

# Knowledge Acquisition and Learning by Experience - The Role of Case-Specific Knowledge

Agnar Aamodt

*University of Trondheim, Department of Informatics, N-7055 Dragvoll, Norway*  
(Email: [agnar@ifi.unit.no](mailto:agnar@ifi.unit.no))

## **Abstract**

As knowledge-based systems are addressing increasingly complex domains, their roles are shifting from classical expert systems to interactive assistants. To develop and maintain such systems, an integration of thorough knowledge acquisition procedures and sustained learning from experience is called for. A knowledge level modeling perspective has shown to be useful for analyzing the various types of knowledge related to a particular domain and set of tasks, and for constructing the models of knowledge contents needed in an intelligent system. To be able to meet the requirements of future systems with respect to robust competence and adaptive learning behavior, particularly in open and weak theory domains, a stronger emphasis should be put on the combined utilization of case-specific and general domain knowledge. In this chapter we present a framework for integrating KA and ML methods within a total knowledge modeling cycle, favoring an iterative rather than a top down approach to system development. Recent advances in the area of case-based reasoning provide a suitable basis. Focus is put on the knowledge maintenance part, for which a case-based method for learning from experience is described and exemplified by existing systems. Our own approach to integration of case-specific and general domain knowledge (the CREEK system) is briefly sketched, and used as a context for discussing case-based approaches to knowledge modeling in general.

## **1. Introduction.**

The quality of a knowledge-based system stands and falls with how well relevant parts of the application domain are captured by the system's knowledge. Most knowledge-based systems in use today contain highly domain-specific knowledge, targeted at a particular, narrow application. The knowledge is typically captured in terms of heuristic rules, interconnected in a chained fashion and targeted at a particular, single problem. An advantage of this approach is that knowledge is captured as general experience, based on what humans have found useful for solving problems in the domain. This often leads to a targeted and efficient problem solving process. Another advantage is that the representation language for that knowledge can be rather simple. The development of such systems is often supported by commercial, user-friendly tools. The knowledge acquisition problem, although responsible for a large part of the development effort needed, is coped with by focusing on one type of knowledge (if-then rules, sometimes augmented with object hierarchies of rule terms) for one application task (e.g. diagnosis of heart failure, assessment of credibility for loan, planning of transportation routes). The widespread commercial use of rule-based expert systems (Harmon and Maus, 1988), (Turban, 1992), (CACM, 1994) is a solid manifestation of the success of the rule-based, domain-narrow approach.

However, there are also well-known problems related to the rule-based approach. An example is the lack of robustness and flexibility in problem solving due to the narrow and tailored scope of the knowledge. Another example is the difficulties in maintaining and updating a system's knowledge over time, to cope with the normal development of the subject field and changes in the system's operational environment. This has led to a focus on methods that combine heuristics with more principled and deep knowledge (a collection of recent research can be found in (David et al., 1993)). Hence, *general* domain knowledge can in principle be captured as shallow, heuristic models - e.g. a set of if-then rules - or as models of deeper, more principled knowledge.

Advancements within the machine learning field related to inductive learning in real world domains have moved machine learning methods from the study of concept formation in general, into the development and maintenance of knowledge-based systems. Results from the subarea of multistrategy learning (Michalski and Tecuci) have particularly contributed to this development. However, the generalization problem, i.e. determining what to generalize from a situation, how far to generalize, what type of generalized descriptions to learn, etc. is still a basic and difficult one. In parallel to the developments in this type of learning, recent research has also studied the direct capturing and utilization of the concrete and *specific* knowledge related to problem solving experiences. The growth of research and development in the field of *case-based reasoning*, has thrown light on the significance of specific problem solving episodes for the solving of new problems and for learning (Kolodner, 1993), (Wess et al., 1994). While traditionally viewed as data or information rather than knowledge, concrete descriptions of past problem solving episodes become knowledge for case-based reasoning methods, since these methods are able to use cases for drawing inferences, i.e. for reasoning. This has opened up for AI methods based on *memorizing* as a core process of learning, and *reminding* and analogical *reuse* as core methods of reasoning.

