

Turn Taking in Problem Solving: Architectural Options

Conflict resolution strategies *between* problem solvers vary according to a number of aspects related to the architecture of a system that integrates several problem solvers.

- Turn taking may be failure-driven or achievement-driven.

- Strategies of conflict resolution may be realized in a hierarchical architecture with a higher, or meta-problem solver excercising control over the other, basic ones (case-based, rule-based, etc.). Alternatively, conflict resultion strategies may be realized in a heterarchical architecture, where the different problem solvers settling turn taking among themselves. This way of controlling the behavior of the system is akin to Selfridge's (1959) pandemo-

  model (albeit not to its implementation in those days), or to Minsky's (1985) society of mind.

- Either division of labor is possible among the various problem solvers, or each problem solver works on each step of a problem on its own.

- We need a set of defined points of turn taking. These are points in the flow of control of problem solving where the result of a test is used to decide whether or not the mode of problem solving has to be changed. Points of turn taking have to fulfil three requirements:

  - They have to be distributed across the whole process of problem solving.

  - It is not useful to install too great a number of points of turn taking. This would result in high computational costs, similar to someone who solves a problem and spends a lot of time and energy thinking about possible alternatives to solve the problem all the while.

  - To keep computational costs low, the test used by points of turn taking should be a natural part of the process of problem solving (e.g. computation of the degree of specifity, similarity, estimated distance to the solution, etc.)

control scheme could possibly be implemented, where all the basic problem solvers take turns regularly, and the meta-problem solver pops up and decides upon turn taking among them only when too much computing resources have been used.

A combined scheme seems also possible for failure-driven and achievement-driven control of turn taking. In fact, only an intelligent combination of both options allows for a sufficiently flexible behavior. A purely failure-driven approach is doomed to fail, because each basic problem solver may face problems and - consequently - no problem solver would attempt to solve the problem. This might be called *giving up too early*. Strict failure-driven turn taking will rule out every problem solver, unless it takes into consideration both the extent of the problem and the expected achievements of each basic problem solver. Likewise, a purely achievement-driven approach will not succeed, if a problem solver runs into lengthy computations and does not take into account to change the mode of problem solving. This might be labeled *giving up too late*.

Cognitive Science and Knowledge Engineering

We hope to have shown how methods and results from cognitive science may be applied to the field of knowledge engineering. We have done so, of course, in a selective way. Cognitive science has produced a multitude of papers that might be applied to knowledge engineering for case-based expert systems. Issues of assessing similarity between cases have been treated by Tversky (1977), or Osherson (1987). Modification of cases might profit from research on analogical mapping (Falkenhainer, Forbus & Gentner, 1990; Holyoak & Thagard, 1989). Turning around, we recognize that research in cognitive science is instigated in many ways by 'applied' problems, e.g., in knowledge engineering. Transfer in both directions is the essence of the interplay between applied and basic research.

One of the peculiarities of knowledge engineering as an applied science is that it draws on a variety of basic fields of research in addition to cognitive science, notably computer science/AI and mathematical logic. From the perspective of knowledge engineering, therefore, any offerings from cognitive science have to stand their ground against the alternatives offered by those other basic disciplines. Cognitive science is pressed by that to develop its own profile of research. Although similar in many ways to AI because of the programming techniques used in cognitive modeling, cognitive science is still distinct from AI because it focuses on modeling human information processing and experimental methods for testing its models, thus becoming a natural science rather than a branch of engineering.

Bibliography

Anderson, J. R., Farrell, R., & Sauers, R. (1984). Learning to program LISP. *Cognitive Science*, 8, 87-129.

Bareiss, R., & King, J. A. (1989). Similarity assessment in case-based reasoning. In DARPA (Ed.), *Proceedings of a Workshop on Case-Based Reasoning* (pp. 67-71). Holiday Inn, Pensacola Beach, Florida, May 31 - June 2, 1989. San Mateo, CA: Morgan Kaufmann.

Bartsch-Spörl, B. (1991). KADS for (all) Cases. In B. Ueberreiter & A. Voß (Eds.), *Materialien KADS Benutzer Treffen*. München: Siemens.

Becker, B. (1991). Gibt es eine Methodologie der KI? *Vortrag gehalten auf der 10. Frühjahrsschule Künstiche Intelligenz. 2.3-10.3.1991. Günne.*

Boose, J. & Bradshaw, J. (1987). Expertise transfer and complex problems: using AQUINAS as a knowledge acquisition workbench for knowledge-based systems. *International Journal of Man-Machine Studies*, 26, 3-28.

Brehmer, B. (1980). In one word: Not from experience. *Acta Psychologica*, 45, 223-241.

Breuker, J., & Wielinga, B. (1987). Use of models in the interpretation of verbal data. In A. L. Kidd (Ed.), *Knowledge acquisitionfor Expert Systems: A practical handbook* (pp. 17-44). Pitman: London.

Breuker & Wielinga (1989). Models of expertise in knowledge acquisition. In G. Guida, & C. Tasso (Eds), *Topics in expert system design* (pp. 265-295). Amsterdam: North-Holland.

Breuker, J., Wielinga, B., van Someren, M., de Hoog, R., Schreiber, G., de Greef, P., Bredeweg, B., Wielemaker, J. Billaut, J.-P., Davoodi, M, & Hayward, S. (1987). *Model-driven knowledge acquisition: Interpretation models*. Deliverable task AI, Esprit Project 1098, Memo 87, VF Project Knowledge Acquisition in Formal Domains.

Breuker, J., & Wielinga, B. (1989). Models of expertise in knowledge acquisition. In G. Guida, & C. Tasso (Eds.), *Topics in expert system desing: Methodologies and Tools*. Amsterdam: North-Holland.

Brooks, L. R. (1987). Decentralized control of categorization: The role of prior episodes. In U. Neisser (Ed.), *Concepts and conceptual development* (pp. 141-174). New York: Cambridge University Press.

Caplan, L. J., & Schooler, C. (1990). Problem solving by reference to rules or previous episodes: The effects of organized training, analogical models, and subsequent complexity of experience. *Memory & Cognition*, 18, 215-227.

Carbonell, J. (1983). Learning by analogy: Formulating and generalizing plans from past experience. In R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (pp. 137-161). Palo Alto: Tioga Press.

Ceci, S. J., & Liker, J. K. . (1986). A day at the races: A study of IQ, expertise, and cognitive complexity. *Journal of Experimental Psychology*, 115, 255-266.

Christaller, T., Güsgen, H.-W., Hertzberg, J., Linster, M., Voß, A., & Voß, H. (1988). Expertise und ihre Modellierung auf dem Rechner. *etz*, 109, 1002-1006.

DARPA (Ed.). (1989). *Proceedings of the Second Workshop on Case-Based Reasoning, Holyday Inn, Pensacola Beach, Florida*. San Mateo, CA: Morgan Kaufmann.

di Piazza, J. S., & Helsabeck, F. A. (1990). Laps: Cases to models to complete expert systems. *AI Magazine*, 11 (3), 80-107.

Enzensberger, H. M. (1991). Hitlers Wiedergänger. *Der Spiegel*, 45 (6), 26-28.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1990). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41, 1-63.

Feigenbaum, E. A. (1984). Knowledge engineering: The applied side of artificial intelligence. *Annals of the New York Academy of Science*, 246, 91-107.

Glaser, R. (1986). On the nature of expertise In F. Klix, & H. Hagendorf (Eds.), *Human memory and cognitive capabilities* (pp. 915-928). Amsterdam: Elsevier.

F. Hayes-Roth, D. Waterman, & D. Lenat (Eds.). (1983). *Building expert systems.* Reading, MA: Addison-Wesley.

Hinrichs, T. R. (1988). Towards an architecture for open world problem solving. In *Proceedings of a Workshop on Case-based Reasoning, Holyday Inn, Clearwater, Florida* (182-189). San Mateo, CA: Morgan Kaufmann.

Hoffman, R. R. (1987). The problem of extracting the knowledge of experts from the perspective of experimental psychology. *AI Magazine*, Summer 1987, 53-67.

Holyoak, K. J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science*, 13, 295-355.

Jackson, P. (1986). *Introduction to Expert Systems.* Reading, MA: Addison-Wesley.

Karbach, W. (1989). Modellbasierte Wissensakquisition. *KI*, 4, 13.

Karbach, W., Linster, M., Voß, A. (1989). OFFICE-Plan: Tackling the synthesis frontier. In *Proceedings of GWAI* (pp. 379-387). Berlin: Springer.

Karbach, W., Voß, A., & Tong, X. (1988). Filling in the knowledge acquisition gap: Via KADS's models of expertise to ZDEST-2's expert systems. In *Proceedings of the Second European Knowledge acquisition workshop*, 31:1-17. Bonn: Gesellschaft für Mathematik und Datenverarbeitung.

Kolodner, J. L. (Ed.). (1988). *Proceedings of a Workshop on Case-based Reasoning, Holyday Inn, Clearwater, Florida.* San Mateo, CA: Morgan Kaufmann.

Kolodner, J. L., & Simpson, R. L. (1986). Problem solving and dynamic memory. In J. L. Kolodner & C. K. Riesbeck (Eds.), *Experience, memory and reasoning* (pp. 99-114). Hillsdale, N.J.: Lawrence Erlbaum.

Koton, P. (1988). Reasoning about evidence in causal explanations. In *Proceedings of a Workshop on Case-based Reasoning, Holyday Inn, Clearwater, Florida* (pp. 260-270). San Mateo, CA: Morgan Kaufmann.

Kaplan, D., Leslie, J., & Schooler, C. (1990). Problem solving by reference to rules or previous episodes: The effect of organized training, analogical models, and subsequent complexity of experience. *Memory & Cognition*, 18, 215-227.

Kurbel, K. (1989). *Entwicklung und Einsatz von Expertensystemen*. Berlin: Springer.

Minsky, M. (1985). The society of mind. New York: Simon & Schuster.

Musen, M. A. (1989). *Automated Generation of model-based knowledge acquisition tools.* Pitman: London.

Neustadt, R., & May, E. (1986). *Thinking in time: The use of history for decision makers*. New York: The Free Press.

Norman, D. A. (1983). Some observations on mental models. In D. Gentner, & A. Stevens (Eds.), *Mental Models.* Hillsdale, NJ: Lawrence Erlbaum.

Osherson, D. N. (1987). New axioms for the contrast model of similarity. *Journal of Mathematical Psychology*, 31, 93-103.

Pfitzner, K. (1990). Die Auswahl von Bibliothekslösungen mittels induzierter Problemklassen. In *Beiträge zum 4. Workshop Planen und Konfigurieren, Ulm, Mai 1990.*

Pressman, R. S. (1987). *Software engineering.* New York: McGraw-Hill.

Raaijmakers, J. G. W., & Shiffrin, R. M. (1981). Search of associative memory. *Psychological Review, 88,* 93-134.

Reed, S. K., Dempster, A., & Ettinger, M. (1985). Usefullness of analogous solutions for solving algebra word problems. *Journal of Experimental Psychology: Learning, Memory and Cognition,* 11, 106-125.

Riesbeck, C. K. (1988). An interface for case-based knowledge acquisition. In J. L. Kolodner (Ed.), *Proceedings of a Workshop on Case-based Reasoning, Holyday Inn, Clearwater, Florida* (pp. 312-326). San Mateo, CA: Morgan Kaufmann.

Riesbeck, C. K., & Schank, R. C. (1989). *Inside case-based reasoning.* Hillsdale, NJ: Erlbaum.

Rissland, E. L. (1986). Learning how to argue: Using hypotheticals. In J. L. Kolodner & C. K. Riesbeck (Eds.), *Experience, memory and reasoning.* Hillsdale, NJ: Lawrence Erlbaum.

Rissland, E. L., & Skalag, D. B., (1989). Combining case-based and rule-based reasoning: A heuristic approach. In *Proceedings of the International Joint Conference on Artificial Intelligence 1989,* August 1989, Detroit.

Ross, B. H. (1984). Remindings and their effects in learning a cognitive skill. *Cognitive Psychology,* 16, 371-416.

Selfridge, O. G. (1959). Pademonium. A paradigm for learning. In *The mechanisms of thought processes.* London: H. M. Stationery Office.

Shavlik, J. W., DeJong, G. F., & Ross, B. H. (1987). Acquiring special case schemata in explanation-based learning. *Proceedings of the Ninth International Annual Conference of the Cognitive Science Society* (pp. 851-860). Hillsdale, NJ: Erlbaum.

Schmalhofer, F. & Bergmann, R. (1990). Case-based knowledge acquisition: Extending expert problem solutions for technical planning tasks. *Paper presented at the GI-Workshop Knowledge Engineering. Berlin 26.4-27.4.1990.*

Schulz, A. (1989). Software-Lifecycle und Vorgehensmodelle. *Angewandte Informatik,* 4, 137-142.

Shavlik, J. W., DeJong, G. F., & Ross, B. H. (1987). Acquiring special case schemata in explanation-based learning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society* (pp. 851-860). Hillsdale, NJ: Lawrence Erlbaum.

Strube, G. (1989). Episodisches Wissen. *Arbeitspapiere der GMD,* 385, 10-26.

Strube, G., & Janetzko, D. (1990). Episodisches Wissen und fallbasiertes Schließen: Aufgaben für die Wissensdiagnostik und die Wissenspsychologie. *Schweizerische Zeitschrift für Psychologie,* 49, 211-221.

Thagard, P., & Holyoak, K. J. (1989). Why indexing is the wrong way. In DARPA (Ed.), *Proceedings of a Workshop on Case-Based Reasoning* (pp. 36-40). Holiday Inn, Pensacola Beach, Florida, May 31 - June 2, 1989. San Mateo, CA: Morgan Kaufmann.

Tulving, E., & Thompson, D. M. (1973). Encoding specifity and retrieval processes in episodic memory. *Psychological Review,* 80, 352-373.

Ueberreiter, B. & Voß, A. (1991). (Eds.) *Materialien KADS Benutzer Treffen.* München: Siemens.

Voss, J. F., Greene, T. R., Post, T. A., & Penner, B. C. (1983). Problem solving in the social . In G. Bower (Ed.), *The psychology of learning and motivation: Advances in research theory.* New York Academic Press.

Voß, A., Karbach, W., Drouven, U., Lorek, D., & Schukey, R. (1990). Operationalization of a synthetic problem. *Task I.2.1 Report. ESPRIT Basic Research Project P3178 REFLECT.* Bonn: Gesellschaft für Mathematik und Datenverarbeitung.

Wielinga, & Breuker (1984). Interpretation models for knowledge acquisition. In T. O'Shea (Ed.), *Advances in Artificial Intelligence.* Amsterdam: North-Holland.

Wielinga, B., & Breuker, J. (1986). Models of expertise In *Proceedings of the European Conference on Artificial Intelligence, Brighton, 1986* (pp. 306-318).

Wippich, W. (1982). Erinnerungen erinnern: Ein automatischer Vorgang? *Zeitschrift für experimentelle und angewandte Psychologie, 29*, 517-530.

Dipl. Psych. Dietmar Janetzko
Prof. Dr. Gerhard Strube

Institut für Informatik und Gesellschaft
Abteilung Kognitionswissenschaft
Universität Freiburg
Friedrichstr. 50
D-7800 Freiburg i.Br.

The Refitting of Plans by a Human Expert

Franz Schmalhofer, Christoph Globig, Jörg Thoben

German Research Center for Artificial Intelligence
University Bldg 57
Erwin-Schroedinger Str.
W-6750 Kaiserslautern

email: schmalho@informatik.uni-kl.de

Abstract. During the course of the development of a Case-Oriented Expert System for situated applications additional cases were needed. The required cases were obtained by having a human expert refit old solutions to new problems and the structural relations between source and target cases were analyzed: A higher degree of reuse of the old cases was found when the expert could apply derivational reasoning and a uniform design rationale (i.e. the solution of the source was generated by the expert himself) than when the expert could only analyze structural relationships (i.e. the source solution was constructed by some one else). Except with very obvious cases, it was also found, that different experts perceive different cases as the most similar source to a given target problem. The results also indicate for user-situated applications of expert systems.

1. Introduction

In order to overcome the brittleness of first generation expert systems, it has recently been proposed to develop Case-Oriented Expert Systems (COEx-Systems), which allow situated applications (Schmalhofer & Thoben, 1992). One prerequisite for developing such a system is that a sufficient number of prototypical cases are available for the desired competence of the system. Since originally we had only very few cases, we had an expert generate solutions to additional prototypical problems by having him refit old solutions, so that they would become solutions for those problems.

The current paper first reviews the integrated knowledge acquisition method (Schmalhofer, Kühn & Schmidt, 1991) for COEx-Systems together with their general characteristics. We then present a structural analysis of the refitted plans. Finally several conclusions with respect to the development of expert systems and the situated applications of old cases are drawn.

2. Case-Oriented Expert Systems for Mechanical Engineering Planning Tasks

In the knowledge acquisition phase for such COEx-Systems, model-based abstractions are formed from concrete past experiences, so that they can be reused in novel situations. Human expert judgments concerning the classification and similarities of the concrete past experiences are applied to obtain an abstraction hierarchy of problem classes (Bergmann & Schmalhofer, 1991; Schmalhofer, Reinartz & Tschaitschian, in press) and supplementary knowledge from written materials is used to obtain explicit operator definitions (Schmidt, 1992) so that associated skeletal plans can be constructed (Bergmann, 1992; Friedland, 1985).

The knowledge acquisition for such systems thus yields an abstraction hierarchy of problem classes with associated skeletal plans which allow for a situated utilization of past experiences in future tasks. During the knowledge acquisition phase, these past experiences have been interpreted by one or several experts within some uniform rationale. More details about such systems can be found in Schmalhofer & Thoben (1992). The respective knowledge acquisition procedures and tools were summarized by Schmalhofer, Bergmann, Kühn & Schmidt (1991). The model of expertise or problem solving model (Breuker & Wielinga, 1989), which underlies COEx-Systems for planning tasks has been described by Kühn & Schmalhofer (1992).

Our research group has recently been developing such a system for production planning problems in mechanical engineering. Without going into any details of this application domain, we can state that production planning is a typical planning problem: For example, the mold of the workpiece defines the given state and the goal workpiece defines the goal state of the manufacturing problem. A number of different types of operations (chucking, unchucking, cutting operations) are available for transforming the mold (given state) to the goal workpiece (goal state). The operations themselves are quite complex requiring the specification of a number of different parameters (such as cutting path specification, specific cutting parameters, toolholders, etc.). It is therefore very useful to classify and abstract operations to different types of macro-operators.

Workpiece

	Geometry	Drive Shaft g1				Drive Shaft g2				Pinion Shaft g3				Axle Shaft g4				Axle Shaft g5			
	Workpiece material	w1	w2	w3	w4	w1	w2	w3	w4	w1	w2	w3	w4	w1	w2	w3	w4	w1	w2	w3	w4
Manufacturing machines (lathes)	d1					3	3	1		3 *	3 *	1 *	*	4	4	2		4	4	2	*
	d2	5 *	5	7	9 *	5	5	7 *	9	m3 5	5 *	7	9	6	6	8	10	6	6	8 *	10
	d3	m1 5 *	5 *	7	9 *	m2 5 *	5	7	9 *	5	5 *	7	9	m4 6	6	8	10	m5 6	6	8 *	10

Table 1 (after Schmalhofer & Thoben, 1992): A number of specific problems are used in order to delineate the competence of the future expert system. From the factorial combination of three types of manufacturing machines (d_1,d_2, and d_3), and workpieces with five different types of geometries (g_1, g_2, g_3, g_4, and g_5) and materials (w_1, w_2 w_3, and w_4) fifty-two problems were identified as meaningful. The numbers 1 to 10 indicate the abstract problem classes to which a specific prototypical problem belongs. An abstraction hierarchy for these problem classes is shown in Figure 1. See text for further explanation.

Since a production plan strongly depends upon the specific geometry of the workpiece (**g**), the workpiece material (**w**), and the particular machine (**d**), which are to be used when manufacturing the workpiece, we denote production problems with the descriptors **g**, **w**, and **d**. By using different indices with these descriptors we can thus refer to a given manufacturing problem.

In Table 1 sixty production problems are specified through the factorial combination of 3 manufacturing machines (d_1, d_2, and d_3), five different geometries (g_1, g_2, g_3, g_4, and g_5) and four different workpiece materials (w_1, w_2, w_3, and w_4). Fifty-two of these problems (all problems whose cells are marked by a number between 1 and 10) are the prototypical problems, which delineate the desired competence of the future expert system. Problems with the same number were assigned to the same abstract problem class. The abstraction hierarchy of these ten abstract problem classes is shown in Figure 1.

Since only five production plans were originally available for the 52 prototypical problems, i.e. the cases m_1, m_2, m_3, m_4, and m_5 (see Table 1), an expert refitted these plans (refitting roots) and his subsequently generated plans (refitting children) for 16 of the 52 prototypical problems. He also constructed one production plan from scratch ($g_5w_4d_1$). In Table 1, the problems with associated refitted plans are indicated by the asterisks.

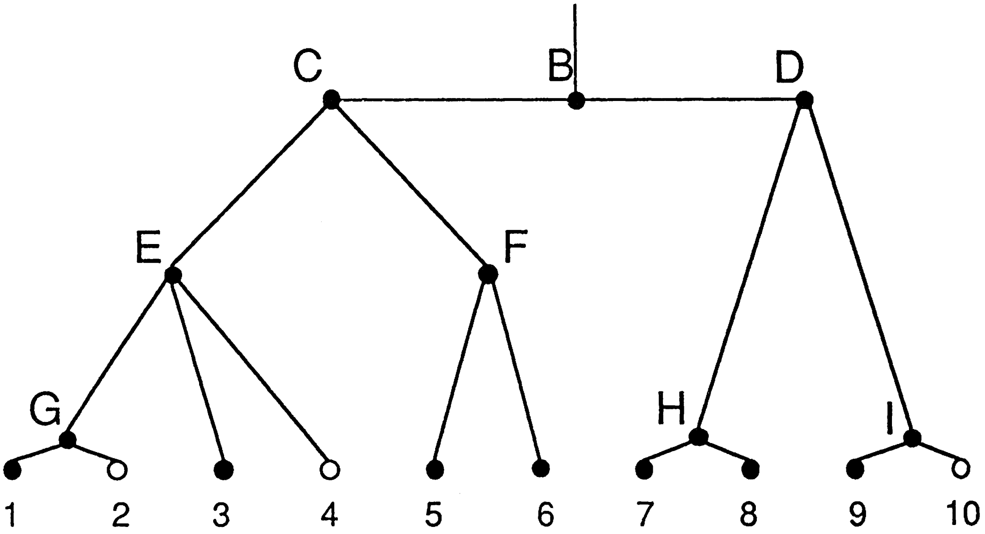


Figure 1: Shown is the problem abstraction hierarchy for 10 abstract problem classes (see Table 1)

3. Plan Refitting

Figure 2 identifies the different source cases which the expert used for finding solution plans for the 16 target problems: The source-target case relation is indicated by an arrow. Whereas case m_3 was five times used as a direct source, the cases m_1, m_2, and m_4 were each only used once as a direct source and case m_5 was never used as a source. On 8 occasions one of the cases which had already been tested in the real world (tested source case or refitting root) were used as source and 8 times a solution plan which the expert had generated himself (i.e. a refitting child) was used as source (self-generated source case).

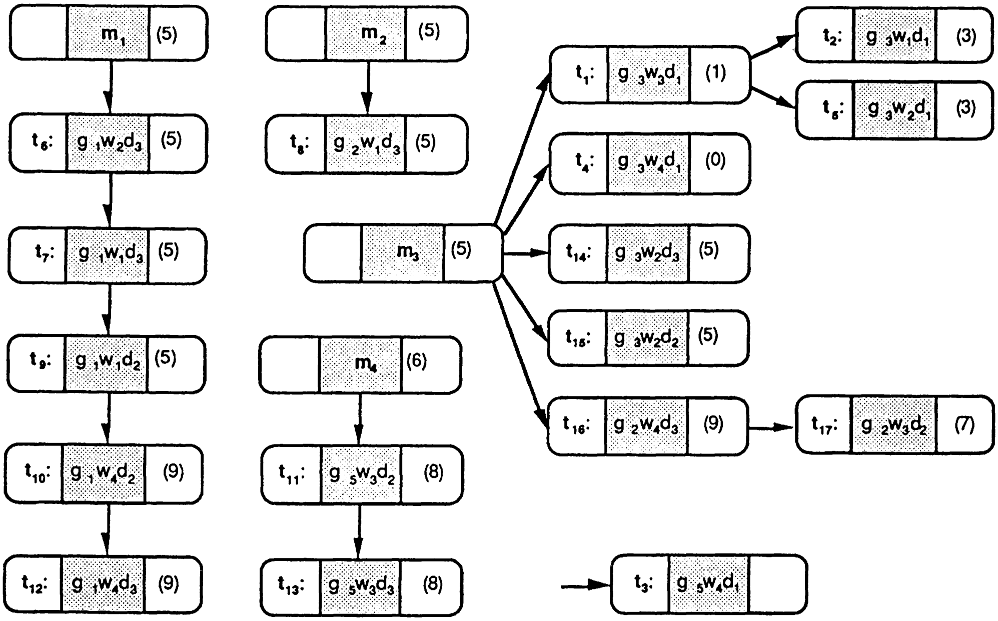


Figure 2 (after Schmalhofer & Thoben, 1992): The source case - target case relation is shown for the 17 tasks (t_1 to t_{17}) which were solved by the expert. In parentheses the abstract problem class that a specific task is associated with (see Figure 1) is noted. Task t_3 was solved from scratch, so that there is no source case associated with it. Whereas the cases m_1, m_2, m_3, and m_4 served as refitting roots, the other cases are denoted as refitting children.

The task numbers t_1 to t_{17} indicate the temporal order in which the 16 refitting and the one plan construction task (t_3) were performed by the expert. These numbers show that the immediately preceding target solution was very often used as the source for the next target problem. For example the solution to task t_6 was used as a source for t_7 and the solution to t_{14} was used as the source for t_{15}. On other occasions somewhat earlier preceding target solutions were used as the source for the current target problem. For example, the last but one target solution was used as the source for task t_{13}. These temporal relationships indicate that for the refitting of old plans, the expert tried to maintain a fresh memory of the modification processes by which he constructed the old plan.

When the expert remembers his reasoning (i.e. the derivations), by which he constructed or modified the old plan, he can perform derivational refitting processes (Carbonell, 1986). When the old cases was generated by somebody else, as for example the tested source cases m_1, m_2, m_3, m_4 and m_5 (i.e. the refitting roots), the expert is more likely to perform only structural refitting processes (Hammond, 1989). Another important observation was: The plans which were obtained by modifying an already existing plan were completed by an order of magnitude faster than the plan which was produced from scratch (t_3).

3.1 Different Types of Modifications between Source and Target Plans

We also analyzed more detailed structural relations between the source and target plans. Thereby it was distinguished between the refitting of tested source cases (i.e. refitting roots), where the expert was very likely to only use structural analogies and the refitting of self-generated source plans (i.e. refitting children), where the expert could at least to a certain degree also apply derivational analogies.

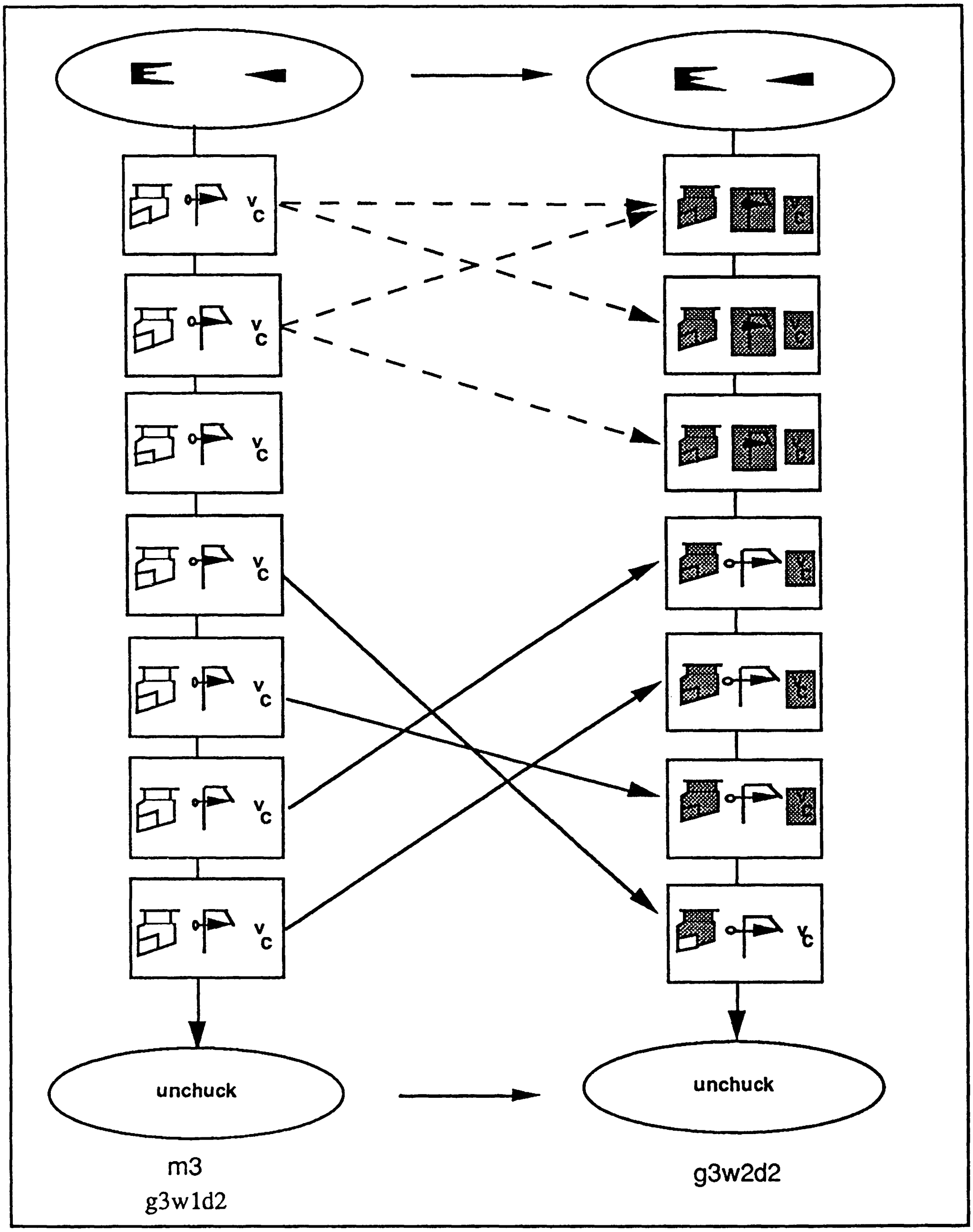


Figure 3: Shown are the structural relations between the tested source case m_3 for problem $g_3w_1d_2$ and the resulting target plan for problem $g_3w_2d_2$. See text for further explanation.

Figure 3 shows the structural relationships between the operators of the tested source case m3 and the refitted plan for the manufacturing problem $g_3w_2d_2$. The Figure shows the structural relationships between corresponding operators of the source plan m_3 for problem $g_3w_1d_2$ and the target plan for problem $g_3w_2d_2$ at the macro level. The ovals represent chucking and unchucking operations. All cutting (macro-)operations are indicated by rectangles. Within these rectangles, 1) the toolholder together with cutting tool, 2) the cutting path, and 3) the cutting parameter v_c are symbolically represented from left to right. Shaded symbols in the target plan indicate changes from the source to the target. The solid lines with arrows indicate which operations of the source were reused in the target plan. The dashed lines indicate substantial changes in the individual operations themselves.

The first two cutting operations of the source plan (see left side of Figure 3) were splitted apart and the resulting components were rejoined across the original operations of the source plan. Two new operations were thus created, which differ in all three parameters from the operations in the source case (see right side of Figure 3). As a consequence, the third cutting operation of the source was completely eliminated from the target.

The execution order of the fourth and the fifth cutting operation of the source was also changed in the target. While the cutting path remained identical, cutting tool and cutting parameters were adjusted to the new workpiece material. The same modification was performed for the sixth and the seventh operation, except that these operations were not reordered in the target plan.

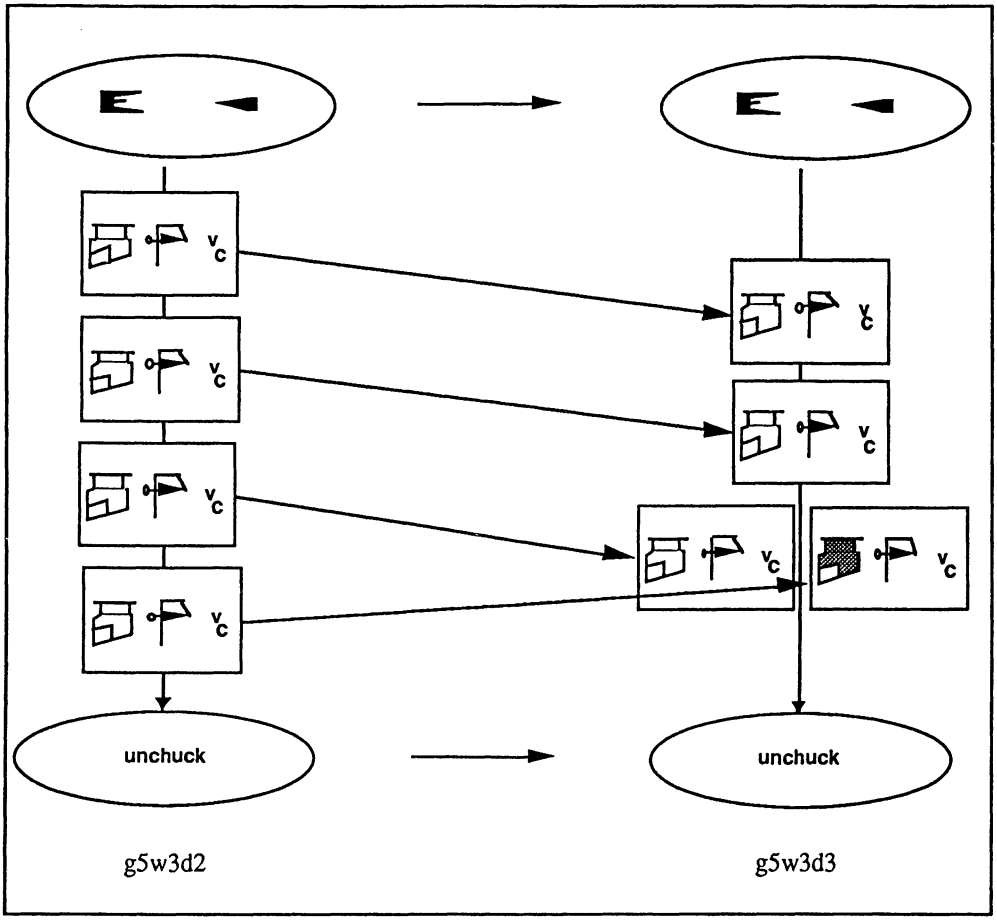


Figure 4: Shown are the structural relations between a self-generated source and the resulting target case. See text for further explanation.

Figure 4 shows the structural relationships between the self-generated source plan $g_5w_3d_2$ and a target plan which is refitted for a machine which allows parallel processing. Whereas the chucking operations as well as the first two cutting operations remain identical, the third and fourth operations of the source are now executed in parallel in the target plan. In addition, one of the toolholders is changed. This source-target pair thus shows a large degree of reuse of the operations and the execution sequence of the old plan.

3.2 Comparison of Structural Relations among Four Different Plan Pair Groups

We compared the structural relations among four different groups of plan pairs. The first group consisted of the 8 pairs, which contained tested source cases (t_1, t_4, t_6, t_8, t_{11}, t_{14}, t_{15}, and t_{16}). The second group of plan pairs contained the 8 pairs with self-generated source plans (t_2,

type of change	The 16 actual modification tasks		The 11most similar case pairs	
	source is refiffing root	source is refitting child	source is refiffing root	source is refitting child
additional chuckings	0.25	0.13	0.00	0.14
eliminated chuckings	0.00	0.13	0.00	0.14
new parallel executions	0.13	0.25	0.25	0.29
new serial executions	0.13	0.25	0.25	0.14
splitted operations	0.75	0.00	0.50	0.00
joined operations	1.13	0.00	1.25	0.00
reordering of operations	0.75	0.50	1.00	0.43
cutting path changes	3.75	0.25	6.75	1.43
cutting parameter changes	5.63	4.25	8.75	2.86
toolholder changes	4.88	2.50	8.50	1.57
cutting tool changes	5.25	4.00	8.00	2.86
total number of case pairs	8	8	4	7
total number of cuts in source	61	57	34	46
total number of cuts in target	53	55	36	41

Table 2: Average number of different types of changes from the source to the target case for the 16 performed modification tasks (see Figure 2) and 11 most similar case pairs from the abstraction hierarchy.

t_5, t_7, t_9, t_{10}, t_{12}, t_{13}, and t_{17}). The abstraction hierarchy of problem classes (see Figure 1 and for more details Schmalhofer & Thoben (1992)) was used for defining the third and fourth group of plan pairs. More specifically, for each of the 16 target plans, the most similar plan according to the abstraction hierarchy was selected as a hypothetical source case and the structural relations of these case pairs were analyzed. Group 3 contains the plan pairs, where the source plan was m_1, m_2, m_3, or m_4 (i.e. the refitting roots): m_3 - $g_1w_1d_3$, m_1 - $g_1w_1d_3$, m_2 - $g_2w_1d_3$, m_3 - $g_3w_2d_2$. Group 4 contains the plan pairs, where the source plan was a self-generated plan: $g_3w_3d_1$ - $g_3w_1d_1$, $g_3w_3d_1$ - $g_3w_2d_1$, $g_1w_1d_3$ - $g_1w_1d_2$, $g_1w_4d_2$ - $g_1w_4d_3$, $g_5w_3d_2$ - $g_5w_3d_3$, $g_5w_3d_2$ - $g_2w_3d_2$, $g_1w_4d_3$ - $g_2w_4d_3$.

The results of this analysis are shown in Table 2. In general, fewer structural changes were observed between the (real or hypothetical) source and the target case, when the source case was also self-generated (i.e. a refitting child) than when the source case was generated by somebody else (i.e. a refitting root). And as expected, changes of the operations themselves occurred less frequently than parameter changes (e.g. cutting parameter changes).

3.3 Assessing the Expert's Consistency in the Source-Case Selections

In order to assess the expert's consistency in selecting the same source case as the most similar one to a given target problem, further data were collected from the expert who had performed the 16 refitting tasks (HW). In addition an additional expert (RL) had to perform the same task. The task consisted in selecting the most similar source from the cases m_1, m_2, m_3, m_4 and m_5 to each of the 16 target problems, for which a plan modification was performed. In addition, the similarity between the source and target problem had to be estimated by a number between 1 and 7. Whereas 1 meant the lowest similarity, 7 indicated the highest possible similarity. Table 3 shows the results in comparison to the actually used source case. For self-generated source cases, the refitting roots (see Figure 2) were also determined.

From Table 3 it can be seen that the cases which were identified as most similar by HW correspond in only 50 percent to the actually selected source case or root of the source case (i.e. the refitting root) in the refitting task. There is also only a 47 percent consistency between the two experts. However, when only those cases, which were identified as most similar with a similarity rating of 7 are considered, the two experts agreed in 100 percent of the cases. More details have been reported by Thoben, Schmalhofer & Reinartz.

4. Conclusion

Our main purpose for having an expert refit old plans to new problems was to obtain a sufficient number of cases for developing a Case Oriented Expert System for production planning in mechanical engineering. Although there is now a sufficient number of cases available for constructing skeletal plans for the important set of medium level problem classes (i.e. for all classes with a solid node in Figure 1), further prerequisites must be satisfied. Unlike case-based reasoning which does not make such strong prerequisites, Case Oriented Expert Systems require that all prototypical cases follow the same design rationale. This requirement arises from the fact, that several layers of more and more abstract skeletal plans are to be constructed from these cases, so that deductive justifications will exist for the resulting state and operator sequence abstraction mappings (Bergmann & Schmalhofer, 1992). We will consequently have to test, whether the cases of the refitting roots (m_1, m_2, m_3, m_4, and m_5) follow the same design rationale as the cases generated by the expert HW.

problem		HW		RL
task	target case	actually used source with (refitting root)	most similar case identified	most similar case identified
t_1	g3w3d1	m3	m5 : 5	m3 : 1
t_2	g3w1d1	g3w3d1 (m3)	m4 : 6	m3 : 2
t_4	g3w4d1	m3	m4 : 3	m3 : 1
t_5	g3w2d1	g3w3d1 (m3)	m4 : 4	m3 : 3
t_6	g1w2d3	m1	m1 : 3	m2 : 6
t_7	g1w1d3	g1w2d3 (m1)	m1 : 4	m2 : 6
t_8	g2w1d3	m2	m2 : 7	m2 : 7
t_9	g1w1d2	g1w1d3 (m1)	m1 : 4	m2 : 6
t_{10}	g1w4d2	g1w1d2 (m1)	m1 : 6	m1 : 5
t_{11}	g5w3d2	m4	m5 : 7	m5 : 7
t_{12}	g1w4d3	g1w4d2 (m1)	m1 : 7	m1 : 7
t_{13}	g5w3d3	g5w3d2 (m4)	m5 : 7	m5 : 7
t_{14}	g3w2d3	m3	m3 : 7	m3 : 7
t_{15}	g3w2d2	m3	m3 : 7	m3 : 5
t_{16}	g2w4d3	m3	m2 : 3	m2 : 3
t_{17}	g2w3d2	g2w4d3 (m3)	m2 : 4	m5 : 4

Table 3: Consistency assessment between two experts (HW and RL) and two different tasks: Actual source selection and most similar case identification with similarity judgement.

Our study also yielded a typology for the structural relations between the old and the refitted plans. In some situations refitting purely consisted of small scale modifications (e.g. parameter changes) of the building blocks (i.e. macrooperators) of a plan, while the global structure of the plan (e.g. a complete or partial execution order) was maintained. Under other circumstances, the global structure of the plan was modified according to some well justifiably rationale. In still other situations, rather creative processes were applied: Operations were splitted and rearranged in different ways and the execution order was changed in a quite unpredictable way (see Figure 3). Such changes may be an indication for different underlying design rationales. The inconsistencies between different experts are another indication for idiosyncratic planning rationales.

The expert's refitting task is also similar to the task a user would perform with the expert system. As the expert in our study, the user (or the system) has to select the most similar abstract (or concrete) plan and refine (or refit) it to the problem at hand. From the observation, that different experts preferred different plans to be most similar to a given problem, we may conclude that expert systems should accommodate such differences in personal user preferences. In other words expert systems should be more user-oriented and user-situated applications should also be possible in expert systems.

Acknowledgments

This research was supported by grant ITW 8902 C4 from BMFT (German Ministry for Science and Technology) and by grant Schm 648/1 from DFG (German Science Foundation). We would like to thank Dipl.-Ing. Ralf Legleitner and Hans-Werner Höper for serving as experts in this study. Thomas Reinartz helped in analyzing the recorded materials. The comments of Angi Voss and Stefan Wess on a previous version of this paper are also much appreciated.

References

Bergmann, R. (1992). Knowledge Acquisition by generating skeletal plans from real world cases. In Schmalhofer, F., Strube, G., & Wetter, T. (Eds.), Contemporary Knowledge Engineering and Cognition (pp. $pages). Berlin/Heidelberg: Springer-Verlag.

Bergmann, R. & Schmalhofer, F. (1991). CECoS: A case experience combination system for knowledge acquisition for expert systems. Behavior Research Methods, Instruments, & Computers, 23, 142-148.

Bergmann, R. & Schmalhofer, F. (1992). Learning Plan Abstractions: Formal Model and Method. In Biundo, S. & Schmalhofer, F. (Eds.), Proceedings of the DFKI Workshop on Planning, DFKI-Document D-92nn, pp. 20-24.

Breuker, J. & Wielinga, B. (1989). Models of expertise in knowledge acquisition. In Guida, G. & Tasso, C. (Eds.), Topics in expert system design, methodologies and tools (pp. 265 - 295). Amsterdam, Netherlands: North Holland.

Carbonell, J.G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Michalski, R.S., Carbonell, J.G., & Mitchell, T.M. (Eds.), Machine Learning: An artificial intelligence approach (Vol. 2, pp. 371-392). Los Altos, CA: Morgan Kaufmann.

Friedland, P.E. & Iwasaki, Y. (1985). The concept and implementation of skeletal plans. Journal of Automated Reasoning, 1, 161-208.

Hammond, K. (1989). Case-based planning. London: Academic Press.

Kühn, O. & Schmalhofer, F. (1992). Hierarchical skeletal plan refinement: Task-and inference structures. In Bauer, C. & Karbach, W. (Eds) Proceedings of the 2nd KADS User Meeting (pp. 201-210) München: Siemens AG.

Schmalhofer, F. & Bergmann, R. (1992). Plan Recognition by Constructing Skeletal Programs as Abstractions from Symbolic Execution Traces, manuscript, DFKI Kaiserslautern.

Schmalhofer, F., Bergmann, R., Kühn, O., & Schmidt, G. (1991). Using integrated knowledge acquisition to prepare sophisticated expert plans for their re-use in novel situations. In Christaller, T. (Ed.), GWAI-91: 15. Fachtagung Künstliche Intelligenz (pp. 62-73). Berlin: Springer-Verlag.

Schmalhofer, F., Kühn, O., & Schmidt, G. (1991). Integrated knowledge acquisition from text, previously solved cases, and expert memories. Applied Artificial Intelligence, 5, 311-337.

Schmalhofer, F., Reinartz, T. & Tschaitschian, B. (in press). Intelligent documentation as a catalyst for developing cooperative knowledge-based systems. In Wetter, Th., Althoff, K.D., Boose, J., Gaines, B. Linster, M. & Schmalhofer, F. (Eds) Current Developments in Knowledge Acquisition: EKAW-92 Heidelberg: Springer-Verlag.

Schmalhofer, F. & Thoben, J. (1992). The model-based construction of a case oriented expert system. AI-Communications, 5, 1, 3-18.

Thoben, J., Schmalhofer, F., & Reinartz, T. (1991). Wiederholungs- Varianten- und Neuplanung bei der Fertigung rotationssymmetrischer Drehteile (DFKI-Document No. D-91-16). Kaiserslautern, Germany: German Research Center for Artificial Intelligence.

Knowledge Acquisition by Generating Skeletal Plans from Real World Cases

Ralph Bergmann
German Research Center for Artificial Intelligence
University Bldg 57
Erwin-Schroedinger Str.
D-6750 Kaiserslautern

email: bergmann@informatik.uni-kl.de

Abstract. Although skeletal plan refinement is used in several planning systems, a procedure for the automatic acquisition of such high-level plans has not yet been developed. The proposed explanation-based knowledge acquisition procedure constructs a skeletal plan automatically from a sophisticated concrete planning case. The classification of that case into a well-described class of problems serves as an instrument for adjusting the applicability of the acquired skeletal plans to that class. The four phases of the proposed procedure are constituted as follows: In the first phase, the execution of the source plan is simulated, and explanations for the effects of the occurred operators are constructed. In the second phase, the generalization of these explanations is performed with respect to a criterion of operationality which specifies the vocabulary for defining abstract operators for the skeletal plan. The third phase, a dependency analysis of the resulting operator effects, unveils the interactions of the concrete plan which are substantial for the specified class. In the forth phase, the concept descriptions for the abstract operators of the skeletal plan are formed by collecting and normalizing the important constraints for each operation that were indicated by the dependencies. With this procedure sophisticated planning solutions from human experts can be generalized into skeletal plans and consequently be reused by a planning system in novel situations.

1. Introduction

Many planning problems can be subdivided into more or less specific classes of problem types, in which each class has its own general solution plan (Tu, Kahn, Musen, Ferguson, Shortliffe & Fagan 1989). All plans of one class can thus be said to use the same solution idea, viewed from a higher level of abstraction. Such an abstract plan is called *skeletal plan* by Friedland and Iwasaki (1985) and is defined as follows:

A skeletal plan is a sequence of abstract and only partially specified steps which, when specialized to specific executable operations, will solve a given problem in a specific problem context. Skeletal plans exist at many levels of generality and are applicable to different classes of planning problems. They capitalize on beneficial interactions and avoid detrimental interferences between the single operations of a plan. Skeletal plan operations just need the right level of generality in the respective situation, to maintain and replay important interactions and eliminate irrelevant details which are easy to reconstruct.

Classical planning mechanisms such as specialization rules, (Shortliffe, Scott, Bischoff, Campbell, Melle & Jacobs 1981) or heuristic approaches (Friedland & Iwasaki, 1985) may be applied for obtaining a concrete plan from a good skeletal plan. A skeletal plan is refined to a concrete plan by specializing abstract operators independent of each other, so that the search in the space of concrete operators becomes feasible. This requires that the important interactions between operators, which always occur even in simple realistic planning tasks, must be taken into account during the construction of the skeletal plans. Therefore, the quality of the skeletal plan data base mainly determines the quality of the results of such a planning system.

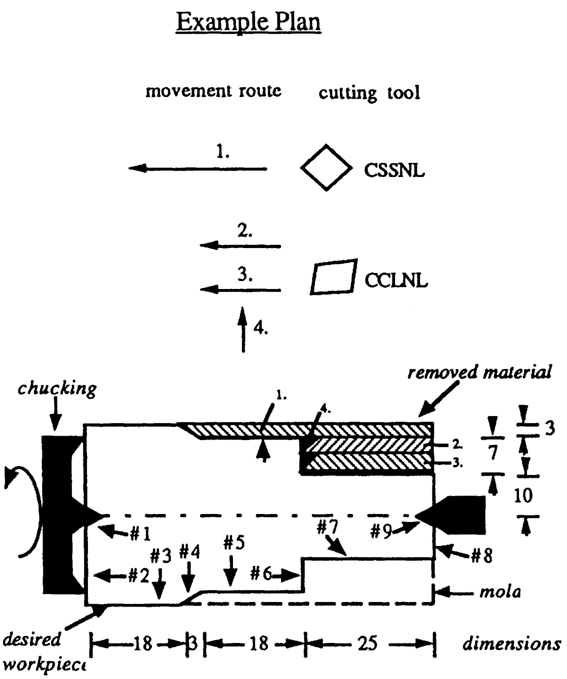
Nevertheless, the problem of the acquisition of skeletal plans has not yet been solved. In OPAL (Musen, Fagan, Combs & Shortliffe 1987), the knowledge acquisition system for the ONCOCIN expert system, oncological therapy protocols, which function as skeletal plans in this domain, must be constructed and entered manually with support of a graphical editor. For Friedlands MOLGEN planner the situation is similar. The acquisition and debugging of skeletal plans has been identified as a major problem (Stefik, 1981). This is because constructing skeletal plans is a modelling task which requires the definition of a terminology sufficiently abstracting from details which are irrelevant for the planning task. Usually neither an abstract planning terminology nor skeletal plans described in terms of such a terminology are directly available in real world domains. Schmalhofer and Thoben (this volume) have studied domain experts who are requested to construct skeletal plans for mechanical engineering. Their approach to the manual construction of skeletal plans seems to be successful but is quite time-consuming for the expert.

This current paper investigates explanation-based learning (EBL) (Mitchell, Keller, Kedar-Cabelli 1986; DeJong & Mooney 1986) in order to establish an *automatic* knowledge acquisition method for obtaining skeletal plans from real world problem solutions for a given class of problems. A theory that describes important aspects of the operator effects is used to simulate the execution of the plan and to derive an explanation of how the plan solves those problem features that define the problem class.

In the following sections, the real world application domain of mechanical engineering is introduced together with an example, and the representation of skeletal plans is discussed. The proposed method for an automatic knowledge acquisition is subdivided into four phases: a plan simulation and explanation phase, a generalization phase, a dependency analysis, and a normalization into the skeletal plan representation. In the final discussion the benefits and the limitations of this approach are evaluated in a general manner.

2. Production Planning in Mechanical Engineering

The planning domain used as the field of demonstration is mechanical engineering, more specifically the production of rotational parts on a lathe. Presently, the design of mechanical work pieces is widely supported by CAD systems, and computer controlled lathes (CNC-machines) are used for manufacturing such parts. The planning process itself cannot be performed automatically since a lot of different kinds of domain knowledge are required for the construction of good plans. The characteristics of the complexity of this domain and the planning process are presented in detail by Schmalhofer and Thoben (this volume) through a set of planning tasks from a catalogue of a supplier of machining centers and tools. For the demonstration of the automatic approach to the acquisition of skeletal plans, only a simplified version of an already existing real world planning problem is introduced for clarity. Figure 1 shows an example of the work piece to be produced together with a production plan which consists of one chucking and four cutting operations.



Formalization of the plan:

chuck (lathe_dog, '30Kg', left)
cut('CSSNL',
form(linear,(64,17), (21,17)),
speed(450,0.45))
cut('CCLNL',
form(linear,(64,14),(39,14)),
speed(450,0.45))
cut('CCLNL',
form(linear,(64,10),(39,10)),
speed(450,0.45))
cut('CCLNL',
form(linear,(39,10),(39,17)),
speed(150,0.25))

Formalization of the desired workpiece

surface(#1,form(linear, (3,0), (0,3))) ∧
centerhole(#1,'Zen3mm') ∧
surface(#2,form(linear, (0,3), (0,20))) ∧
facing_area(#2) ∧
surface(#3,form(linear,'(0,20), (18,20))) ∧
surface(#4,form(linear, (18,20), (21,17))) ∧
surface(#5,form(linear, (21,17), (39,17))) ∧
surface(#6,form(linear, (39,17), (39,10))) ∧
tolerance(#6,low) ∧
surface(#7,form(linear, (39,10), (64,10))) ∧
surface(#8,form(linear, (64,10), (64,3))) ∧
facing_area(#8) ∧
surface(#9,form(linear, (64,3), (61,0))) ∧
centerhole(#9,'Zen3mm',) ∧

Figure 1:
In this example, a mold is fixed with a lathe dog, and material is removed in four steps, in which two cutting tools with different shapes are used. The material that is removed by each step is indicated by the shaded areas on the sketch of the mold. Note that this graphical sketch is only a two-dimensional sectional drawing of the three-dimenstional rotational part.

The formal description of the example is presented on the right side of this Figure. In this representation, the plan is defined as a sequence of operators together with their parameters. The workpiece is represented, similar to world states in STRIPS (Fikes & Nilsson, 1971), as a conjunction of predicate facts, in which each fact expresses one isolated attribute of the workpiece, such as a coherent surface area or a technological feature (centerhole, facing area or material) (Bergmann, Bernardi, Klauck, Kühn, Legleitner, Schmalhofer & Schmidt 1990). This representation of cases, which can easily be derived from the data representations CAD systems employ, is used as the input for the skeletal plan construction procedure.

2.1 Skeletal Plans in Mechanical Engineering

A skeletal plan consists by definition of a sequence of abstract operators or operator classes. Extensionally, an operator class is formed by grouping some concrete operators together. Intentionally, an operator class needs to be described by a combination of relevant attributes that all operators of that class have in common. A conjunction of constraints to some operator features, such as inductive methods of concept formation usually construct, are a useful manner for defining operator classes for a skeletal plan. In mechanical engineering, for example, a subclass of chucking operators may be formed by the following conjunctive description of three operator features:

chucking with centerholes **and**
chucking position on the left side **and**
two fixations.

2.2 Acquisition of Skeletal Plans

Automatic acquisition of skeletal plans by analysis of cases is itself a knowledge-intensive process. Knowledge is required to explain the functioning of the problem solution, to identify interactions between the operators, and a terminology is needed to construct the descriptions of operator classes. Therefore this automatic process is embedded into an integrated knowledge acquisition method, which has been described by (Schmalhofer, Schmidt & Kühn, 1991b; Schmalhofer, Bergmann, Kühn & Schmidt, 1991a).

Within this integrated knowledge acquisition method, an interactive tool named COKAM (Case-Oriented Knowledge-Acquisition Method from Text) (Schmidt & Schmalhofer, 1990) is used to extract information from text and the expert's common sense knowledge, guided by cases of problem solution. The formalization of this elicited knowledge serves as a model of operators which already contains the basic terms on which operator abstractions can be composed. This operator theory is required to be mostly complete and tractable to enable the application of the explanation-based learning procedure.

Another interactive knowledge acquisition tool named CECoS (Case-Experience Combination System) (Bergmann & Schmalhofer, 1991) yields a hierarchically structured set of problem classes from a set of prototypical cases through human expert judgements. The expert judgements are obtained so that a useful skeletal plan exists for each of the classes. An intensional definition of this class hierarchy, together with the classification of the origin case used for skeletal plan generation, is used to adjust the level of generality for the generated skeletal plan.

3. The Generation Procedure

The automatic generation of skeletal plans is based on an understanding of how a specific plan solves the given problem, and on recognizing those dependencies between the actions of the plan that are significant for a general solution for the whole problem class. A sequence of operator classes is constructed, so that the significant dependencies are maintained. The following detailed description of this approach is divided into four distinct phases.

3.1 Phase-I: Simulation and Explanation

This phase uses a domain theory that describes the applicability and the effects of operators for simulating the execution of the target plan. This theory formally represents each operator as a set of rules, in which the successor world state is created by the execution of the STRIPS like add- and delete actions of the rules' consequences. For example, two rules like *R1* and *R2* describe two effects of chucking operators with different generality. *R1* models the general effect of the execution of a chucking operator, whereas *R2* models the more specific consequence of a special chucking operator that requires centerholes.

R1: **IF** operator(chuck(x_{tool},x1,x2)) **THEN**
DELETE(unchucked),
ADD(chucked)

R2: **IF** operator(chuck(x_{tool},x1,x2)) $\wedge$
requires_centerholes(x_{tool}) $\wedge$
two_centerholes **THEN**
ADD(chuck_precision(high))

Additionally, a set of axioms is provided to infer the conditions of these rules from the descriptions of the world states. With the axioms *A1* and *A2* the condition of rule *R2* can be inferred to be true for the first operator in the example in Figure 1.

A1: (∃c1,c2,x1,x2,x3,x4. (centerhole(c1,x1,x2) ∧ centerhole(c2,x3,x4) ∧ c1 ≠ c2)) → two_centerholes.

A2: requires_centerholes(lathe_dog).

With a complete theory for all operators in the target plan, its execution is successfully simulated by sequentially applying all rules for the operators $Op_1,...,Op_n$ of the plan:

$$S_{ini} \xrightarrow{Op1} S_1 \xrightarrow{Op2} S_2 \text{ .... } S_{n-1} \xrightarrow{Opn} S_{goal}$$

From the initial state S_{ini} (the mold in mechanical engineering) all intermediate states that result after the execution of each operator, and the final state S_{goal} (the target workpiece) are computed. The proofs that exists for the applicability of each operator rule can now be seen as an explanation of each effect, which depends on operator attributes as well as on world state attributes.

3.2 Phase-II: Generalization

These proofs are generalized using EBL (Mitchell, Keller & Kedar-Cabelli, 1986; DeJong & Mooney 1986), which yields separate and more general concepts for the produced effects. The operationality criterion for EBL determines the vocabulary for expressing the generalized concepts and establishes the constraints from which the operator classes are constructed. These terms are initially provided through the application of COKAM. As a result of this generalization, a set of conditions is found which ensures more generally that the situations S1,...,Sn-1 and S_{goal} are created in the same manner. Look at the following example of four EBL generalized concepts for the effects of the first operator from Figure 1.

C1: chucked ←
operator(chuck(x_{tool}, x1,x2)).

C2: chucking_position(x_{pos}) ←
operator(chuck(x_{tool},x1,x_{pos})).

C3: chucking_precision(high) ←
operator(chuck(x_{tool}, x1,x2)) ∧
requires_centerholes(x_{tool}) ∧
centerhole(c1,x3,x4) ∧
centerhole(c2,x5,x6) ∧
c1 ≠ c2.

C4: chucking_fixations(x_n) ←
operator(chuck(x_{tool},x1,x2)) ∧
number_of_fixations(x_{tool},x_n).

3.3 Phase-III: Dependency Analysis

The task of the dependency analysis is to identify those effects of the operators which are necessary to guarantee that those features of the workpieces which are named as relevant for the classification of the workpiece, are created by every specialization of the skeletal plan. Therefore, the interconnections between the separate concepts which were identified in the second phase are determined and analyzed. A directed graph is constructed in which all existing dependencies between the concepts are explicitly noted as arcs. A dependency arc between two concepts Cx and Cy exists, if the concept Cx describes an effect which is a necessary condition which occurs in the formation of concept Cy. Figure 2 shows a graphical representation of the dependency graph that results from the analysis of the example in Figure 1. For example, the dependency of the concept "tolerance(#6,low))" on the concept "chucking_precision(high)" states that it is necessary to have a high chucking precision in order to produce surface #6 with a low tolerance. Note that all the concepts are always related to one operator and usually require certain constraints on them. Thereby, the dependencies between two concepts also express dependencies between two operators. In the example mentioned above, the cut-operation which creates the surface #6 is dependent on the chucking operator.

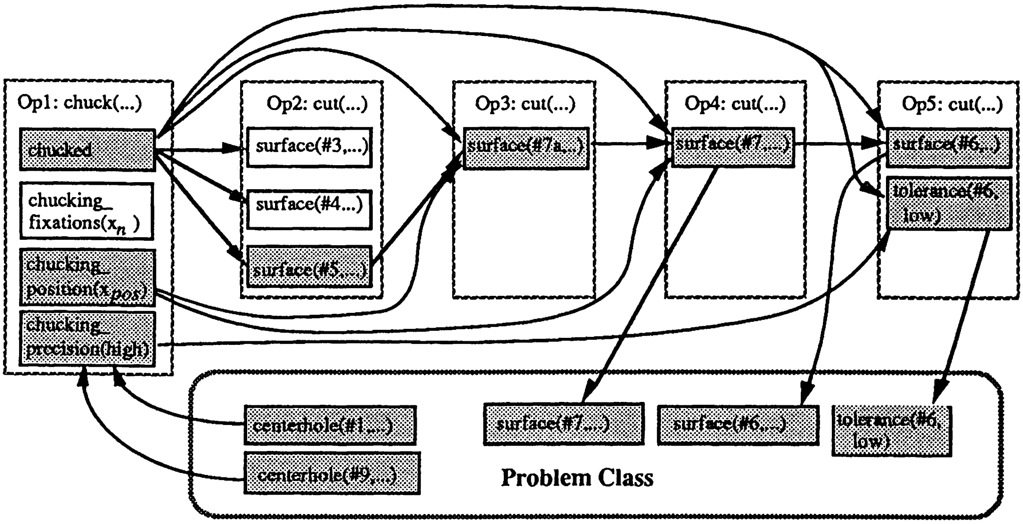


Figure 2: Dependency Graph for example case

For generating a skeletal plan, which is tailored to a definite problem class specified by important features of workpiece and mold, all concepts on which the class-relevant features are dependent, have to be identified. This is achieved by computing the least subgraph which contains all relevant features of the problem class, and in which all dependency predecessors of the concepts in the subgraph are themselves part of the subgraph. In Figure 2, the gray marked concepts, together with their links, form the subgraph, which results from a description of the classification of the examples in Figure 1. An overgeneralization that may have resulted from the independent treatment of the operators must be avoided in order to ensure that the operator classes which are to be generated for the skeletal plan can in fact be specialized independently for the solution of a new planning problem. Therefore, the concepts of the determined subgraph are unified along their dependency arcs, which yields one general concept of the plan for the whole problem class. For the example in Figure 1 a fragment of this concept is sketched as follows:

surface(#6,....) $\wedge$ surface(#7.....) $\wedge$ low_tolerance(#6) $\leftarrow$
operator(1,chuck(x_{tool1},x1,x2)) $\wedge$; *From concept C1: "chucked"*
operator(1,chuck(x_{tool2},x3,left)) $\wedge$; *From concept C2 : "chucking_position"*
operator(1,chuck(x_{tool3}, x4,x5)) $\wedge$; *From concept C3: "chucking_precision"*
requires_centerholes(x_{tool3}) $\wedge$; *From concept C3: "chucking_precision"*
centerhole(c1,x6,x7)) $\wedge$; *From concept C3: "chucking_precision"*
centerhole(c2,x8,x9)) $\wedge$; *From concept C3: "chucking_precision"*
c1 $\neq$ c2 $\wedge$; *From concept C3: "chucking_precision"*
operator(2,cut(...)) $\wedge$ ..... ; *From concept "surface(#5...)"*

3.4 Phase-IV: Normalization into Skeletal Plan Representation

This phase builds the skeletal plan in its final representation by identifying independently solvable sub-formulas from the concept of the plan which expresses only local constraints on one operator. By analyzing the occurrence of variables in the conditions of the plan concept, all conditions are separated into:

- one set $\mathfrak{R}_{Enable}$ which collects all conditions that only relate to features of the problem description,
- one set $\mathfrak{R}_{Opi}$ for each operator Op_i where the conditions only specify parameters which directly correspond to one operator.

The set of constraints $\mathfrak{R}_{Enable}$ formally describes the class of problems for which the skeletal plan can be used, and thus functions as an application condition for the skeletal plan. The skeletal plan itself is built of the sequence of constraints $\mathfrak{R}_{Opi}$, which exactly describe the required classes of operators. A further simplification of the constraint set is performed by the application of some rewrite reduction rules. Thereby, more operational descriptions of the operators classes are obtained.

For the example in Figure 1 the following skeletal plan with application conditions is generated.

1. Skeletal plan: operator(1,chuck(x_{tool}, x1,left)) $\wedge$ requires_centerholes(x_{tool})
operator(2,cut(...)) $\wedge$ ...
. . .
operator(5,cut(...)) $\wedge$ ...
2. Application condition: centerhole(c1,x3,x4)) $\wedge$ centerhole(c2,x5,x6)) $\wedge$ c1 $\neq$ c2 $\wedge$ ....

A prototype of the described method was implemented on a Apple-Macintosh-II computer using the LPA-PROLOG environment (Bergmann 1990). This prototype creates skeletal plans for cases like the one in Figure 1.

4. Discussion

The automatic knowledge acquisition approach presented in this paper makes use of the idea to automatically prepare large amounts of already formally available knowledge for further use in an expert system. Especially for real world planning tasks such as mechanical engineering, the reuse of manually optimized plans in a more general way becomes possible without involving a domain expert in a time-consuming knowledge acquisition process. Qualitatively high skeletal plans can be generated if the origin plans are qualitatively good. Because of the explanation of the goal achievement, the beneficial interactions between operators are discovered and can be maintained for the abstract solution.

Since a knowledge-intensive learning paradigm such as explanation-based learning is the core of this method, a large amount of knowledge has to be provided to enable its application. The requirements on the available domain theory are very high in the sense that a correct and tractable theory is needed which is complete enough to allow the simulation of the full plan. It seems hopeless to acquire such a theory automatically, even if inductive learning methods were applied. Therefore the skeletal plan generation procedure has to be integrated with other, non-automatic methods such as COKAM and CECoS and works well if the requirements mentioned on the theory can be fulfilled in the application domain at hand.

Another question is concerned with the usefulness of the skeletal plans that are automatically acquired by this procedure. A skeletal plan is useful if it provides an abstraction that reduces the computational complexity of a planning process (Korf, 1988), and if it can be applied to a large class of problems. Since complexity reduction and wide applicability are somehow competing properties for a single abstraction we are engaged to find a hierarchy of skeletal plans. Therefore the utility based on the generality of a skeletal plan should be judged with respect to the problem class, for which the skeletal plan is constructed. If automatically obtained skeletal plans are compared with those that were acquired manually with considerable effort, a major weakness of generalization procedure can be identified. The described procedure is able to construct skeletal plans by abstracting single operations of a plan as far as the domain theory contains abstract descriptions of operator effects. If such a sufficient operator model can be supplied, the automatically generated operations can compete in utility with those constructed manually. If only a shallow operator theory such as in the case of the STRIPS domain is provided, the resulting skeletal plan for the most specific problem classes is the same as a macro-operator composed of all operators in the plan (Fikes, Hart & Nilsson, 1972).

Another kind of abstraction that appeared important for planning could not be performed by the proposed skeletal plan generation procedure. It is unable to collapse sequences of concrete operations into one single abstract operation. The bounds of the operations are always transfered from the concrete plan to the skeletal plan. Knoblock's (1990) approach to operator abstraction shows exactly the same deficits while Mädler (1991) tries to find "eyes of a needle" in the state space to combine sequences of operations into one single abstraction.

Further research to improve this automatic knowledge acquisition should consequently deal with the problem of finding more appropriate abstractions, for example by changing the plan representation language. Since this seems to be a knowledge-intensive process that cannot be applied in a isolated fashion, the interactions between such automatic and manual knowledge acquisition methods must be further examined and developed.

Acknowledgments

I would like to thank Franz Schmalhofer for many helpful discussions and for significantly contributing to this paper.

This research was supported in part by grant ITW 8902 C4 from the BMFT (German Ministry for Science and Technology) and by grant Schm 648/1 from the DFG (German Science Foundation).

References

Bergmann, R. (1990). Generierung von Skelettplänen als Problem der Wissensakquisition. *Unpublished masters thesis*. Universität Kaiserslautern.

Bergmann, R., Bernardi, A., Klauck, C., Kühn, O., Legleitner, R., Schmalhofer, F., & Schmidt, G. (1990). Formulierung von Anforderungen zur Darstellung von Werkstücken und Spezifikation einer Makrorepräsenation. *Internes ARC-TEC Diskussionspapier Nr. 8*.

Bergmann, R. & Schmalhofer, F. (1991). CECoS: A case experience combination system for knowledge acquisition for expert systems. To appear in: *Behavior Research Methods, Instruments and Computers.*

DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning,* **1**, pp. 145-176.

Fikes, R.E., & Nilsson, N.J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence,* **2**, pp. 189-208.

Fikes, R.E., Hart, P.E., & Nilsson, N.J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence,* **3**, pp. 251-288.

Friedland, P.E., & Iwasaki, Y. (1985). The concept and implementation of skeletal plans. *Journal of Automated Reasoning,* **1**, pp. 161-208.

Knoblock, C. A. (1990). Learning abstraction hierarchies for problem solving. *Proceedings Eight National Conference on Artificial Intelligence,* **2**.

Korf, R.E. (1988). Optimal path-finding algorithms. In: *Search in Artificial Intelligence.* Kanal, L., Kumar, V. (eds.), Springer, NY.

Mädler, F. (1991). Problemzerlegung als optimalitätserhaltende Operatorabstraktion. In: GWAI-91, 15. Fachtagung für Künstliche Intelligenz. Th. Christaller (ed.). Springer, Berlin.

Mitchell, T.M., Keller, R.M., & Kedar-Cabelli, S.T. (1986). Explanation-based generalization: A unifying view. *Machine Learning,* **1**, pp. 47-80.

Musen, M.A., Fagan, L.M., Combs, D.M., & Shortliffe, E.H. (1987). Use of a domain model to drive an interactive knowledge editing tool. *International Journal of Man-Machine Studies,* **26**, 1, pp. 105-121.

Schmalhofer, F., Bergmann, B., Kühn ,O. & Schmidt, G. (1991a) Using integrated knowledge acquisition to prepare sophisticated expert plans for their re-use in novel situations. In: GWAI-91, 15. Fachtagung für Künstliche Intelligenz. Th. Christaller (ed.). Springer, Berlin.

Schmalhofer, F., Kühn, O., & Schmidt, G. (1991b). Integrated knowledge acquisition from text, previous solved cases and expert memories. *Applied Artificial Intelligence* , **5**, pp. 311-337.

Schmalhofer, F., & Thoben, J. (this volume). The model-based construction of a Case-Oriented Expert System. *Contemporary Knowledge Engineering and Cognition.* F. Schmalhofer, G. Strube & Th. Wetter (eds.).

Schmidt, G., & Schmalhofer, F. (1990). Case-oriented knowledge acquisition from text. In: *Current Trends in Knowledge Acquisition*, Wielinga, B., Boose, J., Gaines, B., Schreiber, G., van Someren, M. (eds.), IOS Press, May 1990, pp. 302-312.

Shortliffe, E.H., Scott, A.C. Bischoff, M.B., Campbell, A.B., Melle, W., & Jacobs, C.D. (1981). ONCOCIN: An expert system for oncology protocol management. *Proceedings of 7th International Joint Conference on Artificial Intelligence, Vancouver, Canada,* pp. 878-881.

Stefik, M. (1981). Planning with constraints (MOLGEN: part 1). *Artificial Intelligence* **16,** pp.111-139.

Tu, S.W., Kahn, M.G., Musen, M.A., Ferguson, J.C., Shortliffe, E.H., & Fagan, L.M. (1989). Episodic skeletal-plan refinement based on temporal data. *Communications of the ACM,* **32,** 12, pp. 1439-1455.

Knowledge Acquisition from Cases

Sonja Branskat
Institute for Applied Information Technology
German Research Center for Mathematics and Data Processing
PO Box 1240 5205 St. Augustin 1

Abstract. This paper presents a hypermedia, domain independent system supporting the acquisition, formalization, and representation of cases. The system assists a knowledge engineer in a step by step transformation of an informally represented case into a formally represented one. Each case consists of five components: context, task, solution trace, solution and evaluation of the solution. The knowledge engineer builds a formalization sequence of case descriptions. An attribute/value description is considered to be formal. The data structure is derived from Memory Organization Packages. The sequence itself documents the formalization process.

1. Introduction

Many techniques for knowledge elicitation are based on protocoling the expert when he works on a case (Neale, 1988). A knowledge engineer (KE) analyses the transcript and transforms it into a formal knowledge base. While the KE creates different knowledges bases (protocols on video- or tape-records, transcripts and models of expertise on paper and a knowledge base in a computer-based programming-language), only the formalized knowledge base on a computer is accessible by a user (expert, KE or customer). Due to the media break during the formalization the process itself is not transparent and cannot be re-examined. There is no access provided from formal knowledge elements to any underlying informal elements. As a consequence a user cannot look up the meaning or the context of the concepts that the interviewed expert had in mind (Karbach and Linster, 1990).

The system UFA, presented in this paper, promises to substantially reduce if not discard the media break. UFA, implemented in HyperCard, is designed to systematically support the formalization of cases. The support facilitates integrated documentation, transparency of the formalization process, and avoids any media break.

Further approaches using hypermedia for knowledge acquisition are described in (Gaines and Linster, 1990), (Linster and Gaines, 1990).

2. Representation of Cases in UFA

In UFA the KE represents cases in a formalization sequence analogous to the generalization and specialization hierarchy of the Memory Organization Packages (MOPs) (Schank, 1985). MOPs are widely used to represent cases in case-based reasoning systems. While Schank's norm-based MOP-hierarchy consists of formal cases in different generalization states, a sequence in UFA consists of different formalization states of the same case. In the formalization sequence successive descriptions of the same case

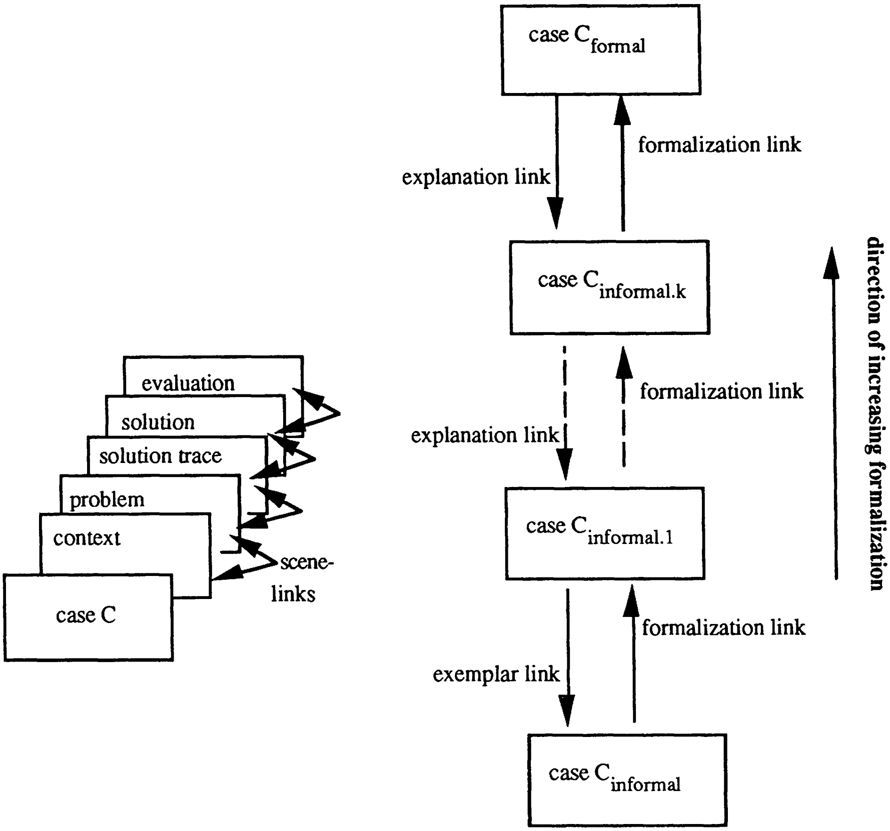


Figure 1: Case model and formalization sequence.

C: case $C_{informal}$, *case*$C_{informal.1}$, ..., *case* $C_{informal.k}$, ... , *case* C_{formal} (see Figure 1) have an increasing degree of formalization. This sequence documents the formalization process. The sequence starts with an informal case description and ends with a complete formal description.

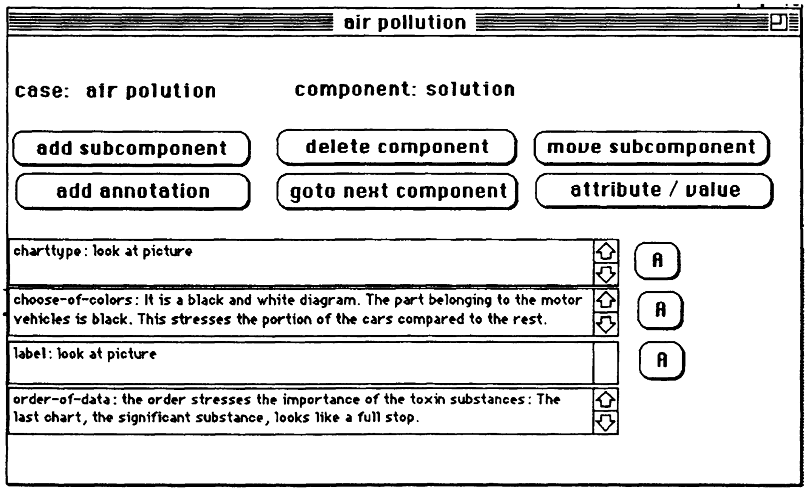
A case model is proposed, in which each case consists of five components: context, task, trace of the problem solving process (called *solution trace*), solution and evaluation of the solution as shown in Figure 1. Each case is represented in a MOP. Scene-links connect the components of a case. Formalization links connect a case description (e.g. case $C_{informal.k)}$ with its more formal case description (e.g. case $C_{formal.k+1}$). Explanation links connect a case description (e.g. case $C_{informal.k}$) with its next less formal description (e.g. case $C_{informal.k-1}$). Exemplar links connect the case description of the first formalization state (case $C_{informal.1}$) with the initial informal description (case $C_{informal}$).

3. Formalization of Cases in UFA: an Example

To demonstrate UFA an example is given in the domain of business graphics. In business graphics one visualizes quantitative relations between data with graphical techniques to obtain optimal information presentation. For the case *air pollution* the scanned graphic is a first informal description of the component *solution*. Four circular charts present the contribution of cars to air pollution differentiated into four different groups of chemicals.

In the first step the KE identifies the components: context, problem, solution trace, solution and evaluation and generates a new MOP in UFA (case $C_{informal}$) to describe the components. He describes the component *solution* with the scanned graphic as an informal description.

In the second step the KE generates a new MOP (case $C_{informal.1}$) and connects it with a formalization link to the initial case description (case $A_{informal}$). On the first formalization layer he differentiates the domain independent structural component *solution* into domain dependent structural subcomponents: the charttype that is used, the colors that are used and the labelling of the chart. Using the function *add subcomponent* (see Figure 2) he generates new fields to describe these subcomponents. Using the function *add annotation* (see Figure 2) he creates an annotation card for the selected subcomponent of the component *solution* and describes it with a picture and free text. With the function *go back* UFA returns from the annotation card to the component *solution*. Buttons labelled *A* (see Figure 2) access any annotation card for this subcomponent.



case: air polution

component: solution

charttype: look at picture

choose-of-colors: It is a black and white diagram. The part belonging to the motor vehicles is black. This stresses the portion of the cars compared to the rest.

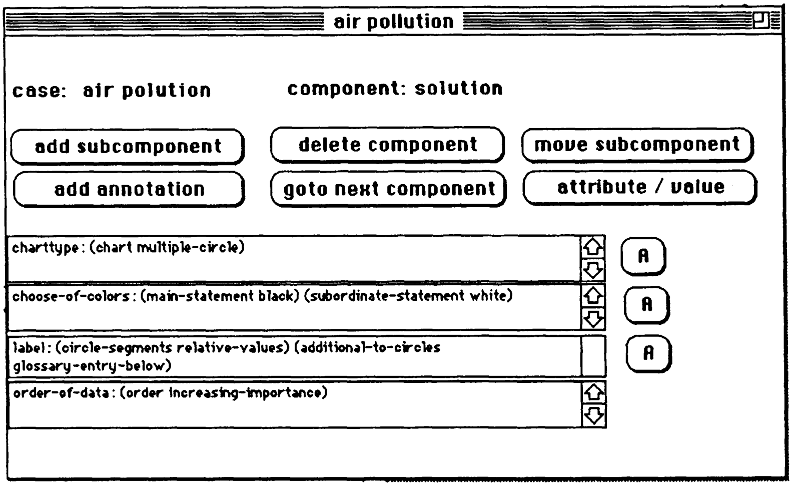
label: look at picture

order-of-data: the order stresses the importance of the toxin substances: The last chart, the significant substance, looks like a full stop.

Figure 2: The component solution of the case air pollution*: the formalization links and the textfields for the already generated subcomponents.*

At the outset of the third step the KE creates a new MOP (case $C_{informal.2}$) and connects it with an explanation link to its more informal version (case $C_{informal.1}$). He uses a formalization link to connect (case $C_{informal.1}$) to its more formal version (case $C_{informal.2}$). He decides to copy the subcomponents of the old description (case $C_{informal.1}$) to the components of the more formalized case description (case $C_{informal.2}$). The KE creates a new subcomponent of the component *solution*, in order to describe the solution in a more formal way by adding the order the data are presented. He describes the new structural component *order-of-data* directly on the solution card using the text field popping up with the creation of a new subcomponent.

A few formalization steps later the KE describes the subcomponent *charttype* with *circle-chart* and decides, that *multiple circle-chart* should be a value of the attribute *charttype.* He declares this in UFA using the function *att/value* (see Figure 2) and expanding the range of values of this attribute. The case is formalized if all components are described with attributes and values (see Figure 3). The formalization process is finished, as now all relevant knowledge enclosed in graphics and text has been transformed into a formal, computer accessible and retrievable notation.



case: air polution component: solution

charttype: (chart multiple-circle)

choose-of-colors: (main-statement black) (subordinate-statement white)

label: (circle-segments relative-values) (additional-to-circles glossary-entry-below)

order-of-data: (order increasing-importance)

Figure 3: The component solution of the case air pollution*: a complete formalized version.*

Evolving a linear sequence of case descriptions the KE uses further functions to develop a domain dependent model for cases. He can:

- explain the concepts and formalizations he has used in a hypermedial terminological dictionary,
- define terms of concepts as synonym and replace them by a term he chooses,
- define domain dependent subcomponents as attributes and define values for them; he can use functions to check, if no attribute is associated with a value that is not allowed,
- determine the order of all domain dependent characteristics of a component,
- get an overview over the degrees of formalization that the system provides.

4. Conclusion

UFA shows a way how to bridge the media break. It supports the KE by stepwise transforming a "natural" representation of cases into a formal MOP like representation using only one media. The formalization process is completely documented within UFA. UFA can easily be extended to handle video or tape-records. However, it is necessary to gain more experience using UFA to give a more detailed evaluation.

5. Acknowledgement

I am grateful to Marc Linster, Barbara Becker, Thomas Christaller and all the other members of the Artificial Intelligence Research Division of the German National Research Centre for Computer Science (GMD) and Brian Gaines for making valuable comments on a previous version of this article.

Literature

Karbach W., Linster M..; *Wissensakquisition für Expertensysteme;* Carl Hanser Verlag, Bonn (1990).

Linster, Marc and Gaines, Brian (1990)*Supporting Acquisition and Interpretation of Knowledge in a Hypermedia Environment,* Tech. Rept. 455, Arbeitspapiere der GMD, GMD, July 1990

Gaines, Brain and Linster, Marc (1990) Development of Second Generation Knowledge Acquisition Systems; Introduced to EKAW90.

Neale, I.M. (1988) First generation Expert Systems: a review of Knowledge Acquisition Methodologies; *The Knowledge engineer review 3* , pp. 105-146.

Schank, R.C.; *Dynamic memory: A theory of reminding and learning in computers and people;* Cambridge University Press (1985).

Transforming examples into cases

Peter Reimann & Thomas J. Schult

Psychologisches Institut der Universität
Niemensstr.10
D-7800 Freiburg

e-mail: reimann@cogsys.psychologie.uni-freiburg.dbp.de

Abstract: Our work is concerned with a central process in knowledge acquisition for case-based expert systems: understanding examples of expert's problem solving traces, in our case worked-out examples in physics textbooks. Based on evidence from psychological research, an active, expectation-driven strategy for example processing is developed. The strategy deals with the initial phases of learning, exploiting case knowledge before relying on general knowledge. We report on first attempts to realize this strategy as a cognitive computer model.

1. Introduction

Knowledge acquisition was and will for some time be the major impediment to the wide-spread application of knowledge-based systems in real-world domains. Several ways were suggested to automate this task using machine learning techniques. The traditional approach uses induction to abstract principles out of a large set of examples (e.g., Quinlan, 1983). Recently, it has been suggested to circumvent the knowledge acquisition bottleneck to a certain extent by equipping expert systems directly with one of the main aspects of expertise: experience in form of *cases*. Case-based expert systems do not need difficultly to acquire rules, whether they stem from interviews with an expert or from the application of machine learning techniques. Instead, they attempt to solve new problems by mapping to solutions of former problems and adapting the old solution to a new problem. First commercial expert system shells including CBR components are, for example, ART-IM and ReMind.

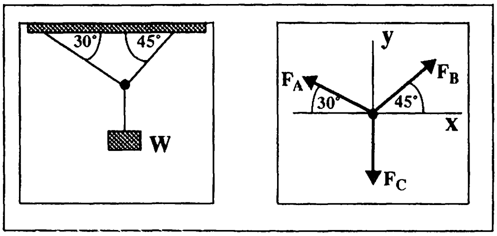
The power of a case-based expert system lies mainly in the amount and quality of the cases stored as well as in the indexing of the cases. The knowledge acquisition process for a CBR expert system amounts to provide the cases and fine-tune the indexing. This task is not necessarily easier than pressing an expert for the verbalization of rules. The complexity of this kind of knowledge acquisition is directly proportional to the structure of the cases one wants to store in case memory. If this structure is simple - for example, flat attribute-value lists - then case acquisition is almost trivial. If the structure is more complex - for example, if cases contain descriptions of causal relations - then case acquisition is more demanding and may require application of knowledge acquisition and machine learning techniques similar to those that have been developed for rule-based expert systems.

In any case, it is desirable to automate the case acquisition process as much as possible. This is a challenging problem if cases have a complex structure. For instance, Redmond (1989a) describes the difficulties involved in transforming records of experts' troubleshooting behavior into a case description that is useful for a case-based troubleshooting expert system. In our research, we study how humans acquire case knowledge not from observing an expert directly but from solution traces as they are provided by worked-out examples in textbooks. Insights into the process of transforming worked-out solution examples into cases is relevant for research in knowledge acquisition for case-based expert systems to the extent that (a) learning from experts and learning from solution examples impose similar problems (Redmond 1989b), or (b) that learning from solution examples is a component of the knowledge acquisition task.

2. Models of Example Elaboration Strategies

In order to shed light on the acquisition of cases from solution examples, we consider how humans tackle this learning task. We identify successful human strategies and model components of these strategies with computer simulations. The examples are solutions to mechanics problems as they appear in a textbook on college-level physics widely used in the US. Successful learning from these examples can best be understood as an active elaboration of the solutions steps, as an empirical study revealed (Chi, Bassok, Lewis, Reimann & Glaser, 1989).

Worked-out examples may be difficult to understand because they do not contain the necessary information to perform sensible generalizations. In particular, examples do often not contain the reasons for why a certain step in the solution is performed. Consider the example solution provided in Figure 1. In statement 6, it says that the forces shown in the force diagram are "all the forces acting on the body". Yet, the example contains no procedure that describes how this could be determined by the student. In the absence of rationales for problem solving decisions, it is hard to decide what the essential features of an example are and what the superficial ones are, i.e., those one can generalize over.



(1) The left figure shows an object of weight W hung by massless strings. (2) Consider the knot at the junction of the three strings to be "the body". (3) The body remains at rest under the action of the three forces shown in the right figure. (4) Suppose we are given the magnitude of one of the three forces. (5) How can we find the magnitude of the other forces? (6) F_A, F_B and F_C are all the forces acting on the body. (7) Since the body is unaccelerated, $F_A + F_B + F_C = 0$. (8) Choosing the x- and y-axes as shown, we can write this vector equation as three scalar equations: (9) $F_{Ax} + F_{Bx} = 0$, and (10) $F_{Ay} + F_{By} + F_{Cy} = 0$. (11) The third scalar equation for the z-axis is simply: (12) $F_{Az} = F_{Bz} = F_{Cz} = 0$ (...)

Figure 1: A worked-out example for a mechanics problem (Halliday & Resnick, 1985)

Even so this learning problem is typically one of novices (students) who want to acquire problem solving knowledge by studying a textbook, a knowledge engineer may encounter similar problems when observing an expert's problem solving behavior without being able or willing to interrupt and ask for the reasons for decisions taken by the expert, or when studying written materials such as textbooks to become more familiar with a domain. On the following pages, we will outline a computational model of how human problem solvers analyze worked-out examples so that they can use them effectively for problem solving by analogical transfer to similar problems. To a certain extent, this model can also be seen as a method for automatic knowledge acquisition from text. It should be mentioned that as these pages are written this model of apprenticeship learning is yet a conceptual one, only parts of which are in the stage of becoming implemented.