There has been a clear development trend over the last years from single-task expert systems to multi-functional knowledge-based systems: A medical knowledge-based system, for example, instead of only suggesting a diagnosis should also be able to support the user in setting up an examination protocol, revising the test scheme according to results, deriving consequences of measurements, criticizing the user's decisions, pointing out missing risk-checks, etc. The history of AI strongly indicates that no single type of knowledge is sufficient to realize this range of functionality, and that combined methods are called for. Knowledge bases are required that can be used for multiple tasks (sometimes referred to as multifunctional knowledge bases), associated with methods that are able to utilize the different types of knowledge in a combined way. There is also a trend towards a merging of knowledge-based and other system types, such as information systems, databases, user-interfaces, hypermedia systems, and decision support systems in general. Knowledge-based systems - or system components - are to an increasing degree viewed as *intelligent decision support* systems, i.e. heavily *user-oriented* systems that cooperate with the human user in the solving of problems<sup>1</sup>, rather than as expert systems in the classical sense (Boy, 1991), (Aamodt, 1994). Improving the methods for development and maintenance of this type of knowledge-based systems is what this paper is about.

The main issue we address is: How can recent advances in knowledge acquisition (KA) and machine learning (ML) be combined, so that the development and continued maintenance of such systems become more feasible? A basic assumption made is that knowledge-based systems need to acquire two types of capabilities: First, they need a certain level of *competence*, i.e. a body of knowledge that is sufficiently deep to cope with the complexity of real world problems, and sufficiently broad to cope with the variety of tasks in the domain. They should also exhibit a smooth decay for problems outside the core domain. Second, as systems solve problems or assist the user in other ways, they should gradually become more *experienced*. While competence refers to a potential ability based on an established platform of knowledge, experience refers to operational capabilities based on the continually growing number of problems that a system has solved.

As human beings, we are competent and robust problem solvers in our domains of competence since we possess a coherent body of general knowledge, and a huge library of specific past problem solving cases (see for example (Schank, 1982)). We base our methods on a general understanding of the task environment in which we operate, and we maintain a memory of past problem-solving episodes integrated into this fundamental knowledge structure. Our ability to maintain and coherently utilize a variety of knowledge types has been documented by research in cognitive psychology (Tulving, 1972), (Anderson, 1990) and within the interdisciplinary field of cognitive science (Simon, 1989), (Strube, 1991). When solving problems or trying to understand situations, we are frequently reminded of similar previous experiences. Using more general knowledge as support, we are able to adapt the solution (or solution path) of a previous case to the solving of a new problem. Through success and failure in achieving our task, we learn to do things better the next time. Although artificial intelligence methods do not necessarily have to duplicate methods of human intelligence, cognitive science has provided highly valuable input to the development of knowledge-based systems in general, and to knowledge acquisition and learning methods in particular (see for example the collection of papers in (VanLehn, 1990)). This is particularly relevant for the type of systems discussed here, since cooperative man-machine problem solving should be based on a common ‘understanding’ of domain terms and input data, which in turn is facilitated by some degree of similarity of knowledge organization, interpretation methods, etc., between human and computer agents.

An important challenge for future AI methods will be to combine and synergetically utilize general and case-specific domain knowledge. This is therefore a major issue with respect to the integration of KA and ML methods. The challenges of more complex and user-oriented systems call for a change of view on knowledge acquisition and machine learning. The two main issues motivated and described in this chapter are:

- Knowledge acquisition research has until now mainly focused on the ‘first round’ of system development, i.e. up to the delivery of the system to the end-user. A stronger emphasis on continued ever-lasting knowledge maintenance (refinement, extension, adaptation) is needed. On this basis, we advocate that systems development in general should be more of an iterative process than a top-down one, and that the integration of KA and ML should be driven by the needs of such iterative approaches.

- Knowledge-based systems has until now mainly focused on utilizing general domain knowledge for solving problems. The difficulty of modeling this type of knowledge, and its insufficiency for a range of problem solving and learning tasks, has lead to the increased interest in case-based problem solving and learning methods. On this basis, we advocate that a stronger focus needs be put on the synergetic combination of case-specific and general domain knowledge, with implications for knowledge modeling, problem solving methods, and learning methods.