The general design of the strategy is based on insights into the process of text comprehension and understanding (e.g., Brown, Collins, & Harris, 1978) as well as on research concerning the dynamic character of human memory and its influence on problem solving and understanding (Schank, 1982). To account for the specific sort of text on which learning is based in our case, worked-out examples in mechanics, we further rely on observations made by Chi et al. (1989). There it was analyzed how students acquire problem solving knowledge concerning mechanics by studying worked-out examples. The study revealed important differences between successful and not so successful students, success measured in terms of correct solutions to problems. Successful students mentioned more often that they didn't understand a certain part of the worked-out example. Besides this difference in monitoring understanding of the example text, successful students also engaged in a series of activities to overcome their problems: They elaborated on the relations between a particular step in the example and the goals behind that step. They further attempted to come up with a specification of conditions that could explain why the operator under question was applied. Finally, they elaborated on the effects the application of an operator had beyond those mentioned in the example. The not so successful students displayed either none or considerably less of these elaborative inferences.

To capture the essential differences between successful and less successful learning from examples in form of computational models, we treat the process of example elaboration as a plan recognition task where the understanding system has to encode a given example solution in terms of problem solving goals, operators, and relations to domain concepts. Figure 2 shows the main components of the model.

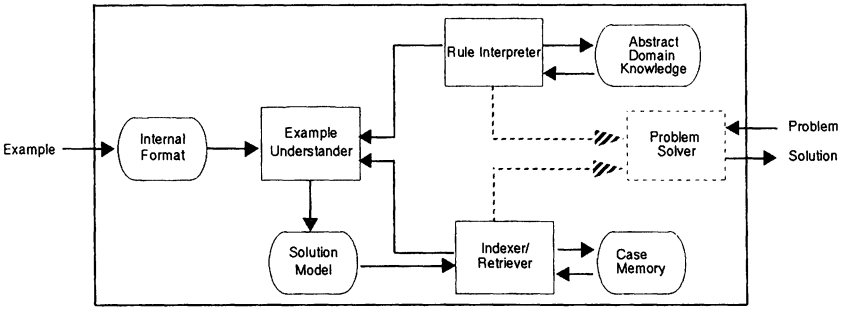


Figure 2: Model Components

It comprises modules for example comprehension and problem solving. The problem solving component demonstrates the competence of the system. It attempts to solve new problems building on knowledge mainly acquired by learning from examples. Both the problem solving module and the example understanding module are hybrid systems, combining a case-based approach with general but weak inference methods. The abstract domain knowledge comprises an object hierarchy for problems concerning particle dynamics and a rule-based problem solver for this domain.

Most important for the current discussion is the example understander. This component takes a worked-out example as input (in a propositional format) and generates a model of the example solution that is then stored into case memory. To construct the example solution model, the system calls up case knowledge about former examples and abstract domain knowledge. The general control decision is to rely on case knowledge first. Two strategies are contrasted: an active, expectation-driven one that leads to a hierarchical representation of an example and allows for problem solving by derivational analogy (Carbonell, 1986), versus a passive example processing strategy leading to a flat representation that enables the problem solver to use an example by transformations based on syntactical similarities. We want to illustrate the details of how these knowledge sources are used in the context of the active comprehension strategy.

3. Active Example Elaboration Strategy

In order to motivate our active elaboration strategy, we have to introduce the case structure that the elaboration strategy has to acquire from examples. We adopt the case model as described in Alterman (1988) because this case representation supports flexible analogical problem solving. If cases are represented in such a form and if corresponding retrieval and adaptation processes are defined, the case-based reasoner can deal with problems such as Steps-Out-Of-Order, Failing-Preconditions, Failing-Outcome, and Differing-Goals. In Figure 3, the main features of this case structure are depicted. The task of the elaboration strategy is to build a case representation that contains as many as possible of the nodes and links which are part of the fully developed case structure from a solution example . Many of these features have to be inferred by the example understander since they are not mentioned explicitely in the example text.

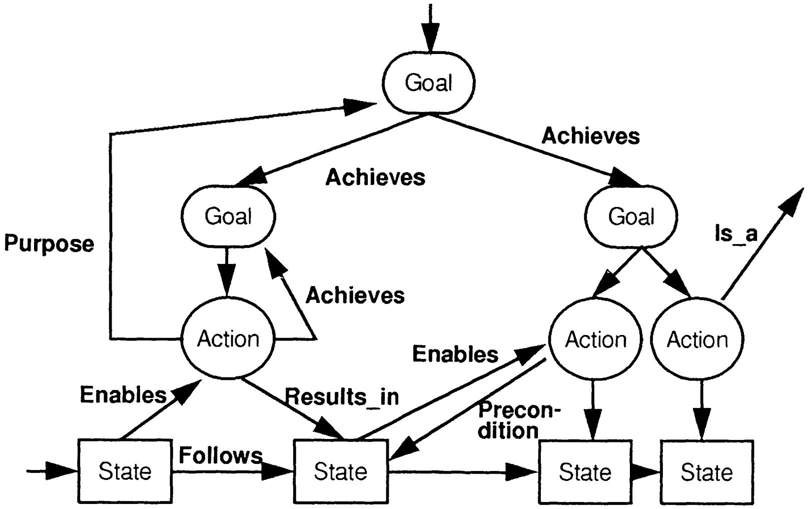


Figure 3: Elaborated case structure

Under the active strategy, the system monitors its understanding of the example by checking for the steps in the example solution whether they cohere with its knowledge about problem solving in the domain. It tests its understanding in an active, expectation-driven manner: The system predicts the next example step and compares its prediction with the step actually appearing in the example. Problems in understanding are identified by false or missing predictions. Having identified an understanding problem the system tries to explain the offending part of the example, thus resorting to a cognitively more demanding mode of example processing. This includes attempting to derive the action(s) observed in the example from its concrete and abstract knowledge about the domain as well as to come up with alternate solution plans. In summary, this strategy reflects many aspects of the performance of successful subjects from the study of Chi et al. (1989) and incorporates general principles of comprehension (e.g., Brown et al., 1978).

We are currently implementing this strategy in a system called AXE, the Active eXample Elaborator, using KEE and CommonLisp. AXE "reads" an example statement by statement and attempts to formulate expectations about the content of the next statement. A statement comprises propositions describing either (a) a goal selection, or (b) an operator selection (law, equation, inference rule), or (c) a problem solving state. Thus, AXE tries to predict (a) which goal will be worked on next, (b) which operator will be applied, or (c) what effects a goal or operator selection will have on the solution. These predictions - or expectations - are generated as follows.

Step 1 - Case-based expectation formation: After having read the first lines of an example where the problem is described in terms of objects and their relations, values given, and values desired, AXE performs a look-up in its case memory to find out whether it has encountered a similar example before. Case memory has the structure of a MOP-based memory (Schank, 1982) that indexes problem statements and solution steps as they occur in worked-out examples. If AXE finds a similar example, it will be used to derive expectations for the new one, e.g., to predict the next state of the current example. For instance, after statement (7) in the example in Figure 1, AXE might expect a rotation of the reference frame, if in a similar case the frame did not remain in the default orientation. Later this expectation would turn out to be inappropriate.

Step 2 - Rule-based expectation formation: If no similar example is found, AXE attempts to formulate a prediction by using its abstract domain knowledge in a forward-reasoning mode. AXE can at times predict the class of operators instead of a specific operator since its knowledge about domain operators is organized in a rule hierarchy. After Step 1 or 2, AXE does either have an expectation about the next state of the example solution or has failed to derive an expectation. Having no expectation is considered a failure and the system continues with Step 5, trying to repair this failure. If an expectation has been derived, it continues with Step 3.

Step 3 - Comparison: In the next step, the expectations about the example solution state are compared with the actual situation as described in the example statement. This comparison leads to an evaluation of the expectation. It may be (a) specific, (b) too general, or (c) insufficient, depending on whether a specific state can be predicted (a), or whether only a general state of affairs can be predicted (b), or whether the expectation covers only part of the effects resulting from the application of the operator (c). It is further determined whether the expectation is correct or wrong. For the example mentioned in step 1, a specific but wrong prediction would be identified.

Step 4 - Continuation: In the case of a specific and correct hypothesis, AXE goes on and tries to formulate expectations for the next example state. Before that, the correctly predicted operator instantiation is stored into case memory in terms of its differences to the operator instantiation retrieved.

Step 5 - Case-based repair: In all the other cases (no expectation, wrong, insufficient or too general expectation) AXE tries to learn from the expectation failure. The failure-driven repair is performed first in a case-based manner, relying on case knowledge before attempting to use more general, but search-intensive repair strategies. AXE looks into case memory to see whether it has made a similar expectation failure before and has encountered a similar example state before. This information is used to update the example solution model. The now successfully explained solution step is stored into case memory, and the system goes on predicting the next example statement.

Step 6 - Rule-based repair: If an expectation failure cannot be explained using case memory content, the system attempts to reconstruct the plan that might underlie the actions taken in the example part. This mode of example processing is particularly important during initial learning, since in this stage the learner possesses only a few cases and therefore cannot always recognize an example as the "same old story". During plan reconstruction, the system resorts to its abstract domain knowledge and tries to find an operator or a sequence of operators that could have generated the problem solving state as displayed in the current example statement. If AXE finds a sequence that connects the statement in the example with a currently active problem solving goal, it stores the sequence into case memory. In effect, the system can from now on substitute matching for search when encountering similar example parts.

For the example mentioned in steps 1 and 3, AXE would learn that the default frame must not necessarily be rotated in order to fulfill the goal to orientate the reference frame so that the resulting equations become simplified.

Step 7 - Copying the example solution step: If the plan reconstruction attempts fail, the system will store the specific representation of the example statement (an operator and its bindings) under the recent goal it was working on. Although AXE does not understand why this step was taken, it can at least copy the solution step when encountering a similar situation in a problem solving context.

An additional step - counterfactual reasoning: Whenever AXE succeeds either in predicting a part of the example solution (Step 1 or 2) or in recovering from an expectation failure (Step 5 or 6), it engages in a kind of counterfactual reasoning. After having established goal-action links by solution step recognition or reconstruction, starting from the goal it is asked: What other means exist to realize this goal? And why were they not chosen in the example solution? To come up with answers to these two questions, alternative solution paths are generated by working forward form the respective goal and applying domain rules. Doing that, AXE looks for problems with the alternative solution plans, problems that can be classified into categories such as failed constraint, excessive costs, failed result, or bad side effect (Collins, 1987). The fitst step of the alternative sequence with the explanation of why it encountered a problem is stored in case memory. By learning from counterfactual reasoning, the system acquires decision heuristics, thus enriching the necessary conditions of operators with further justifications. Evidence for reasoning similar to this counterfactual inference component was found in the most successful student of the Chi et al. (1989) study.

Having implemented the central components of an active example processing strategy, we hope to be able to demonstrate in the next step that a passive strategy can be modeled as a subset of the components making up the current learning model and that these differences in learning will lead to expected differences in analogical problem solving. That is, the active learner when working on a new problem should be able to retrieve from its case memory example solutions or parts of them based on features that go beyond literal similarity and should be able to adapt the example steps in a more flexible and correct manner than it is possible for the passive learner.

Acknowledgement: The research reported here is supported by a grant from the Deutsche Forschungsgemeinschaft to the first author (Re 814/1-1).

4. References

Alterman, R., Adaptive planning. *Cognitive Science, 11*, (1988), 393-421.

Brown, J.S., Collins, A., & Harris, G., Artificial intelligence and learning strategies. In *Learning Strategies*, H. O'Neill (Ed.), New York, Academic Press, 1978, pp. 107-139.

Carbonell, J.G., Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In *Machine Learning. An artificial intelligence approach, Vol. 2*, R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), Los Altos, CA, Kaufmann, 1986, pp. 371-392.

Chi, M.T.H., Bassok, M., Lewis, M., Reimann, P., & Glaser, R., Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science, 13*, (1989), 145-182.

Collins, G.C., Plan creation: Using strategies as blueprints. Ph.d. thesis, Yale University, 1987.

Halliday, D., & Resnick, R., Fundamentals of Physics. New York, John Wiley & Sons, 1985.

Quinlan, J.R., Learning efficient classification procedures and their application to chess end games. In *Machine Learning*, R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), Palo Alto, Tioga Press, 1983, pp. 463-482.

Redmond, M., Learning from others' experience: Creating cases from examples. In *Proceedings Case-Based Reasoning Workshop*, San Mateo, CA, Morgan Kaufmann, 1989a.

Redmond, M., Combining explanation types for learning by understanding instructional examples. In *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, Hillsdale, NJ., Erlbaum, 1989b.

Schank, R.C., Dynamic memory. A theory of reminding and learning in computers and people. New York, Cambridge University Press, 1982.

Case-Based Reasoning and Expert System Development[1]

Klaus-Dieter Althoff, Stefan Weß

University of Kaiserslautern
Dept. of Computer Science
P.O. Box 3049, D-6750 Kaiserslautern
Federal Republic of Germany
e-mail: *lastname*@informatik.uni-kl.de

Abstract. As a supplementation to other papers within this chapter on case-based approaches to Knowledge Engineering, we discuss some general aspects of case-based reasoning. We differentiate it from other case-using approaches and argue for the use of case-based reasoners within integrated knowledge engineering environments.

1. Introduction

Developing expert systems which can solve complex real world problems is still a difficult task. Therefore, knowledge engineering people need flexible methods and powerful tools which support them in doing this hard work. Within this paper we give a short introduction to such a flexible method, namely *case-based reasoning*, which might be one key issue in building, e.g., integrated knowledge engineering environments to offer the support needed. *Cases* are examples which have occurred in reality and consist of a problem description, a solution, and the underlying justification (derivation) for that solution. From a simplifying point of view, case-based reasoning means solving novel problems based on the adaptation of already known similar problem solutions. For being able to improve the problem solving capabilities of a system, cases must be memorized and integrated with already available empirical knowledge.

As concerned with problem solving, learning, and the acquisition of cases, case-based reasoning is within the focus of different fields of research, e.g. Cognitive Psychology, Machine Learning, and Knowledge Engineering. Apart from these strong commonalities, all those fields have their own view on the case-based reasoning approach. From a *Cognitive Psychology* point of view, it can be seen as a model of human problem solving. Within the *Machine Learning* community, case-based learning means an inductive learning method with a special kind of hypotheses generation. Verbatim examples are collected to learn (mainly) implicit concept descriptions which are then processed by the use of analogical reasoning. For the field of *Knowledge Engineering,* case-based reasoning implies a dynamic view on knowledge model-

[1] The work presented herein was partially supported by the Deutsche Forschungsgemeinschaft, SFB 314: "Artificial Intelligence - Knowledge-Based Systems", projects X6 and X9.

ing which overcomes the strict distinction between knowledge acquisition and application which, actually, is the underlying assumption of the model-based approach to knowledge acquisition. The automation of the knowledge acquisition and adaptation processes is the transition to learning. In the sense of automatic knowledge modeling, this has already been suggested by Morik [Mor87]. Thus, the case-based reasoning approach can be roughly characterized by the notions of learning ability, adaptation, and integration of knowledge acquisition and application. Case-based reasoning is a well-suited method for dealing with any kind of inhomogeneous solution spaces.

In this paper, we discuss some general aspects of case-based reasoning. Since case-based reasoning is a hot research topic many scientific contributions within this field have to be considered. Many different research communities have, at least partially, similar interests and/or methods, e.g. Machine Learning, Cognitive Psychology, Statistics, Pattern Recognition, Neural Networks, and Knowledge Engineering. In the next section we summarize the basic characteristics of case-based reasoning. Commonalities and important distinctions between case-based reasoning and other approaches are presented in section 3. Finally, we argue for the use of case-based reasoning within integrated knowledge engineering environments.

2. Case-Based Reasoning

Introduced to the community by Kolodner [Kol80, KSS85] and Schank [Sch82], the basic problem solving model of case-based reasoning grew out of several projects at Yale University. There exists a strong overlapping with research work done so far in the field of analogical reasoning. In its simplest form, case-based reasoning is similar to approaches known from statistics and pattern recognition (e.g. nearest neighbor classification) [cf. e.g. Tou81]. A general overview of case-based reasoning is given in [Sla91] and [RS89]. Important research goals concerning case-based reasoning from a Cognitive Psychology point of view are presented in [SJ90].

2.1. Cases

What is meant by the notion of ´case´ is one of the central questions in case-based reasoning. From a psychological point of view, cases are abstractions of events or processes which can be limited within space and time. Such knowledge is also known as *episodic knowledge* [cf. Str89]. Once the abstraction mapping is fixed, cases are often identified with their underlying events or processes.

For us, a case is an "example which has occurred in reality", i.e. a problem that occurred and has been solved by a certain kind of problem solving mechanism (human expert, expert system etc.). Therefore, the "observed" solution is empirically justified. Such cases are then mapped onto the respective case representation which, of course, reflects only a part of the "problem solving reality". In this sense, cases include implicit problem solving heuristics which can be interpreted with respect to different purposes.

For being able to describe cases in more detail, at least three different levels of abstraction should be differentiated [cf. Ric89 and And89]:

- a cognitive level (knowledge level)
- a representational level (algorithmic level)
- an implementational level

Within the context of diagnosing an engineering system, a case is the behavioral result of processes that have their origin on the cognitive level. On the representation level, this could be abstracted into a sequence of attribute-value pairs. Finally, on the implementation level a case is implemented using lists, structured objects, or a special subgraphs.

Since there is no general agreement concerning formal descriptions of cases, we give a definition which is very general but, nevertheless, sufficient for our purposes here [cf. also VC89].

Definition
A case is a triple (P,S,J) where P is a problem description, S the solution for the described problem, and J the justification of the solution. A case corresponds to a real event or process which can be limited within space and time.

Justifications are an explicit representation of the problem solving process. They can be more or less complex. The simplest kind of justification is an "empty" one resulting in a case-based reasoner which could only find solutions for problems it has "seen" before. For classification tasks this approach is often sufficient and known as *case-matching* (*classification/interpretive/precedent-based*) case-based reasoning [cf. Ham89a]. E.g., in a simple diagnostic situation a case might read as follows: the problem is described by means of the observed symptoms, the solution is the achieved diagnosis, and the justification is empty.

If more than transfer of unmodified solutions is needed, justifications, as an additional knowledge source, must be available. They can range from a simple problem solving trace to a complete explanation using some kind of deep reasoning model. Thus, a justification always includes a procedure or a theory which allows the interpretation of the (static) trace. This approach is often called *case-adaptation* (*problem solving*) case-based reasoning [cf. Ham89a]. For a diagnostic task, a justification could be the temporal order by which the symptoms have been ascertained, and for a planning task, a more or less complete dependency graph.

2.2. Problem Solving

We now describe the basic problem solving cycle which characterizes the case-based reasoning paradigm (retrieve, compare, adapt, repair, generalize; cf. [Syc91]). Cases are knowledge sources as well as rules or deep models and, therefore, have to be considered during expert system development, too. Once a case has been acquired, it is stored in a case library (case memory). During problem solving it might be retrieved from the memory if its problem description is similar (enough) to the actual problem at hand. If the case can be applied to the current problem its solution must be adapted based on some simple strategies (identical solution transfer, "patching", etc.), or on a more complex underlying domain theory using the available justifications. If the adaptation has been successful the completed case can be incorporated into the case memory. Thus, if the same problem occurs again it can be directly solved by retrieving this case and applying its stored solution. If the adaptation process has not been successful this case can be stored as a negative example to warn the problem solver not to go this direction if, e.g., the same problem has to be solved again. Additionally, if the system can find out the cause of the failure (explain the failure), it might be able to correct (repair) the wrong solution.

Both the adaptation and the repair processes require a problem solver of their own. Such problem solvers can use general strategies and a more or less complex domain theory to reach their respective goals. In the worst case, they must be as powerful as from-scratch problem solvers. Therefore, the integration of case-based reasoners into broader problem solving architectures is an important research goal (cf. section 4).

2.3. Similarity and Retrieval

Besides the underlying case representation, storage and retrieval of cases are of fundamental importance for the quality and efficiency of case-based problem solving mechanisms. Cases should be stored in memory such that fast retrieval of sufficiently similar cases is possible. They can be organized using a simple list, a data base, a discrimination [cf., e.g., Kol83a+b], or dependency graph. Similar cases can then be found by means of a similarity measure. This could be realized as an explicit mathematical function, as a pair of insert and retrieval procedures for the case memory, or as a combination of both.

2.4. Learning

A case-based reasoning system has to handle, at least, three different learning tasks. This encompasses learning from positive examples which might have been presented by an expert, learning from its own problem solving success, as well as from failure. Within the case-based reasoning community many different learning strategies have been used to handle these tasks. This includes rote learning for the integration of new cases or problem solving experiences into the case memory, explanation-based generalization to single out relevant features to be used as indices [RS89, Ham89b, BM88], generalization of implicit concept descriptions by means of partial matching (indexing, similarity functions) [Kol83a+b, PBH90, PG91], specialization of implicit concept descriptions (forgetting of cases according to certain selection criteria [AKA91], or competitive learning of feature relevances [AW91]), and generalization of feature values [Sal91].

3. Other Case-Using Approaches

Up to now, cases as a knowledge source for solving certain kinds of problems have been used in many different fields. We want to give an overview together with a rough classification of the respective approaches. This allows for an easy differentiation between them. Since many underlying notions of and connections between these approaches are not well understood up to now, we will not introduce a formal framework. Here, much work is still to be done.

Additionally, we do not want to differentiate between the case-based approach and approaches known as exemplar-based or instance-based.

One main aspect of case-based reasoning is that the underlying basic problem solving method is analogical reasoning. In general, analogical reasoning means transforming and extending existing domain knowledge to solve a similar task within another domain using similar methods. The known domain is often called *base* and the new one *target*. Fundamental characteristics of the analogical process are the mechanisms which determine the similarity of the tasks and transfer the methods and/or features from the base to the target domain, respectively. In principle, case-based reasoning can be seen as a special kind of analogical reasoning.

Historically, different research communities have concentrated on these inference mechanisms. For instance, Kolodner [cf. Kol89] points out that the focus within case-based reasoning has been mainly on case representation and retrieval, whereas within analogical reasoning the solution transfer has been treated in more depth. This is due to different basic assumptions concerning base and target domain. For case-based reasoning, they are normally identical, for analogical reasoning, on the other hand, it is mostly an essential feature to have different base and target domains [cf. Bur89, SD90]. For the rest of the paper we will not differentiate between these two approaches.

	Real-Life Connection	**Kind of Heuristic**	**Interpretation**
Rules	Abstract	Explicit	Single
Cases	Concrete	Implicit	Multiple

Fig.1 - Contrasting Cases and Rules

Case-based reasoning and inductive reasoning have in common that both reason from cases, and that the conclusions achieved are normally uncertain. Case-based, inductive, and explanation-based learning all learn from cases and can use preexisting domain knowledge for hypotheses generation. For the pure form of explanation-based learning the domain theory is assumed to be complete and correct. Here cases are used to focus the deductive process. Case-based reasoners mainly learn from the comparison of two cases (i.e. the learning procedure is fundamentally incremental) whereas inductive learners often compare several cases during one learning step. Some inductive learning systems are also able to learn incrementally. While most case-based reasoners store all the cases verbatim within an abstraction hierarchy (case memory) [cf. Sal91], most inductive learners forget all the cases which have been the basis for the generated hypotheses. Other machine learning approaches do both the learning of explicit concept descriptions, and the verbatim storing of cases [cf., e.g., SS88, Fis89]. Additionally, some case-based reasoning approaches try to improve their implicit concept descriptions by selectively removing cases from the case library [cf. KA88, AKA91].

Cognitive Level	**Protocol of a process** Diagnostic process of Friday, the 6th of August, to find out why the lamp in our living-room was not shining.

Representation Level	**Sequence of attribute-value pairs**
	lamp-12 <- off
	switch-3 <- on
	bulb-7 <- okay
	voltage <- not available
	defect <- short-circuit-6

Implementation Level	**List of the respective implementation language** ((lamp-12 off) (switch-3 on) (bulb-7 okay) (voltage none) (defect short-circuit-6))

Fig.2 - An Exemplary Case

From a Machine Learning point of view, case-based reasoning is not so well understood as, e.g., inductive learning. Up to now, there is no general agreement concerning the overall learning task which is addressed by case-based reasoning. Rather, there is a focus on defining and understanding particular mechanisms like reasoning by analogy and reasoning from cases. As a reason for this, Shavlik and Dietterich point out in [SD90] that research work in the field of case-based reasoning has been mainly motivated by concerns for cognitive plausibility rather than by a desire to construct practical systems.

Another reason is that most machine learning systems make a (strong) separation between learning and problem solving [cf. SD90]. Learning involves analyzing training examples or problem solving experiences to extract functions or rules, problem solving involves applying the learned functions or rules to solve new problems. In case-based reasoning, by contrast, problem solving is performed by directly inspecting the training examples (cases) and solving new problems by analogy with these past cases. This appears to be a major distinction of case-based reasoning and other machine learning approaches. However, there are also strong similarities between case-based problem solving and the well-known rule-based approach, because often it is not possible to differentiate between cases and rules (including their processing) on the levels of representation and implementation. Therefore, we suggest to define on a cognitive level what should be the difference between cases and rules. This allows some simple classifications which, as we hope, are helpful to answer some basic questions.

Cognitive Level	**Rule of Thumb** If you turn on a lamp and it does not shine, probably the bulb is defect.

Representation Level	**Sequence of attribute-value pairs**
	lamp <- off
	switch <- on
	defect <- bulb
	probability <- high

Implementation Level	**List of the respective implementation language** ((lamp off) (switch on) (defect bulb) (probability high))

Fig.3 - An Exemplary Rule

A production rule is a well-known knowledge representation scheme and most implemented systems within the Artificial Intelligence community have used it. We will give a very general definition of what a rule (of thumb) is, because we need it for contrasting purposes only. In section two, cases have been defined as episodic knowledge which consists of a problem description, a solution, and a justification for that solution. Normally, rules do not appear to be episodic knowledge but, rather, have been extracted from such knowledge, i.e. rules are more general than cases. Thus, a rule does not necessarily have a direct correspondence to one specific event, but is the result of a generalization process based on a number of different events.

Definition
A rule is a pair (C,A) where A is an action and C a condition which must be fulfilled to do action A.

Compared to the definition of a case, there is a correspondence between C and A, on the one hand, and problem description P and solution S, on the other hand. From another point of view, a rule could be described as an explicit kind of problem solving heuristic which can be contrasted by the more implicit heuristics being included in a case. Thus, the intended use of a rule (normally) is clear whereas a case can be applied in many different ways to solve similar problems. The reason for this is that a case includes a justification which can be interpreted with respect to a current purpose whereas rules (normally) have lost their justification. All these aspects are summarized in figure 1.

Though cases and rules differ concerning their complexity on the cognitive level this is not necessarily reflected on the representation and implementation levels. Therefore, figures 2 and 3 present an exemplary case as well as an exemplary rule which, in principle, differ on the cognitive level only.

Of course, case representations are often much more complex (cf., e.g., [Ber91]) and, additionally, other representational and implementational descriptions would have been possible.

Based on the above definitions, figure 4 gives a rough classification of methods which use cases and/or rules. Apart from the differentiation between cases and rules, we think that the distinction of exact and partial matching is of importance as well. An underlying assumption is that the analogy-based approach applies reasoning between different domains and, therefore, needs more general knowledge than it is offered by cases. For instance, the approach Michalski describes in his paper on two-tiered concept meaning [Mic89] would be classified as an analogy-based (matching) approach.

	Exact Matching	**Partial Matching**
Rules	Standard Rule-Based Approach	Analogy-Based Approach
Cases	Standard Data Base Approach	Case-Based Approach

Fig.4 - Matching of Cases versus Matching of Rules

Using the table given in figure 4, an inductive learning system could be classified as a standard rule-based or analogy-based approach (we do not want to differentiate between the processing of decision trees and rules here). Additionally, approaches known as instance- or exemplar-based as well as those known from statistics, pattern recognition, or neural networks would be classified as case-based approaches.

The above classification can be refined by differentiating between two kinds of partial matching, namely matching based on generalized indices (as it is used in most case memories [cf. Sch82, Kol83a+b, RS89]) and graded matching based on similarity measures [cf. SW88, AKA91, AW91]. While the motivation for the indexing approach is more oriented to cognitive psychology, the second one has its roots in mathematics/statistics. It applies to both approaches that one part of important information is represented explicitly, and another part not (cf. Fig. 5). Thus, their transparency and understandability cannot be evaluated independent from the used application.

	Similarity of Cases	Computation of Similarity
Case Memory	Explicit Neighbors are similar	Implicit By insert and retrieval procedures
Similarity Function	Implicit By computed value	Explicit By used Function

Fig.5 - Similarity: Computation versus Representation

In the past, many statistical and pattern recognition procedures have been developed which use similarity functions, as well as instance- and exemplar-based (case-based) reasoning approaches, but only apply pure syntactical methods for clustering or classification tasks. For a closer inspection of the relations between similarity, uncertainty, and case-based reasoning cf. [RW91].

4. Conclusions

Case-based reasoning represents a specific method for solving a certain class of problems, especially for the treatment of inhomogeneous solution spaces. Within such solution spaces, cases correspond to homogeneous (i.e. "small" changes of the problem descriptions result in "small" changes of the solutions/justifications) subspaces.

Case-based reasoning is a well-suited approach if cases are an important knowledge source within the underlying domain, and the available experts reason from cases (even a formal discipline as mathematics uses case-based reasoning, e.g. to find a certain proof [Ker89]). In addition, many domains are "case-based" in their overall structure, e.g. law, medicine, economy. Within these domains often a lot of "softcases" exist which can be easily adapted to solve novel problems. On the other hand, case-based reasoning is not well-suited in domains mainly consisting of "hardcases" (cases which can only be treated by heavily using common sense knowledge, or a huge amount of domain knowledge).

Partly in response to this problem, it is now widely recognized that a case-based reasoner can "play" different "roles" (the added lists of implemented systems are not intended to be complete, rather they represent an exemplary selection and classification) within a knowledge engineering environment:

- *Case-based reasoning can be used as a stand-alone problem solver* (no cooperation, e.g. CYRUS [Kol80], MEDIATOR [Sim85, Kol89], PROTOS [Bar89, PBH90], CASEY [Kot88], CHEF [Ham89b], PATDEX [AdM+89, Weß91, AW91])
- *Case-based reasoning can be combined with several other separate problem solvers* (input-output cooperation, e.g. GREBE [BP91], JULIA [HK91])
- *Case-based reasoning can be one among several cooperating completely integrated problem solvers* (cooperation at all levels of problem solving, e.g. PRODIGY (?) [VC91a,b], CABARET (?) [RBD+91], CREEK (?) [Aam90,91], D3 (?) [PG91, Pup90], MOLTKE (?) [AMR90, AW91, Alt91])

The first role reflects the early phase of case-based reasoning research where a lot of stand-alone systems have been implemented. Those systems cannot meet all the requirements which normally are posed by real world applications. For overcoming these shortcomings, actually the combination with other problem solving mechanisms (reasoning from rules, constraints, deep models etc.) is a hot research topic ("mixed paradigm reasoning", cf. [RSk89]). Up to now, such combinations are normally restricted to cooperations in an input-output manner. A deeper integration is an important research goal of many groups but, currently, no completely integrated systems are available. All the systems within the third list are only examples which try to achieve this goal (and, therefore, are (question-) marked). Thus, Knowledge Engineering researchers are asked to develop integrated architectures which make use of case-based reasoning.

A first suggestion for the integration of case-based reasoning and model-based knowledge acquisition is given in [JS91], whereas an overview of the integration of case-based, model-based, and compiled knowledge is given in [SZP90]. Schmalhofer et al. make a suggestion concerning the use of cases within an integrated knowledge acquisition process for the preparation of expert plans which can be reused in novel situations [SBK+91]. The MOBAL system is an interesting example for the integration of manual and automatic knowledge acquisition methods [Mor90]. In [dlO91] de la Ossa presents an approach for the automatic adaptation of a given diagnostic knowledge base with respect to changes in the physical system which is to be diagnosed. A case-based approach to theory revision using self-questions and experiments has been suggested by [Oeh91].

We mentioned above that, from a Machine Learning point of view, it is difficult to classify case-based reasoning, because its learning task is not well-defined. Shavlik and Dietterich [SD90] argue that the reason for this has been the motivation of case-based reasoning by concerns for cognitive plausibility rather than by a desire to construct practical systems. However, from a Knowledge Engineering point of view, case-based reasoning has some important advantages over standard Machine Learning approaches, namely, apart from a strong focus on cognitive plausibility, the overcoming of the separation of learning and problem solving.

5. Acknowledgement

Thanks go to especially to Prof. Dr. Michael M. Richter, and our research group at Kaiserslautern, for many very engaged discussions which have been very helpful for our work on case-based reasoning. Alvaro de la Ossa, Dietmar Janetzko, and Franz Schmalhofer have given helpful comments to earlier versions of this paper. Additional insights have come from discussions with Dieter Fensel, Katharina Morik, Stefan Wrobel, and Angi Voss.

6. References

[Aam90] Aamodt A. A Computational Model of Knowledge-Intensive Learning and Problem Solving. In: [WBG+90], pp 1-20

[Aam91] Aamodt A. A Knowledge-Intensive, Integrated Approach to Problem Solving and Sustained Learning. *PhD. Thesis*, University of Trondheim, 1991

[AdM+89] Althoff K-D, De la Ossa A, Maurer F, Stadler M, Weß S: Adaptive Learning in the Domain of Technical Diagnosis. *Proc. Workshop on Adaptive Learning*, FAW Ulm, 1989

[Aha91] Aha DW. Case-Based Learning Algorithms. In: [Bar91], pp 147-158

[AKA91] Aha DW, Kibler D, Albert MK. Instance-Based Learning Algorithms. *Machine Learning , 6*, pp 37-66, 1991

[Alt91] Althoff K-D. *Eine fallbasierte Lernkomponente als integrierter Bestandteil der MOLTKE-Werkbank zur Diagnose technischer Systeme.* Dissertation, University of Kaiserslautern (forthcoming)

[AMR90] Althoff K-D, Maurer F, Rehbold R. Multiple Knowledge Acquisition Strategies in MOLTKE. In: [WBG+90], pp 21-40

[And89] Anderson JR. A Theory of the Origins of Human Knowledge. *Artificial Intelligence - Special Volume on Machine Learning -*, 40, pp 313-352, 1989

[AW91] Althoff K-D, Weß S. Case-Based Knowledge Acquisition, Learning and Problem Solving for Diagnostic Real World Tasks. *Proc. EKAW-91*, 1991

[Bar89] Bareiss R. *Exemplar-Based Knowledge Acquisition.* Academic Press London, 1989

[Bar91] Bareiss R (ed). *Proc. 3rd DARPA Workshop on Case-Based Reasoning*, 1991

[Ber91] Bergmann R. Knowledge acquisition by generating skeletal plans from real world cases. In this volume

[BM88] Barletta R, Mark W. Explanation-Based Indexing of Cases. *Proc. AAAI-88*, 1988

[BP91] Branting LK, Porter BW. Rules and Precedents as Complementary Warrants. *Proc AAAI-91*, pp 3-9

[Bur89] Burstein MH. Analogy versus Case-Based Reasoning. In: [Ham89a], pp 133-136

[dlO91] de la Ossa A. Integrating Strategic Knowledge Acquisition and Reasoning About Change for Knowledge Adaptation. *Proc. KAW-91*, 1991

[Fis89] Fisher D. Noise-tolerant conceptual clustering. *Proc. IJCAI-89*, pp 825-830. Morgan Kaufmann, 1989

[Ham89a] Hammond K (ed). Proc. of the 2nd DARPA Workshop on Case-Based Reasoning. Holliday Inn, Pensacola Beach: Morgan Kaufmann, 1989

[Ham89b] Hammond K. *Case-Based Planning*. Academic Press London, 1989

[HK91] Hinrichs TR, Kolodner JL. The Roles of Adaptation in Case-Based Design. In: [Bar91], pp 121-132

[JS91] Janetzko D, Strube G. Case-based Reasoning and Model-based Knowledge Acquisition. In this volume

[Ker89] Kerber M. Some Aspects of Analogy in Mathematical Reasoning. *SEKI-Report SR-89-12*, University of Kaiserslautern, 1989

[KA88] Kibler D, Aha DW. Learning Representative Exemplars of Concepts: An Initial Case Study. *Proc. Fifth International Workshop on Machine Learning*, Morgan Kaufmann, 1988

[Kol80] Kolodner JL. Retrieval and organizational strategies in conceptual memory: A computer model. *Ph.D. Thesis*, Yale University, 1980

[Kol83a] Kolodner JL. Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, 7, pp 243-280, 1983

[Kol83b] Kolodner JL. Reconstructive Memory: A Computer Model. *Cognitive Science*, 7, pp 281-328, 1983

[Kol84] Kolodner JL. *Retrieval and organizational strategies in conceptual memory.* Hillsdale, NJ: Lawrence Erlbaum Associates, 1984

[Kol88] Kolodner JL (ed). *Proc. of a DARPA Workshop on Case-Based Reasoning.* Morgan Kaufmann Palo Alto, 1988

[Kol89] Kolodner JL. The Mediator: Analysis of an Early Case-Based Problem Solver. *Cognitive Science, 13*, pp 507-549, 1989

[Kot88] Koton P. Reasoning about evidence in causal explanations. *Proc. AAAI-88*, 1988, pp 256-261

[KSS85] Kolodner JL, Simpson RL, Sycara KP. A process model of case-based reasoning in problem solving *Proc. IJCAI-85*, pp 284-290. Los Angeles, CA: Morgan Kaufmann, 1985

[Mic89] Michalski RS. Concept meaning, matching and cohesiveness. In: [VO89]

[Mor87] Morik K. Sloppy Modeling. In: [Mor89], pp 107-134, 1987

[Mor89] Morik K (ed). *Knowledge Representation and Organization in Machine Learning.* Springer Berlin Heidelberg New York, 1989

[Mor90] Morik K. Integrating Manual and Automatic Knowledge Acquisition - BLIP. In: McGraw & Westphal (eds). *Readings in Knowledge Acquisition - Current Practices and Trends*, pp 213-232, Ellis Horwood, 1990

[Oeh91] Oehlmann R. Case-Based Theory Revision: Learning from Self-Questions and Experiments. Talk given at the University of Kaiserslautern, August 1991

[PBH90] Porter BW, Bareiss R, Holte RC. Concept Learning and Heuristic Classification in Weak-Theory Domains. *Artificial Intelligence, 45*, 1990

[PG91] Puppe F, Goos K. Improving Case Based Classification with Expert Knowledge. *Proc. GWAI-91*, Springer, 1991

[Pup90] Puppe F. *Problemlösungsmethoden in Expertensystemen.* Springer Verlag, 1990

[RBD+91] Rissland EL, Basu C, Daniels JL, McCarthy J, Rubinstein ZB, Skalag DB. A Blackboard-Based Architecture for Case-Based Reasoning: An Initial Report. In: [Bar91], pp 77-92

[Ric89] Richter MM. *Prinzipien der Künstlichen Intelligenz.* Teubner Verlag, 1989

[RS89] Riesbeck CK, Schank RC. *Inside Case-Based Reasoning.* Lawrence Erlbaum Associates, 1989

[RSk91] Rissland EL, Skalag DB. Combining Case-Based and Rule-Based Reasoning: A Heuristic Approach. *Proc IJCAI-89*, pp 524-530, 1989

[RW91] Richter MM, Weß S. Similarity, Uncertainty and Case-Based Reasoning in PATDEX. *Festschrift for Woody Bledsoe*, Kluwer Academic Publishers, 1991

[Sal91] Salzberg S. A Nearest Hyperrectangle Learning Method. *Machine Learning, 6*, pp 251-276, 1991

[SBK+91] Schmalhofer F, Bergmann R, Kühn O, Schmidt G. Using Integrated Knowledge Acquisition to Prepare Sophisticated Expert Plans for Their Re-Use in Novel Situations. *Proc. GWAI-91*, 1991

[Sch82] Schank RC. *Dynamic Memory: A Theory Of Learning in Computers and People.* Cambridge, UK: Cambridge University Press, 1982

[SD90] Shavlik JW, Dietterich TG (eds). *Readings in Machine Learning.* San Mateo: Morgan Kaufmann, 1990

[Sim85] Simpson RL. A Computer Model of Case-Based Reasoning in Problem Solving. *Ph.D. Thesis*, Techn. Rep. GIT-ICS/85/18, Georgia Inst. of Technology, 1985

[SS88] Sharma S, Sleeman D. REFINER: A Case-Based Differential Diagnosis Aide for Knowledge Acquisition and Knowledge Refinement. *Proc. EWSL-88*, pp 201-210, 1988

[Sla91] Slade S. Case-Based Reasoning: A Research Paradigm. *AI Magazine* Spring 1991

[Str89] Strube G. Episodisches Wissen. *Arbeitspapiere der GMD*, 385, pp 10-26, 1989

[SW88] Stanfill C, Waltz D. The memory based reasoning paradigm. In: [Kol88], pp 414-424

[SJ90] Strube G, Janetzko D. Episodisches Wissen und fallbasiertes Schließen: Aufgaben für die Wissensdiagnostik und die Wissenspsychologie. *Schweizerische Zeitschrift für Psychologie*, 49 (4), pp 211-221, 1990

[Syc91] Sycara KP. Case-Based Reasoning. *Overview of an course in case-based reasoning at the European Summer School on Machine Learning*, 1991

[SZP90] van Someren MW, Zheng LL, Post W. Cases, Models or Compiled Knowledge: a Comparative Analysis and Proposed Integration. In: [WBG+90], pp 339-355

[Tou81] Tou JT. Application of pattern recognition to knowledge system design and diagnostic inference. *Pattern Recognition - Theory and Application.* Reidel D Publishing, 1981

[VC89] Veloso M, Carbonell JG. Learning Analogies by Analogy - The Closed Loop of Memory Organization and Problem Solving. In: [Ham89a]

[VC91a] Veloso M, Carbonell JG. Learning by Analogical Replay in PRODIGY: First Results. In: *Proc. EWSL-1991*, pp 375-390

[VC91b] Veloso M, Carbonell JG. Variable-Precision Case Retrieval in Analogical Problem Solving. In: [Bar91]

[VO89] Vosniadou S, Ortony A (eds). *Similarity and Analogical Reasoning.* Cambridge University Press, 1989

[WBG+90] Wielinga BJ, Boose J, Gaines B et al. *Current Trends in Knowledge Acquisition* (Proc. EKAW-90). IOS Press Amsterdam, 1990

[Weß91] Weß S. PATDEX/2: ein System zum adaptiven, fallfokussierenden Lernen in technischen Diagnosesituationen. SEKI Working Paper SWP-91-01, University of Kaiserslautern, 1991

Part 3:

Cognitive Adequacy of Expert Systems

The Role of Cognitive Science in Knowledge Engineering

Gerhard Strube
Department of Cognitive Science
Institute of Computer Science and Social Research
Albert Ludwig University, Freiburg
Friedrichstr. 50, D-7800 Freiburg i.Br., Germany
strube@cognition.iig.uni-freiburg.de

Abstract. It is argued that knowledge engineering should take a cognitive stance, i.e. it should aim for cognitively adequate systems. The notion of cognitive adequacy is unfolded from an idealized, absolutely strong meaning (i.e., a complete model of a human expert) down to the very weak notion of conforming to recognized ergonomic standards. Various ways are proposed to enhance cognitive adequacy in the model-based framework of knowledge engineering. Finally, the relevance of these concepts for expert system application is discussed.

Knowledge engineering, as the name says, is an engineering science. As such, the objective for knowledge engineering is to develop techniques to elicit knowledge from experts, or to acquire knowledge from texts, cases, and other sources (automatically, or through the mediating work of a knowledge engineer), to organize it and thus render domain-specific knowledge ready for knowledge-based systems, to validate the knowledge-base, and to maintain its usability and integrity over the years.

This task is certainly demanding enough to stifle ideas that would lead to further degrees of complexity. To insist that knowledge engineering should aim for a (psychologically valid) *cognitive model* of expert knowledge therefore seems an altogether unsane recommendation. Cognitive modeling should be left to psychology, and to cognitive science in general, unless - well, unless it could be shown to significantly enhance the quality of knowledge engineering with respect to its own genuine objective. Costs have to be compared, of course, to the return on investments that can be expected. Moreover, the question of the feasibility of cognitive modeling, by itself and in the context of knowledge engineering, has to be adressed. Loosely spoken, why bother with cognitive modeling?

I believe that there are three good reasons to give knowledge engineering a 'cognitive' orientation, namely, because it provides knowledge-based systems with

□ enhanced validity,
□ added flexibility and stability, and
□ better security and ease of use.

Validity. It is important to recognize that validity refers to both declarative and procedural knowledge, i.e., to the facts and rules that comprise the knowledge base as well as to the kind of reasoning employed in the inference component of the system. To my mind, knowledge engineering has concentrated too much on the content of knowledge, leaving the reasoning process to the programmers. (Take conflict resolution in production-rule systems as an example. Because it is a technical problem, it has received much attention in computer science. But no one, to my knowledge, has ever demonstrated that kind of conflict, let alone the resolution strategies, in human experts.) On the other hand, we know far too little about the kind of reasoning employed by experts. Still, we have evidence for mental simulation (Stigler, 1984), evidence of experts running scenarios in their heads (Ceci & Liker, 1986). I believe that knowledge engineering must try to understand which ways of reasoning are employed by experts in the field, and when. Hybrid systems may encompass different problem solvers as well as different representations of domain knowledge, and the question is which kind of reasoning is to be applied at a given time (Janetzko & Strube, this volume).

Therefore, validity refers to the content and representational format of domain-specific knowledge, to the strategies of reasoning employed, and to the conditions of use for both. In today's reality, however, knowledge bases are often confined to only a narrow facet of domain-relevant knowledge. The usual perspective taken by the knowledge engineer, which is shaped by the tools available, as well as by the means provided for knowledge representation, is prone to ignore relevant areas of expert knowledge altogether. Conceptual models and a cognitive modeling approach, on the other hand, ensure an initial broadness that, although it must be narrowed down eventually, helps to think about which areas might be left out, and which are indispensable. Even where relevant aspects have to be omitted, the cognitive modeling approach ensures that this will not go unnoticed. In short, the cognitive stance in knowledge engineering is bound to yield systems that are more valid with respect to real, i.e., human expert knowledge, and whose 'blind zones' are known and can be taken into account when putting the system to use.

Flexibility & stability. Human experts command a variety of ways to reason from a given set of facts, rules, and the like. The flexibility gained through selective application of different reasoning strategies, which in turn utilize different parts of knowledge - e.g., general rules, or specific cases - is one of the hallmarks of human cognition and perhaps the most important single cause of our success in thinking and problem solving. It follows that the cognitive approach to knowledge engineering must strive for variety in reasoning, and must likewise try to embody domain-specific knowledge in multiple representation formats in the knowledge base. This is certainly demanding, and costly as well. But it pays by providing a means to the solution of a problem where simple, single-minded approaches get stuck. Modern expert system technology aims for enrichment of systems through 'deep' models of parts of the domain (e.g., models of apparatus), and through incorporation of examples or libraries of cases already solved.

Expert system technology, at this point, is more cognitive than knowledge engineering. Take CBR (case-based reasoning) as an example: There is already a considerable literature on representation of cases, and on reasoning from cases (e.g., DARPA, 1988; Riesbeck & Schank, 1989), but we lack techniques for assessing and structuring episodic knowledge. We even lack systematic evaluation of well-known methods like protocol analysis when applied to episodic knowledge.[1]

[1] To name but one of the cognitive problems that are involved, remembering a case results in a verbal protocol that looks like a protocol from 'thinking aloud' during solving an actual problem. But memory may act as a filter, or worse, may give rise to reconstruction, thus mixing general knowledge with specific recall. Therefore, a memory protocol may provide even more indirect evidence as the thinking-aloud technique (Ericsson & Simon, 1984).

In addition to the flexibility gained through enrichment by CBR and the like, enhanced stability can be the result. (It need not be so, because added complexity may bring along problems of coordination, arbitration, etc.) Multiple knowledge enables not only us humans, but any system to cross-check results, which amounts to an evalutation of the solution proposed, and makes the system more robust and more dependable.

Ease of use. A system that models the reasoning of real experts provides a sound base for giving explanations to its user, explanations that are both correct and understandable. By contrast, it is difficult to see how a system whose reasoning is utterly un-human might arrive at explanations that fulfil both requirements. (It is, however, easy to conceive that a system of that kind could give 'intuitive' explanations that are fictitious, or correct ones that are cryptic.) Researchers agree that giving adequate explanations is perhaps the crucial feature of a good expert system[2]. A cognitive approach is necessary, at least to some degree, to make the explanation component of a system useful. Of course, explanation makes further requirements, like natural language interaction (which in turn gains much from a cognitive approach).

Although I would agree that consistency is the key feature to user-friendliness in knowledge-based systems, I feel that cognitive adequacy brings us a step further. The added security and ease of use that result from a cognitively adequate explanation facility are well worth the effort.

It would be wrong to conclude that the cognitive stance in knowledge engineering is all pro's without the con's. Here are some cautions against:

□ Cognitive modeling is an extremely expensive endeavour, costly in time, and hence, money. The gains must be substantial indeed to warrant the effort. Yet my opinion is that there is much to be gained: theoretically, because a cognitive model helps us to understand the expert, and human expertise in general, and practically, because a valid model of expertise is bound to score better in solving real-world problems.

[2] 'Explanatory capabilities are crucial to the ultimate acceptance of expert systems' (Buchanan & Shortliffe, cited after Swartout, 1990, p. 298).

- □ It may simply be impossible to construct a cognitive model in many projects. We still know very little about the nature of expertise, and practical reasons may often forbid the attempt to construct a full model. Sometimes, however, much is won by even little steps in the right direction. Whenever we know that we cannot obtain the ultimate goal, it may be advisable to take a step or two in that direction.

- □ A cognitive model may not be necessary, indeed, it may turn out to be adverse, or harmful, to the task. Of course, that might come true for very specialized, very technical domains only. Still, we must remember that our prime objective is not to model the expert, but to provide expert system tools for novices and experts alike. Therefore, diagnosing faults in a complex device may be more efficiently done by means of a (correct) engineering model than by modeling the cognitive processes of human experts. Still, I believe that modeling the expert is a commendable strategy in most of the cases.

To sum up, it costs, but (at least usually) pays, to strive for a system that employs cognitively adequate representations of knowledge and problems, and equally adequate ways of processing that knowledge. Yet if we make a stand for cognitive modeling in knowledge engineering, we must try to define what it is that we aim at. Therefore:

What does it mean to be cognitively adequate?

There is certainly room for disagreement when answering the above question. For instance, does the term *cognitive* pertain to human thinking exclusively, or not? Whole wars of definition could be envisaged. Yet my objective in this chapter is much more humble than that. The following is no more than an attempt to scale different degrees of 'nearness' to human cognition, and to explore the consequences of each, if attained in knowledge engineering and expert systems.

Strong and weak cognitive adequacy. If strong adequacy is claimed, the system is supposed to function like a human expert, at least in a circumscribed domain. In short, strongly adequate systems employ the very same principles of cognitive functioning as human experts do. If this is not the case, but the system has been carefully built with the human user in mind, it may be credited with weak cognitive adequacy. In this sense, 'weakly adequate' means the same as 'well-adapted to the user', or 'easy to use'.

Weak cognitive adequacy. This is, or rather, should be, the trivial case. A system so characterized is ergonomic and user-friendly. Consistency of the user interface, recognized graphical standards, a clear language for commands, help texts, etc., make the system easy to use. Note, however, that the system may differ considerably from the experts (whose knowledge it attempts to represent) and from its users (if those are different from the expert group). Still, the systems tries to give users a comfortable feel, which may be achieved through symbols or words familiar to the user. The drawback is that the user's understanding of system messages may be at odds with what the system actually does.

Remember the Apple MacIntosh's first version of the trash can, for instance. Its behavior - it worked simply as a delete command, so files could not be regained - irritated many users and was corrected in later versions. In expert systems, the possible damage is vastly greater. Users may interpret the systems explanations wrongly; they may, for instance, arrive at inadequate degrees of confidence in the system. Users of expert systems, if they are not experts themselves, might be lured by 'intuitive' surfaces into unwarranted conclusions about the scope of the system's knowledge. Indeed, the more 'intelligent' a system behaves on the outside, the less it is expected to lack common sense.

Much the same difficulties that emerge in the translation process from system operations to explanations given to the user, also arise in the process of knowledge engineering, i.e., translation of human expert knowledge into the formalism used by the system to represent and utilize it. If the basic notions of the expert are incompatible with those that the system can represent, knowledge engineering becomes complicated, to say the very least. In addition, verification of the knowledge base is next to impossible, because the system's knowledge entities cannot be compared with those of the expert.

My opinion is that weak cognitive adequacy is something every good program attains to, and that it is definitely not enough for expert systems. Indeed, I believe that to remain at that very low level of cognitive adequacy would become an obstacle to knowledge engineering, and a source of severe errors to the user.

Strong cognitive adequacy, absolute and relative. Strong cognitive adequacy comes in (at least) two degrees, absolute and relative. This is to mean, that an absolutely adequate system of the 'strong' family claims to be a model of the human expert's knowledge and way of reasoning in every relevant aspect. For relatively strong adequacy, that claim is reduced to the assertion that the kind of knowledge representation and reasoning as used in the system can be found in human experts, too.

The notion of absolute cognitive adequacy is, of course, an ideal not even fulfilled in the most ambitious projects of cognitive science, for which it amounts to the long-range strategic goal of cognitive modeling. But it is not an objective for AI systems, although some authors, like Charniak and McDermott in their well-known textbook, claim that for the ultimate achievement of AI. To the contrary, it is obvious that expert systems need not (nor should they) have all human faculties at their disposal, not even all cognitive abilities. It would be nice if expert systems had some common sense, however, and the little we know about that suggests that we would need all human knowledge, and, perhaps, the human body even, to arrive at common sense, therefore it does not seem feasible to have common sense (which we would like) without all the complexity of human cognition (which is a highly impractical, if not impossible task to do). This line of argument serves to reduce the quest for cognitively adequate systems to what I have termed relatively strong cognitive adequacy above.

To construct cognitively adequate systems in the strong sense, then, amounts to employ knowledge representations and methods of reasoning that can be demonstrated to be used by human experts, too. As we have argued, this will not amount to a complete model of the human expert, and it need not amount to that. Still, we should take care not to ignore kinds of reasoning and forms of representation that seem essential for natural expertise.

As far as we know, human experts have insight into the causal relationships in the domain, and they use it mainly when other, easier approaches fail, or when they are asked to explain their results. This kind of knowledge may be captured by causal, or 'deep' modeling. Apart from that, their experience yields heuristic rules that may reflect statistical properties, e.g., the relevance and typicality of certain symptoms with respect to some diagnostic category. Rule-based systems usually represent this kind of expert knowledge.

But experts, as far as we know, often and in many domains, like to think 'analogically', i.e., use visual imagery. This aspect is not easily captured in technical systems, which by definition lack the sophisticated visual apparatus shaped in billions of years of evolution. Expert systems, as far as I know, have not yet been enriched with this kind of representation, although there are quite a few experimental approaches to imagery. Finally, experts often work along the same lines as they did when they solved a similar problem, and they like to refer to specific case studies in knowledge engineering interviews in order to explain procedures, or to illustrate certain principles. This characteristic trait of human expertise has attracted much attention recently, and given rise to the construction of systems that employ case-based reasoning.

An important, though mostly neglected, characteristic of natural expertise is meta-knowledge, or reflection. Human experts generally differ from the inexperienced, because they are always 'oriented' during their work. They do not lose sight of the solution they seek, as novices often do, they seldom make errors, and they are quite good at estimating how near they are to a solution (Gruber & Strube, 1989). Meta-knowledge also lets them switch between strategies, an ability that is highly desirable in technical systems, too.

To sum up, I believe that being cognitively adequate in the weak sense is mandatory, yet not sufficient, for modern expert systems. We must strive to use representations of domain-specific knowledge that are both practical and understandable to human users. Not surprising, it turns out that this is what human experts usually have arrived at. We should further try to implement kinds of reasoning that are used by human experts. At least some combinations, like using a case library of previously solved problems together with case-based reasoning techniques, promise to cut down the potentially enormous amount of search space and time, required by purely rule-based systems. We should also try to include at least a moderate degree of reflection, or meta-knowledge, in the system, which is both a hallmark of human expertise and a necessity for control in hybrid systems. Admittedly, we cannot make an expert system into a complete model of the human expert, but we should attempt to include representations and reasoning strategies that are essential to human experts.

Towards cognitively adequate systems

Starting with ergonomics. Aiming at weak adequacy first, let us start with ergonomic design of the knowledge base and its user interface. It needs consistency. It also needs flexibility, which in turn requires 'intelligent' adaptation of the system, hence, a certain degree of AI. User modeling, in the meantime, has become a field of its own (e.g., Kobsa, 1985), but the approach of classifying users and designing the interface according to some typology is not sufficient and in need of enhancement. – Modern user interfaces are also characterized by windowing and graphics. Pen-based input will spread rapidly, while I doubt whether speech output will ever become popular. (Hearing a soft female voice saying 'Your prin-ter ist out of pa-per' repeatedly is not going to make your computer interaction more pleasant, after the initial surprise wears off.) The use of natural language, however, is bound to increase as even portables include the high-performance processors and huge amounts of memory needed for the task. Here is another field where AI comes in.

'Knowledge ergonomics'. The next step will not take us further from weak cognitive adequacy, yet it is a crucial step because it will bring us to knowledge. Naturally enough, the kind of knowledge used by an expert system, even if its machine representation may differ from the one we use in our heads, must be linked to our way of thinking. There are many means that together serve to accomplish this task, although no single one is sufficient.

The basic requirement is that the terminology used by the system must be consistent (see above) and should agree as well as possible with the terminology used by experts in the field. The usual techniques of extracting terminological knowledge from textbooks fulfil that requirement only partially. For instance, important concepts may lack verbal or formal definition, as in the field of interpreting aerial pictures a 'gently rolling plain' (Hoffman, 1987). The basic problem, however, applies to the status of terms or concepts with respect to their role in problem solving. Here is where proposals like KADS come in (Breuker & Wielinga, 1985; KADS-II, 1990). Although KADS is not primarily aimed at being cognitively adequate, its meta-terminology serves to structure a given domain and define the functional role of the concepts, too.

In addition, tools should be available to make the system understandable to its users. This concerns mostly what is called the explanation facility of an expert system. Explanation, of course, should not be limited to a 'rule 463 fired' style of system messages. Use of the domain terminology in explanations is important. The system's line of reasoning should be displayed, as well as the causal dependencies in the knowledge base used by the system. Graphical components may enhance that facility through visualization of causal dependencies, and can in turn be integrated with browsers.

A further tool to enhance the cognitive ergonomics and general usefulness of an expert system is an ' as if' mode of functioning. This function is common in spreadsheets, where you can run simulations, and compare the effects of certain modifications. The same should be extremely useful for knowledge-based systems in order to assess the effects of certain pieces of knowledge, of different kinds of reasoning, or just different preference orders in reasoning. An 'as if' mode opens a system to the user for exploration. It also enhances a system' s usefulness in training.

What else? In order to make a system approximately cognitively adequate, we must embed the characteristics of human problem solving in its knowledge base and inference engine.

□ ***Deep modeling.*** Pure surface modeling, i.e., reliance on statistical rules, is not enough. Although this kind of rules is used by experts, and serves to get quickly at 'most probable' diagnoses, etc., human experts also have the capability to generate causal explanations for any problems using 'deep', functional models of the domain. Especially for untypical problems, deep modeling is a must for expert systems. Apart from considerations of efficiency, functional models of the domain are a necessary part of cognitively adequate systems, because only functional models serve to generate true explanations of a problem and its solution.

□ ***Episodic knowledge and case-based reasoning.*** Human knowledge acquisition makes good use of examples, and learning proceeds fastest when rules and examples are combined (Schmalhofer & Kühn, 1988). Human experts are known to rely on episodic knowledge almost exclusively in domains like law and judgement, and substantially in domains like medical treatment (Strube & Janetzko, 1990). Case-based reasoning has therefore become a major field of research, as conferences (e.g., DARPA, 1988) and recent textbooks (Schank & Riesbeck, 1989) show. Integration of CBR into traditional, rule-based approaches to expert system construction has become an important step in the direction of systems that are cognitively more adequate (see Janetzko & Strube, this volume).

□ ***Meta-knowledge.*** Cognitive control of one' s strategies of reasoning, monitoring of one's own way toward a solution, and careful deployment of cognitive resources is one of the hallmarks of human experts (Gruber & Strube, 1989). The present state of expert system technology, however, largely ignores this aspect. Still, I hope that recent developments in software engineering, above all the debate on OOP and multi-agent systems, may stimulate discussion of intelligent distribution of resources.

□ ***Learning.*** Although machine learning and expert systems have remained separate fields up to now, automatic knowledge acquisition, and hence, incorporation of components for learning, has become a hot issue for expert system technology. I'll focus on automatic knowledge acquisition during the system's active life time, i.e., on acquiring new and additional knowledge.

Human experts continuously change and amplify their knowledge. This is done via two different processes: (1) Explicit communication (mainly verbal) of rules or equivalent (i.e., general) knowledge, and (2), learning from experience through re-use of solution paths already successful in previous cases, and through generalization from specific experience. Much the same classification holds for machine learning. Modern expert systems should provide means for learning directly from the user. That feature should also include checking procedures for integrability of new rules into the knowledge base, i.e., checks for logical consistency (still a hard problem), and in case of inconsistencies, prompting the user to specify conditions of applicability in order to circumvent inconsistency. (I am assuming here that AI's means of dealing with inconsistency do not yet approach the power of our abilities to be both rational and inconsistent.) In addition, an expert system should include facilities to generalize from experience (at least some simple, well-known algorithm, like EBG), and the capability to adapt and re-use previously computed solutions to similar problems, in other words, case-based reasoning. The German Ministry of Research and Technology's project FABEL (1991) is an example of building systems in that spirit.

Approximating strong cognitive adequacy in knowledge-based systems, as may be guessed from the paragraphs above, is neither easy, nor is it cheap. Even those factors necessary for weak cognitive adequacy, i.e., the ergonomic features, are on a par with the most ambitious features to be found in other software with respect to complexity and demands on the hardware. So my message is once more to take at least some steps into the direction of greater cognitive adequacy, for this will enhance usability, and therefore acceptance. This brings us to our last issue, viz., concepts of expert systems usage that fit with the design goal of cognitive adequacy.

How to use cognitively adequate expert systems

Design of information-processing systems must include a philosophy of its application. This means that the designer must consider the role of the system in the organization of work on the problems the system is intended to help with. In other words, the system' s integration into the greater mixed man-machine system has to be thought about.

Expert systems technology started out with the goal of systems that could replace costly experts. While this is still a valid conception in certain environments (e.g., for doing routine diagnostics in computer-integrated manufacturing), the general trend has shifted to emphasize expert systems as general tools in the hands of experts (Becker & Paeteau,1991). This philosophy of expert system application makes communication between man and machine much easier, since the gap in terminology and knowledge between expert and non-expert is eliminated. In addition, non-expert users are usually not able to assess the limitations and constraints of an expert system's reasoning. Experts, however, to have the necessary prerequisites to evaluate systems, and to take their basic presuppositions into account. An expert system is a complex tool that needs sophistication on the side of the user, too. Seen as sophisticated tools, expert systems are also more benevolent in social perspective, since they do not aim at replacing highly skilled workers, but enhance the quality of work. Still, in order to become a tool as good as possible, expert systems must be given the ability to explain their ways of reasoning and, most importantly, evaluate alternative solutions (see the 'as if'-mode discussed earlier in this chapter). Cognitively adequate systems will simply be the better tools.

Goals for knowledge engineering

How can knowledge engineering help us to get systems that are cognitively more adequate? I believe that we should follow three promising lines at least:

□ continue to follow the modeling approach (e.g., KADS), which are not implementation-specific, in order to provide a framework for terminological classes that could perhaps be modified to become cognitively more adequate in itself - a cognitve orientation need not be adverse to enginnering needs (see Linster, this volume),

□ enrich present-day knowledge engineering techniques with methods specifically devised or adapted for episodic knowledge, thereby integrating case-based approaches with more traditional lines of expert systems design, and finally,

□ integrate knowledge engineering and machine learning. This is not meant to take a stance in the long-standing discussion about the benefits and shortcomings of automatic knowledge engineering v. knowledge acquisition as mediated by knowledge engineers. Indeed, I believe that machine learning methods are a desirable complement to work that can and should be done by knowledge engineers.

All this, and still more so the issues to be tackled along the way to strongly adequate cognitive systems, is expensive with respect to both effort and costs. But the benefits to be expected from those efforts might be even greater: It pays to have cognitively adequate systems. I wish knowledge engineering would follow that direction.

References

Becker, B., & Paeteau, M. (1991). Von der kognitiven zur interaktiven Adäquatheit? In: T. Malsch (Ed.), *Informatisierung und gesellschaftliche Arbeit.* Berlin: Edition Sigma.

Breuker, J.A., Wielinga, B. W., et al. (1987). *Model-driven knowledge acquisition: Interpretation models.* Deliverable task A1, Esprit Project 1098. University of Amsterdam.

Charniak, E., & McDermott, D. (1985). *Introduction to Artificial Intelligence.* Reading, MA: Addison-Wesley.

DARPA (1988). *Case-Based Reasoning. Proceedings of the 1st Workshop on Case Based Reasoning.* Clearwater, FL.

Ericsson, K. A., & Simon. H.A. (1984). *Protocol analysis. Verbal reports as data.* Cambridge, MA: MIT Press

[FABEL](1991). FABEL: *Intergration von modell- und fallbasierten Entwicklungsansätzen für wissensbasierte Systeme.* Antrag für ein Verbundvorhaben an den Bundesminister für Forschung und Technologie (BMFT), Bonn.

Gruber, H., & Strube, G. (1989). Zweierlei Experten: Problemisten, Partiespieler und Novizen beim Lösen von Schachproblemen. *Sprache und Kognition, 8, 72-85.*

Hoffmann, R. (1987). The problem of extracting the knowledge of experts from the perspective of experimental psychology. *Al Magazine, 8 (2), 53-67.*

Janetzko, D., & Strube, G. (this volume). Case based reasoning and model-based knowledge acquisition.

[KADS-II] (1990). KADS-II. Esprit II Technical Annex for Project P5248. Meylan, France: Cap Gemini.

Kobsa, A. (Ed.)(1985). *Benutzermodellierung in Dialogsystemen.* Berlin: Springer.

Riesbeck, C., & Schank, R.C. (1989). *Inside case-based reasoning.* Hillsdale, NJ: Erlbaum.

Schmalhofer, F., & Kühn, O. (1988). *Acquiring computer skills by exploration versus demontration.* 10th Annual Conference of the Cognitive Science Society. Montreal. Hillsdale, NJ: Erlbaum.

Stigler, J. F. (1984). 'Mental abacus': The effect of abacus training on Chinese children's mental calculation. *Cognitive Psychology, 16,* 145-175.

Strube, G., & Janetzko, D. (1990). Episodisches Wissen und fallbasiertes Schließen: Aufgaben für die Wissensdiagnostik und die Wissenspsychologie. *Schweizerische Zeitschrift für Psychologie, 49,* 211-221.

Swartout, W. (1990). Explanation. In S.C. Shapiro (Ed.), *Encyclopedia of Artificial Intelligence* (vol. 1, pp. 298-300). New York: Wiley.

Acknowledgements. *I wish to thank all the participants of the KEKOG workshop for an inspiring discussions of the ideas presented here, and especially Angi Voß, who commented on the first draft of this paper.*

Knowledge Acquisition as an Empirically Based Modelling Activity

Beate Schlenker and Thomas Wetter

IBM Germany, Scientific Center
Institute for Knowledge Based Systems
Wilckensstr. 1a
W-6900 Heidelberg
Germany
WETTER@DHDIBM1.bitnet

When knowledge engineering is consequently looked upon as scientific discovery, certain prescriptions can be derived about how to observe expertise and how to deal with intermediate stages of the modelling process. These prescriptions concern roles played by underlying assumptions, theories and paradigms of cognitive science and their respective implications upon conduction and interpretation of individual experiments, and the language(s) used in the modelling process. Most essentially, scientific discovery is characterized by planned feedback, including theory based identification of contradicting or falsifying evidence. Some of these characteristics clearly differ from principles suggested in KADS for the respective modelling activities.

1.0 Considerations on Theory of Science and Underlying Assumptions

The process of knowledge acquisition can be characterized as the knowledge engineer's attempt to get to know the epistemological structure of the domain to be modelled (Woodward, Shaw, and Gaines 1991). To this end the knowledge engineer has to learn something about the concepts and problem solving strategies used by the expert in the respective domain. One problem is that the knowledge engineer is a novice in the respective domain of expertise. Furthermore the expert's concepts and problem solving strategies are not directly observable, but have to be derived from (mainly verbal) data. In cognitive psychology (and AI) it is widely respected (e.g. Gigerenzer 1981; Clancey 1989) that modelling empirical phenomena requires both, specific assumptions about mechanisms underlying the phenomena and a foundation of the modelling process in theory of science. If the knowledge engineer does not make these theoretical concepts explicit, he will interpret the expert's observable behavior with uncontrolled bias and his interpretations will not be reproducible by others.

The specific assumptions we make draw upon specific theories or paradigmata from cognitive psychology as background knowledge. This knowledge is not further put under question during the modelling process. One possible background or basic assumption (the one we use) may be to describe higher cognitive processes such as learning as information processing behavior (*information processing paradigm*). Methods used in cognitive psychology to elicit data and to infer knowledge about the underlying cognitive processes are considered as background knowledge as well. The methods come along with their respective refinements of the information processing paradigm. Concurrent think aloud protocols are e.g. in accordance with the "working memory"-refinement of the information processing paradigm. In the framework of the so defined knowledge the knowledge engineer constructs models about entities whose existence "on the expert's mind" would explain his behavior. [1]

Concerning the foundations of the process to arrive at such a model one possibility to reduce the knowledge engineer's bias and to make the modelling process more objective is to apply the general "empirical method" (cf. Schulz, Muthig, and Koeppler 1981) to knowledge acquisition. This implies the use of Popper's falsification strategy (1966), according to which hypotheses or theories have to be formulated in a way that they can be falsified in principle. The knowledge engineer's misconceptions can thus be identified during the phase of model construction. The proposed "empirically based method" of knowledge acquisition consists of a sequence of operations and related rules that are applied to test the appropriateness of models used to reconstruct part of the perceivable reality. The process of model construction can be characterized as a process of incremental refinement from a rough general to a complex and more detailed and accurate one.

Since model development includes correction by falsifying evidence, the knowledge engineer has to define observable evidence against his hypothesis before having any data extracted from the expert. This implies both the selection of a data extraction method and the a priori definition of an encoding schema. An encoding schema is principially determined by the observation method used and the theoretic foundation which it is built upon. Contrary to methods like KADS, where the static models are

[1] We might speak of concepts, relations, processes ... instead of entities. But this would constrain the variation of model structures admitted in our approach more than necessary (given the information processing paradigm) and desirable given the subsequent considerations about language.

imported in order to form the model of expertise, using the strategy of planned feedback means dynamic model development.

2.0 Planning the Process

We don't treat a requirements analysis as part of the following process. Given the requirements the following activities occur several times in a cyclic process until a result meets external criteria, which we assume to have been specified as part of the requirements. Furthermore the information processing paradigm and the relatedness of observation and underlying theory are not put under question. As to the rest, the strategy of planned feedback means dynamic model development. In the following the modelling steps are listed.

1. Construction of hypotheses forming a theory that explains the knowledge underlying the expert behavior. This can be knowledge concepts as well as knowledge structures or problem solving strategies.
2. Formulation of a discriminating experiment. This includes the derivation of observable behavioral consequences that would be supported by the hypothesized underlying cognitive structures or processes as well as the behavior representing the falsification of the hypothesis. It also includes how to encode the raw observations.
3. Definition of a criterion for empirical evidence to distinguish between support and falsification of hypotheses.
4. Evaluation of a hypothesis according to empirical data. That is rejection of the hypothesis in case of falsification.
   a. In case of falsification: Continue with 1; construct new hypothesis
   b. In case of support: Continue with 6
5. Check whether the model meets external criteria.
   a. If not: Continue with 1; refine hypotheses
   b. If yes: stop.

2.1 Falsification

We suggest to distinguish *essential falsification* from *incidential falsification.* We suggest to call Popper's falsification in the strict sense using only one contradicting evidence *incidential falsification.* Following Popper in a strict sense would mean that a hypothesis or theory would have to be given up in case of only one unexpected item occurring in an observation. For reasons which are probably inherent to the field of human cognition, incidential falsification will always occur. That is, theories will permanently be falsified before they can be applied. Therefore criteria have to be established according to which sets of observations are considered as falsifications. Such an essential falsification provides the feedback that the model has to be rejected or modified. The term essential falsification means that a considerable amount of falsifying items is needed in order to reject a hypothesis or theory. Statistical evidence cannot be used in the usual sense of observing a random sample of subjects, since the investigations made are mainly single case studies aiming at modeling individual expertise. Hence, criteria have to be established which are plausible with respect to the method applied (e.g. its characteristic noise). One could also think of external criteria for essential falsification, such as ratings of another expert or consent from the party that ordered the knowledge acquisition activity. The empirical data are then compared to the evidence criterion and the hypothesis or theory is refuted in case of essential falsifying evidence. Otherwise the hypothesis or theory can be kept until a better one is found.

It should be pointed out that our strategy of planned feedback and the incorporation of falsifying experiments can be seen as competing with an approach which might be derived from Woodward et al. (1992), where the observer considers possible causes of misconception, bias, etc. and tries to avoid them by compensating means either in planning investigations or interpreting observations. This anticipatory attitude or procedure is, however, hypothesis-driven. Hypotheses are not made about the model itself but about factors influencing the construction of the model. And hypotheses are never systematically challenged or tested in the process; they are just imported.

3.0 Development of the Modelling Language

It has turned out as another essential side condition of such an approach that the language used for denoting the model must not be prescribed in the beginning. As Heisenberg (1989) has pointed out, the need for the means to describe the phenomena under study emerges from the process of investigating them. E.g. quantum mechanics would not have reached its present state of cognition, if the term ″impulse″ would not have been discovered and understood as being applicable also in situations where mass, speed, or both don't make sense[2] . In our case this means that we must avoid to prescribe a formal language for the model of the phenomenon to be investigated. We must even try to be mentally independent of such languages (to the extent that humans are capable of thinking independently of any language). It must be part of the investigator's attitude and awareness that situations will occur where he has to overcome the language he used in former iterations. On the other hand, a formal language must be arrived at in the end. For the model to be executed on a computer,

[2] i.e. essentially in such situations where the wave aspect is more useful than the particle aspect and the impulse of a wave emerges more or less naturally from its spectrum.

it must be written in a language whose formal semantics is known. For the present purpose this should be operationalized in two ways. A denotational semantics allows to argue about properties such as truth conditions of sentences that might be part of the model. A procedural semantics allows to map the elements of the model onto operations offered by some abstract machine, which then simulates the behavior of the model. Agreement between denotational and procedural semantics in the example of truth conditions means that the simulation supplies "true" exactly for the sentences found true on the basis of the denotational semantics.

For practical purposes the direction into which the language should be developed, or maybe better, the sector in which the intermediate languages should lie, is thus constrained by the need for formal semantics.

4.0 Skill Requirements for Empirically Based Modelling

Reviewing all these requirements, it becomes obvious that the task we are describing is nothing less than scientific discovery. That means that the knowledge engineer doing his work according to these suggestions needs to have at least the following skills:

- sufficient psychological knowledge to design experiments for a large number of purposes
- some basic knowledge of theory of science and theory of cognition to do observation, interpretation, hypothesis formation, and experiment planning in a principled way
- sufficient knowledge in semantics of formal languages to proceed from informal first notations to full fledged formal languages.

In any case it is obvious that the suggested approach needs both very high skills and very much effort. So it may be asked whether less expensive methods should be generally preferred. In practice many projects which can be carried through by straightforward use of KADS would never have been approved on the basis of cost estimate of what we are suggesting. There are, however, general considerations and specific situations where our suggestion should nevertheless be applied.

5.0 Comparison of Empirically Based Modelling and Modelling in KADS

In the following the "empirically based method" as described above is compared with some modelling aspects of KADS. Although in KADS no explanation of the underlying cognitive structures but functional models are intended, Breuker & Wielinga (1987) claim to explore the nature of expertise. Another intention of KADS is to support systematic knowledge engineering. Since data used for model verification are extracted by psychological methods, a critical analysis of some specific assumptions and actions in KADS seems helpful. The general consideration addresses the restriction by language. If we understand the KADS layers as constituents of a language and the interpretation models (IMs) as building blocks for future applications, then the scope of what can be expressed is definitively fixed. Whatsoever detail identified as characterizing an application under study has to be expressed in KADS layers and IMs. Details not fitting into this frame can either not be expressed or have to be deformed.

5.1 Variety of Models in the Approaches

The process of empirically based modelling may in some case well arrive at a model which replicates the KADS layers (domain, inference, task, strategy), because these are not implausible and definitely within the information processing paradigm. They do, however, not exploit its variation - frames with their tight coupling of entity and inference would not be covered by the KADS layers. Hence the empirically based modelling might arrive at models outside the scope of KADS.
Obviously the IMs further constrain the search space (which is the intention in KADS and not per se a deficiency). However, among those empirically developed models which do replicate the KADS layers, only some will also replicate one of the existing IMs.

5.2 Aspects of Validation and Control in the Process

What is more severe than the limitation of expressiveness is the principle lack of feedback indicating that something has been missed. I.e. KADS representations which look appealing may be essentially deficient without any systematic chance to detect the deficiency.

5.2.1 Model selection vs. model construction

In contrast to the dynamic process in the empirically based approach with its means of feedback, modelling in KADS is intended as top-down process driven by the interpretation models. Interpretation models are matched with verbal data gathered by psychological methods in order to find the appropriate interpretation model for the domain. This is critical since the method of verification does not imply any possibility to control the knowledge engineer's bias in interpreting the verbal data. Plausibility is used as the only test criterion for the appropriateness of the constructed conceptual model. Therefore model selection according to plausibility can lead to a situation where different knowledge engineers select different interpretation models for the same domain with every knowledge engineer having a plausible explanation for his decision. This reveals a certain arbitrariness of model selection in KADS.

5.2.2 Observation and interpretation

First of all it has to be clarified that elicited data are directly used in KADS, while encoded items are generated in the empirically based approach. The KADS view implies that verbal data have an inherent truth. It contradicts the empirical view of data as verbalizations interpreted on the predefined background knowledge (cf. Clancey 1989).

5.2.3 Justification of elicited material

For the selection of methods to elicit verbal data, Breuker & Wielinga (1987) have developed a table (p.23) which statically assigns methods to required data types such as procedural or static knowledge. This again reflects the attitude that elicited data can be both collected from the expert and applied in selecting or filling an IM irrespective of their context of elicitation and without need of theory and situation based interpretation. In other words: elicited data are justified in themselves and as they occur.

In the empirically based approach elicitation methods and their respectice encoding prescriptions are dynamically designed or selected according to the needs to support or falsify an actual hypothesis. Each method entails guidelines to remove subjectivity and bias as far as possible based on known characteristics of the methods. As an example of such criteria, Ericsson & Simon (1984) name protocol segmentation and separate encoding of the segments in arbitrary order as a means to achieve objectivity. An encoded item is hence justified by having been produced from an authentic recording of verbal data by means of theoretically justified procedure. As the next step will show, this does not mean that such an item is never put under question again.

5.2.4 Justification of the use of elicited material in model formation

Use of verbal data in KADS means model matching as long as an IM still has to be selected or slot filling once the decision for an IM has been made. In the latter case the attitude is confirmatory, i.e. the use of verbal data is justified when the knowledge engineer managed to use it for his model construction.

In the empirically based approach the use of encoded items is never finally justified. The use of items to develop a theory respectively its model formulation in some direction may be hypothetized at some stage in the development and may have to be withdrawn at later stages by falsifying evidence concerning those hypotheses of the model, to which the items contributed. Therefore, the use of items can be described as temporarily justified as long as no falsifying evidence has occured.

5.2.5 Justification of models of expertise

Principally, in both approaches the justification of a full model is the sum of the individual justifications outlined above. This does not yet entail that a justified model is correct or useful. There is a weak notion of correctness in KADS, of the kind that all verbal data can be explained by a model. In the empirically based approach there is no claim that a model will ever be correct. It would only be used as an acceptable approximation as long as there is no falsifying evidence. Usefulness has to be judged on the basis of requirements, which have not been part of this outline.

Breuker, J.A. & Wielinga, B.J. (1987) *Use of models in the interpretation of verbal data.* In A. Kidd (ed.), Knowledge Acquisition for Expert Systems: A Practical Handbook, (p. 17-44). New York: Plenum.

Clancey, W. (1989) *The frame of reference problem in the design of intelligent machines.* To appear in K. van Lehn & A. Newell (eds.), Architectures for Intelligence: The Twenty-Second Carnegie Symposium on Cognition. Hillsdale: LEA.

Ericsson, E.A. & Simon, H.A. (1984) *Protocol Analysis. Verbal reports as data.* Cambridge, Mass.: The MIT Press.

Gigerenzer, G. (1981) *Messung und Modellbildung in der Psychologie.* München: Reinhardt.

Heisenberg, W. (1989) *Ordnung der Wirklichkeit.* München: Piper.

Popper, K.R. (1966, 1989) *Logik der Forschung.* 9. Auflage. Tübingen: Mohr.

Schulz, Th., Muthig, K-P., Koeppler, K. (1981) *Theorie, Experiment und Versuchsplanung in der Psychologie.* Stuttgart: Kohlhammer.

Woodward, B.J., Shaw, M.L.G., Gaines, B.R. (1992) *The Cognitive Basis of Knowledge Engineering.* in this volume

Shifting Positions: Moving From a Cognitive Science Point of View to a Knowledge Engineering Stance

Marc Linster
AI Research Division
GMD
PO Box 1240
D-5205 St. Augustin 1

Abstract

After a short overview of knowledge acquisition highlights, we review experiences that we had in our knowledge acquisition project. We conclude that automated knowledge acquisition does not work without a documentation of the purpose that the knowledge will fulfill once it is acquired. This can be done for example through a description of a method of problem-solving. The remainder of the paper gives a more detailed account of the motives (outside the actual experiences with KRITON) that lead to these conclusions. After outlining several requirements, we delineate the role of cognitive science research in our current approach.

1 A Review of Work in Knowledge Acquisition that Influenced Us

We give a short overview of developments in knowledge acquisition that we consider as cornerstones. There is no intention of completeness. We only want to situate our work, and explicate our bias.

The systematic construction of knowledge-based systems—henceforth called *knowledge engineering*—started out from early work on TEIRESIAS [Davis, 1982] and first developments of engineering guidelines for knowledge-based systems, for example [Buchanan *et al.*, 1983]. A considerable effort has been invested to obtain systematic support for the development of knowledge-based systems. To widen the *knowledge-acquisition bottleneck* researchers have investigated a multitude of methods: cognitive-science oriented techniques for the systematic elicitation of knowledge, for example protocol analysis, or personal-construct analysis; dedicated tools for the acquisition of knowledge for special tasks, for example OPAL [Musen *et al.*, 1987]; method-specific knowledge-acquisition tools such as ETS [Boose, 1985]; workbenches to support conceptualization, such as ProtoKEW [Shadbolt, 1992]; learning systems to develop domain-models, for example, BLIP [Morik, 1987]; and knowledge acquisition methodologies, such as KADS [Wielinga *et al.*, 1992].

In the course of these developments, a change of focus occurred. Whereas early work viewed the development of knowledge-based systems as a transfer activity from a human mind into a computer-accessible representation—this is best illustrated by Feigenbaum's bottleneck metaphor or Boose's expertise *transfer* system ETS—current work views knowledge engineering as a process of constructing models [Clancey, 1989]. At the same time, the *mining view* of knowledge acquisition, whose optimistic variant states that extensive and deeper knowledge elicitation provides the key to expertise, gave way to a more pragmatic view, stating that the knowledge acquisition process is guided by the requirements of system building. *If we were able to obtain the detailed data of the knowledge* *that drives human expert behavior, we would not know how to handle it* [Breuker and Wielinga, 1989, p. 267].

2 Experiences in Our Knowledge Acquisition Project

This shift of positions can be illustrated very well using GMD's knowledge-acquisition project. Initially we focussed on automated knowledge-elicitation tools that relied heavily on cognitive-science motivated techniques. The first such tool that we developed at GMD was called *KRITON*. KRITON made it obvious that automated knowledge acquisition without an underlying method of problem-solving does not work, no matter how pretty and elaborated the knowledge-acquisition techniques or interfaces are.

The next sections give a more detailed account of the reasons and intermediary steps which lead to that conclusion.

2.1 The Approach Taken in KRITON

The knowledge acquisition tool KRITON [Diederich and Linster, 1989; Linster, 1989] includes several knowledge elicitation techniques that cooperate with the goal to acquire a large coherent body of knowledge: (1) TEXT ANALYSIS and INTERVIEW for the acquisition of static domain features; and (2) PROTOCOL ANALYSIS for the acquisition of procedural and associative knowledge.

The TEXT ANALYSIS component reads texts from a file. Nouns are highlighted and made mouse-sensitive. The user can include them into a hierarchy describing the way the text presents the organization of the concepts of the domain, or the way the expert sees their organization. The text analysis is a computer support to get started in a knowledge acquisition process.

The INTERVIEW component edits and completes the initial information gathered by the TEXT ANALYSIS. Relying heavily on the repertory-grid technique [Gaines and Shaw, 1981], it acquires attributes describing the concepts of the domain.

Several other techniques are used in the INTERVIEW to complete the description of the concepts, such as *explicit inheritance*, *explicit generalization*, and *explicit completion*.

PROTOCOL ANALYSIS helps acquire procedural and associative knowledge. It is based on the work of Ericsson and Simon [1984]. It concentrates on the analysis of transcripts of recordings of thinking-aloud protocols. To analyze protocols in KRITON, they are transcribed with the pauses of speech. Using the assumption that pauses of speech represent delimiters that separate the transcript into coherent segments, these segments are transformed into operator-argument structures that are then combined into rules to make the inference steps, implicitly contained in the protocol, explicit.

TEXT ANALYSIS and INTERVIEW work on a common data structure: a semantic net consisting of labeled edges and attributes. The user has total freedom to define relationships as edges or as attributes. PROTOCOL ANALYSIS produces rules describing how the concepts in the semantic net are used in a problem-solving process.

In a prototypical acquisition session the user starts out with analyzing a background text and building a taxonomy representing the most important and frequently used concepts of a domain. These will then be attributed and described in the INTERVIEW. The taxonomy can be edited to fit new facets of the domain unraveled during the INTERVIEW. After this one can go back to the TEXT ANALYSIS to complete the taxonomy or go directly into the PROTOCOL ANALYSIS. New concepts appearing in rules are introduced into the taxonomy in the INTERVIEW. The result of an acquisition process with KRITON is a dense network describing structural relations (stemming from INTERVIEW and TEXT ANALYSIS) or associative relations (stemming from PROTOCOL ANALYSIS).

2.2 Consequences and Subsequent Work

KRITON provides good analysis facilities to define an initial vocabulary and to get a first start when building a knowledge-based system.

Experience though showed that to move beyond the initial knowledge acquisition, a tool needs strong guidance—it must be goal driven. In KRITON it is not clear in which directions the taxonomies and the protocols have to be elaborated, as no meta-knowledge about the purpose of the knowledge is available to the system. As the system has no information about the role an element of the semantic net plays, it cannot inquire about the typical relations that are needed between problem-solving concepts of certain roles.

In subsequent developments we focussed on making KRITON knowledge-based, to give it explicit information about which kind of knowledge to acquire [Diederich and Linster, 1989]. We introduced pre-defined template-like domain structures, which had to be filled by the knowledge acquisition process. The process is guided by a program called *WATCHER*, using meta-knowledge about expected domain structures to trigger elicitation tools. For example, the WATCHER uses the rule *"classes and instances must be discernible on the basis of their attributes"* to trigger a differentiating repertory-grid–based interview for elements of the knowledge base requiring attributes or values. However, a reconsideration showed that the previous rule expresses knowledge structures needed by a *classification problem-solving method.* This led us to an analysis of the relation between problem-solving method and knowledge-acquisition tool. For example we analyzed the combination of the problem-solving method *heuristic association* with the repertory-grid based interview tool of KRITON. It showed that strong guidance for the acquisition tool emanates from the problem-solving method.

First experiences with KADS [Karbach *et al.*, 1989] led to the assumption that flexible, configurable problem-solving methods are needed, instead of pre-defined, selectable ones. The knowledge-needs, derived from the methods of problem-solving, seemed a good basis to guide the knowledge acquisition tools in their task. This set the stage for the research questions that we are exploring today: the exploitation of the properties of explicit problem-solving methods for (automated) knowledge acquisition.

3 How Does This Relate to Our Positions on Cognitive Science?

Initially, we aimed at developing descriptions of human reasoning, and transform these into operational systems. This was the purpose of the elicitation tools in KRITON. It left us with unstructured *heaps* of knowledge bits that each, in context, made a lot of sense. However, as decontextualized knowledge units, they are worthless.

Thus, we changed our positions, and moved from a cognitive-science oriented point of view to a stance that emphasizes the engineering aspect of the development of knowledge-based systems. We decided not to view the knowledge of a system as consisting of discrete, atomic elements that each had the property of being knowledge. We assumed Clancey's point of view, stating that *knowledge is something that an observer ascribes to a human agent in order to describe and explain recurring interactions that the agent has with its environment* [Clancey, 1989, p. 288]. For us this meant that the results of knowledge acquisition systems must be viewed in the context of the purpose of the agent. We are fully aware that we cannot capture the totality of this purpose. However, we can try to define certain frameworks explicating those parts of the purpose that we are aware of and that we can express.

The decision that we took, to frame the acquisition of knowledge with methods of problem-solving, is thus motivated by three arguments:

1. Knowledge is not composed of discrete elements that each on their own, independently of purpose, context, or interpreter are *knowledge.*

2. It is pointless to try to acquire knowledge that one cannot represent in a machine [Breuker and Wielinga, 1989, p. 267].

3. Knowledge elicitation, without the documentation of a purpose, gets stuck all too soon.

This entailed a change in focus in our project. We are now using the framework of a problem-solving method to give meaning to knowledge-bits such as rules, clauses, or frames; and we are using that framework to guide automated knowledge acquisition.

3.1 Where Does that Put Cognitive Science?

At first sight, one may think that cognitive science is out of the ball park now. This is not true at all. Even if today we focus on the engineering of intelligent systems, and even if we say that we are building these systems in ways that must not necessarily coincide with human mechanisms of intelligence, then the process of building these systems is nevertheless a process of *construing* the ways one or several intelligent agents solve the problems of a real-world task.

Thus our engineering implements must satisfy requirements from two sides: (1) they must be computer accessible, so that we can closely link their role in the process of *modeling-to-make-sense* [Clancey, 1989, p. 289] with their task in *modeling-to-implement-systems*; and (2) they must make sense as mediating devices in the social interaction processes between knowledge engineer and human expert(s).

We discuss the second point to explain in some detail why we still need cognitive science, even if we tackle the problem from an engineering point of view.

In a knowledge-acquisition situation observed behavior (e.g., thinking-aloud protocols, video footage, repertory grids or answers to focussed questions) is being analyzed and rationalized using knowledge-structuring primitives, such as rules, frames, knowledge sources, or tasks. This is a constructive process of building a model. The model is the result of construing behavior with the help of primitives aiming at making sense. Knowledge acquisition is a creative process of discussion between knowledge engineer and expert. The model is the result of this interaction.

To enable this process, the meaning of the knowledge-structuring primitives must be accessible to all participants of the discussion. Especially, they must be able to see that a model accounts for observed behavior. This is easier for the knowledge engineer than for the domain expert, as the primitives do have well-defined semantics in terms of the underlying operational knowledge-representation system. One cannot expect the domain expert to be a programmer. This implies that the semantics of the terms of the model must be intuitively accessible to the expert, that is, they must be adequate for the task.

To make the discussion more proficient, the meaning of the primitives should be such that a model can be tested against new situations, that is, that it can be validated. Potential contradictions between the model and a new situation lead to reconsideration, extension or structural changes of the model.

Furthermore, an ideal set of primitives should be such that a model provides guidance in the discussion, just as an agenda keeps track of unresolved issues. A formal analysis of a model might point to open ends, ambiguous or underconstrained decision points.

This implies that even though the model must not necessarily be a truthful reconstruction of human reasoning processes, the primitives that are used to build the models must in a certain sense be *cognitively adequate* to mold human knowledge and problem-solving processes, even if these are created in the actual knowledge engineering process.

3.2 Today's Situation

Today we are analyzing modeling frameworks like KADS, to find out whether they suffice these requirements. For example when re-modeling the cancer-chemotherapy administration task of ONCOCIN with KADS [Linster and Musen, 1992] we analyzed how KADS knowledge-structuring primitives (i.e., concept, instance, and relation on the domain layer; meta class, knowledge source, and inference structure on the inference layer; task and task structure on the task layer) can be used as rationalization tools for observed behavior (i.e., medical documents of cancer chemotherapy and knowledge structures of ONCOCIN). Furthermore we examined how KADS models guide the acquisition process, and in how far they support a constructive discussion between expert and knowledge engineer. An analysis of this modeling process, the cognitive processes and potential biases involved is presented in [Woodward, 1991].

To close the loop of conceptual modeling à la KADS (i.e., modeling to make sense) and modeling to implement systems several implementations have been developed, such as MODEL-K [Karbach *et al.*, 1991] or OMOS [Linster, 1991]. The operational modeling language OMOS bridges the gap between modeling as a process of making sense and modeling as a process of implementing knowledge-based systems, by providing KADS-oriented knowledge-structuring primitives for methods and domains. They are operational and provide immediate feedback for automated knowledge acquisition tools [Kühn *et al.*, 1991] (see Figure 1).

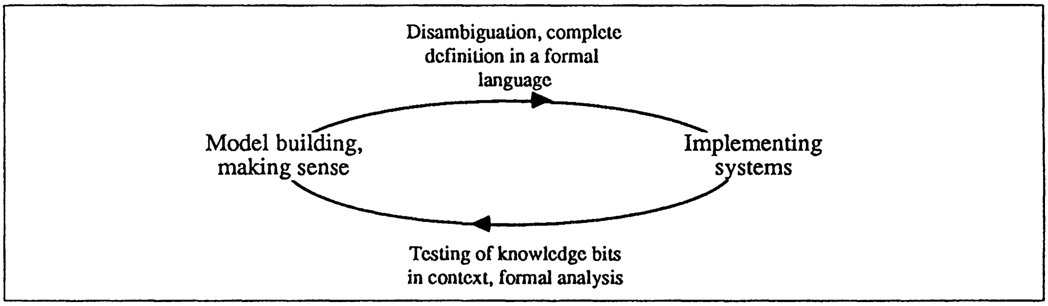

Disambiguation, complete definition in a formal language
Model building, making sense
Implementing systems
Testing of knowledge bits in context, formal analysis

Figure 1: The interaction between model building to understand ill-structured situations and behaviors, and modeling to implement systems.

Our recent work on OMOS and MODEL-K has emphasized the system-building aspect of knowledge engineering. We have been looking for knowledge-structuring primitives that are operational and that provide guidance in the construction process. We now must analyze these results again from the cognitive science point of view. We need to know in how far our terms are cognitively adequate building blocks for the modeling of expertise. Furthermore we want to analyze the knowledge-*creation* aspect of our current view on knowledge engineering. We want to know in how far a knowledge engineering process relying on these primitives creates the knowledge for a knowledge-based system, and how these knowledge-structuring primitives influence the discussion process between expert and knowledge engineer. This involves the analysis of the cognitive activities of knowledge engineering [Woodward *et al.*, 1992] in KADS-like frameworks, and the role of epistemic knowledge-structuring primitives in frameworks that were devised to analyze the model-building processes (e.g., [Wetter and Woodward, 1990]). This will help us answer questions like: What kind of knowledge can we acquire into our models? How will our models be biased? How can we use well-defined techniques from cognitive science to support model-building and the creation of operational systems?

Acknowledgements

This work has been influenced strongly by discussions led in the AI Research Division of GMD. Barbara Becker, Werner Karbach, and Angi Voß were involved in the development of many of the arguments of this paper. Thomas Christaller and Angi Voß commented on earlier versions.

References

[Boose, 1985] John H. Boose. A knowledge acquisition program for expert systems based on personal construct psychology. *International Journal of Man-Machine Studies*, 23:495 – 525, 1985.

[Breuker and Wielinga, 1989] 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*, Studies in Computer Science and Artificial Intelligence, pages 265 – 295. North-Holland, Amsterdam, 1989.

[Buchanan *et al.*, 1983] B. G. Buchanan, D. Barstow, R. Bechtal, J. Bennett, William Clancey, Casimir Kulikowsky, Tom Mitchell, and Donald Waterman. Constructing an expert system. In F. Hayes-Roth, D. Waterman, and D. Lenat, editors, *Building Expert Systems*, pages 127 - 167. Addison Wesley Publishing, London, 1983.

[Clancey, 1989] William J. Clancey. The knowledge level reinterpreted: Modeling how systems interact. *Machine Learning, Special Issue on Knowledge Acquisition*, 4(3, 4):285 - 292, 1989.

[Davis, 1982] R. Davis. Application of meta-level knowledge to the construction, maintenance. In R. Davis and D.B. Lenat, editors, *Knowledge Based Systems in Artificial Intelligence*. McGraw-Hill, New York, 1982. Doctoral dissertation, Computer Science Department, Stanford University.

[Diederich and Linster, 1989] Joachim Diederich and Marc Linster. Knowledge-based knowledge elicitation. In Giovanni Guida and Carlo Tasso, editors, *Topics in Expert System Design*, pages 323 - 352. North-Holland, Amsterdam, 1989.

[Ericsson and Simon, 1984] A. Ericsson and Herbert Simon. *Protocol-Analysis - Verbal Reports as Data*. MIT Press, Cambridge, 1984.

[Gaines and Shaw, 1981] Brian R. Gaines and Mildred L.G. Shaw. New directions in the analysis and interactive elicitation of personal construct systems. In Mildred L. G. Shaw, editor, *Recent Advances in Personal Construct Technology*, Computers and People, pages 147 – 182. Academic Press, London, 1981.

[Karbach *et al.*, 1989] Werner Karbach, Marc Linster, and Angi Voß. OFFICE-Plan, tackling the synthesis frontier. In Dieter Metzing, editor, *Proceedings of GWAI89*, volume 216 of *Informatik Fachberichte*, pages 379 – 387, Heidelberg, Septembre 1989. Gesellschaft fuer Informatik, Springer Verlag. Also published as WEREX-Bericht nbr. 23.

[Karbach *et al.*, 1991] Werner Karbach, Angi Voß, Ralf Schukey, and Uwe Drouven. MODEL-K: Prototyping at the knowledge level. In *Proceedings of the First International Conference on Knowledge Modeling and Expertise Transfer*, Sophia Antipolis, France, 1991.

[Kühn *et al.*, 1991] Otto Kühn, Marc Linster, and Gabi Schmidt. Clamping, COKAM, KADS and OMOS. In Duncan Smeed, Marc Linster, John H. Boose, and Brian R. Gaines, editors, *Proceedings of EKAW91*. University of Strathclyde, 1991. Also published as Technical Memo TM-91-03 of DFKI, Kaiserslautern.

[Linster and Musen, 1992] Marc Linster and Mark Musen. Use of KADS to create a conceptual model of the ONCOCIN task. *Knowledge Acquisition*, Special Issue on KADS, 1992.

[Linster, 1989] Marc Linster. Towards a second generation knowledge acquisition tool. *Knowledge Acquisition*, 1(2):163 - 183, 1989.

[Linster, 1991] Marc Linster. *Knowledge acquisition based on explicit methods of problem-solving.* PhD thesis, University of Kaiserslautern, Kaiserslautern, 1991. Submitted.

[Morik, 1987] Katharina Morik. Acquiring domain models. *International Journal of Man-Machine Studies*, 26:93–104, 1987.

[Musen *et al.*, 1987] Mark A. Musen, L. M. Fagan, D. M. Combs, and E. H. Shortliffe. Use of a domain-model to drive an interactive knowledge-editing tool. *International Journal of Man-Machine Studies*, 26:105 - 121, 1987.

[Shadbolt, 1992] Nigel R. Shadbolt. Facts, fantasies and frameworks: The design of a knowledge acquisition workbench. In Franz Schmalhofer, Gerd Strube, and Thomas Wetter, editors, *Contemporary Knowledge Engineering and Cognition*, Lecture Notes in Computer Science. Springer, Heidelberg, 1992.

[Wetter and Woodward, 1990] Thomas Wetter and Brian Woodward. Towards a theoretical framework for knowledge acquisition. In John H. Bosse and Brian R. Gaines, editors, *Proceedings of the 5th Banff Knowledge Acquisition Workshop*, pages 35/1 - 35/25, Calgary, 1990. AAAI, University of Calgary.

[Wielinga *et al.*, 1992] Bob Wielinga, Guus Schreiber, and Jost Breuker. KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition*, Special Issue on KADS, 1992.

[Woodward *et al.*, 1992] Brian Woodward, Mildred Shaw, and Brian Gaines. The cognitive basis of knowledge engineering. In Franz Schmalhofer, Gerd Strube, and Thomas Wetter, editors, *Contemporary Knowledge Engineering and Cognition*, Lecture Notes in Computer Science. Springer, Heidelberg, 1992.

[Woodward, 1991] Brian Woodward. Developing K-ONCOCIN: A case study in the cognitive processes of knowledge engineers. In John H. Boose and Brian R. Gaines, editors, *Proceedings of the Knowledge Acquisition Workshop 91 (KAW91)*, Calgary, 1991. AAAI, University of Calgary.

Two Questions from Expert System Developers to Cognitive Scientists

Frank Puppe, Ute Gappa

Universität Karlsruhe
Institut für Logik, Komplexität und Deduktionssysteme
Postfach 6980, W-7500 Karlsruhe
Germany

Abstract: (1) Are the well-known "strong" problem solving methods in expert systems cognitively adequate enough for the experts which have to formalize their knowledge accordingly? and (2) How significant are adequate graphical representations offered by some knowledge acquisition tools for the internal model of the experts?

In this paper we concentrate on one of the many forms of cooperations between the displicines of knowledge engineering and cognition: Proposals from the knowledge engineering field should be evaluated with respect to their cognitive validity. In particular, we exemplify this approach by asking two questions: about the cognitive significance of strong problem solving methods and of graphical display forms of the knowledge. Both questions reflect basic approaches of our research. We also give some reasons, why we are asking these questions. However, we don´t provide any answers.

There are three basic knowledge acquisition types for expert systems:

- *Indirect knowledge acquisition:* A "knowledge engineer" acquires the knowledge from experts ("knowledge holders") and formalizes it for the expert system.
- *Direct knowledge acquisition:* The knowledge holders formalize their knowledge by themselves.
- *Automatic knowledge acquisition:* The knowledge is transformed automatically from already existing knowledge (e.g. from the literature or from cases).

Indirect knowledge acquisition is susceptible to communication problems and quite expensive, in particular with respect to the maintenance of the knowledge bases. High quality automatic knowledge acquisition is impossible with the current state of the art. Therefore, direct knowledge acquisition seems to be the most promising path. It has also the motivational advantage, that the knowledge holders are given complete authorship and reputation for their knowledge bases. However, direct knowledge acquisition can only succeed, if the expense of the knowledge holders for learning and using given knowledge acquisition and maintenance facilities remains acceptable low. This requires:

- cognitive adequacy of the underlying problem solving methods,
- simple to use and effective knowledge acquisition tools.

Although it is not necessary, that the problem solving methods of expert systems are those of the experts, the methods should be at least easily understandible. Otherwise experts will have much difficulty to express and formalize their knowledge. Therefore, an exchange with cognitive science could be quite useful: On the one hand, the existing problem solving methods should be tested for their cognitive plausibility, and on the other hand, constructive results from cognitive science could lead to the development of easy to understand problem solving methods.

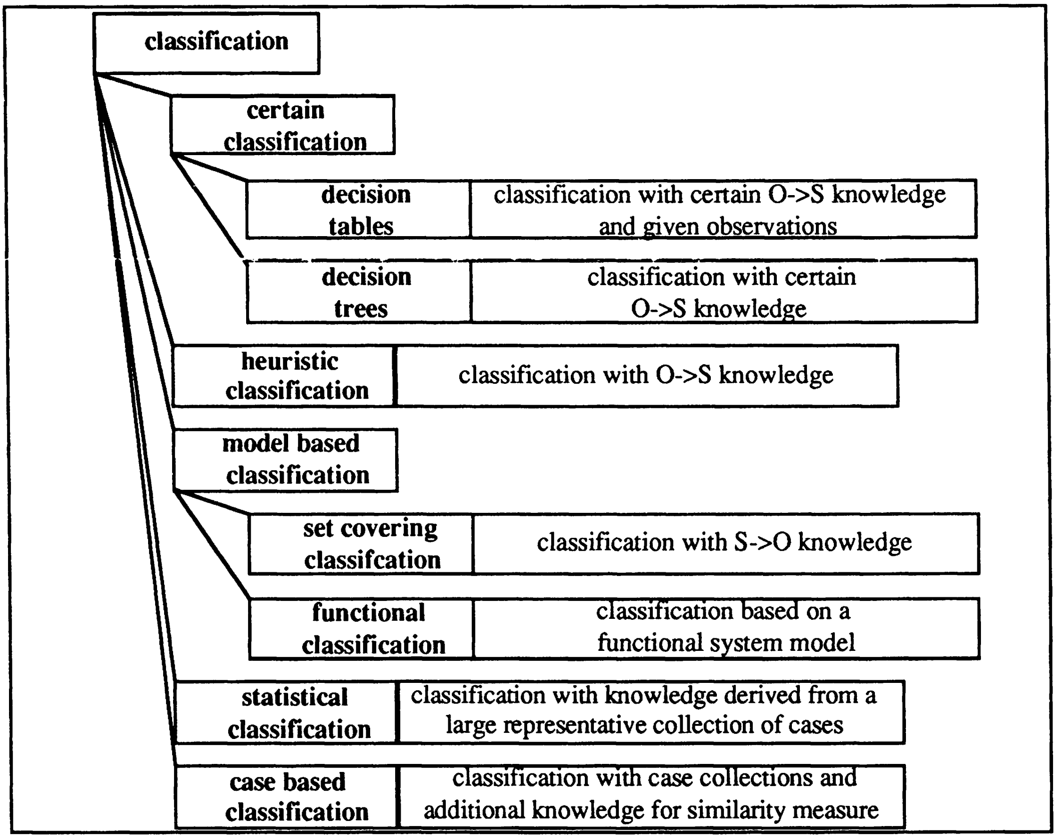
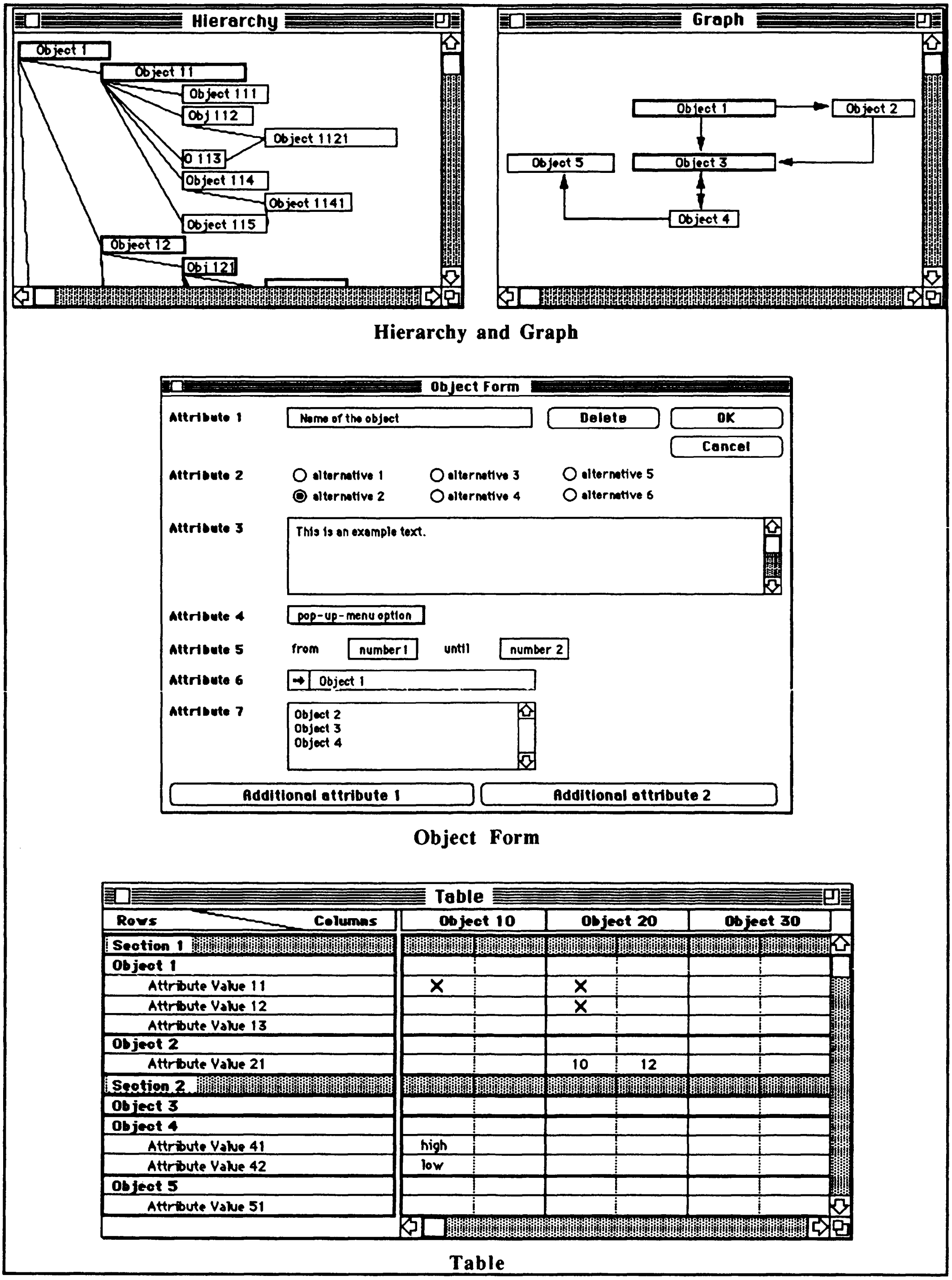

classification
certain classification
decision tables
classification with certain O->S knowledge and given observations
decision trees
classification with certain O->S knowledge
heuristic classification
classification with O->S knowledge
model based classification
set covering classifcation
classification with S->O knowledge
functional classification
classification based on a functional system model
statistical classification
classification with knowledge derived from a large representative collection of cases
case based classification
classification with case collections and additional knowledge for similarity measure

Fig. 1: Overview on well known classification problem solving methods (from [Puppe 90]). Abbrevations: O = Observations; S = Solutions.



Hierarchy and Graph

Attribute 1 Name of the object

Attribute 2 ○ alternative 1 ● alternative 2 ○ alternative 3 ○ alternative 4 ○ alternative 5 ○ alternative 6

Attribute 3 This is an example text.

Attribute 4 pop-up-menu option

Attribute 5 from number1 until number 2

Attribute 6 → Object 1

Attribute 7 Object 2
Object 3
Object 4

Object Form

Rows Columns	Object 10		Object 20		Object 30	
Section 1						
Object 1						
Attribute Value 11	X		X			
Attribute Value 12			X			
Attribute Value 13						
Object 2						
Attribute Value 21			10	12		
Section 2						
Object 3						
Object 4						
Attribute Value 41	high					
Attribute Value 42	low					
Object 5						
Attribute Value 51						

Table

Fig. 2: Generic graphical primitives (from [Gappa 91])

Examples for such an exchange are the psychological studies about the decision making process of physicians [Elstein 78, Kassirer 78, 82, Feltovich 84]. Among other things, they revealed the consistent use of the hypothesize-and-test strategy (early generation and goal-directed evaluation of diagnostic hypotheses) and of the differential-diagnosis strategy (taking into account always several competitive hypotheses simultaneously) by physicians. In accordance with the progress of expert system research, it would be desirable to test complete problem solving methods, of which the above mentioned strategies are only parts. An overview on well known classification problem solving methods is shown in Fig. 1 from [Puppe 90], where the methods also are described in detail.

One precondition for examining hypothesized cognitive models of experts is the formalization and implementation of their problem solving models, which we tried in our classification expert system shells MED2 resp. D3 [Puppe 87, D3 91].

However, adequate problem solving methods are only part of the story. In addition adequate, maybe graphical knowledge acquisition tools are needed in order to help domain experts to understand the underlying problem-solving model and to structure and formalize their knowledge by themselves with only limited assistance from "knowledge engineers". Graphical knowledge acquisition environments should allow to directly enter knowledge in specialized versions of basic graphical primitives like hierarchies, graphs, forms and tables shown in Fig. 2 from [Gappa 91], where the primitives are also described in more detail. Our other question to cognitive scientists concerns the significance of such graphics for the internal model of the experts. Do they normally visualize their knowledge in such graphical representations and are there other basic primitives than those in Fig. 2?

References:

[D3 91] Bamberger, S., Gappa, U., Goos, K., Meinl, A., Poeck, K., and Puppe, F.: The Diagnostic Expert System Shell D3, Manual, Version 1.0 (in German; translation to English in preparation), Universität Karlsruhe, Institut für Logik, Komplexität und Deduktionssysteme, 1991.

Elstein, A., Shulman, L., and Sprafka, S.: Medical Problem Solving, Harvard Univ. Press, 1978.

Feltovich, P., Johnson P., Moller, J., and Swanson, D.: LCS: the Role and Development of Medical Knowledge in Diagnostic Reasoning, in Clancey, W. und Shortliffe, E.(eds.): Readings in Medical Artificial Intelligence, Addison-Wesley, 1984 (1980).

Gappa, U.: A Toolbox for Generating Graphical Knowledge Acquisition Environments, in: Proc. of The World Congress of Expert Systems, Vol. 2, 797-810, Pergamon Press, 1991.

Kassirer, J., Kuipers, B., and Gorry, G.: Towards a Theory of Clinical Expertise, American Journal of Medicine 73: 251-259, 1982.

Kassirer, J. and Gorry, G.: Clinical Problem Solving: a Behavioral Analysis, Annals of Int. Med. 89, 245-255, 1978.

Puppe, F.: Requirements for a Classification Expert System Shell and Their Realization in MED2, Applied Artificial Intelligence 1: 163-171, 1987.

Puppe, F.: Problem Solving Methods in Expert Systems (in German; translation to English in progress) Springer, 1990.

The Cognitive Basis of Knowledge Engineering

J. Brian Woodward, M.L.G. Shaw, and B.R. Gaines
Knowledge Science Institute
Department of Computer Science
University of Calgary
2500 University Dr. NW.
Calgary, Alberta
CANADA, T2N 1N4

e-mail: woodward@cpsc.ucalgary.ca

Abstract. The goal of knowledge engineering is to create an artificial system which reflects knowledge-like qualities. Current tools, techniques and procedures in knowledge engineering concentrate on the elicitation and representation of knowledge structures. This concentration of effort reflects the current emphasis on the epistemological and computational/representational characteristics of knowledge engineering. A different, yet complementary perspective is offered in this paper. Knowledge engineering is defined as a human activity system, characterized as a cognitive environment or network which deals with complex epistemological domains. Rather than viewing knowledge engineering as entirely concerned with knowledge content, those processes which produce the knowledge in the knowledge engineering environment are viewed as the focus of attention. Memory, judgment and choice, text comprehension, and social cognition and communication represent a selection of cognitive science domains which offer research findings of importance for knowledge engineering. Based on these research findings, the groundwork is laid for the development of cognition-support tools for knowledge engineering.

1. Introduction and Purpose

The domain of knowledge engineering has focused on the acquisition and modelling of knowledge. This is not a surprising statement but it raises an important issue which this paper addresses. The terms ‘acquisition’ and ‘modelling’ would first suggest an emphasis on the processes and procedures related to knowledge engineering but a closer look suggests that the focus is rather on the outcome of these processes. The ‘knowledge’ focus dictates that whatever processes are developed, or adapted, they must result in structures which somehow ‘capture’ or ‘represent’ identified forms of ‘knowledge’. The tools, techniques and procedures developed to acquire knowledge are classified as either ‘domain-specific’ or as ‘generic’. The term ‘domain- specific’ denotes a tool, technique or procedure which has developed out of knowledge engineering within a defined domain (eg. Opal: see Musen, 1989). The content, or the specific knowledge structures of the domain, guide and determine the tool, technique or procedure to be used. Generic tools (eg. KSSO: Gaines, 1988) are those which are considered to be useful and appropriate in a wide variety of domains. In this case, the tool guides the type of content which will be specified from the domain. *The results of these tools are knowledge-specific structures.*

This paper presents arguments and research support for the development of a different category or class of knowledge engineering tool: cognition-based tools. This type of tool is characterized by a de-emphasis of the 'knowledge' itself with a greater emphasis on the cognitive activities abundant in knowledge engineering activity which produce knowledge structures. These cognitive processes inherent in the knowledge engineering process establish the cognitive basis for knowledge engineering. The selection is based on sites of cognitive activity not solely on domain tasks or knowledge centred generic processes. Rather than emphasizing processes which identify domain tasks and knowledge structures, this approach emphasizes the support for the cognitive process activity in knowledge engineering.

Knowledge engineering is viewed as a body of activity which is characterized by intense cognitive activity as well as a strong emphasis on knowledge structures. Section 2 presents the main premises for viewing knowledge engineering from a cognitive processing perspective rather than from a knowledge perspective. Knowledge engineering is defined as a human activity system consisting of different sites and levels of cognitive activity. The cognitive processes occurring at each site and between sites constitutes part of the cognitive environment. The focus of this paper is on the cognitive processes of the knowledge engineer. Section 3 presents research on the processes involved in understanding new, complex domains (a usual experience for knowledge engineers). Sections 4 through 7 present cognitive research relevant for knowledge engineering: memory, text comprehension, judgment and choice, and social cognition.

2. Knowledge Engineering Viewed as a Collection of Cognitive Processes

2.1 The Current Focus in Knowledge Engineering

The domain of knowledge engineering reflects a strong emphasis on the acquisition and modelling of knowledge structures and the processes to manipulate and transform them. These aspects may be categorized as the epistemological nature of the domain. Epistemological and accompanying ontological discussions are concerned with the concepts selected to express the various structures for acquisition and modelling. This emphasis reflects the importance of concepts like the 'knowledge level' (Simon, 1988) and the research on 'expertise' and expert knowledge (eg. Johnson, 1986). Results of these studies centre predominantly on expert-novice differences in conceptualizing the domain problem, in the knowledge structures used in finding a solution, and, to a lesser extent, in the different methods used to solve the problems.

Discussions concerning the nature of knowledge also characterize the domain of knowledge engineering. These discussions include the role of different types or forms of knowledge in meeting tasks demands. The KADS (Wielinga, et al, 1989) methodology in knowledge acquisition is based on a strong epistemological foundation. A parallel approach to the development of re-useable problem solving structures and processes is represented by role-limiting methods (see Marcus, 1988) and generic tasks (Chandrasecaran, 1988). This research points to the need for identifying usable knowledge structures and the processes by which these structures are manipulated.

A related issue is that of knowledge representation. Representation frameworks abound and they will not be discussed here (see Brachman and Leveque, 1985). The discussion of forms and structures of knowledge logically lead to the issues of how to represent that knowledge in a form useful for modelling. Selected epistemological constructs act as the basis for knowledge structures and are used to impart meaning. Formal languages and representational frameworks determine what is computationally tractable. Epistemic concepts are used to develop formal structures and these structures are used to order and arrange the knowledge of a given domain.

2.2 The Cognitive Aspect of Knowledge Engineering

Knowledge engineering activity has been largely focused on the epistemological and representational aspects of knowledge. A different, yet complementary view would identify knowledge engineering as the interaction of cogniting agents with the emphasis on the cognitive processes used in knowledge engineering. The knowledge-intensive approach in current knowledge engineering forces us to view the outcome or the product of our efforts as most important. The 'knowledge' is 'acquired', 'transferred', 'captured' and 'modelled'. This emphasis directs our attention away from the processes that 'produce', 'organize', and 'represent' the knowledge: away from notions of learning, comprehending, and communicating. Perhaps these processes are taken for granted; perhaps they are considered invariable. This emphasis is clearly displayed in our tools, techniques and procedures in that they focus on structures of knowledge and the processes used to manipulate them. Few of our tools address the complexity of the cognitive processes used to develop the knowledge structures although many of our knowledge acquisition tools support them.

Perhaps another reason for our present emphasis is the inconclusive nature of cognitive processes. Perhaps we understand these processes far too little or view them as problems and sources of noise to use them effectively in our work in knowledge engineering. We try to avoid the biasing nature of the knowledge engineer's presence in the development of knowledge-based systems. Our approach is to develop methods to 'capture' the expert's knowledge automatically and directly, thus eliminating the influence of the knowledge engineer. This view might alter if cognitive processes were viewed as a necessary component to knowledge engineering in that we cannot fully understand knowledge and its re-creation if we fail to understand those cognitive processes involved.

Wetter and Woodward (1990) have identified the importance of incorporating an understanding of cognitive processes in the development of a theory of knowledge acquisition. In order to develop a method of knowledge acquisition in a principled manner, the epistemic concepts must be clearly defined, the epistemic concepts must have a basis in psychological occurrence, and the representational formalism must reflect the intended epistemic and psychological meaning.

Rather than viewing the domain of knowledge acquisition from the epistemological or representation perspective, this paper addresses the domain from the cognitive perspective. This perspective is viewed as complementary to the epistemological and representational ones in that it helps to complete the picture. The cognitive perspective is characterized by an emphasis on cognitive processes, their identification and support, rather than on the identification and production of knowledge structures. The question here is not 'Where's the knowledge ?' but 'What cognitive processes produce the knowledge?'

2.3 Knowledge Engineering as a Set of Cognitive Processes

This section sets out the premises for viewing knowledge engineering as a set of cognitive processes rather than as a set of activities which acquires and models knowledge in one form or another.

First, knowledge engineering represents an ordered collection of cognitive activities. From an information processing point of view, cognitive activities are explained as processes of a "physical symbol system (eg. human brain) consisting of a representation system and the processes to manipulate it" (Aitkenhead and Slack,1985). This is a broad definition which includes human cognition as well as physical symbol systems which display cognition-like processes. Cognitive activities may be ordered and contained within one physical symbol system or shared between interacting physical symbol systems.

Second, the purpose of knowledge engineering is to use cognitive processes to produce a model which demonstrates cognitive properties. This premise suggests a comparison point between the goals of cognitive psychology and those of artificial intelligence in that the outcomes of each are considered models. Aitkenhead and Slack (1985) point out that in those disciplines concerned with cognitive processing and modelling the formulation of information processing models are evaluated against a body of experimental data. In artificial intelligence, the goal is to build computer-based models of performance which are evaluated against criteria such as computational efficiency and logical coherence. The focus of the former is on the final model and its ability to account for evidence of cognitive structures and processes. The focus of the latter is on the computational attributes and usefulness of the final model. Viewing knowledge engineering as a set of cognitive processes means that we emphasize the information processing activities which lead to a final model rather than the characteristics and content of the final model.

Third, knowledge engineering is done in a cognitive environment. The environment defines a set of situations in which information is generated and produced, provided, manipulated, organized, re-organized and re-represented. The environment also defines those processes and structures which are used to generate, manipulate, comprehend, and organize the information. These processes are concerned with meaning, not just information. In this sense, the environment we are defining here might be called a 'knowledge' environment but that would suggest we are interested more in the outcome of the meaning or in asking 'what is the meaning' of a certain piece of information. However, we are more interested in asking 'how is the meaning' of a certain piece of information determined. Consequently, our interest is on the cognitive processes of developing meaning. In this environment, humans are involved and they supply many of he cognitive processes.

Fourth, the cognitive environment is characterized by sites and levels of cognitive activity. The environmental structure determines the locations of activity and the types of cognitive activity which occur at those locations. Sites refer to those clearly distinguishable points of cognitive activity (eg. the knowledge engineer, expert). Level refers to various combination of sites which are in interaction. Interaction of cognitive processing sites entails another level of cognitive activity which is distinguishable from cognitive activity occurring at a single site. For example, single site activity (eg. the expert) may reflect processes leading to making a choice about a course of action. Interaction of sites requires the use of social cognition processes. Presently, our field is much more fascinated with the cognitive processes of the expert but less so with those of the knowledge engineer and the user. In fact, we try to eliminate the cognitive effect of the former and ignore that of the latter. Why do we wish to develop our understanding of cognitive processes at one significant site in knowledge engineering and not others ? The expert's cognitive processes as well as those of the knowledge engineer and the user are open for our observation and study because they a part of the knowledge engineering environment. Addressing cognitive processes separately and in interaction may bring us better understanding of the knowledge engineering process.

Fifth, an understanding of the cognitive environment acts as a basis for developing tools, techniques, and procedures to support and/or replace these processes. With our emphasis on the cognitive aspects of knowledge engineering, the focus of potential support processes is based on our understanding and careful elucidation of those cognitive processes which are characteristic of sites and levels of cognitive activity. Rather than emphasize knowledge support we are suggesting a complementary view of cognition support as a basis for tool, technique, and procedure development. For example, in addition to asking questions based on the domain content and problem-solving heuristics, it is useful to ask questions which are formed to reflect different cognitive processes (eg. LaFrance, 1986).

3. Cognitive Characteristics of the Domain of Knowledge Engineering

3.1 Understanding Complex Domains

The role of the knowledge engineer is cognitively demanding in that the knowledge engineer is expected to enter into the epistemological structure of a domain far enough to build a model of that domain. Even if the domain is restricted in scope, the knowledge engineer must become familiar with the problem-solving structures in the domain and the underlying assumptions of the domain of interest. These demands are similar to expert development within that domain in that the knowledge engineer, as a novice, must quickly become familiar with the necessary epistemological concepts and problem-solving processes in order to adequately model the domain or part thereof. When a knowledge engineer enters a new, complex, epistemological domain, he/she uses a variety of cognitive processes to first understand and comprehend the basic structures and assumptions of the domain and second, to model this understanding. It is these cognitive processes which require identification and description so that tools, techniques, and procedures may be developed as aids and supports to the knowledge engineer alone or in interaction with the expert.

The development of problem-solving skill in medicine acts as a case in point. Research on how medical students and interns come to develop expertise in a complex epistemological domain parallels the cognitive demands placed on the knowledge engineer when he/she enters a new domain. Evans (1989) stresses that addressing the complex domain of medical problem-solving reflects four critical issues:

> "1)the relation between domain knowledge and problem-solving behaviour; 2) the sources of bias and misconception in problem representation and decision-making; 3) the role of discourse in structuring the acquisition of data..., and 4) the identification of formal methods for the evaluation of performance" (p. 2).

These same issues are of importance to the knowledge engineer and to knowledge engineering activity. The point here is that the knowledge engineer is required to learn something about the domain in order to make knowledge engineering decisions. The cognitive demands placed on a knowledge engineer when he/she enters a new domain are similar to novices in that domain who are in the process of developing a greater understanding of the domain. The rest of this section presents research support for a number of cognitive processes involved when an individual is developing understanding in a complex domain.

3.2 Cognitive Processes in Understanding

The biomedical domain is characterized as epistemologically rich. Many of the concepts and families of concepts are highly interdependent, forming extremely complex epistemological networks. Meaning and understanding of the concepts alone as well as the interactions of these concepts are necessary for understanding the domain. Understanding is often hampered not only by the complexity but by the ill-structured nature of the domain. A number of researchers have reported the difficulties of understanding complex systems

(Flood, 1987) as well as the requirements and methods for overcoming these difficulties (Cairns and Woodward, 1988). The novice and the knowledge engineer are similar when they embark on conceptual analyses of clusters of complex concepts.

Patel et al (1989) identified a number of cognitive processes used in developing a functional relationship between biomedical knowledge and clinical reasoning. The following cognitive processes were viewed as necessary for developing an understanding of how to incorporate models of biomedical knowledge in diagnosing clinical cases. These processes included comprehension of text-based propositions, construction of consistent and coherent explanations, and selection of relevant and irrelevant facts based on clear discrimination principles. These findings are based on novice-expert interactions and they represent points of distinction between novice and experts in the domain. Knowledge engineers resemble novices (or even sub-novices) when they first enter a new domain. The cognitive processes used by novices, which distinguish them from experts, are identified as correlates of poor understanding, diagnostic performance, and explanatory performance.

Another set of studies (Feltovich et al, 1989, 1984; Spiro et al, 1988, 1990) addresses the nature of conceptual understanding in biomedicine. More specifically, the studies address how deep models of complex ideas develop and how problems may arise with these models, during their development, which lead to misconceptions. The results have shown developmental patterns for a variety of biomedical misunderstandings. The results support three main reasons for misunderstandings: multiplicity, interdependency and oversimplification. The first refers to the fact that many influences contributed to the acquisition and maintenance of misconceptions such as learner style and ability, the methods of education, and practices of biomedical science research. The second characterizes misconceptions as reciprocating networks based on basic component ideas which are faulty. These networks tend to support each other or positively reinforce each other so that misconceptions are promulgated throughout the network. The final pattern suggests that due to human information processing capacity, the complexity of the concepts is reduced by selecting, collapsing or relating concepts in a way which changes the meaning.

3.3 Cognitive Demands on the Novice

Complex concepts make unusual cognitive demands on the novice. Feltovich et al (1989) describe four main demands on:

1. working memory because of the large number of nested or looped steps or goals, or because of the large number of variables to be considered, processed, and/or reconciled;
2. formal representation in the sense that the degree of abstraction necessary for understanding is often onerous (eg. representing the concept of rate or acceleration);
3. intuition or prior knowledge in the sense that new concepts can conflict with prior meanings and/or relationships among concepts; and
4. notions of regularity because many concepts can be ill-structured and highly variable in how they are used or they may be very strongly dependent on other concepts for their meaning.

These same demands are placed on the knowledge engineer. These cognitive demands affect knowledge engineering activity by leading to misconceptions which may be perpetuated into the final system if not ‘caught’ by the expert (assuming there exist specific procedures for the expert to use to ‘catch’ the misconception). Also, when complex conceptual domains are addressed, complexity reduction occurs. Many researchers have studied the processes whereby humans use particular procedures to reduce the complexity of conceptual information. (Kahneman and Tversky, 1973; Kahneman et al, 1982; Coulson et al, 1986; Spiro, 1980).

Feltovich et al (1989) distinguishes conceptual bias from those biases in judgment and decision-making (Kahneman et al, 1982) in that the family of 'reductive biases' are used solely to reduce complexity rather than reflect judgment processes. Six reductive biases were identified. 'Static' bias occurs when a dynamic system is viewed as a static model. 'Step-wise' bias describes the situation where a continuous process is broken down into discrete, identifiable steps resulting in a loss of meaning. 'External agent' bias is the attribution of essential intrinsic characteristics of entities or processes to external influences. 'Prior analogy' bias occurs when new concepts are given meaning based on already held concepts or simple models within the domain or from other domains. 'Common connotation' bias occurs when technical terms are reduced in meaning based on the day-to-day use of the term or concept. Finally, 'restriction of scope' bias refers to the belief that general principles are taken to apply only under certain circumstances based on a repeated co-incident relationship.

3.4 Methods of Information Presentation

One important issue addressed in the research in the biomedical domain is the need to select the 'right' concepts for presentation (Feltovich et al, 1989). Due to the complex, ill-structured nature of certain facts of a domain, the tendency is to present information initially in a simple overview format and then to incrementally increase the complexity so that models are built to a point of greater sophistication (Glaser, 1984). The reason to select the right concepts is due to a perceived failure of this incremental approach. These researchers found that students used the simplified models to 'filter-out' or arrange later material so that developing an understanding of the complexity did not occur. Also, they found that the initial simplification promoted further simplification of ideas. The tendency was to under-dimensionalize. The recommendation is to select key concepts, to ensure that they are understood, and then to build around the key concepts. Criteria for selection of these key concepts are the perceived importance of the concept by the community, the degree of centrality of the concept in the literature and for clinical cases, a high degree of difficulty in understanding the concept, and the concept has cross-context application.

A second issue emerging from the biomedical research suggests the use of abnormality models of information processing. This type of model is extremely useful in identifying, describing and correcting processes which result in some form of misunderstanding. These models frame a conceptual misunderstanding in terms of the processes used to develop it. These types of models appear to have a real application in knowledge engineering. Often the knowledge engineer needs to use the expert as the best guide to conceptual verification; however, only content is addressed. Procedural models of common processing biases or distortions (abnormalities) might be used by the knowledge engineer in verification of knowledge-based systems.

3.5 Summary

The biomedical research demonstrates the value of studying the cognitive processes of learning in the context in which they occur. In most cases the research was carried out in the classroom and teaching hospitals associated with medical schools. If we are to learn more about the cognitive processes involved in knowledge engineering, the actual context becomes the laboratory.

Also, the research presented from the biomedical field provides an example of the importance of understanding the cognitive processes involved when an individual attempts to comprehend a complex epistemological domain. Knowledge engineers are constantly confronted with this task and the cognitive demands placed on them are onerous. This research has indicated that identifiable cognitive processes are involved in the understanding of complexity and has provided some direction for the development of cognition support tools, techniques and procedures for knowledge engineering.

The following sections of the paper identify and describe other sets of cognitive processes apparent in knowledge engineering activity.

4. Memory

4.1 Types of 'Memory'

Commonly agreed upon mechanisms of memory (Potter, 1990) support memory as a three phase phenomenon. Processes are postulated for information registration or encoding, for retrieval or remembering, and for forgetting. Temporal architectures have also been postulated (Crowder, 1982). Gorfein and Hoffman (1987), Klatzky (1980), and Crowder (1976) provide good evaluations of the memory research. For knowledge engineering, specific issues and questions arise from this research.

Snodgrass (1989) asks 'How many memories?'. This question was originally asked by Tulving (1985) but Snodgrass developed a memory arrangement based on 3 types of memory tasks. The first is called 'episodic' memory which is essentially 'remembering that' something had occurred or that something was said. The memories reflect traces of autobiographical events, important to the individual and retained by the individual because of some relevance pointer. The second memory task is that of 'semantic' memory or 'knowing that' something is true or false or something is known or believed to have occurred. This type of memory is more encyclopedic and is indexed by a number of conceptual pointers so information is retained because it points to some other piece of information. The third memory is that for 'knowing how'. Snodgrass describes this memory as a 'procedural' memory in that what is retained is a perceptual/motor sequence of actions which have been acquired and demonstrated by completing actions, by doing.

The concept of multiple memories suggests that events, occurrences and information are, in some manner and form, 'retained' by the individual in different memory 'locations'. The processes of representing and retaining information in each of the three memories are considered different tasks, but Snodgrass admits to knowing little about the degree of independence or interdependence among the memory types. The concept of different memories also brings up other questions: under what circumstances are the different memory tasks activated (eg. recall, cued recall, recognition tasks), how is information in each memory represented, and what processes are used to access the different memories? Answers to these questions may lead directly to processes which may be supported in the knowledge engineering process.

4.2 Environment X Memory Research

Smith (1988) presented anecdotal evidence for episodic memories which were cued or triggered by environmental stimuli. Bjork and Richardson-Klavehn (1989) present research on the context-memory relationship directly. They studied aspects of various contextual influences on how information is initially retained and how information is available, based on contextual retrieval cues. They contend that generating ideas or becoming aware of new information is done within a defined, current, episodic context. Features of the context are incorporated with the new information in memory. Recall probabilities are seen as a function of the strength of association to the current context. Matching episodic contexts ensures better recall of information.

It is necessary also to distinguish between aspects of the episodic context that are meaningfully related to the information retained and those aspects of the context which are independent and incidental to that information. Baddeley, (1982) distinguished 'interactive' and 'independent' forms of contextual information or patterns. 'Integrated' and 'isolated' distinctions were made by Eich (1985) and Hewitt (1980) categorized contextual patterns as 'intrinsic' or 'extrinsic' to the meaning of the retained information. Bjork and Bjork (1988) proposed three aspects of context. That information which was clearly and strongly associated with the target information was labelled 'integrated' information. 'Influential' information was that which had strong associations to the target information in the sense that it influenced the degree of meaningfulness of the target information. "Incidental' information was not only independent or isolated from the target information but did not influence the interpretation of that information. These three distinctions are used to describe context-target relationships. Finally, these researchers postulate two types of cognitive processing during recall of information. The data driven process (also see Jacoby, 1983) posits a type of task which forces the individual to respond to perceptual information only (eg. identify words or word fragments or supply missing structures). This is usually considered an indirect form of memory. Conceptually driven tasks (see Roediger and Blaxton, 1987) demand constructive, semantically-based processes as a basis for direct recall of information. These three main dimensions (concept-target relationships, type of context, and type of recall task) form a taxonomy of recall which suggests 12 different methods of structuring a recall situation to increase the effectiveness and efficiency of recall activity.

Memory recall situations are common ones in knowledge engineering activity in the knowledge acquisition phase. The concept of multiple memories has application (eg. La France, 1986) in that structured recall situations established by the knowledge engineer affects the retrieval processes of the expert. Also, structures such as the Bjork and Richardson- Klavehn's taxonomy and can act as a basis for developing support tools and procedures to aid in recall for the expert, the knowledge engineer and even for the user.

4.3 Retrieval and Organizational Strategies

Kolodner (1984) outlines a number of principles of memory retrieval and storage. Those of particular interest to knowledge engineers are presented and discussed. The first principle states that human remembering is often a process of re-construction. An individual must re-assemble or recreate the necessary structures and events which 'must have happened' rather than directly retrieving what actually did happen. The 'actual' event, with all its salient features, is not 'stored' in memory. When asked to recall a situation, an individual will produce a cogent assembly of plausible items. A second principle suggests that the process of remembering mirrors a progressive narrowing or focusing-in on a description of the recalled event. In other words, an individual begins with salient features (as perceived in the recall context) and begins a type of choice or selection task in order to eventually select the full event (for reconstruction) to meet the current recall demands (eg. E-MOPS are considered by Kolodner to be the basic structures which organize events in memory).

The issue of contextual information is raised in a third principle. In order to retrieve the appropriate information from memory, an individual first requires some information or knowledge about the contexts associated with the target items. The original context is seen as having a powerful link with other salient features of information (previously discussed). A fourth principle suggests that retrieval often requires a search for something other than what was requested. In order to successfully retrieve and re-construct an event or episode, it is often necessary to locate intermediate or ancillary information which may act as indices for the required information. Finally, a fifth principle suggests that conceptual categories contain or hold generalized information only and that details may be generated at the time of recall.

Each of these principles raise issues of interest for the knowledge engineering environment. Naive assumptions about what an expert offers in response to questions lull the knowledge engineer into believing that he/she has 'captured' the necessary knowledge from the expert. Tools, techniques and procedures based on these principles may aid the knowledge engineer to clearly identify what knowledge has been gained, what further knowledge is necessary and what knowledge is suspect. The effects of the knowledge engineer's questions are not studied yet this process is the main method for gathering expert information.

A more detailed analysis of retrieval processes is necessary to identify supportable processes. Kolodner (1984) suggests two main types of retrieval activity. Processes for constructing and further specifying the context for search represent the first type. She calls these processes 'executive' strategies. 'Instantiation' strategies represent the second type of process. These processes direct search and the application of constructive strategies. Elaboration, transformation, and relation have been identified. One event may enable another, act as a precondition for another, may result from another, act as a reason for another. An event can be linked to another event in a larger episode of events, related in a sequence of events, act as a preceding or following event or act as a standard event. Similar relationships have been distinguished by Schank (1975) and by Graesser and Clark (1985).

Kolodner (1984) also discusses the important aspect of event 'features'. These features are aspects of events which are associated with the target information and unique features usually make the best indices for memory retrieval strategies. Also, good features are able to relate events and provide context relationships. The indices are then used for searches. Access to information retained by an individual can be more efficient through the use of key features. One method of retrieval is through responses to questions (eg. Lehnert, 1978) which are based on key features and which emphasize the appropriate format or form of the answer as opposed to the content of the answer. Good answers are seen as responding to the intent and content of the question. Using the model of instantiations (Kolodner, 1984), experts may be given tasks where they are required to identify a variety of relationships between events as well as the features of events.

Transformational processes Kosslyn (1980) themselves deserve a closer look for their particular relevance to knowledge engineering. In contrast to comparison processes, which juxtapose memory structures and return a match/mismatch or measure of similarity, transformational processes change the contents of a structure. 'Alterations' are one form of transformation which add or delete structures or parts thereof or they act to reorganize the structure. 'Productions' use old data structures to generate new structures or replace old ones. 'Derivations' are new structures which have been generated by inferential processes.

The preceding discussions of retrieval and organizational processes have reflected a set of 'content-free' processes. Retrieval processes based on semantic meaning also provide a basis for understanding how some structures are implicitly activated. The basic assumption (Nelson, 1979) is that words are tokens which are connected to a larger number of related concepts but that these concepts can be connected to a larger number of related concepts. Hitting the trigger activates a network of concepts. The questions arise, however, as to what information is actually stored and can be activated quickly (implicitly) and what must be re-generated. Graesser and Clark (1985) have reported that some semantic forms of knowledge are not likely to be stored (see section on text comprehension). For implicitly activated concepts, the cue 'set' is of great importance. The larger, more diverse, the set of cues, the greater the network of related entities and the greater the time and confusion in re-generating the unstored information. A complex stimulus triggering multiple representations in memory can lead to a wealth of information but the representations themselves may be of a different form and structure (Snodgrass, 1989; Collins and Loftus, 1975).

Do our knowledge engineering methods attempt to access different memory structures and do they support an understanding of how these structures may change during the knowledge engineering process ? Our borrowed tools, techniques and procedures were not generally developed with an understanding of memory structures and processes in mind. Also, the issue of context-memory relationships has been neglected even though its effect is of great importance to knowledge engineering. An understanding of how new structures are developed and incorporated into new episodic experiences or how related concepts become incorporated into representations is not supported in knowledge engineering tools. Determining which concepts are related directly and which indirectly to decision making are of concern to the knowledge engineer but the tools used are rudimentary and are focused on the knowledge, not the processes.

5. Text Comprehension

5.1 Cognitive Processes in Text Comprehension

The reader appears an active participant in text comprehension. Perceptual processes (processes which react to visual stimuli and which reflect modular stimulus processing) occur during comprehension and are considered mandatory, automatic and immediate (Swinney and Osterhout, 1990). Cognitive processes, on the other hand, reflect inference processes based on prior world knowledge, plausibility, and pragmatics. The cognitive activity drives analysis and acts as the basis for viewing readers as constructive learners (Wittrock, 1979) or 'active agents' (Anderson, 1970). These cognitive processes and their influences are of interest to better understand the cognitive demands placed on the knowledge engineer.

Inferential processes constitute a class of cognitive processes used in text comprehension to understand concepts, to differentiate between concepts, and to make connections between and among concepts. As someone reads, he/she attempts to make the text meaningful, usually by making predictions (Smith, 1982) and then by determining if the predictions hold true. The elimination of possible meanings until the correct one is found is a process of inferencing based on prior knowledge and on the goal of the reader (van Dijk and Kintsch, 1983; Lesgold and Perfetti, 1978). These types of processes are of interest to us because to make meaningful the information contained in domain textual material, onerous inferential demands are placed on the knowledge engineer.

It is very common for readers to fail in making all the necessary inferences during text comprehension (Britton et al, 1990). Comprehension failure increases as the text becomes less of a narrative, with common inference patterns, and more expository. It is common for a knowledge engineer to read expository and or instructional texts. Making the appropriate inferences depends in large part on prior knowledge of the domain, so considerable inferential demands are placed on the knowledge engineer. The more inferences required of the reader by the text, the less likely the inferences will be made and the less likely the meaning captured. Making the necessary distinctions of concepts and making the necessary inferences of these concepts requires familiarity with the domain: something which the knowledge engineer does not initially have and may never develop sufficiently during the knowledge engineering activity of the project.

5.2 Inference Making During Text Comprehension

Content-based inferences are those which can be made on the basis of perceived word meaning (semantics), as the word is used in the sentence (syntax), or in the context of understanding (pragmatics). Seifert (1990) distinguishes three types of inferences: anaphoric inferences or inferences based on pronouns (see Clark and Haviland, 1977), instrumental inferences or inferences about action (see Singer, 1980), and pragmatic inferences or contextually driven connections (see Graesser, 1981). These three types of content inference are generated on the bias of the kind of world knowledge (and specific domain knowledge) needed to understand a certain kind of text.

Thematic inferences (Seifert et al, 1986) and schematic structures (Schank, 1982; Schank and Abelson, 1977; and Rumelhart and Ortney, 1977) are those inferences generated less on the basis of content and more on the basis of abstract patterns of goals and plans suggested by the text (Seifert, 1990). The relations between concepts are implied more by an abstract conceptualization or structure than by the semantic links in the text. In many cases, these themes are determined by the processing goals (Holyoak and Novik, 1988) of the comprehender. Skimming a text vs searching a text for a specific point may lead to different inferencing paths (Seifert, 1990). The processing goals of the knowledge engineer may vary at different points in the knowledge engineering process (i.e. during each pass through the text). Misconstruing the meaning, generating incomplete or erroneous inference paths, and selecting or constructing partial or incomplete thematic and schematic structures hampers the knowledge engineering process.

The difficulties of the knowledge engineer are further defined by Garner (1985) who notes that the comprehension of technical, expository prose is "little assisted by content-schematic knowledge, in that the very novelty of the material ensures that the readers will be unable to fit much of the new information into their old in-head information"(p.10). Van Dijk and Kintsch (1983) suggest a series of comprehension stages for technical prose. The first step involves a focus on specific content (based on single, individual propositions) to build a 'microstructure. Then, the task is to produce or derive a 'macrostructure' of important content through the application of three types of inferencing rules - deletion, construction and generalization. Procedures for supporting this set of inferencing activity may help overcome the need for domain knowledge in initial comprehension activity.

Types of content-based inferences are more dependent on stored knowledge than on generated or infered knowledge. When reading technical prose, comprehension is strongly based the capacity of the reader to make the correct content-based inferences. Knowing what types of structures are usually stored versus those that are usually generated from stored information helps in determining what processes may be constructed to aid the knowledge engineer. Graesser and Clark (1985) have reported a systematic study of those structures which appear to be stored and those which are not. Those concepts that are known to be directly stored may be elicited. Those concepts can then act as anchors to derive those concepts which have not been directly stored. Knowing which concepts are likely derived gives the knowledge engineer the opportunity to challenge, test or validate them in another manner. Different processes are also necessary for eliciting different types of concepts (eg. causal, goal).

Causal inferences are also of interest. Myers and Duffy (1990) describe a popular mechanism for the development or generation of causal inferences during text comprehension. First, a network representation is constructed. This network results from those initially identified concepts in the text Tentative relationships are constructed on the basis of the surface meaning of the text. Sets of inferences are then generated from these antecedents and they become part of the network. More inferences are drawn as more related information becomes available through more text. In particular, causal antecedents are believed held in short term memory awaiting a link or connection to another concepts which satisfies causal chain properties (Fletcher and Bloom 1988; Trabasso and Sperry, 1985). These causal inferences display four particular properties (Trabasso et al, 1989). The first property is 'temporal priority' in that a cause never happens after the effect. The property of 'operativity' states that the cause must be active at the time when the consequence occurs. The third property suggests that a cause must be seen as necessary for the event to occur. Finally, given a set of antecedent conditions, a cause must be 'sufficient' for the consequent to occur.

For text comprehension in knowledge engineering, it is important for the knowledge engineer to generate the appropriate causal inferences while reading the text. However, given the complexity and newness of the content in the domain, it is entirely likely that the knowledge engineer will fail to make the necessary causal inferences. Some research findings (Graesser and Clark, 1985; Liu, 1989; Trabasso et al, 1988) suggest that it is possible to improve text comprehension by inviting causal inferences by such procedures as systematically posing questions to the reader. Procedures which flag the need for a causal inference or to identify when one is missed would aid the knowledge engineer during comprehension.

5.3 Role of Text Structures in Comprehension

Grammatic and semantic structures in the text have been shown to influence text comprehension. Rumelhart (1980) suggests that when text comprehension is appropriately completed, it is done so because structures in the text have helped to organize knowledge into units which activate the required cognitive processes so that new information is easily processed. When comprehension fails, it is partially due to the reader not having the appropriate schemata for the text content; the reader having the appropriate schemata but no textual cues to trigger its generation or retrieval; or, the reader finds or constructs a consistent interpretation of the text but not the one which was intended (see also Spilich et al, 1979).

Haberland and Graesser (1990) have found that when goal hierarchies were easily portrayed in the text passages subsequent statements were easier to comprehend. When readers are forced to assume or identify goal hierarchies on their own, there tends to be a greater failure rate in making the appropriate inferences. Poor text construction leads to poor text comprehension in a number of other ways as well (Garner, 1988):

Expert generated text comes in many forms. Highly structured text often reflect assumptions that the reader has the necessary background knowledge for sufficient comprehension. In this type of text, goal hierarchies and cause-effect indicators may not be as prevalent as they may be in more introductory material. Procedures which flag these types of indicators in richly technical material (Woodward, 1990) support the knowledge engineer's inferencing processes.

5.4 Meta-Cognition and Comprehension Failure

Comprehension of technical, expository text can also be affected by factors or cues external to the text (Rothopf, 1982). Metacognition, or stable knowledge the reader has about him/herself, about the comprehension task, and about the strategies employed affect the success of comprehension. Flavell (1981) identified four metacognitive components and their relationships to success in text comprehension. Meta-cognitive experiences of the reader refer to those thoughts of how the reader is approaching the goal of reading, questions about how the comprehension is progressing, or the intrusion of other thoughts into the comprehension process. Cognition 'goals' refer to the variety of reasons for reading (eg. skimming, identifying assumptions). Finally, the cognitive strategies of the reader refer to the types of processes the reader uses while reading the text (eg. highlighting passages or writing down key words). Steinberg (1984) offers a compendium of cognitive processes which may be used during text comprehension.

Another critical meta-cognitive process for a reader is that of detecting comprehension failures (see Baker and Brown, 1984a, 1984b; and Brown et al, 1986 for reviews). Often readers do not pay attention to all the words of a sentence or passage. Also, sentences are often mis-parsed due to a partial match between text-based terms and the relevant knowledge structures (Reder and Cleermans, 1990; and see also a summary of error detection research, Garner, 1988). Finally, the necessary inferences may not be made by the reader. Under these circumstances, comprehension will be incomplete and the reader may fail to note the discrepencies. Comprehension failure can only be rectified or corrected if the reader becomes aware of the errors or aware of the partial nature of the inferences. If detection fails as well,then few connections or inferences are likely to be made. Markman (1977) has called this state of affairs the 'delusion of comprehension'.

5.5 Summary

Most of the research in natural language processing in the field of knowledge engineering addresses the understanding and use of natural language as a basis for knowledge acquisition for knowledge-based systems. Wetter and Nuse (1992) have identified many conceptual and practical difficulties with this approach. Essentially, text-based information used for knowledge engineering purposes is removed from its pragmatic context. Knowledge structures present in the pragmatic context for writing the text may or may not relate directly with the knowledge structures in the domain or with the pragmatic context in the knowledge engineering domain. The resulting formal knowledge has experienced two critical transformations: the domain knowledge into natural language and the natural language into a formal language. This approach to natural language focuses on generated knowledge structures and their relationship to domain knowledge.

This paper suggests an alternate, but complementary, approach to the used of text-based natural language by suggesting the development of tools, techniques and procedures which aid and/or identify the cognitive inferencing processes during text comprehension. Rather than focusing on 'extracting' the knowledge from natural language, the focus is on identifying and supporting the cognitive processes involved in knowledge engineering activity. There appear to be well defined and substantiated processes which can focus tool, technique and procedure development.

6. Judgment and Choice

6.1 Judgment and Choice in Knowledge Engineering

Making judgments and choosing alternatives are complicated and complex processes. In many cases, humans complete these complex processes quickly, with less than complete information, and with little consideration of their own thought processes. Limited human information processing capacity requires that we reduce the complexity of the information environment by a number of methods, which consequently result in loss of information and meaning. The expert's domain displays its own inherent patterns but the knowledge engineer and the expert (and the user) structure their own patterns for the domain to make evaluative and predictive judgments and to deal with their own uncertainty about the patterns displayed by the domain.

A common type of judgment made in knowledge engineering is that of 'probable cause'. Einhorn and Hogarth (1981) have identified a number of factors which affect this type of judgment. Aspects of the context in which the judgment is made is considered a major factor. The causal context sets the stage for how various causal candidates emerge as distinct from the entire causal context. Causal clues such as temporal order, co-variation, contiguity in time and space, and the similarity of cause and effect reflect a number of imperfect indicators of causal relations which influence judgment. A third major factor is that of the judgmental strategy used to combine the information from the causal context with the causal clues. A final factor is the role of alternative explanations which may act to discount or reduce the strength of particular causal patterns.

Probability theory has been used as a contrast point to human causal reasoning processes. Tversky and Kahneman (1980) have shown that 'base rate' information is often ignored when making predictive judgments unless it is seen as relevant in the causal context. Judgment errors such as reversing the probabilities of two related events (Eddy, 1982) have also been identified. Issues of plausibility of combined, predicted events present the effects of taking multiple connected predictions into account (Kahneman and Tversky, 1973; Slovic et al, 1976). Accurate judgments become scarce as multiple predictions are requested. Finally, research has shown how people update judgments upon receiving new information (Edwards, 1968). When new information forces a review of a prior judgment, it has proven difficult to evaluate the new evidence against two alternative hypotheses, especially when one has not been well specified. When updating beliefs (Fischhoff and Beyth-Marom, 1983), it is not common to make simultaneous evaluations of that evidence for all alternative hypotheses.

The interaction effects between task demands and human judgment processes helps explain some of the difficulties individuals have with making judgments under certain circumstances. Cue availability constitutes one contextual variable reflecting aspects of task demand (Hogarth, 1987) which strongly influence predictive and evaluative judgments. In the knowledge engineering domain, judgments are made by the expert, knowledge engineer and the user. Availability of appropriate cues (Tversky and Kahneman, 1973) can affect judgments of relative frequency and information presentation patterns can affect the salience of context and cues. Often the degree of variability of a variable is strongly distorted by contextual effects, thus affecting resulting judgments. For example, people tend to base their estimates of frequencies and probabilities on absolute vs relative frequencies (Estes, 1976)

Within the knowledge engineering domain it is possible to identify and outline those judgment processes which are not normally completed effectively by the expert, knowledge engineer and the user. Statistical models have been developed or adapted (Meyer and Booker, 1991) but other types are possible. Hogarth (1987) has described different methods for combining information to stabilize judgment processes. Processes of judgment and choice are viewed as conflict resolution models in that the selection of competing alternatives requires a resolution of conflicting information. 'Compensatory' models directly address the conflict created by the differences by allowing trade-offs on the choice dimensions. Examples are the linear model, additive difference model, and the ideal point model. Each model sets a tactical approach to combining information. 'Non-compensatory' approaches are those which do not allow trade-offs on choice dimensions. Examples of these types of models are the conjunctive, disjunctive, lexicographic, and elimination-by-aspects models.

The field of decision analysis focuses on identifying the intellectual tasks required for making decisions and the field has developed a variety of mathematically and logically based theories, procedures and techniques. This approach stresses the organization of information in a manner which is more conducive to making correct judgments and decisions. A main problem is figuring out what the information means and what relevance it has to the decisions under investigation. The intellectual tasks are identified as systems analysis or decision analysis. The main steps are to identify the problem (its overall analytic structure) and to formalize parts of it. Secondly, the analyst picks an appropriate subset of analytic tools and structures. Finally, the elements and relations identified in the first step are refined (von Winterfeld and Edwards, 1986).

Based on this approach, a number of techniques and procedures have been developed to aid decision making. Decision trees identify steps, sequences of judgment points, and the possible paths (see Holloway, 1979; Behn and Vaupel, 1982). Event and fault trees are similar techniques. Influence diagrams (Howard and Matheson, 1980) present a graphic picture of interactions of variables in a model without imposing a tree structure. The study of uncertainty measurement has also produced a number of models of how to model and represent uncertain information (eg. value and utility measurement, group probability assessments). Finally, this approach has spawned such techniques as multiattribute utility theory (Edwards and Newman, 1982) and sensitivity analysis (von Winterfeld and Edwards, 1986). Techniques in knowledge acquisition based on a decision analysis approach have been developed (Bradshaw and Boose, 1989a; Bradshaw et al, (1989b).

6.2 Judgmental Bias and Decision Aids

Hogarth and Makridakis (1981) have listed a number of sources of bias and have identified their effects on judgment and decision making processes. In the information acquisition stage of decision making, cue clarity and availability, selective perception, cue frequencies, concrete information domination, illusory correlations, data presentation form (primacy, recency, mode, mixture, display and context), and framing are listed as sources of bias. In the processing stage of decision making, inconsistency (of strategy application), conservatism (low revision), non-linear extrapolation, heuristics, anchoring and adjustment, representativeness, law of small numbers, justifiability, regression bias, 'best-guess' strategy, complexity of relationships, emotional stress, social pressures, and inconsistent information sources are identified as sources of bias. In the output stage of decision making, question format, scale effects, wishful thinking and illusion of control are seen as biasers. Finally, in the feedback stage, incomplete outcomes, misperception of chance fluctuations, success/failure attributions, logical fallacies in recall and hindsight bias contribute to faulty judgments.

Specifically in the knowledge engineering environment, Meyer and Booker (1991) and Cleaves (1986) have identified expert bias under two categories: motivational and conceptual. Cleaves developed a number of monitoring and corrective procedures for use with experts to reduce bias. 'Mechanical' procedures manipulate the task or adjust the judgments after knowledge elicitation. 'Behavioural' procedures use interviewing and group interaction techniques to encourage full understanding, identification and control of biasing processes. Visual props to emphasize visual patterns rather than verbal expressions, varying the format of the requested judgment, using differences between actual and essential values, and combining judgments from several individuals constituted mechanical means of bias reduction. Behavioural means included focusing on specific biases in interviews, breaking down complex relationships, training experts in the types of biases, exhorting experts to give reasons for judgments and using group settings for developing consensus. Meyer and Booker (1991) developed similar methods to reduce elicitation biases and they have developed a variety of statistical methods for combining judgments and for reducing biasing effects with groups of experts.

7. Social Cognition and Communication

7.1 Effects of Domain Content on Cognitive Processes

In scientific and technical domains judgments are based on ideas, concepts and/or concrete objects. Using concepts and ideas about concrete entities or well established principles and laws requires a set of cognitive processes or inference processes as a basis of making decisions about something external to the decision maker him/herself. In social domains where decision making needs to include judgments about people a different set, or perhaps an additional set, of cognitive processes come into play. In domains such as law or banking, judgments are made by people about other people. In addition to inferences about ideas, concepts, concrete objects, inferences are made about intentions, characteristics, and motivations of people. This section presents a sampling of research from the areas of social cognition and communication and addresses its application to knowledge engineering. Knowledge engineers may make social decisions about the expert and/or the user. The expert makes social judgments about the knowledge engineer's abilities and extent of domain knowledge. If the expert's domain involves making judgments about people then social cognition processes also come into play in the expertise. In this sense, then, the social characteristics of the domain affect the types of cognitive processes involved in decision making.

7.2 Causal and Dispositional Attributions

This area accounts for a full 11% of all social psychological research (Kelley and Michela, 1980), consequently, a wealth of models and a strong theoretical foundation exists for this area of social cognition. With respect to the knowledge engineering domain, this field of research addresses the question of how the expert, knowledge engineer and user infer a correspondence between observed behaviour and the intentions that produced it. Correspondent inference theory (Jones, 1985) provides a basis for understanding the cognitive processes involved in attributing intentions and dispositions. Attribution theory (Kelley, 1983) discusses those cognitive structures and processes involved in attributing cause.

Kelley (1983) offers two sets of inference processes and structures which account for how people arrive at causal attributions. The first assumes there is information from multiple sources and that covariation of an observed effect and possible causes can be perceived. The second assumes that the perceiver is faced with a single observation and must then use the configuration of factors that are plausible causes of the observed effect. This first set of processes is named the 'covariation' principle. Kelley likened this set of cognitive processes to an ANOVA analysis. 'Consensus' information is required in that the number of people who make similar decisions is taken into account in the decision making processes. 'Consistency' information about how often the single decision maker in question made similar judgments under similar conditions is used as well. Finally, information concerning the 'distinctiveness' of the choice which provides a sense of whether or not a different choice is made due to small changes in one factor. If each of these three factors is considered as having two values (high and low), then a 2 X 2 X 2 ANOVA pattern helps explain this principle. Shaver (1981) points out however, that this approach does not help in distinguishing the truly causal from covarying but non-causal relationships and Langer (1978) has suggested that causal attributions are elicited by types of questions and context demands and are not usually emitted.

The second set of processes describes the 'configuration' principle. This principle bases its strength on proposed underlying structures or causal schemata to help explain casual attributions. Fiedler (1982) points out that a causal schemata does not really exist but it acts as a useful psychological construct for explanation and prediction. One schemata is that of Multiple Sufficient Cause which is evoked when the decision maker can predict effects from the presence or absence of causal contenders. Depending on the presence or absence of multiple contenders, this schemata helps to sort out and order the various combinations of causes. Two important principles are connected with this particular schemata. The 'discounting' principle outlines when one cause or set of possible causes has sufficient strength to discount others. The 'augmentation' principle outlines under what conditions causal contenders support one another.

Schemata of this type support decision making when information is incomplete, when causal 'shorthands' are required for complex situations, and when similar decisions are required across different content areas (Fiske and Taylor, 1984). Reeder and Brewer (1979) postulate 'implicational' schemata to refer to the perceiver's prior conceptions about categories of behaviours. 'Partially restricted' schemata reflect extremes of behaviour. If a person's behaviour is seen as an extreme type of behaviour then it is difficult for the person to be seen acting in the opposite extreme. 'Hierarchically restricted' schemata assign a behaviour or individual to a category which keeps the individual from being seen as better or worse that than category. For example very skilled people experience a range of outcomes depending on motivation and task demands but low skilled people are never seen as having greater aptitude or skill. 'Fully restrictive' schemas are categorical attributions which identify stable levels of an attribute (eg. neatness).

Other research has supported the free occurrence of causal attributions. This research applies to knowledge engineering activity. Lau and Russell (1980) devised a method for identifying attributional structures through content analysis of newspaper stories. Think aloud protocols were analyzed for causal attributions from the responses of parole decision makers while they were reviewing cases (Carroll and Weiner, 1982). During problem solving, when prior expectations were not confirmed, the causal attributions made by subjects were identified and analyzed (Pyszczynski and Greenberg, 1981; Hamilton, 1988).

7.3 Cognitive Components of Causal Attribution

In the previous section the structures of attribution were presented. This section reports on the research which identifies the processes of attribution. The logic of attributions is addressed, the inference processes as well as the knowledge used when making causal and dispositional attributions is summarized.

Hansen (1985) distinguishes attributional content from process. Concepts like covariation and configuration (causal schemata) are considered aspects of logic. How the actual attributions are made constitute the process (see also Newscombe and Rutter, 1982a, 1982b). The knowledge structures used when making attributions is seen as the content (see also Galambos et al, 1986). Kelley's (1983) covariation model has been contrasted with Hewstone's and Jaspars (1987) logical model for identifying necessary and sufficient conditions for causal attribution. The logical model helps delineate all possible combination of causes for consideration. The 'abnormal conditions' model identifies 'counterfactual' criteria and 'contrastive' criteria to establish an opportunity to compare the normal to the abnormal.

Cognitive processes for causal attribution are based largely on the research exemplified by Tversky and Kahneman (1980). Attributional heuristics for causal information processing such as representativeness, anchoring and adjustment, and availability have been identified and are discussed in more detail in another section of this paper. The concept of attributional salience is addressed by Taylor and Fiske (1978). This work identifies the process by which causal attributions refute judgments based only on immediately perceivable information. The mechanism suggests that this type of information can become overly represented in subsequent explanations. In other words, new information which is drawn directly from the immediate context may, under certain circumstance, become over represented in subsequent explanations and render a prior attribution null and void.

Causal schemata are assumed to affect perception, memory and inference (Fiske and Taylor, 1984). They are viewed as working in a top-down manner to simplify and/or amplify information and they act to identify salient information for attributional processes. Knowledge-based causal attribution (Abelson and Lalljee, 1988; Leddo and Abelson, 1986) focuses on the content of causal schemata. This approach tries to draw distinctions between the process of explaining events and the process of understanding them. It has been found that this process requires an understanding of how an individual forms an action plan, the goals of the plan sequence, how that particular plan could be seen to achieve the goal and those particular conditions which initiate the goal (Read, 1987).

7. 4 Meaning Interpretation in Communication

Most of the work on meaning interpretation emphasizes the semantic aspect of language generation and comprehension. This approach is based heavily on the semantic qualities of memory (Kolodner, 1984). Semantically-based categories hold conceptual items which are related through inferences that can be drawn about the categories and concepts. Semantic-based meaning is coupled with words and phrases. In any context, meaning is generated through the use of stored concepts in interaction with a variety of inference processes. Frames (Minsky, 1975), scripts (Schank and Abelson, 1977), schemas (Rumelhart, 1980), mental models (Johnson-Laird, 1980, 1983), causal memory (Genter and Stevens, 1983), situation models (van Dijk and Kintsch, 1983), and E-MOPS (Kolodner, 1984) are examples of semantic memory structures and their accompanying inference processes. These semantic approaches have influenced knowledge engineering computational and representational structures.

However, Grice (1975) places the emphasis not on word meaning but on implications of utterances. In studying communication, linguistic mechanisms have remained front stage in the form of much research in phonology, syntax and semantics. These three fields have produced a number of approaches which concentrate on the meaning and knowledge structures in linguistic forms. The assumptions of these approaches state that knowledge is encoded in the utterances or with the utterances. However, semantic representations of sentences cannot be regarded as corresponding very closely to to thoughts because sentences can be used convey a great number of different thoughts (Sperber and Wilson, 1988). Semantics provide a basis for understanding some of the message but little of the meaning. Semantics do not cover the time and place of the utterance, the identity of the speaker or the speaker's intention. Semantics, as studied by the grammar, only help define the range of possibilities of interpretation. The pragmatics of communication, the interpretation of utterances, helps to choose among the possibilities.

Sperber and Wilson (1988) put forward the mutual knowledge hypothesis. In order to reduce misunderstandings and to increase the recreation of meaning between two people, the context must be shared. Does the expert assume that the knowledge engineer knows X? Does the knowledge engineer know that the expert knows that he knows X? To generate the correct interpretation of an utterance, the one intended by the speaker (Grice, 1975), every item of contextual information used in the interpretation of the utterance must be known by both individuals and each must know that they know. The speaker and the listener must use specific strategies to interpret utterances so to identify the intent of the communication. Misunderstandings commonly occur not only because of semantic confusion but because of pragmatic confusion as well. It is critical for the knowledge engineer to establish an environment with the necessary contextual attributes to ensure that meaning is not impaired.

Without capturing the intention, mistaken facts are no different from true facts. The goal of communication, especially in a domain like knowledge engineering which demands that knowledge is understood, is to ensure that the intentions of statements are clear. When the expert gives an explanation of an event, what are his/her intentions? It is to inform, to teach, to impart understanding, is it to fill-in the background? Knowledge engineers attribute intentions to the experts and users and the experts attribute intentions to the knowledge engineers. A knowledge engineer may review written protocols for communication intentions or he/she can ask the expert for his/her intentions. In a shared cognitive environment like knowledge engineering, all participants have the capacity for making the correct assumptions about the meaning of utterances but they do not always do so. Sperber and Wilson (1988) remind us that the current task determines what information is used for communication. When situational demands trigger cognitive processes, stored information will only be used if it is necessary, the default is to use information directly inferable from the immediate context.

8. Elaborative Paths

The purpose of this paper was to develop a view of knowledge engineering as a set of cognitive processes to complement the predominant 'knowledge' perspective. Knowledge engineering was characterized as a set of cognitive activities used to understand complex, epistemologically rich domains. Selected research in the fields of text comprehension, memory, judgment, and social cognition provided examples of a variety of cognitive processes which have a high probability of occurrence in knowledge engineering activity. The question now is how can this perspective be developed and tested? What are the promising paths?

One of the basic premises outlined in section 2 of the paper described the concept of a cognitive environment. Viewing knowledge engineering as occuring in such an environment requires much more elaboration as to what constitutes such an environment. Identifying and defining the various sites of cognitive activity and the manner in which these sites interact appears to be an essential component of the cognitive basis of knowledge engineering. One of the main issues reflects the need to develop a common understanding of how meaning is produced within, among, or between interacting sites. Also, within this framework, the user of the final system becomes a much more important factor. A more elaborate and systematic description of the cognitive network of knowledge engineering is necessary.

A second path is suggested by the need to empirically identify the critical cognitive processes involved in knowledge engineering. Four likely candidates of cognitive research were presented in this paper but little evidence is available to substantiate the use of many of the cognitive processes suggested in the presented research. Also, this paper addressed only those processes attributed to the knowledge engineer, not the other cogniting agents involved in knowledge engineering. A much more systematic endeavour is required which will result in a system of cognitive processes for knowledge engineering. The various models of cognitive processes from the cognitive science research domain need to be applied in the knowledge engineering domain to determine those which best explain the cognitive processes of knowledge engineering.

A parallel approach to the one mentioned in the previous paragraph is to test the utility of cognitve process models by developing tools, techniques, and procedures for use in knowledge engineering. This approach is a pragmatic one and may appeal more to knowledge engineers than to cognitive scientists. The development of this type of artefact would support (and may, in some cases, replace) the cognitive processes identified in knowledge engineering. The emphasis of these tools would centre on the cognitive activity in knowledge engineering rather than on the development of knowledge structures. Surveying the tools, techniques, and procedures already in use (formally and informally) by knowledge engineers and experts (even users) which could be classified as cognition-support tools would aid in this approach.

Another very critical issue raised in this paper focused on the very onerous tasks faced by the knowledge engineer when entering a new domain. The issue of how an individual deals with complexity, especially epistemological complexity, is central to the cognitive process view of knowledge engineering. Research on expertise sheds some light on this issue for the expert, but we have very little information on how the knowledge engineer and the user deal with the same complexity from a cognitive process point of view. The biomedical research presented in this paper addresses some of the same issues for knowledge engineers. Among other ideas is the importance of understanding how misconceptions develop and spread. For example, the process of comprehending technical prose might be enhanced by structuring the prose in ways which reduce the inference demands on the reader. Some clear paths for research and study have been identified but what is needed is to view knowledge engineering as a laboratory for the study of applied cognitive processes.

Finally, there appears a wealth of cognitive science knowledge available to knowledge engineers: however, challenging problems exist. The first observation is that few of the studies reported in cognitive science are in a form directly useful to knowledge engineers. This is not surprising but it is problematic for knowledge engineering. Methods are required for translating single cognitive science research results and for organizing and presenting a variety of studies into a form useful to knowledge engineers. Another related issue includes developing a valid approach to selecting relevant and promising cognitive science work for application in knowledge engineering. This development requires a thorough understanding of the theoretical foundations of cognitive science and the pragmatic demands of knowledge engineering.

Once cognitive models have been identified for translation into knowledge engineering, another serious problem arises. How does a cognitive science model produce an operational tool, technique, or procedure in knowledge engineering? Does the model retain its meaning in the new context? Does the 'engineering' alter or negate the model's meaning and application? Also, even if the result is successful, other issues arise. These issues are related to the unprincipled importing and combination of knowledge tools which may have contradictory or incompatible assumptions about knowledge. Engineering cognitive science models into knowledge engineering tools, techniques, and procedures requires a principled model or methodology of knowledge engineering.

Cognitive Science offers a great deal of well-developed knowledge to the domain of knowledge engineering but a principled, coordinated framework for identifying, selecting, organizing, translating, engineering cognitive science results into useful knowledge engineering tools, techniques, and procedures is necessary for the full benefits to be realized.

9. References

Abelson, R.P. and Lalljee, M. (1988). Knowledge structures and causal explanations. In D. Hilton (Ed.), **Contemporary Science and Natural Explanation: Commonsense Conceptions of Causality.** Harvester Press: Brighton.

Aitkenhead, A.C. and Slack, J.M. (1985) **Issues in Cognitive Modeling.** Lawrence Erlbaum Associates: London.

Anderson, R.C. (1970). Control of student mediating processes during verbal learning and instruction. **Review of Educational Research,** 40, 349-369.

Baddeley, A.D. (1982). Domains of recollection. **Psychological Review,** 89, 708-729.

Baker, L. and Brown, A.L. (1984a) Cognitive monitoring in reading. In J. Flood (Ed.) **Understanding Reading Comprehension: Cognition, Language and the Structure of Prose** (pp. 21-44). DE: International Reading Association: Newark.

Baker, L. and Brown, A.L. (1984b) Metacognitive skills in reading. In P.D. Pearson (Ed.) **Handbook of Reading Research** (pp. 353-394). Longman: New York.

Behn, R.D. and Vaupel, J.W. (1982) **Quick Analysis for Busy Decision Makers.** Basic Books: New York.

Bjork, R.A. and Richardson-Klavehn, A. (1989) On the puzzling relationship between environmental context and human memory. In C. Izawa (Ed.) **Current Issues in Cognitive Processes: The Tulane Flowertree Symposium on Cognition** (pp. 313-344). LEA: Hillsdale, N.J.

Bjork, E.L. and Bjork, R.A. (1988). On the adaptive aspects of retrieval failure in autobiographical memory. In M.M. Gruneberg, P.E. Morris, and R.N. Sykes (Eds.). **Practical Aspects of Memory II.** Wiley: London.

Brachman, R.J. and Levesque, H.J. (Eds.). **Readings in Knowledge Representation.** Morgan Kaufman Publishers: Los Alto, California.

Bradshaw, J.M. and Boose, J. H.(1989a) Decision analysis techniques for knowledge acquisition: combining information and preference models using Aquinas. **International Journal of Man-Machine Studies,** 32, 121-186.

Bradshaw, J. M., Covington, S.P., Russo, P.J., and Boose, J.H., (1989). Knowledge acquisition techniques for intelligent decision systems: integrating Axotl and Aquinas in DDUCKS. Proceedings of the **AAAI Uncertainty Workshop,** August 18-20, Windsor, Ontario, Canada.

Britton, B.K.,Van Dusen, L., Glynn, S.M., and Hemphill, D. (1990) The impact of inferences on instructional text. In Graesser, A.C. and Bower, G.H. (Eds.), **Inferences and Text Comprehension** (pp. 53-70). Academic Press: New York.

Brown, A.L., Armbruster, B.B., and Baker, L. (1986) The role of metacognition in reading and studying. In J. Orasanu (Ed.), **Reading Comprehension: From Research to Practice** (pp. 49-75). Erlbaum: Hillsdale, N.J.

Cairns, K.V. and Woodward, J.B. (1988) Life Choices simulation: model and methodology. **Systems Practice,** 1(1), 47-64.

Carroll, J.S. and Weiner, R.L. (1982) Cognitive social psychology in court and beyond. In A.H. Hastorf and A.M. Isen (Eds.), **Cognitive Social Psychology.** Elsevier/North-Holland: New York.

Chandrasecaran, B. (1988) Generic tasks as building blocks for knowledge-based systems: the diagnosis and routine design examples. **The Knowledge Engineering Review,** Vol. 3(3), p 183-210.

Clark, H.H. and Haviland, S.E. (1977) Comprehension and the given new contract. In R.O. Freedle (Ed.), **Discourse Production and Comprehension (Vol. 1).** Ablex: Norwood, N.J.

Cleaves, D.A. (1986). Cognitive biases and corrective techniques: proposals for improving elicitation procedures for knowledge-based systems. In proceedings of the **First**

AAAI Knowledge Acquisition Workshop for Knowledge-Based Systems: Banff, Canada.

Collins, A.M. and Luftus E.F. (1975) A spreading activation theory of semantic processing. **Psychological Review**, 82, 407-428.

Coulson, R.J., Feltovich, P.J., and Spiro, R.J. (1986) Foundations of a Misunderstanding of the Ultrastructural Basis of Myocardial Failure: A Reciprocating Network of Oversimplifications. **Report #1 Conceptual Knowledge Research Project**, Southern Illinois University School of Medicine, Springfield, IL

Crowder, R.G. (1976). **Principles of Learning and Memory**. Erlbaum: Hillsdale, N.J.

Crowder, R.G. (1982). The demise of short term memory. **Acta Psychologica**, 50, 291-323.

Eddy, D.M. (1982) Probabalistic reasoning in clinical medicine: problems and opportunities. In D, Kahneman, P. Slovic, and A. Tversky (Eds.). **Judgment Under Uncertainty: Heuristics and Biases**. Cambridge University Press: New York.

Edwards, W. (1968) Conservatism in human information processing. In B. Kleinmuntz (Ed.) **Formal Representation of Human Judgment**. Wiley: New York.

Edwards, W. and Newman, J.R. (1982). **Multiattribute Evaluation**. Sage: Beverly Hills, CA.

Eich, E. (1985) Context ,memory, and integrated item/content imagery. **Journal of Experimental Psychology: Learning, Memory and Cognition**, 11, 764-770.

Einhorn, H.J. and Hogarth, R.M. (1981). Behavioral decision theory: processes of judgment and choice. **Annual Review of Psychology**, 32, 53-88.

Estes, W.K. (1976) The cognitive side of probability learning. **Psychological Review**, 83, 37-64.

Evans, D.A. (1989) Issues of cognitive science in medicine. In D.A. Evans and V.L. Patel (Eds.), **Cognitive Science on Medicine: Biomedical Modeling** (pp. 1-16). MIT Press: London.

Fiedler, K. (1982) Causal schemata: review and criticism or research on a popular construct. **Journal of Personality and Social Psychology**, 42, 1001-13.

Feltovich, P.J., Spiro, R.J., and Coulson, R.L. (1989). The nature of conceptual understanding in biomedicine: the deep structure of complex ideas and the development of misconceptions. In D.A. Evans and V.L. Patel (Eds.), **Cognitive Science on Medicine: Biomedical Modeling** (pp. 113-172). MIT Press: London.

Feltovich, P.J., Johnson, P.E., Moller, J.H., and Swanson, D.B. (1984) LCS: the role and development of medical knowledge in diagnostic expertise. In W.J. Clancey and E.H. Shortcliffe (Eds.) **Readings in medical Artificial Intelligence: The First Decade**. Addison-Wesley: Reading, MA.

Fischhoff, B. and Beyth-Marom, R. (1983) Hypothesis evaluation from a Bayesian perspective. **Psychological Review**, 90, 239-260.

Fischhoff, B., Slovic, P, and Lichtenstein, S. (1978) Fault trees: sensitivity of estimated probabilities to problem representation. **Journal of Experimental Psychology: Human Perception and Performance**, 4, 342-355.

Fiske, S.T. and Taylor, S.E. (1984). **Social Cognition**. Random House: New York.

Flavell, J.H. (1981) Cognitive monitoring. In W.P. Dickson (Ed.), **Children's Oral Communication Skills** (pp. 35-60). Academic Press: New York.

Fletcher, C.R. and Bloom, C.P. (1988). Causal reasoning in the comprehension of simple narrative texts. **Journal of Memory and Language**, 27, 235-244.

Flood, R.L. (1987). Complexity: a definition by construction of a conceptual framework. **Systems Research**, 4(3), 177-185.

Gaines, B.R. (1988) Knowledge acquisition systems for rapid prototyping of expert systems. **INFOR**, 26(4), 256-285 (Nov.).

Galambos, J.A., Abelson, R.P., and Black, J.B. (Eds.) (1986). **Knowledge Structures**. Erlbaum: Hillsdale.

Garner, R. (1985). **Metacognition and Reading Comprehension**. Ablex: Norwood, N.J.

Garner, R. (1988). Strategies for reading and studying expository text. **Educational Psychologist**.

Gentner, D. and Stevens, A.L. (1983) **Mental Models**. Erlbaum: Hillsdale, N.J.

Glaser, R. (1984) Education and thinking: the role of knowledge. **American Psychologist**, 39, 93-104.

Gorfein, D.S. and Hoffman, R.R. (1987). **Memory and Learning: The Ebbinghaus Centennial Conference**. Erlbaum Associates: Hillsdale, N.J.

Graesser, A.C. and Clark, L.F. (1985) **Structures and Procedures of Implicit Knowledge**. Ablex: Norwood, N.J.

Graesser, A.C. (1981) **Prose Comprehension Beyond the Word**. Springer-Verlag: New York.

Grice, H.P. (1975). Logic and conversation. In P. Cole and J. Morgan (Eds.), **Syntax and Semantics 3: Speech Acts**. Academic Press: New York.

Haberland, K, and Graesser, A.C. (1990) Integration and buffering of new information. In Graesser, A.C. and Bower, G.H. (Eds.), **Inferences and Text Comprehension** (pp. 71-88). Academic Press: New York.

Hamilton, D.L. (1988). Causal attribution viewed from an information processing perspective. In D. Bar-Tal and A.W. Kruglanski (Eds.), **The Social Psychology of Knowledge**. Cambridge University Press: Cambridge, UK.

Hansen, R.D. (1985). Cognitive economy and commonsense attribution processing. In J.H. Harvey and G. Weary (Eds.), **Attribution: Basic Issues and Applications**. Academic Press: Orlando, FL.

Hewitt, C. (1985) The challenge of open systems. **Byte**, 10, 223-42.

Hewstone, M. and Jaspars, J.M.F. (1987). Covariation and causal attribution: a logical model of the intuitive analysis of variance. **Journal of Personality and Social Psychology**, 53, 663-72.

Hogarth, R. (1987) **Judgment and Choice (2nd Ed)**. Wiley: New York.

Hogarth, R., and Makridakis, S. (1981) Forecasting and planning: an evaluation. **Management Science**, 27(2).

Holloway, C.A. (1979). **Decision Making Under Uncertainty: Models and Choices**. Prentice-Hall: Englewood Cliffs, N.J.

Holyoak, K. and Novik, L. (1983) Unpublished manuscript. Referenced in Seifert, C.M.Content-based inferences. In Graesser, A.C. and Bower, G.H.(Eds.),**Inferences and Text Comprehension** (p. 118). Academic Press: New York.

Howard, R.A. and Matheson, J.E. (1980) **Influence Diagrams**. Stanford Research Institute International: Menlo Park, CA.

Jacoby, L.L. (1983). Remembering the data: analyzing interactive processes in reading. **Journal of Verbal Learning and Verbal Behaviour**, 22, 485-508.

Johnson, P.E. (1986) Specification of expertise: Knowledge acquisition for expert systems. In proceedings of the **First AAAI Knowledge Acquisition Workshop for Knowledge-Based Systems**: Banff, Canada.

Johnson-Laird, P.N. 91983) **Mental Models**. Cambridge University Press: Cambridge, UK.

Jones, E.E. (1985) Major developments in social psychology during the past five decades. In G. Lindzey and E. Aronson (Eds.), **Handbook of Social Psychology (Vol.1)(3rd Ed)**. Random House: New York.

Kahneman, D.L. and Tversky, A. (1973). On the psychology of prediction. **Psychological Review**, 80, 273-251.

Kahneman, D.L., Slovic, P. and Tversky, A. (Eds) (1982) **Judgment Under Uncertainty: Heuristics and Biases**. Cambridge University Press: New York.

Kelley, H.H. and Michela, J.L. (1980). Attribution theory and research. **Annual Review of Psychology**, 31, 457-503.

Kelley, H.H. (1983). Perceived causal structures. In J.M.F. Jaspars, F.D. Fincham, and M. Hewstone (Eds.), **Attribution Theory and Research: Conceptual Development and Social Dimensions**. Academic Press: London.

Klatzky (1980). **Human Memory: Structures and Processes (2nd Ed.)** W.H. freeman: San Fransisco.

Kolodner, J.L. (1984) **Retrieval and Organizational Strategies in Conceptual memory: A Computer Model.** LEA: Hillsdale, N.J.

Kosslyn, S.M. (1980) **Image and Mind**. Harvard University Press: Cambridge, MA.

La France, M (1986) The knowledge acquisition grid: a method for training knowledge engineers. In proceedings of the **First AAAI Knowledge Acquisition Workshop for Knowledge-Based Systems**: Banff, Canada.

Langer, E.J. (1978). Rethinking the role of thought in social interaction. In J.H. Harvey, W.J. Ickes and R.F. Kidd (Eds.), **New Directions in Attribution Research (Vol. 3)**. Erlbaum: Hillsdale, N.J.

Lau, R.R. and Russell, D. (1980) Attributions in the sports pages. **Journal of Personality and Social Psychology**, 39, 451-63.

Leddo, J., and Abelson, R.P. (1986). The nature of explanations. In J.A. Galambos, R.P. Abelson, and J.B. Black (Eds.), **Knowledge Structures**. Erlbaum: Hillsdale, N.J.

Lehnert, W.G. (1978) **The Process of Question Answering.** LEA: Hillsdale, N.J.

Lesgold, A.M. and Perfetti, C.A. (1978) Interactive processes in reading comprehension. **Discourse Processes**, 1, 323-336.

Liu, L. (1989) Reading Between the Lines: The Assessment and Promotion of Comprehension by the Use of Questions. **Unpublished Doctoral Dissertation**: University of Chicago.

Marcus, S. (Ed.)(1988) **Automated Knowledge Acquisition for Expert Systems.** Kluwar Academic Publishers: Norwell, Mass.

Markman, E.M. (1977) Realizing that you don't understand: elementary school children's awareness of inconsistencies. **Child Development**, 50, 643-655.

Meyer, M.A. and Booker, J.M. (1991). **Eliciting and Analyzing Expert Judgment: A Practical Guide**. Academic Press: New York.

Minsky, M. (1975) A framework for representing knowledge. In P.H. Winston (Ed.), **The Psychology of Computer Vision**. McGraw-Hill: New York.

Musen, M.A. (1989) Conceptual models of interactive knowledge acquisition. **Knowledge Acquisition**, Vol. 1(1), p 73-88.

Myers, J.L and Duffy, S.A. (1990) Causal inferences and text memory. In Graesser, A.C. and Bower, G.H. (Eds.), **Inferences and Text Comprehension** (pp. 159-174). Academic Press: New York.

Nelson, D.L. (1979) Remembering pictures and words: appearance, significance, and name. In L. Cermak and F. Craik (Eds.). **Levels of Processing in Human Memory**. Erlbaum: Hillsdale, N.J.

Newscombe, R.D. and Rutter, D.R. (1982a). Ten reasons why ANOVA theory and research fail to explain attribution processes: 1 conceptual problems. **Current Psychological Reviews**, 2, 95-107.

Newscombe, R.D. and Rutter, D.R. (1982b). Ten reasons why ANOVA theory and research fail to explain attribution processes: 2 methodological problems. **Current Psychological Reviews**, 2, 153-70

Patel, V.L., Evans, D.A. ,and Groen, G.J. (1989). Biomedical knowledge and clinical reasoning. In D.A. Evans and V.L. Patel (Eds.), **Cognitive Science on Medicine: Biomedical Modeling** (pp. 53-112). MIT Press: London.

Potter, M. (1990). remembering. In D.N. Osherson and E.E. Smith (Eds.), **Thinking** (pp. 3-32). MIT Press: Cambridge, MA.

Pyszczynski, T.A. and Greenberg, J. (1981). Role of disconfirmed expectancies in the instigation of attributional processing. **Journal of Personality and Social Psychology**, 40, 31-8.

Read, S.J. (1987) Constructing causal scenarios: a knowledge structure approach to causal reasoning. **Journal of Personality and Social Psychology,** 52, 288-302.

Reder, L.M. and Cleermans, A. (1990). The role of partial matches in comprehension: the moses illusion revisited. In Graesser, A.C. and Bower, G.H. (Eds.), **Inferences and Text Comprehension** (pp. 233-258). Academic Press: New York.

Reeder, G.D. and Brewer, M.B. (1979). A schematic model of dispositional attribution in interpersonal perception. **Psychological Review,** 86, 61-79.

Roediger, H.L. and Blaxton, T.A. (1987). Retrieval modes produce dissociations in memory for surface information. In D.S. Gorfein and R.R. Hoffman (Eds.). **Memory and Cognitive Processes: The Ebbinghaus Centennial Conference.** LEA: Hillsdale, N.J.

Rothkopf, E.Z. (1982) Adjunct aids and the control of mathemagenic activities during purposeful reading. In W. Otto and S. White (Eds.) **Reading Expository Material** (pp. 109-138) Academic Press: New York.

Rumelhart, D.E. and Ortney, A. (1977) The representation of knowledge in memory. In R.C. Anderson, R.J. Spiro, and W.E. Montague (Eds.), **Schooling and the Acquisition of Knowledge**. Erlbaum: Hillsdale, N.J.

Rumelhart, D.E. (1980) Schemata: The building blocks of cognition. In R.J. Spiro, B.C. Bruce and W.F. Brewer, (Eds), **Theoretical Issues in Reading Comprehension** (pp. 33-58). Erlbaum: Hillsdale, N.J.

Schank, R.C. (1975). **Conceptual Information Processing.** American Elsevier: New York.

Schank, R.C. and Abelson, R.P. (1977) **Scripts, Plans, Goals, and Understanding.** Erlbaum: Hillsdale, N.J.

Schank, R.C. (1982) **Dynamic Memory: A Theory of Reminding and Learning in Computers and People.** Cambridge University Press: Cambridge.

Seifert, C.M. (1990) Content-based inferences in text. In Graesser, A.C. and Bower, G.H. (Eds.), **Inferences and Text Comprehension** (pp. 103-122). Academic Press: New York.

Seifert, C.M., Abelson, R.P., and McKoon, G. (1986) The role of thematic knowledge structures in reminding. In J.A. Galambos, R.P Abelson, and J.B. Black (Eds.), **Knowledge Structures.** Erlbaum, Hillsdale, N.J.

Shaver, K.G. (1981). Back to basics: on the role of theory in the attribution of causality. In J.H. Harvey, W.J. Ickes and R.F. Kidd (Eds.), **New Directions in Attribution Research (Vol. 3)**. Erlbaum: Hillsdale, N.J.

Simon, H.A. (1988) Cognitive architectures and rational analysis:Comment. In K. vanLehn (ed), **Architectures for Intelligence: The Twenty-Second Carnegie Symposium on Cognition.** Lawrence Erlbaum Associates: Hillsdale.

Singer, M. (1980) The role of case-filling inferences in the coherence of brief passages. **Discourse Processes,** 3, 185-201.

Slovic, P. Fischhoff, B., and Lichtenstein, S. (1976). Cognitive processes and societal risk taking. In J.S. Carroll and J.W. Payne (Eds.) **Cognition and Social Behavior**. Erlbaum: Hillsdale, N.J.

Smith, F. (1982). **Understanding Reading** (3rd Ed.). Holt, Rinehart, and Winston: New York.

Smith, S.M. (1988). Environmental context-dependent memory. In D.M. Thomson and G.M. Davies (Eds.) **Memory in Context: Context in Memory**. Wiley: New York.

Snodgrass, J.G. (1989). How many memory systems are there really?: some evidence from the picture fragment completion task. In C. Izawa (Ed.) **Current Issues in Cognitive Processes: The Tulane Flowertree Symposium on Cognition** (pp. 135-174). LEA: Hillsdale, N.J.

Sperber, D. and Wilson, D. (1988) **Relevance: Communication and Cognition.** Basil Blackwell: Oxford, UK.

Spilich, G.J., Vesonder, G.T., Chiesi, H.L., and Voss, J.F. (1979). Text processing of domain-related information for individuals with high and low domain knowledge. **Journal of Verbal Learning and Behaviour**, 18, 275-290.

Spiro, R.J., Feltovich, P.J., and Coulson, R.L. (1988) Seductive reductions: the hazards of oversimplification of complex concepts. **Report #4 Conceptual Knowledge Research Project**, Southern Illinois University School of medicine, Springfield, IL.

Spiro, R.J. (1980) Constructive processes in prose comprehension and recall. In R.J. Spiro, B.C. Bruce, and W.F. Brewer (Eds.), **Theoretical Issues in Reading Comprehension.** Lawrence Erlbaum Associates: Hillsdale, N.J.

Spiro, R.J., Feltovich, P.J., Coulson, R.L., and Anderson, D.K. (1990) multiple analogies for complex concepts: antidotes for analogy-induced misconceptions in advanced knowledge acquisition. In S. Vosniadou and A. Ortony (Eds.) **Similarity and Analogical Reasoning.** Cambridge University Press: Cambridge, UK.

Stanovich, K.E. (1980). Toward an interactive-compensatory model of individual differences in the development of reading fluency. **Reading Research Quarterly,** 16, 32-71.

Steinberg, R.J. (1984) What should intelligence tests test? Implications of a triarchic theory of intelligence for intelligence testing. **Educational Researcher**, 13, 5-15.

Swinney, D.A. and Osterhout, L. (1990). Inference generation during auditory language comprehension. In Graesser, A.C. and Bower, G.H. (Eds.), **Inferences and Text Comprehension** (pp. 17-34). Academic Press: New York.

Taylor, S.E. and Fiske, S.T. (1978). Salience, attention and attribution: top of the head phenomena. In L. Berkowitz (Ed.), **Advances in Experimental Social Psychology (Vol. 11).** Academic Press: New York.

Trabasso, T, and Sperry, L.L. (1985) Causal relatedness and importance of story events. **Journal of Memory and Language**, 24, 419-427.

Trabasso, T, van den Broek, P.W. and Suh, S.Y. (1989) Logical necessity and transitivity of causal relations in stories. **Discourse Processes**, 12, 1-25.

Trabasso, T, van den Broek, P.W. and Liu, L. (1988) A model for generating questions that assess and promote comprehension. **Questioning Exchange**, 2, 25-38.

Tulving, E. (1985). How many memory systems are there? **American Psychologist**, 4, 385-398.

Tversky, A. and Kahneman, D.L. (1973). Availability: a heuristic for judging frequency and probability. **Cognitive Psychology**, 5, 207-232.

Tversky, A. and Kahneman, D.L. (1980). Causal schemas in judgment under uncertainty. In M. Fishbein (Ed.) **Progress in Social Psychology.** Erlbaum: Hillsdale, N.J.

van Dijk, T.A. and Kintsch, W. (1983). **Strategies of Discourse Comprehension.** Academic Press,: New York.

von Winterfeldt, D. and Edwards, W. (1986) **Decision Analysis and Behavioral Research**, Cambridge University Press: Cambridge. UK.

Wetter, Th. and Nuse, R. (1992) Use of natural language for knowledge acquisition: strategies to cope with semantic and pragmatic variation. **IBM Journal of Research and Development: Special Issue on AI.**

Wetter, Th. and Woodward, J.B. (1990) Towards a theoretical framework for knowledge acquisition. Proceedings of the **Fifth AAAI Knowledge Acquisition Workshop for Knowledge-Based Systems**: Banff, Canada.

Wielinga, B., Akkermans, H., Schreiber, G. and Balder, J. (1989) A knowledge acquisition perspective on knowledge-level models. Proceedings of the **Fourth AAAI Knowledge Acquisition Workshop for Knowledge-Based Systems**: Banff, Canada.

Wittrock, M.C. (1979). The cognitive movement in instruction. **Educational Researcher**, 8, 5-11.

Woodward, J.B. (1990) Knowledge acquisition at the front end: defining the domain. **Knowledge Acquisition**, 2(1), 73-94.

Concluding Remarks

A Comparative Assessment of Selected Approaches in the Focal Area of Knowledge Engineering and Cognition

Thomas Wetter

IBM Germany, Scientific Center
Institute for Knowledge Based Systems
Postfach 10 30 68
W-6900 Heidelberg
Germany
WETTER@DHDIBM1.bitnet

Introduction

A workshop on knowledge **engineering** and **cognition** by its very nature brings together researchers and practitioners with quite different perspectives. Each of the perspectives sets the stage for a specific way to approach problems. The perspective taken may be as crucial for the solutions that are achieved as the problem itself is. As a result, it may turn out – as was the case concerning some of the approaches presented at the workshop – that different groups arrived at contradictory results for seemingly identical problems.

It is the intention of this analysis to identify reasons for such apparent contradictions. I am not about to remove contradictions but hope to trace them back to their causes and hence clarify them. This basically involves an analytical approach, namely understanding why something happens. But it does also have a constructive facet: given a situation where a researcher or practitioner has to make a selection to investigate, refine, or apply an approach, the situation so far has shown that in many cases he will find positive and negative votes concerning whether the approach works juxtaposed in an unrelated way. The aim of the analysis is to distinguish situations where an approach does or does not work.

The prevailing structure of this article will be to examine a number of criteria and to characterize as many individual contributions according to as many criteria as seem to apply.

Among the reasons for different researchers and practitioners to proceed fundamentally differently and to arrive at different results that are hard to compare is that both knowledge engineering and the study of cognition are imbedded in a cluster of different disciplines and their respective methods and criteria. Among them are

- informatics [1] ([Dallemagne], [Manago], [Branskat])[2]
- artificial intelligence ([Bergmann], [Puppe])
- cognitive and social psychology ([Fensel], [Woodward] [Thoben], [Glowalla])
- cognitive modelling ([Reimann], [Janetzko])

This list is not exhaustive, since some of the disciplines tend to lie somewhere in between. According to their traditions, the approaches may be focussed upon

- software development and its tools, re-usable building blocks and results, and the process of managing software projects,
- sophisticated computational methods to solve complex problems,
- individual and social processes and their impact upon applying and communicating knowledge,
- computational models of knowledgeable behavior.

Consequently, the approaches may judge their respective results according to

- exploitation of human and computational resources,
- appropriateness of representation and inference,
- assessment of processes and contents addressed by the involved agents,
- plausibility of models, similarity with observed phenomena.

[1] Some readers may expect to find the term "computer science" here. But we are not really dealing with a science of computing machinery but with a science of the well founded use of information. Therefore, the term "informatics" that is more common in Europe has been preferred to the traditional American "computer science".

[2] Arguments presented in this text may draw upon oral contributions during the workshop in February, 1991 or papers in this textbook prepared after the workshop. In the latter case, the source is referenced as [First_author92]. In the former case, when there is no corresponding written contribution, or when it does not play a role, the respective workshop participant is referenced as [Participant]. Of course I take full responsibility for possibly having misunderstood or over-interpreted some detail of the oral presentations.

If we consider this topic more generally again, some of the approaches are predominantly product oriented (informatics and artificial intelligence), others address the extent of capturing the processes involved in human knowers (cognitive and social psychology), and a third group mainly aims at abstract formal models of the real world (cognitive modelling). To label these approaches in a much too rough but still suggestive way, we could talk about engineering, formal theories, humanities, and science approaches. Formal theories may of course play a role in all four. In AI, their inherent properties are an essential point of interest. An engineer may use theories for constructing objects in the real world with desired properties. In the humanities and more so in science, the descriptive and predictive power of formal theories with respect to phenomena in the real world is the most essential criterion. Given this, it would be more of a surprise than an expected result that the different approaches arrive at comparable or even similar results. One of the major outcomes for the workshop participants, and hopefully for the readers of this book, might be to understand which role each of the disciplines plays with respect to the other disciplines involved.

Consequently, the strategy of presentation in this chapter is to try and pick up one point of contrast after the other and to collect the controversial opinions and results from the individual contributions. This may draw upon oral contributions at the workshop or written material in the respective text in these proceedings. In the latter case I only briefly refer to a segment from the individual contribution.

The different attitudes that the involved disciplines take and the methods they apply form one organizational scheme for the subsequent text. There is a second scheme orthogonal to this one, namely to identify objects and individuals playing certain parts in the process of knowledge engineering. It is their properties, activities, contributions, etc. that can be analyzed from the different perspectives of the disciplines. Individuals obviously are experts and knowledge engineers but also researchers in knowledge engineering [3]. Objects may be tools or methods that knowledge engineers use.

Outline of Presentation

The text is organized as follows: there will be two major parts centering around the two groups of human protagonists of the play: the experts and the knowledge engineers. The part concerning the characteristics of the expert's role in the play will address the three aspects of:

[3] The role of users of the final systems has not been an issue at the workshop and is only addressed in [Strube]. Therefore a comparative assesssment is not required.

1. development of expertise, including the aspect of intermediate states
2. change of expertise through the effects of knowledge acquisition activities
3. representation of expertise, with special emphasis on
   - case vs. theory based representation
   - common sense foundation of expertise

The part analyzing the knowledge engineer 's role has two sections:

4. his possibilities, limitations, skills etc., including
   - details of how to play the role
   - the degree to which he must become a domain expert of his own
   - his biases
5. methods and tools that can support his activity, with the *models of problem solving* as the most prominent examples:
   - their origins in artificial intelligence, psychology, or directly from knowledge engineering
   - their nature
   - the activities involved in using them
   - the required skill

Finally, we will discuss shifts of positions that happened in long ranging projects and speculate about reasons why different approaches arrived at different and seemingly contradictory results.

1. Development and Nature of Expertise

A major group of arguments can be gathered around ″encoding specifity″ ([Tulvi73]). Encoding specifity roughly means that memorizing some information cannot be separated from memorizing the context (place, mode of presentation, etc.) of the information. Encoding specifity has been used by [Glowalla] and can explicitly be found referenced by Janetzko and Strube ([Janet92]). Glowalla also draws upon neural architecture considerations in this context, which make it plausible that the full setting of the ″information itself″, the way it comes across the individual and the situation or context where it is embedded, influence the way it is encoded. [Woodward] also uses related arguments under the terminology ″environment vs. memory″ (also ″ecologic memory″).

Taking these arguments in their full strength would imply that the individual does not have much of a choice regarding the way his knowledge becomes represented in his mind - one might even use the term ″brain″ here.

On the other hand, [Schmalhofer] reports experimental evidence for a learning process of going gradually into the depth of a ″body of knowledge″ as opposed to going gradually into the breadth. Since there is no reason to believe that systematically ″shallow details″ from the full horizontal range of a domain precede ″deep details″, i.e.

that all environments "choose" to "present" first shallow and then deep material, a second active mechanism (aside from the more passive specific encoding) may exist. According to [Schmalhofer], deep representations can arise from application of prior knowledge or from being confronted with a combination of material (cases, justifications, theory, etc.). The former does not lend itself for a systematic argument (we will return to this point when discussing common sense parts of expertise), because it leads to an infinite regress when trying to clarify the origin and nature of the prior knowledge. The latter is further specified by [Schmalhofer] as mechanisms for filling holes in representations. It remains open as to how much this process is active and how much is passive. [Schmalhofer] also encountered the correction of misconceptions which again can be interpreted as due to active or passive processes.

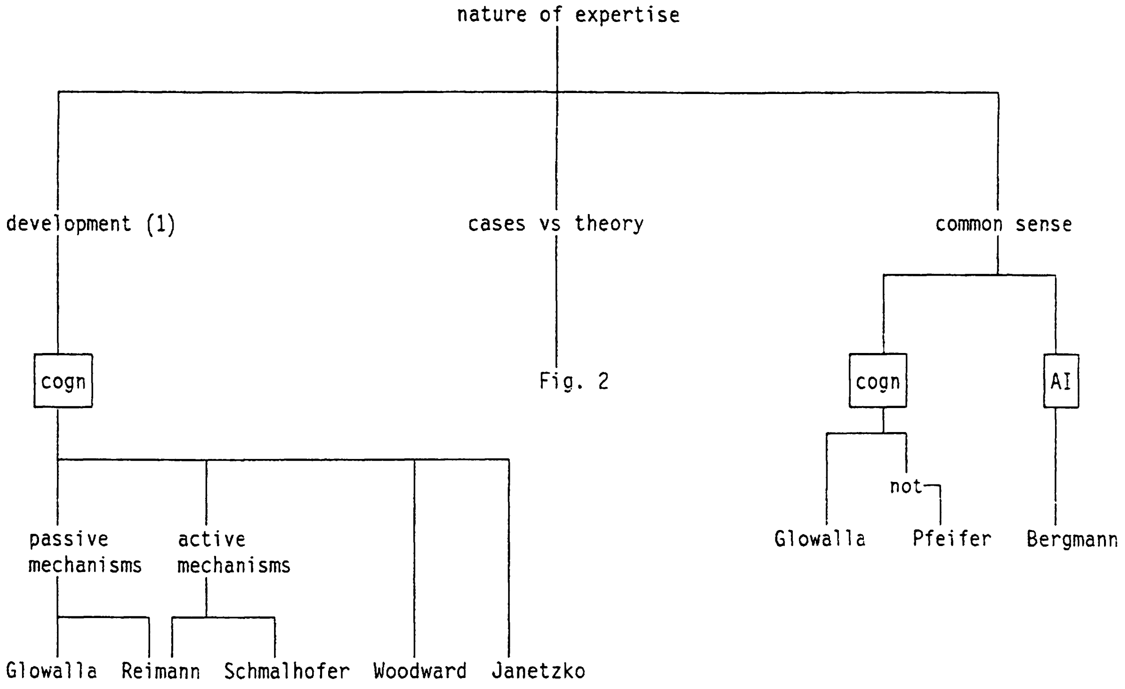
For the focal area of cognition and knowledge engineering, two implications from the recent discussion are important.

First, if only passive mechanisms occur, it is hard to argue that basic representations can change. This is crucial for the subsequent discussion as to whether cases or theories are the basic representation. If e.g. material only occurs in the form of cases, there is no reason to believe that the individual can arrive at a theory, as has been postulated in [Janet92].

Secondly, we can ask whether intermediate representations persist or are extinguished when more advanced ones emerge in the individual. If we have to be aware of active revisions, as postulated for the process of developing mental models, early representations may vanish. If a metaphor of adding and compiling applies, early representations would not be extinguished but just "buried", and could be retrieved by adequate "digging", as postulated in Kolodners models of memory and recall (for references to both schools of thought, cf. [Janet92]).

2. Change of Expertise under Knowledge Acquisition Activities

It has been claimed (e.g. by Becker and Bartsch-Spörl ([Becke90])) that being exposed to knowledge acquisition activities changes the knowledge in the expert. This aspect has not been addressed at the workshop. But a posteriori reflection of its contributions supplies systematic support for some of the above arguments namely about encoding specifity. If indeed the way how the material is presented to the knowledge engineer co-determines the representation, knowledge engineering as a discipline must be aware that its activities fundamentally determine its results. If certain predominant elicitation methods are applied over and over again, experts may be incapable of resisting the format of presentation associated with the methods. In other words, the methods will start to encode into the expert that which they claim to elicit from him.



(1) including "change of expertise under knowledge acquisition activities"

Figure 1. Nature of expertise

This is different and more critical here than in cognitive science because of its impact on software products.

In Figure 1 and in the following figures, a topic of the size of approx. one section is broken down into several steps. Leaf notes of the resulting trees represent individuals and their contributions. Abbreviations in boxes (***cogn***(itive science), ***inf***(ormatics), ***A***(rtificial) ***I***(ntelligence), ***psy***(chology) and ***soc***(ial) sciences) denote the disciplines from which the results have been derived. The depth of the tree may vary according to the degree of differentiation of the positions. ″not″ above Pfeifer denotes that common sense as part of expertise is refuted in the contribution of Pfeifer et al. (see below).

3. Forms and Representation of Expertise

Cases vs. Theories

In this section, ″theory″ will be used in the sense of Janetzko′s and Strube′s ([Janet92]) ″semantic knowledge″. For ″case″ we follow Althoff′s and Weß′ ([Altho92]) definitions. Concerning cases, an important aspect of differentiation will be to what extent abstraction takes place; we will speak of authentic cases when no conscious or explicit reduction, omission, or comprehension of detail occurs.

Apart from the transient role of cases or episodes during development of expertise (cf. section 1.) these same cases and episodes are also increasingly discussed as constituents or carriers of fully developed expertise. This is the subject of the present section. In a related and more fundamental text, Althoff and Weß ([Altho92]) provide the terminological basis for the distinction between case based reasoning and inductive learning, between case and rule based reasoning, and between case based and analogical reasoning. Their terminological and partly mathematical apparatus will be used here and there in the subsequent set of direct comparisons between approaches presented in this book.

We follow the top level distinction of [Altho92] that cases can be used directly for reasoning (case based or analogical reasoning) or as material from which to construct general knowledge by induction.

To continue with the distinctions that we encounter:

- Case and rule based reasoning may occur in isolation or they can be combined or fully integrated in different ways.

- Cases can be taken as such or as concrete exemplars from which abstractions are generated and used for reasoning.

Let us set out with reasoning from authentic cases, i.e. a more or less pure approach according to the first two distinctions. Reimann and Schult ([Reima92]) supply considerable detail from the cognitive psychology literature and from their own work about learning from the solutions of exemplary tasks. They have isolated those behavioral patterns that characterize successful learners. Their methodological contribution is to describe these patterns and to work towards formal and executable models and simulation of such models.

Successful learning involves more elaboration, reflection, and self assessment of the learner and results in hierarchical arrangements of the example tasks or cases. These observations from mechanics resemble results known from text comprehension. In contrast, Reimann and Schult find that passive learning supplies flat case collections organized by similarity measures.

While [Reima92] classify the form of learning that results in a hierarchical representation as successful, and the form resulting in flat collections as less successful, Althoff and Weß([Altho92]) report other psychological investigations without any preference for one of the forms. This deviation between the findings reported in [Reima92] and [Altho92] can be interpreted in several ways but can probably not be decided at present. The hypothesis of [Reima92] in favor of hierarchical organization may turn out to be generally valid and happened not to be detected in the investigations quoted in [Altho92]. In this case, cognitive science comes up with a well founded recommendation for architectures of case based reasoners. This architecture may resemble Kolodner's (for more detail and references, cf. Janetzko and Strube, [Janet92])

It may, however, turn out as well that there are fields where hierarchical organization is superior and others where similarity based retrieval should be preferred. This is not implausible, because in contrast to some other domains, the rather regular domain of classical mechanics used by [Reima92] may have intrinsic structures supporting the development of a hierarchical indexing structure. In this case, both types of architectures will coexist, and it now becomes part of the knowledge engineer's skill to determine the demands of the domain he is encountering. [Reima92]'s contribution to the resolution of this question could consist in a move towards testing (in domains other than mechanics) whether the cognitive models correctly predict the retrieval of cases.

While Reimann and Schult ([Reima92]) are looking deeply into human cognition and how it acquires and represents cases, Branskat ([Brans92]) rather assumes the practical informatics perspective to provide a knowledge engineer's workplace. Her hypercard implemented tool supports the step by step transition from informal case descriptions to MOPs to be used in a case based reasoner. In other words, [Branskat] sets out from one of the above plausible hypotheses how cases may be stored, namely through hierarchical indexing in Kolodner's sense, and supplies a tool for the generation of respective knowledge bases.

The common feature of these three contributions, which also distinguishes them from all others presented aside from a short part of Manago and Conruyt ([Manag92] see below) is that they focus on sets of authentic cases, as opposed to abstractions from cases or theories derived from cases.

We now vary the pure form of solely reasoning from non-abstracted cases. The first variation is brought into play in Janetzko and Strube ([Janet92]). They refer to research results in cognitive psychology which imply that human individuals apply both cases or episodes and general knowledge, each at its respective time, and report about conditions when it is necessary to "take a turn" from case to theory based reasoning or vice versa. This contribution suggests both, a hybrid architecture where cases and a theory co-exist and a special form of inference is to take a turn from one form of reasoning to the other, and a KADS interpretation model of such a hybrid system, i.e. a form that has proved useful for the ***process*** of engineering systems.

This theoretical concept for a hybrid architecture should not be mistaken for the fully implemented two systems KATE and CAS-SYSTEM in [Manag92]. KATE is an induction system enhanced by pre-structuring of the cases. CAS-SYSTEM is a case based reasoner based on authentic cases. KATE and CAS-SYSTEM exist next to each other, not as an integrated hybrid reasoner. ***But*** KATE has proved in large real applications (with certain drawbacks to be reported below), and CAS-SYSTEM has been created to overcome some of these practically relevant shortcomings. Both these systems in [Manag92] are precisely described in data processing terminology ("...dynamically builds a path...", "...computes the tree..."), as opposed to the psychological backbone of [Janet92]'s argumentation.

The second variation away from pure case based reasoning does not change the architecture but proceeds from authentic cases to forms of systematic useful abstractions. The two contributions of Schmalhofer et al ([Schma92]) and Bergmann ([Bergm92]) complement each other in isolating the essence of a case from incidental detail to support knowledge engineering for a system architecture that reasons from abstract cases, which in this case are skeletal plans.

Let us start with [Schma92] who supply a frame in which [Bergm92] fills a certain slot. Starting point of the work of [Schma92] are concrete observations in a domain of mechanical engineering about how humans memorize problem categories and partly specified solution "skeletons". Concretely, a technical drawing of a rotational part is a problem description, for which the solution in form of a fully specified working plan to produce the part is required.

A possibly surprising result is that *problem* classes and skeletal *solutions* are memorized - or at least can be elicited - separately. One reason may be that the attribution of a skeletal solution plan to a problem definition is not just to match the visual pattern of the drawing to an equally visual skeletal plan from a library but involves considerations about material properties, the number of pieces to produce, etc.

Secondly, problem classes result in a hierarchical organization, which is, however, fully different from Kolodner's indexing in both contents and method of construction. While Kolodner's indices are the extensions of concrete memorizing activities for individual events, [Schma92]'s classes and their distinctions each reflect whole categories of similar problems characterized e.g. by workpiece material or machine type. Furthermore, an index in Kolodner's sense comes into existence by having to distinguish a concrete new individual element from concrete known elements. The classes in [Schma92]are derived by psychometric methods, which determine distances between individual yet prototypical elements based on similarity assessments of pairs ("hierarchical clustering"). As an independent criterion that the clusters in [Schma92] make sense, it turns out that production plans follow naturally (although not trivially) and exemplars from a library of solutions can be identified for classes that make sense, whereas such workplans do not exist for senseless classes.

All activities described in [Schma92] are supported by dedicated tools. We are facing an informatics style workplace, whose only shortcoming might possibly be that it has so far only proven useful for the one application outlined, namely working plans for rotational parts. Generalization may be feasible, but there is no evidence yet.

For the transition from expert generated concrete plans to abstract skeletal plans, Bergmann ([Bergm92]) adds an AI based set of automatic tools which realize explanation based learning to derive abstract descriptions of operations concretely encountered in workplans and dependency graphs between the abstract operators. This allows for the final skeletal plans to have beneficial dependencies (e.g. required sequences in order to ensure preconditions for operators) preserved and detrimental ones removed.

In this context, [Thoben] has pointed out at the workshop that the term "variant planning" from engineering sciences is, according to his observations, misleading for the description of the mental use of known cases. While variant planning in engineering modifies fully worked out cases to satisfy a slightly deviating requirement, human use of cases more resembles the application of partially instantiated abstract patterns and their instantiation, as can be realized on the basis of Bergmann's abstracted skeletal plans.

Bergmann's tools draw upon theories from AI, including some common sense ideas. They provide results of well understood formal quality: for a generated skeletal plan

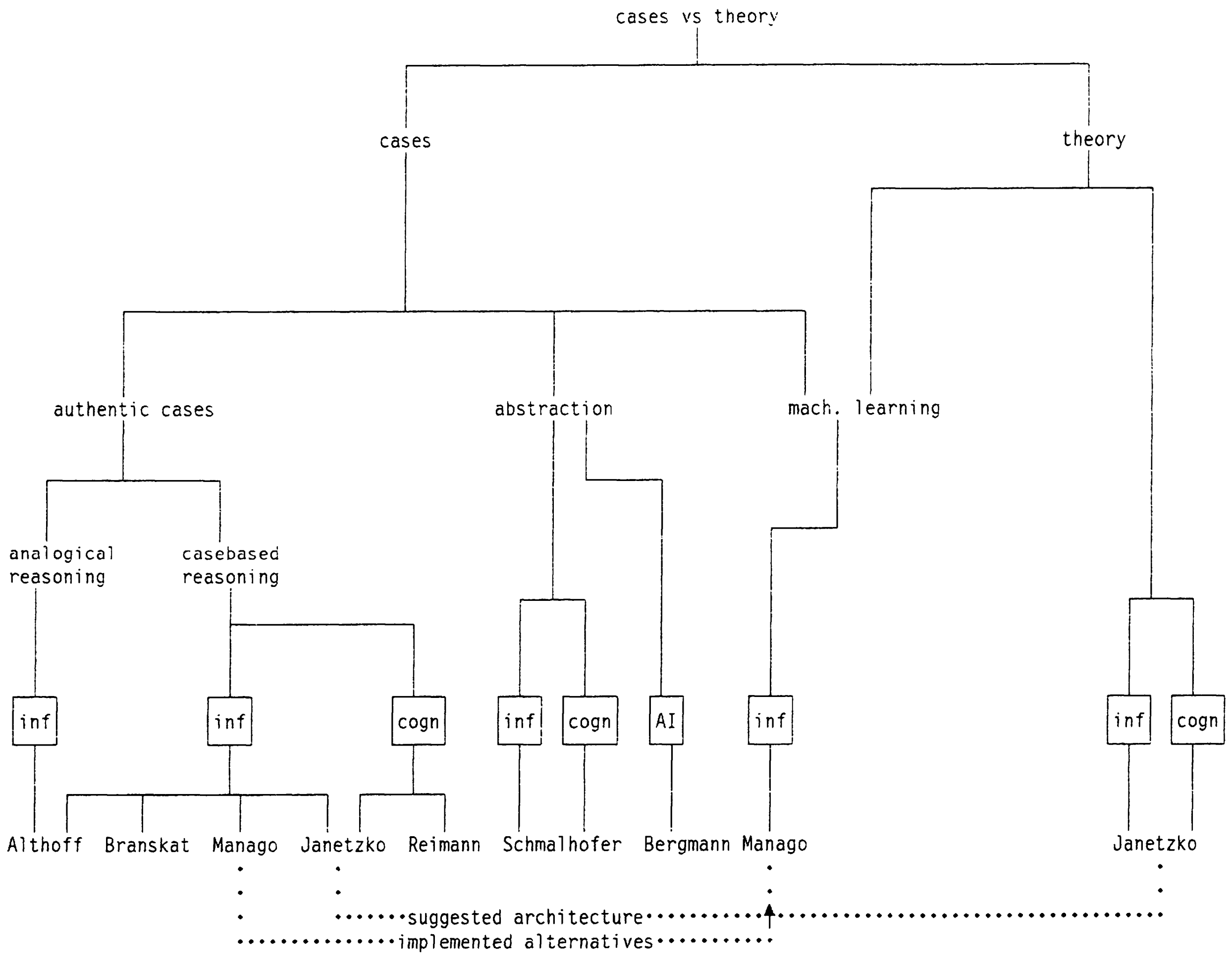


Figure 2. Cases vs. theory

it can be guaranteed that all specific plans which it represents can be derived as specializations. At present, the automatic procedure cannot, however, "collapse sequences of concrete operations into a single abstract operation", a capacity which the human expert has.

For the comparison with the next approach it makes sense to rephrase this work as follows: general theoretical considerations are applied to improve the usefulness of case collections. At first sight, the opposite seems to be true for the core part of Manago and Conruyt ([Manag92]): cases are used for automatic induction of general regularities.

Practical experience with predecessors of KATE have, however, led to a transition from "blind" induction to the prestructuring of the case format. The major argument is a practical one: purely mechanical induction is not feasible beyond a certain number of features of the objects to be learned. Common sense is used to reduce the number of combinations that make sense. Although formally different, the approaches of [Bergm92] and [Manag92] hence share the strategy of reducing their respective "search spaces" by making use of human knowledge about implausible or physically impossible combinations before the initiation of mechanical processes.

The positions that the contributions in this book take about case vs. theory based reasoning are not as controversial as we will find later in this text concerning different model based approaches to knowledge acquisition. They nicely exploit large parts of the possible spectrum, but all except [Manag92] have not yet undergone sufficient comparative or large scale practical evaluation to reveal far ranging differences.

Common sense

Positions at the workshop widely diverged about the common sense contents of expertise.

Fundamentally, [Glowalla] claimed that a considerable part of expertise is made up of common sense. This would be supported by [Bergmann]'s observation that the constraints superimposed on skeletal plans to transfer them into realistic, fully specified ones carry many traits regarding what has been discussed as formal theories of common sense ([Hobbs85]), such as that there can only be one form of fixture (chucking) at a time (in the domain of production plans for manufacturing rotational parts).

Another support comes from observations of [Schmalhofer] already briefly discussed above that deep knowledge (inasmuch as expertise is deep) can develop on the basis of prior knowledge, which ends up in common sense sooner or later. I.e. expertise itself would at least be founded on common sense. And given the position of development as adding and compiling, presented above as one plausible alternative for development of expertise, common sense would remain present in expertise one way or the other.

In contrast [Pfeifer] argued that expertise is rather specialized. It is acquired by fully developed cognitive individuals. Hence it is bound to be different than the normal repertory of cognitive science methods as applied, e.g. by [Glowalla].

Support for this position comes from the use of KADS [Schre88]. A survey of existing conceptual models, i.e. specifications of domain knowledge and problem solving processes occurring in an application, reveals that the strategic layer of KADS which may be understood as representing flexible human management of new or unexpected situations, is hardly ever required in KADS conceptual models, i.e. the comprehensive notation of a model of expertise. An interpretation by KADS researchers is that *"expert - i.e. highly overlearned - problem solving skills* compiles out the flexibility available as part of *"'intelligence' in the psychometric sense"* [Schre88].

Although it has proved successful in practical engineering domains, one should be careful about the latter argument as being valid as a scientific statement about the nature of expertise. For applying KADS means to assess human capacities from the informatics perspective, which is more or less negligent of the intellectual and experimental repertoire of cognitive science. KADS has **not** claimed to be able to deal with expertise in the ways of psychological investigation - as would be required to prove [Pfeifer]'s conjecture - but to strive for usefulness for the purpose of running knowledge acquisition projects. Although KADS models may well serve the purpose of being a humanly *conceivable* notation of how expert problem solving can take place, there is neither a claim nor a proof that these models describe the manner of problem solving that *actually* happens in the expert.

4. Roles and Capacities of Knowledge Engineer and Expert

As indicated we will organize this section by role and - inasmuch as they involve a knowledge engineer - by the capacities, limitations, and possibilities that support his activities.

Roles

Only in Puppe's and Gappa's ([Puppe]) approach do we find the position that a knowledge engineer is not required. This is based on the claim that CLASSICA is a tool that allows the expert to construct knowledge based systems on his own within the scope of the shell underlying CLASSICA. This does avoid serious questions and problems raised in the sequel of this text (e.g. concerning biases of the knowledge engineer). But [Puppe] himself raised the question of cognitive adequateness of

CLASSICA as a tool for novice users of electronic data processing equipment. Furthermore, the question must be raised whether an expert is able to (re-)structure his knowledge according to the fixed requirements of CLASSICA, with or without being educated in the theory underlying CLASSICA. This can be rephrased as to whether the expert has to become a knowledge engineer. This is rather like the mirrorimage of a question that we will have to discuss later, namely whether the knowledge engineer has to become a domain expert. This makes evident why the usability question of CLASSICA and comparable tools is not the classical one of software ergonomics: it is not a question of whether an individual is efficiently supported in his *normal* job (namely acting as an expert) but in an *unusual* one, namely reflecting and formalizing his knowledge. To summarize, using tools of the type of which CLASSICA is an example requires of the expert the coordination of different unfamiliar activities, namely using the tools and reflecting his knowledge under constraints of having to formalize it.

All other approaches presented involve a knowledge engineer, and obviously all involve experts. The expert can be more an actor or the object of observation, his role can be precisely specified in advance or may evolve in the process. In my opinion, the most liberal attitude towards the expert is the one of [Fensel], who refers to the interpretative paradigm of qualitative social science. For initial phases of a knowledge engineering process communication and interaction between knowledge engineer and expert without restriction by any modelling or other paradigm is a prerequisite for guaranteeing (or at least for enabling) coverage of the full spectrum of expertise. Any early determination of having to model into some representation bears the risk of overseeing important details next to those that can be subsumed under the model of Strube (cf. [Strub92]).

After this interaction and communication process, the knowledge engineer applies his data reduction and interpretation methods to discover as much as possible from the traces he has. In other words, both expert and knowledge engineer are first involved in the full spectrum of both their technical and communicational capacities. Then the knowledge engineer evaluates the interactions in the light of his methods.

The next two approaches place the expert in two differently restricted roles in comparison to [Fensel] or [Puppe]. First, Schlenker and Wetter ([Schle92]) take a scientific discovery perspective of the following form: there is a natural phenomenon named knowledge. Knowledge engineering means to proceed towards a model of the phenomenon through a cyclic process of experimentation, interpretation of observations, hypothesis forming, and designing of further experiments to support or refute the present state of cognition. In [Wetter]'s exposition of this approach, the expert is comparable to subjects in psychological experiments. He undergoes an experimental setting, is observed, and that's it. The attempt to apply this approach and to supply rationales for all its activities has meanwhile revealed to Schlenker and Wetter (cf.

[Schle92]) that the pure objectivist observer - observed object relation cannot or should not be maintained between knowledge engineer and expert. Instead, the setting of elicition co-determines the concretely observed (verbal) data. In analogy to the above mentioned encoding specifity, one might think of a "recall specifity". Nevertheless, the knowledge engineer controls the process and hence is as responsible for the result as is the natural scientist for his physical, etc. model. On the other hand, ideally he can draw upon all the methodological and theory of science knowledge that guards and criticizes the process of scientific progress in natural sciences.

Apart from the possible criticism as to whether knowledge can be "delivered" in such a sterile situation, it is obvious that such a process is extremely time consuming. Both, [Fensel] and [Schlenker] may end up with highly detailed, well suited, and objectively (in their respective theories of science traditions) justified results. Or they might end up running out of resources. Or they might not even get started because knowledge engineers with the required skills are not available and no one is willing to pay for their education except for within narrow academic environments.

[Bergmann] defines a restricted role of the expert in a more realistic, practical, down to earth way. In his environment of specific plans being generalized by means of formal domain theories, it is a common observation that plans are over-generalized due to an underspecified domain theory. It is now the role of the expert to detect over-generalizations, to mend them, and to enlarge or modify the domain theories accordingly. In this case both the communicational role and the content related role of the expert is very narrowly restricted, but in accordance with what practical expertise really consists of. A question may arise as to whether the outlined role can be generalized to other areas of application.

Does the knowledge engineer have to become a domain expert?

[Woodward] takes a clear position in understanding knowledge engineering to be the study of a complex domain and draws analogies to learning biases etc. Studying the domain can almost be re-phrased as becoming an expert.

Although [Woodward] differs from [Schlenker] in a number of other aspects, there is the following similarity: if we interpret the scientific discovery of [Schlenker] as a process at the end of which the scientist is an expert in his domain, the knowledge engineer in this framework would be an expert of the expertise that he has discovered. In the case of [Fensel], it is not so clear whether the knowledge engineer must become a domain expert. The first interaction phase seems to tend toward this direction, whereas the second (at least potentially) operationalized interpretation phase might have the character of mechanical labelling, ordering etc., based on linguistic markers. It appears that the second phase of [Fensel] can be labeled as "labelling" in which the individual labels arise from the domain under study, whereas the way in which a label

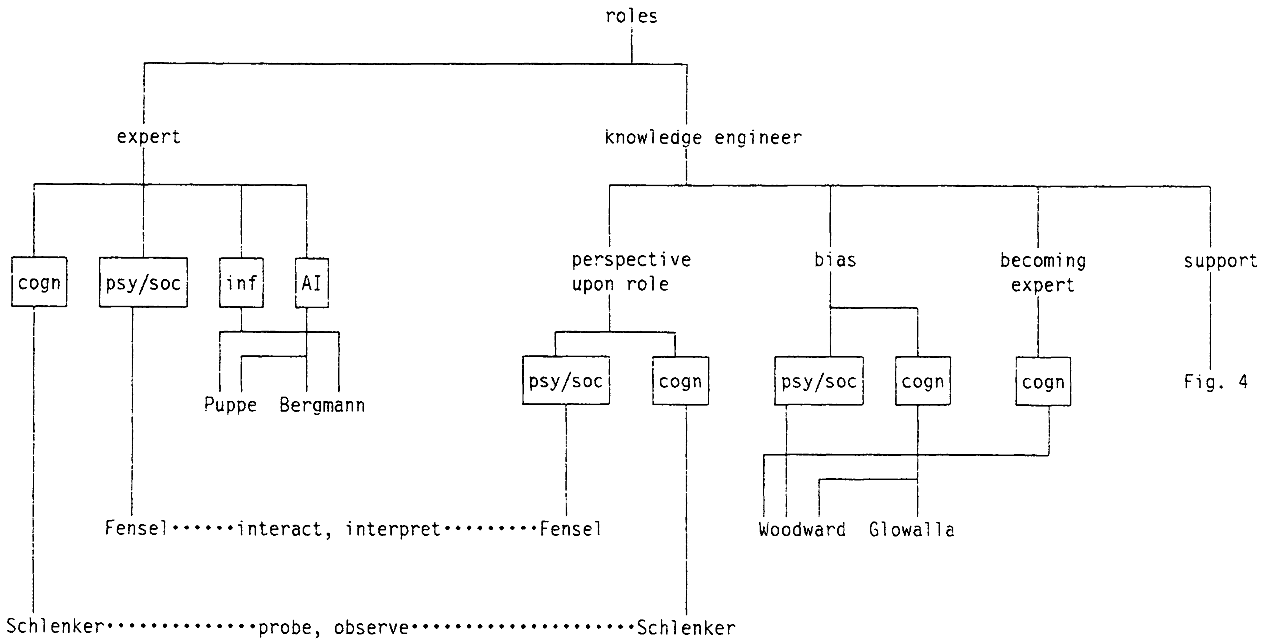


Figure 3. Human roles in knowledge engineering

is attached to a recorded detail is prescribed by the method. Labelling principles immanent to a method might reduce the need to become a domain expert, but also the precision and security of match of the resulting model might be less in the method of [Fensel] than in the feedback controlled one of [Schlenker].

Biases of the knowledge engineer

According to [Woodward], cognitive biases inevitably co-determine the work of the knowledge engineer. Starting from "knowledge engineering as learning a complex domain", we have to be aware of learning biases (e.g. an over-simplification bias), biases of memorizing and recall (e.g. the whole environment vs. memory types of problems) (cf. [Woodw92]). Further support for the existence of recall biases comes from [Glowalla], who reveals e.g. considerable omissions of presented detail in a way which can easily be identified in a controlled psychological experiment where the investigator typically knows all the detail of material presented. This does not, however, apply to knowledge engineering where the knowledge engineer typically does **not** know the material to be elicited from the expert.

Biases can be dealt with in a number of ways:

1. trying to be aware of the mechanisms and to avoid or reduce them ([Woodward])
2. trying to be aware of the context for possible intervention or compensation ([Fensel])
3. hoping for laws of larger numbers, i.e. of a cancelling out when a large number of knowledge engineering efforts is taken (multiple knowledge engineers, elicitation methods, interpretation methods, experts, etc.)
4. trying to detect the biases by means of feedback ([Schlenker]).

The third of these is probably not really feasible. The first might appear most feasible, but it bears the risk of being trapped in wrong hypotheses about where the biases come in and by what methods they can be reduced or compensated for. The second, taken as a goal, reflects one of the strongest present trends in knowledge acquisition but lacks any operationalization in the approach of [Fensel]. An instance of the fourth point is [Schle92], which may, however, not be feasible except for extremely critical bodies of knowledge.

5. Supporting the Knowledge Engineer

After having outlined the inevitable constraints under which knowledge engineering has to take place, we now describe cognitive and engineering aids in support of the knowledge engineer's work. By far the longest part of this section deals with approaches dominated by their respective types of pre-existing models and the ways in which they are used. But let us start with the few alternatives that have been addressed at the workshop.

Fully bottom up approaches

Principally, all knowledge engineering work involves some manifestation of knowledge and knowledgeable behavior and an executable formalization being in some relation to the manifestation. Support can consist of the systematic treatment of manifestations (bottom up) or desired target structures of formalizations (top down). The first set of approaches will depend on interpretation aids, whereas the latter will depend on formalization aids. In practice, both aids will occur in both types of approaches, with stronger emphasis on one or the other.

Obviously [Fensel] has a strong sense of how to interpret. His way of interpreting is not determined by the form of the model to be arrived at, but aims rather at making the best (justified) use of the detail. The same applies to [Schlenker], who takes different measures to make the best use of the detail (operationalized interpretation rules in the approaches of [Fensel] and [Schlenker] plus planned feedback by further observation in [Schlenker]). So primarily both approaches are open as to which kind of model is to be arrived at.

[Fensel] aims at arriving at KADS conceptual models at a later stage. Although very critical about the present stage of KADS interpretation models, he makes the point that his highly consuming bottom up approach can profit from some top down guidance.

While [Fensel] assumes full spectrum communication as a starting point, [Schlenker] and [Woodward] share a view of reducing the communication spectrum. However, while this is done in the case of [Schlenker] specifically with an aim at modelling, in the case of [Woodward] it respects more the known phenomena about communication and its failures or biases.

Libraries of models

Models support communication and the mental capture of the details of a domain much better than isolated "pieces of knowledge". This applies to some extent to the previous approaches which develop new models, and probably more to the following ones that try to make use of existing models. It has also been claimed that the communicational value of models supports maintenance. Depending upon their provenance some classes of models may be better suited for capture and communication of expertise than others.

Origins of models

Models and libraries of models can originate from the disciplines involved in different ways. The usability of models for human users on one hand and as computational formalizations or preforms of such on the other hand can be expected to be highly correlated to the origin of the models.

Puppe's and Gappa's ([Puppe92]) models e.g. of *cover and differentiate* (in diagnosis), etc. originate from artificial intelligence, more concretely from the discussion of models of problem solving and partly from naive physics or studies about the relation between structure and behavior. These models draw on first principles of reasoning and their respective knowledge structures. They are fully computationally implemented templates, including a user interface or modelling workplace to be used by the domain expert for form filling. Aside from this, they exist without any further guidance, instruction, etc.

KADS also draws upon research about humanly comprehensible generic elements of representing knowledge and inference such as the KL-ONE research and Clancey's work. These latter roots rather emphasize characteristics of representation and inference whereas the ancestors of the models of [Puppe] are rather those of the characteristics of the domains themselves. But this difference in origin is probably much less pronounced than the difference in what the two approaches did with the models: Puppe implemented them, whereas the KADS group elaborated on how to apply them.

Coming back now to more detail on ancestry, the following is true for the spirit of the conceptual ("knowledge level") modelling: empirical evidence in a number of diverse fields of early KADS applications was reflected and cast into semiformal descriptions of the inference processes that had been encountered. This is not too far from the manner in which early expert system shells evolved; having modelled an application, its knowledge representation and inference technology were isolated and sold as a software product. The difference with KADS is that a software product is not sold but rather a methodology, which includes experiences that have been reflected and cast into some humanly conceivable notation of the abstracted models.

The next prominent modelling approach represented at the workshop is McDermott's and coworkers' Spark-Burn-Firefighter (SBF) of Dallemagne et al. ([Dalle92], [Klinker]). At the beginning (see paragraph *Are libraries and tools feasible and usable?* below), the attempt was made to provide a meta-tool making available the technologies of approved knowledge acquisition tools, supporting the process of selecting among them in order to allow the non-programmer to write knowledge based systems. The starting point was *empirical* in the sense of collecting what had proved useful in practice and not *theoretical* in the sense of supplying the formal detail of theories of reasoning, etc.

When models arise from cognitive psychology ([Schlenker], Janetzko and Strube ([Janet92])), their intention is to best capture the processes in humans when solving a problem. They would, hence, not model domains as such but human conceptualizations of domains. By this token they would have the best chances of being humanly conceivable (see Strube's discussion on cognitive adequacy in this book) but would offer the least chance of detecting and getting rid of individual misconceptions (biases) of the expert from whose conceptualization the model is derived.

Nature of the models
Axes of distinctions between the different approaches are

- whether models are intended as mental tools or code fragments
- which grain size they and their elements have
- to what extent and detail they pre-exist, or rather how much refining and instantiating is required in the individual knowledge engineering process
- which stages a model goes through.

To start with the most obvious approach, Puppe's and Gappa's ([Puppe]) D3 offers executable versions of the artificial intelligence models of diagnostic problem solving. The individual engineering activities consist of selecting the most appropriate diagnostic model and supplying domain knowledge in the respective format. There is no operationalized process of supporting the selection, but once it has been arrived at, the rest is more or less routine form filling. The models are specified on all levels of grain size. Every model is a complete monolithic building block, which precisely complements a domain theory to be added by the expert.

[Klinker] distinguishes two grain sizes of constituents. The smaller ones — mechanisms — serve as both mental and technical building blocks and are designed for re-use. The process of selecting and configuring the appropriate mechanisms is widely supported by analytic and attribution aids in the tool Spark. The tool Burn generates acquisition tools to elicit the required detail for customizing the configuration of mechanisms. The fully configured combinations — the methods or workflows — being the larger constituents in SBF are understood to be specific for the application and not intended for re-use. In other words, we find here no match for what will be introduced as the interpretation model in KADS.

The model structure and building block perspective of KADS plays different roles in the approaches of [Fensel], Pfeifer et al. ([Rothenfluh]), [Shadbolt], and

[4] For an introduction to the KADS terminology, cf. Janetzko and Strube ([Janet92]). The KADS expression "domain layer" should not be confused with the standard use of "domain" in contexts such as domain expert, complex domain, etc. where static and inferencing aspects are normally included. In some cases where confusion might easily occur,

[Linster][4]. In KADS, re-occurring patterns of inferencing (the larger units) have been abstracted from the domains they operate upon. They have been named interpretation models and have been collected in a library. Typical elements of the library are methods of problem solving such as heuristic classification. Interpretation models have been provided in verbal and pseudocode notation including individual knowledge sources as smaller building blocks. In pure KADS, interpretation models are primarily meant — as the name already suggests — as mental help in interpreting data from knowledge elicitation activities. They help in constructing conceptual models, but they do not become parts of them.

Activities using models and the required skills

Having introduced the KADS modelling *elements*, we now begin the section on *using* models with KADS and will come back to the other types of models later, i.e. we will proceed in the reverse order now.

[Fensel] takes KADS as such (at least in the oral presentation; in the text, the reference to KADS is only marginal and the modelling terminology is more like that of [Musen89]). After the initial steps of arriving at a common understanding between knowledge engineer and expert, he tries to come up with a principled formulation of the process model [5]. Further processes use elicited data to add the domain information to the selected process model. Models do not control the process but filter recorded interpretations in the end. The required skills for this final filtering are in understanding the model constituents as mental tools, in understanding results of interpretation by means of methods from qualitative sociology, and relating them to each other. While the latter draws upon skills that are part of an education in the humanities, the former presently seems to be based on intuition.

While [Fensel] is not fully determined about modelling in KADS but arrives at structures in a way which seems related to KADS, [Rothenfluh] explicitly refers to KADS but comes up with a different method of arriving at KADS conceptual models. Based on protocol analysis data, the method uses the degree of variation among knowledge engineers in phrasing their interpretations in terms of the KADS interpretation models as a measure for the adequateness of the individual interpretation model. Basically [Rothenfluh] understands the KADS conceptual models as computational, and his question is whether they are also mental tools. To answer this, he draws upon intuition just as [Fensel] did (besides the skill of experimental psychology of conducting think-aloud experiments), but at least he assesses the result by the un-

I use the expression "application" instead of "domain" to denote that all static and inferencing aspects are to be subsumed.

[5] It can be assumed that it roughly resembles the task and inference layer of a KADS conceptual model. But to preserve authenticity, Fensel's [Fense92] terminology is used.

specific feedback of similarity or dissimilarity of the conceptual models produced by different knowledge engineers. He tests for inter-individual reliability among knowledge engineers, not for the validity of models. This is in good accordance with the communication role of models.

[Shadbolt] works on a method of precisely determining the selection of an interpretation model. Based on an informal task analysis, he offers elicitation tools whose use results in a suggested interpretation model. For this purpose he has complemented the set of interpretation models (for the selection of which in KADS there is only an intuitively defined decision tree) by a set of formal criteria, whose evaluation is supported by the tools.

[Linster] keeps the static knowledge / inference knowledge distinction, but starts from individually tailored models rather than from surrogates derived from pre-formed patterns. Essentially, the model serves as a medium of communication between humans. [Linster] has meanwhile extended his approach that starts with a thorough investigation of the structure of a new domain by meta-tools to generate tailor suited tools which enable the experts to enter the details of the model themselves. In the latter sense, the approaches of [Linster] and [Puppe] resemble each other. However, in Linster's case, a knowledge engineer is heavily involved in forming the model which then determines the functionality of the customized tool. No mental support other than the KADS recommendation of layers is given. In other words, skill in computer science and intuition or modelling skill form the intellectual bases of knowledge engineering.

Coming back now to the non-KADS building block approaches, the approach of [Klinker] may seem to be perfect: his process is widely operationalized, his smaller building blocks (mechanisms) are both mentally embedded and technically available as building blocks (code fragments) whose instantiation supplies operational code. However, obviously and admittantly the building blocks of Klinker's approach are considerably simpler than those of KADS or in Puppe's case. Assuming that a well educated facilitator (=knowledge engineer?) is involved, more or less standard procedures (or their abstractions in forms of the mechanisms) can be identified and instantiated, the result being a software module which has properties that enable easy integration in an environment of similarly produced modules. In summary, the approach has been discussed more as KASE (knowledge aided software engineering) than as knowledge engineering. The key role is that of the facilitator, who has to match abstract descriptions of mechanism functionality (provided by programmers) with the information processing needs of (non-programmer) users. If this succeeds, the users can influence their own application programs to a considerable extent without requiring essentially different skills than they need for their normal profession.

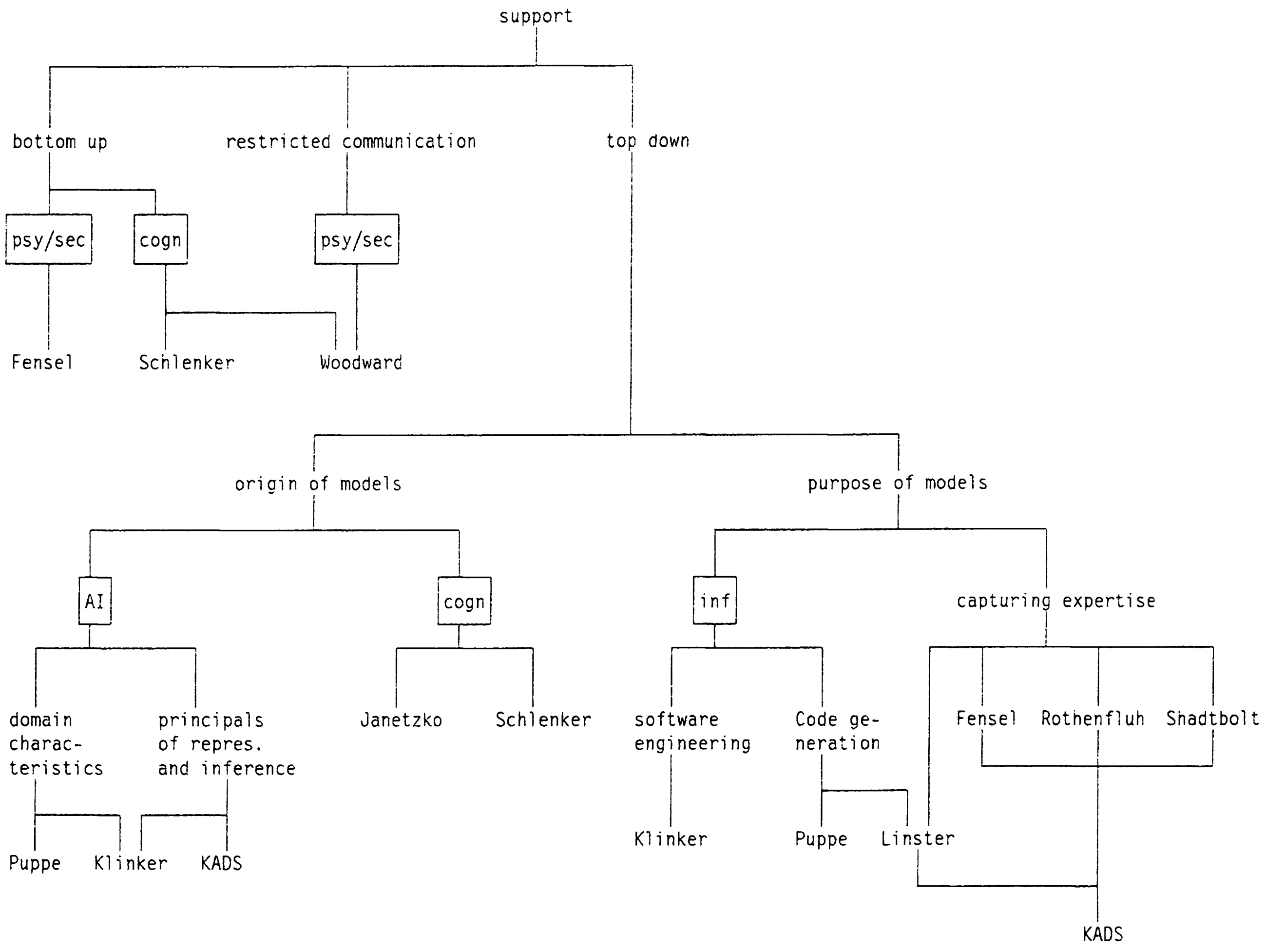


Figure 4. Ways of supporting the knowledge engineer

To finish this reverse list of skills required to apply the models, we find the trivial fact that in the case of [Puppe], the knowledge engineer has finished his work when he has implemented the computational form of the model. In particular, he is not involved in any interpretation activity and his biases (cf. [Woodward]) are not an issue. (However, what about expert biases?) The requirements on the domain expert have already been discussed above. Given the origin of the models from artificial intelligence research, Puppe's observation is probably correct that it would be too time consuming for the domain expert to create them himself.

Fig. 4 illustrates the diverse aids that are being supplied for knowledge engineering. For the top down part, KADS plays a distinguished role indicated by the connection of the four "leaf" nodes of authors Linster, Fensel, Rothenfluh, Shadbolt. That part of the figure is the only one where an attribution to one of the four disciplines does not seem appropriate. This may be an indication that we are operating in the genuine territory where knowledge engineering and cognition widely overlap.

Discussion

Since this text as a whole has the character of a discussion, this section will only be used as a reminder of the aspects of highest contrast encountered and to speculate about origins and the reasons for the divergences.

Which criteria are met by the systems produced by our methods?

Many of the contributions cannot undergo this comparison yet because they either do not cover the whole process up to running systems or are in a preliminary status. For the purpose of demonstrating extreme positions, we concentrate on Manago and Conruyt ([Manag92]) and Schlenker ([Schle92]).

[Manag92] clearly claim practical applicability of knowledge bases induced by KATE but equally clearly admit that induction algorithms supply sets of rules, some of which are hardly conceivable for the human user. Although this violates Strube's ([Strub92]) criterion of cognitive adequacy, it is not sufficient to fully disqualify KATE. Despite the inconceivability of some rules, the systems can be successfully used, which may mean that fields where KATE has been applied are factually characterized by such rules that can by no means be expressed in a form that is more appealing. Then human expertise cannot really cover such fields. Another interpretation is that inappropriate descriptions have been used as a starting point for KATE and that starting from different formalizations of the cases themselves or more appropriate use of common sense prior to induction may lead to better conceivable knowledge bases. At the present state of cognition, a certain scepticism remains as to whether a system can be valid if it cannot be understood by a human.

In the other extreme, Schlenker [Schle92] supplies some of the detail required to arrive at absolutely cognitive adequate systems in Strube's sense (cf. [Strub92]). We have argued above and also find support in [Strub92] that this is a cumbersome way to proceed, and indeed, in contrast to KATE there are no real size applications of Schlenker's method available yet. At present it is fair to say that large scale application and strong cognitive adequacy are mutually exclusive.

Are KADS conceptual models computational?

From the perspective of the originators of KADS, KADS models are non-formal and non-computational models for making sense, whereas [Fensel] and [Rothenfluh] understand them to be computational models. This may be due to the ***different socializations*** of the groups. The original KADS formulation can be characterized as pseudocode which has a certain formal and computational flavor for researchers with a background in the humanities, whereas — for researchers in formal sciences — it does not satisfy the required standards of precision, rigorosity, and formal foundation.

Are libraries of tools feasible and usable?

The Spark-Burn-Firefighter of [McDermott] has started as an attempt to collect knowledge acquisition tools in a library, to supply a meta-tool that guides the selection of the best suited individual tool(s) from the libraries, and to generate knowledge based systems by means of these tools. This has failed in a certain sense: the meta-tool would have had to ask too many questions of a far too complex nature to allow efficient use. The scope and focus of Spark-Burn-Firefighter was consequently changed towards simpler building blocks (mechanisms instead of artificial intelligence inference methods), towards homogeneous foundations in the sense of standardized description of components ("process", "resources", "outcomes"), and towards far ranging tool support for early parts of the conceptualization process (dictionary, etc.). The present claim, supported by first experimental evidence, is that the effort in producing tailored application systems is considerably reduced for the users of Spark-Burn-Firefighter.

Another claim is that the typical users of Spark-Burn-Firefighter would not be capable of applying KADS; e.g. they would not be able to make appropriate use of KADS interpretation models, while there is evidence that they make appropriate use of SBF mechanisms.

On the other hand there is a considerable number of KADS success stories and the attempt of [Shadbolt] to provide a meta-tool (ProtoKEW) to systematically select an interpretation model and tools to sytematically fill a selected one.

At least two considerations may help to resolve these apparent contradictions. The first draws upon the ***different foundations*** of early Spark-Burn-Firefighter and ProtoKEW. While [McDermott] attempted the technical integration of conceptually unrelated tools, [Shadbolt] aims at a workbench which supports one methodology of knowledge engineering (KADS). It has been a matter of intensive debate in recent workshops on knowledge acquisition whether a combination of inhomogenous tools is worth the effort, or whether a common theory as a basis of integration should precede the technicalities of software compatibility issues. The present trend seems to favor a precedence of theoretical foundation (cf. Wetter and Woodward, [Wette90]), for which [Shadbolt] is a representative.

The second consideration takes the ***education and skills*** of users into account. The typical Spark-Burn-Firefighter user (not the facilitator) is a non-programmer employee in a non-EDP department (such as sales or accounting), who needs software support for some of his work. His skills are those required in his department. The typical KADS ″worker″ has specific skills ranging from system analyst to Ph.D. in computer science with a KADS education. Needless to say, the latter may have wider ranging capacities in handling complex modelling approaches.

Must the knowledge engineer become an expert?

The bottom up approaches and [Woodward] agree upon ″yes″ while the modelling approaches do not make explicit claims but implicitly mean ″no″. This divergence has a shallow and a deep explanation. The ***superficial*** one is that the skill in applying modelling aids and model structures provides a means of penetrating a domain for the purpose of formalizing it without really understanding it: ***modelling skill*** as a ***shortcut towards learning complex subjects***?

Though this may factually be true in the sense that knowledge based systems are generated in such a way, it may ***conceal a deeper aspect.*** Modelling within a modelling paradigm means ***capturing what is within*** the scope of the paradigm but ***deforming, deliberately omitting, or just overseeing what is outside.*** Taking up the above question, modelling skill does ***not*** enable the individual to ***really learn*** a complex subject ***but*** to ***get hold of those aspects*** of the subject that can be ***subsumed under*** some facet of the ***model*** structures or languages he handles. And these are narrower and less flexible the higher the demands on precision and formality are. This is one of the reasons why informality of conceptual models is defended by a number of KADS researchers who primarily see KADS as a research environment.

And how about the personality of the knowledge engineer?

Might it be true, as [Bartsch-Spörl] claimed, that, given all the intricacies revealed during the workshop and partly reported here, the ultimate determining success factor is the intuition or other personal traits of the individual knowledge engineer? And if so, would this mean that projects beyond a certain size or complexity are not feasible because there are not enough such geniuses available? Maybe it is permissible to finish this speculative outlook by paraphrasing M.M. Richter, head of the research group who hosted the workshop: We don't require methods for those who already know, but as a basis to teach those who also need to know.

Acknowledgements

Comments from Klaus-Dieter Althoff on earlier versions have helped me to see several details clearer. Angi Voss has very carefully studied the material. Her suggestions have supported and motivated me very much to extend the scope of the text to its present size and to provide, although technically simple, graphic overviews of the approaches which may serve as an additional form of guidance into the highly diverse subject matter of the book. Last but not least, Mark Beers has supported me in writing it in English.

References

[Altho92] **Althoff, K.D. and Weß, S.** *Case-Based Reasoning and Expert System Development* in this volume

[Becke90] **Becker, B. and Bartsch-Spörl, B.** *Die Veränderung von Expertenwissen durch den Prozeß der Wissensakquisition (Modification of expertise during the process of knowledge acquisition)* KI **2** (2) 31-36 (1990)

[Bergm92] **Bergmann, R.** *Knowledge Acquisition by Generating Skeletal Plans from Real World Cases* in this volume

[Brans92] **Branskat, S.** *Knowledge Acquisition from Cases* in this volume

[Dalle92] **Dallemagne, G., Klinker, G., Marques, D., McDermott, J., Tung, D.** *Making Application Programming More Worthwhile* in this volume

[Fense92] **Fensel, D.** *Knowledge Acquisition and the Interpretative Paradigm* in this volume

[Janet92] **Janetzko, D. and Strube, G.** *Case-based Reasoning and Model-based Knowledge Acquisition* in this volume

[Hobbs85] **Hobbs, J.R. and Moore, R.C.** *Formal Theories of the Commonsense World* Norwood, NJ 1985 (Ablex)

[Linst92] **Linster, M.** *Shifting Positions : Moving from a Cognitive Science Point of View to a Knowledge Engineering Stance* in this volume

[Manag92] **Manago, M., Conruyt, N.** *Using Information Technology to Solve Real World Problems* in this volume

[Musen89] **Musen, M.** *Conceptual Models of Interactive Knowledge Acquisition Tools* Knowledge Acquisition **1**, 73-98 (1989).

[Pfeif92] **Pfeifer, R., Rothenfluh, T., Stolze, M., Steiner, F.** *Mapping Expert Behaviour onto Task-Level Frameworks: The need for "Eco-Pragmatic" Approaches to Knowledge Engineering* in this volume

[Puppe92] **Puppe, F. and Gappa, U.** *Two Questions from Expert System Developers to Cognitive Scientists* in this volume

[Reima92] **Reimann, P. and Schult, T.J.** *Learning from Problem Solving Traces* in this volume

[Schle92] **Schlenker, B. and Wetter, Th.** *Knowledge Acquisition as an Empirically Based Modelling Activity* in this volume

[Schma92] **Schmalhofer, F., Globig, C., and Thoben, J.** *Refitting of Plans by a Human Expert* in this volume

[Schre88] **Schreiber, G., Breuker, J., Bredeweg, B., and Wielinga, B.** *Modelling in knowledge based system Development* Boose, J., Gaines, B., and Linster, M. (eds.); Proc. European Knowledge Acquisition Workshop EKAW88, Bonn, July 88, GMD-Studien 143, St. Augustin 1988

[Shadb92] **Shadbolt, N.** *Facts, Fantasies, and Frameworks: The Design of a Knowledge Acquisition Workbench* in this volume

[Strub92] **Strube, G.** *Different Types of Cognitve Adequacy.* in this volume

[Tulvi73] **Tulving, E. and Thompson, D.M.** *Encoding Specifity and Retrieval Processes in Episodic Memory* Psychological Reviews *80*, 352-373 (1973)

[Wette90] **Wetter, Th. and Woodward, B.** *Towards a Theoretical Framework for Knowledge Acquisition* in: Boose, J. and Gaines, B. (eds.) Proc. 5th AAAI Knowledge Acquisition for Knowledge Based-Systems Workshop, Banff (Canada) Nov. 1990

[Woodw92] **Woodward, J.B., Shaw, M.L.G., Gaines, B.R.** *The Cognitive Basis of Knowledge Engineering* in this volume

[Glowall] **Dr. U. Glowalla** *Justus-Liebig-Universität Gießen, Fachbereich 06 Psychologie, 6300 Gießen, Federal Republic of Germany*

About the Authors

Klaus-Dieter Althoff studied mathematics and operations research at the Technical University of Aachen/Germany. From 1986 to 1990 he worked as a researcher in the German Sonderforschungsbereich 314 "Artificial Intelligence - Knowledge-Based Systems" in the projects X6 (knowledge acquisition workbench for the fault diagnosis of engineering systems) and X9 (learning and analogy for engineering expert systems). Since the beginning of 1991 he worked as a research assistant at Kaiserslautern University/Germany within the research group of Prof. Richter and is about to finish his doctoral dissertation on "A case-based learning component as an integrated part of the MOLTKE workbench for the diagnosis of engineering systems".

Ralph Bergmann is research assistant at the German Research Center for Artificial Intelligence and is currently pursuing his dissertation (PhD). He received his diploma in computer science from the University of Kaiserslautern in 1990. His primary research interests are machine learning, especially explanation-based methods, knowledge acquisition and cognitive modelling. His current address is: *German Research Center for Artificial Intelligence, University Bldg. 57, Erwin-Schrödinger-Str., D- 6750 Kaiserslautern, Germany.*

Sonja Branskat is currently pursuing her PhD at the Knowledge Science Institute of the University of Calgary. She received her B.Sc. in mathematics from the University of Freiburg, Germany in 1986 and her M.Sc. in computer science from the University Hagen, Germany in 1991. Her research interests are in knowledge acquisition and computer supported cooperative work. Her current address is: *Department of Computer Science, University of Calgary, 2500 University Drive N.W., Calgary, Alberta, Canada, T2N 1N4.*

Noël Conruyt is a research engineer at Acknowledge corporation. He received his Master's in computer science at University of Paris VI (Jussieu) in 1988 and he is currently doing his PhD at the Institut National de Recherche en Informatique et Automatique (INRIA) in collaboration with the Museum of Natural History in Paris. His research interests are in using induction and case-based reasoning to solve problems in biology (identification of marine sponges), building interactive tools to collect case libraries, designing knowledge acquisition tools which assist the user in building domain theories for machine learning and case-based reasoning tools. His current address is: *Acknowledge, 16 Passage Foubert, 75013 Paris, France.*

Geoffroy Dallemagne is a researcher at Digital Equipment Corporation's AI research group. He is currently pursuing a PhD at Ecole Centrale de Paris's Laboratory for Computer Science. He received his M.Sc. in applied mathematics from Ecole Centrale de Paris in 1988. His research interests are workplace analysis and computer supported cooperative work. His current address is: *Paris Research Lab. Digital, 85 avenue Victor-Hugo, 92563 Rueil-Malmaison Cedex, France.*

Dieter Fensel is research assistant at the University of Karlsruhe. He received a diploma degree in sociology from Freie Universität Berlin and a diploma degree in computer science from Technische Universität Berlin in 1989. His research interests are in shifting knowledge acquisition from an art to an engineering discipline and in developing and applying machine learning algorithms. His current address is: *Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, P.O.Box 6980, 7500 Karlsruhe, Germany, e-mail: fensel@aifb.uni-karlsruhe.de.*

Dr. Brian R. Gaines is Killam Memorial Research Professor and Director of the Knowledge Science Institute at the University of Calgary. He received his B.A., M.A. and PhD from Trinity College, Cambridge, and is a Chartered Engineer, Chartered Psychologist, and a Fellow of the Institution of Electrical Engineers, the British Computer Society and the British Psychological Society. His research interests include: the socio-economic dynamics of science and technology; the nature, acquisition and transfer of knowledge; software engineering for heterogeneous systems; and expert system applications in manufacturing, the professions, sciences and humanities. His current address is: *Knowledge Science Institute, University of Calgary, Calgary, Alberta, Canada T2N 1N4.*

Christoph Globig is a student in computer science at the University of Kaiserslautern and a student researcher at the German Research Center for Artificial Intelligence.

Ute Gappa is research assistent at the University of Karlsruhe and is pursuing her dissertation (PhD). She received her Diplom in Informatik (Masters in computer science) from the University of Kaiserslautern in 1988. Her primary research interests are automated knowledge acquisition from experts, graphical user-interfaces and tools for the generation of knowledge acquisition systems. Her current adress is: *University of Karlsruhe, Institute of Logic, PO Box 6980, D-7500 Karlsruhe, Germany.*

Dietmar Janetzko is a doctoral candidate at Bochum University and a research assistant in the department of cognitive science at the Institute for Computer & Social Sciences at Freiburg University. He obtained his diploma in psychology in 1987 from Bochum University. His research interests include case-based reasoning, analogy, metaphor and knowledge acquisition. His current address is: *Department of Cognitive Science, Institute for Computer & Social Sciences, Friedrichstr. 50, D-7800 Freiburg, Germany.*

Georg Klinker joined Digital Equipment Corporation in 1988 as a member of the AI research group. His research interests focus on making it easier to create application programs through reuse of software artifacts. From 1984 to 1988 he was a research scientist at the Carnegie Mellon Computer Science Department. Georg Klinker received his M.B.A. from the University of Hamburg, Germany in 1983. While in Hamburg he worked for the software company IfaD.

Dr. Marc Linster is a member of the AI Research Division of GMD (German National Research Center for Mathematics and Dataprocessing). His work is focussing on problem-solving methods and their influence on (automated) knowledge acquisition. He obtained a doctorate ---concerned with the above-mentioned problems--- from the University of Kaiserslautern, where he had also received his diploma in computer science in 1987. His current address is: *AI Research Division, GMD, PO Box 1240, 5205 St. Augustin, FRG.*

Dr. Michel Manago is the chief scientist at Acknowledge corporation. He received his B.Sc. in computer science from the University of Illinois in Urbana Champaign in 1983 and his PhD at University of Orsay (France) in 1988. His research interests include inductive learning and case-based reasoning technologies, integration of symbolic and numeric methods, theoretical foundations of machine learning and building computer tools for the analysis of the human genome (genetics). He is involved in using machine learning and case-based reasoning for applications which deal with finance (credit assessment), identification (photo-interpretation), analysis in clinical databases, telecommunications and technical maintenance (fault diagnosis, preventive maintenance). His current address is: *Acknowledge, 16 Passage Foubert, 75013 Paris, France.*

David Marques has been a research scientist in Digital Equipment Corporation's Artificial Intelligence Research Group for the past 5 years. His research interests are in enabling end-users to do their own programming (through software reuse), and in understanding (through modeling) the context of end-user activities. He received his A.B. in psychology from Cornell in 1972, PhD in psychobiology from the University of Michigan in 1976, and did research in neuroscience before joining Digital Equipment Corporation in 1982 in the Software Services organization.

Dr. John McDermott is a member of the technical staff at Digital Equipment Corporation. He has been a member of the faculty of the Computer Science Department at Carnegie-Mellon University since 1974. In 1983 he confounded the Carnegie Group, Inc. He received a PhD in philosophy from the University of Notre Dame in 1969. His research interests are the application of AI techniques to industrial problems and artificial intelligence in general.

Dr. Rolf Pfeifer is a full professor of computer science and heads the AI Lab at the Institute for Informatics at the University of Zurich in Switzerland. After receiving his M.Sc. in physics and mathematics and his PhD (with a thesis on AI and psychological modeling) at the Swiss Federal Institute of Technology (ETH), he spent three years in the USA at Carnegy Mellon University in Herb Simon's group and at Yale in Bob Abelson and Roger Schank's cognitive science lab. His main research interests are foundations of AI and cognitive science, autonomous agents, and "situated design". His current address is: *AI Lab, Institute for Informatics, University of Zurich, Winterthurerstrasse 190, CH-8057 Zurich. E-mail: pfeifer@ifi.unizh.ch.*

Dr. Frank Puppe is professor for informatics at Würzburg University. He received his PhD 1986 from Kaiserslautern University and his habilitation about "Problem Solving Methods in Expert Systems" 1991 from Karlsruhe University. His research interests include strong problem solving methods in expert systems, graphical knowledge acqusition, intelligent tutoring systems and machine learning as well as their evaluations in technical and medical applications. His current address is: *Universität Würzburg, Institut für Informatik, Am Hubland, D-8700 Würzburg.*

Dr. Peter Reimann is lecturer at the Department of Psychology, University of Freiburg, FRG. He received his M.Sc. in psychology in 1984 and his PhD in 1989 from the University of Freiburg. His research interests are the computational psychology of learning, memory and problem solving as well as applications in intelligent tutoring and testing systems and man-machine interaction. His current address is: *Dept. of Psychology, Univ. of Freiburg, Niemensstr. 10, D-7800 Freiburg, FRG.*

Dr. Thomas Rothenfluh is currently a Visiting Scholar at The Ohio State University in Columbus, Ohio. He received his PhD in psychology from the University of Zurich in 1988 and is now supported with a 3-year scholarship from the Swiss National Science Foundation. His main research interests are in cognitive science, knowledge systems and psychological modelling. His current address is: *Laboratory for Artificial Intelligence Research, Dept. of Computer and Information Science, The Ohio State University, 2036 Neil Ave. Mall, Columbus, OH 43210-1277.*

Beate Schlenker is currently pursuing her Diplom-Degree in psychology at the University of Freiburg. Her primary research interest is knowledge acquisition. Her current address is: *Am langen Graben 41, 5300 Bonn 3.*

Dr. Franz Schmalhofer is a senior scientist and leader of the knowledge acquisition group at the German Research Center for Artificial Intelligence (DFKI) in Kaiserslautern. He has studied mathematics, psychology and computer science at the University of Regensburg/Germany and the University of Colorado at Boulder, where he earned his PhD in 1982. He has held academic positions at the Universities of Heidelberg/Germany, Freiburg/Germany and the McGill University of Montreal/Canada. His research lies mostly in the areas of cognitive modelling, knowledge acquisition, machine learning, expert systems, planning, human-computer interaction, text comprehension and decision research. *email: schmalho@informatik.uni-kl.de.*

Thomas J. Schult is research scientist at the Department of Psychology, University of Freiburg, Germany. He received his B.Sc. in mathematics from the University of Freiburg in 1988, and his diploma in computer science from the University of Hagen in 1990. His research interests are in case-based reasoning and in intelligent tutoring systems. His current address is: *Dept. of Psychology, Univ. of Freiburg, Niemensstr. 10, D-7800 Freiburg, Germany.*

Dr. Nigel Shadbolt is professor of Intelligent Systems at the University of Nottingham, England. He received his B.A. in philosophy and psychology from Newcastle upon Tyne University in 1978 and his PhD from the Department of Artificial Intelligence at the University of Edinburgh in 1984. His research interests are in the areas of knowledge acquisition, and the foundations of agent design. His current address is: *The Artificial Intelligence Group, Department of Psychology, University of Nottingham, University Park, Nottingham NG7 2RD, England.*

Dr. Mildred L. G. Shaw is Professor of Computer Science at the University of Calgary. She received her B.Sc. and M.Sc. from the University of London, and her PhD from Brunel University and is a Chartered Mathematician and Chartered Psychologist. She is a Fellow of the Institute of Mathematics and its Applications and the British Computer Society and an Associate Fellow of the British Psychological Society. Her research interests include: human-computer interaction; the acquisition and transfer of knowledge; software engineering; and expert system applications. Her current address is: *Knowledge Science Institute, University of Calgary, Calgary, Alberta, Canada T2N 1N4.*

Felix Steiner is currently working on knowledge based systems for customer credit assessment at Credit Suisse, Zurich. He received his degree in clinical psychology from the University of Zurich, Switzerland in 1981. His interests are in knowledge engineering, knowledge acquisition, and connectionism. His current address is: *Credit Suisse, P.O. Box 590 / Okb39, CH-8021 Zurich.*

Markus Stolze is currently persuing his PhD at the AI Lab, University Zurich. He received his M.Sc. in computer science from the University of Bern, Switzerland in 1988. His primary research interests focus on the design of interactive knowledge based systems. His current address is: *Institute for Informatics, Winterthurerstrasse 190, CH-8057 Zurich. E-mail: stolze@ifi.unizh.ch.*

Dr. Gerhard Strube is professor of cognitive science at the University of Freiburg, Germany. He received Mag.theol. and Dipl.-Psych. degrees in 1973 and 1974, Dr. phil. in cognitive psychology 1977, Dr. phil. habil. 1983 (University of Munich). 1982-1987 senior scientist at the Max Planck Institute for Psychological Research, Munich, 1987-1991 full professor of cognition and human information processing at the Ruhr University, Bochum. His research interests focus on computer models of language processes, acquisition and representation of knowledge, and reasoning. His current position is director of the *department of cognitive science, Institut für Informatik und Gesellschaft, Universität Freiburg, Friedrichstr. 50, D-7800 Freiburg i. Br., Germany.*

Jörg Thoben is research assistant at the German Research Center for Artificial Intelligence and is currently pursuing his dissertation (PhD). He received his diploma in psychology from the University of Münster in 1992. His primary research interests are mental models, cognitive modelling and knowledge acquisition. His current address is: *German Research Center for Artificial Intelligence, University Bldg. 57, Erwin-Schrödinger-Str., D- 6750 Kaiserslautern, Germany.*

David Tung is a researcher/principal software engineer at Digital Equipment Corporation, AI Research Group. His research interests include knowledge acquisition, software reuse, and workplace analysis. He received his B.Sc. in electronic engineering from University of London, and M.Sc. in artificial intelligence from University of Edinburgh. He is a member of IEEE, AAAI, ACM, and EE. His current address is: *DEC, 111 Locke Drive, LMO2-1/K11, Marlboro MA 01752.*

Dr. Angi Voss is working at the AI Research Division of the German National Research Center for Computer Science (GMD), where she is responsible for a subdivision on knowledge modelling. She received her diploma in informatics at the University of Bonn, and her doctoral degree at the University of Kaiserslautern. Her current address is: *GMD, AI Research Division, Schloss Birlinghoven, D-5305 St. Augustin 1, e-mail: avoss@gmdzi.gmd.de.*

Stefan Wess studied computer science and operations research at the University of Kaiserslautern. From 1987 to 1990 he worked as a student researcher in the German Sonderforschungsbereich 314 "Artificial Intelligence - Knowledge-Based Systems" in the projects X6 (knowledge acquisition workbench for the fault diagnosis of engineering systems) and X9 (learning and analogy for engineering expert systems). He received his diploma in computer science in 1990. Since 1991 he is working as a researcher in a Case-Based Reasoning Project at the University of Kaiserslautern.

Dr. Thomas Wetter studied mathematics and computer science at the Technical University of Aachen. He investigated several topics in mathematical modelling of physiological and medical phenomena and received his PhD in mathematics about logic based medical diagnosis in 1984. His affiliation now is the IBM Germany Scientific Center, Heidelberg. There he has been working in software ergonomics, expert systems, and knowledge acquisition. He teaches graduate courses in AI at the universities of Heidelberg and Kaiserslautern.

Dr. Brian Woodward is a research associate with the Knowledge Science Institute in the Department of Computer Science at the University of Calgary. His research interests include the design and development of cognition support software, training and educational simulations, and management decision-making. He obtained his PhD in educational psychology from the University of Calgary in 1978 and works as an industrial/organizational psychologist specializing in management selection and development.

Univ.-Bibliothek
Regensburg