The next section introduces the integration issue by describing some commonalties and differences between knowledge acquisition and machine learning methods. This is followed by a general framework for knowledge modeling that identifies basic types of modeling tasks and partial models. The framework provides a discussion ground for KA+ML integration by combining case-specific and general knowledge. In section 4 the case-based reasoning process is elaborated and exemplified by existing methods and systems. A way to integrate KA+ML methods, building upon the assumptions and modeling principles earlier described, is suggested in section 5, followed by a summary of our own integration approach - the CREEK system. In the conclusion section, the results presented are summarized, and consequences for future KA and ML research are discussed.

## **2. Knowledge modeling, knowledge acquisition, and learning.**

### *2.1. General presentation*

The process of constructing an appropriate body of knowledge for a system to perform its tasks is here generally referred to as *knowledge modeling*. Along with (Clancey, 1989) a knowledge-based system can be viewed as a qualitative model of that part of the real world which the system is to reason about. Knowledge modeling is here the whole process that starts with a real world task environment, and realizes a (partial) model of it in a computer system. It is a general term that covers tasks such as elicitation, analysis, construction, representation, implementation, validation, and maintenance of knowledge, and can be viewed as a synonym to knowledge acquisition in the wide sense. By using this term we want to advocate a strong “modeling” view of knowledge acquisition (Ford et al., 1993), i.e. the view that knowledge acquisition is a constructive modeling process (Morik, 1990), as opposed to the “knowledge extraction” or “knowledge transfer” view assumed by earlier accounts (e.g. Boose, 1986). The knowledge is constructed in a cooperative modeling effort between the domain expert(s) and knowledge engineer(s), and may not have explicitly ‘existed’ anywhere before (Clancey, 1993), (Gaines, 1993). Correspondingly, the modeling view also implies that machine learning is regarded as a constructive knowledge modeling process. Any knowledge modeling task may be accomplished by largely manual methods (knowledge acquisition methods), and/or by automatic ones (machine learning methods). We will refer to methods for knowledge analysis and model building that are largely manual, and that assume a cooperation between one or more developers and one or more domain experts (jointly referred to as the “development team”), as *knowledge acquisition*. ‘Largely manual’ also includes computer-assisted methods and the use of automated tools, including inductive

learning methods as support for manual conceptualization. We will refer to methods that are largely automatic, and assume that the computer itself is able to generate useful knowledge on the basis of input data, as *machine learning*. 'Largely automatic' here also includes manual input and assistance in the learning process.

The problems of increased complexity and extended functionality of systems, as pointed out in the introductory section, are being addressed within the AI community in various ways. An important trend is the emphasis on *knowledge level* analysis now adopted by most KA methodologies (Van de Velde, 1993). Other trends are the integration of knowledge-based components with information system and database parts (Wielinga et al., 1993), and the increased emphasis on modeling for highly user-interactive systems (Clancey, 1993). Examples from machine learning include the growing amount of work on knowledge-intensive learning (Kodratoff and Tecuci, 1987a), (Aamodt, 1990b), (Leake, 1993), multistrategy learning methods (Michalski and Tecuci, 1990), methods that incorporate the user in the learning loop (Mitchell et al., 1985), and case-based methods for learning by experience (Kolodner, 1993), (Aamodt and Plaza, 1994). We will describe some of these approaches in more detail below.

A major requirement for the type of systems we are discussing here is their ability to *adapt*, in some way, to a continually *evolving environment*. It is for practical reasons unfeasible to rely solely on manual updating and refinement procedures for maintaining the knowledge over time. If a system is to continually maintain and improve its problem solving competence, it will therefore need to incorporate methods for automated *learning by experience*. An increasing amount of machine learning research is concentrating on incremental methods that enable learning while solving real problems. This includes approaches for learning of generalized knowledge (e.g. (Tecuci, 1988), (Van de Velde, 1988)) as well as for learning of specialized knowledge (Riesbeck and Schank, 1989), (Porter et al., 1990), (Aamodt, 1991). These activities have shown promising results that have encouraged more intensive research into sustained learning methods for real world knowledge-based systems applications. Results achieved within the following three research areas are of particular relevance:

*Knowledge-intensive learning.* Most of the current research on knowledge intensive learning methods is gathered under the term *explanation-based learning*, of which an early overview was given in (DeJong, 1988). Knowledge intensive learning methods differ from approaches based on superficial and syntactic similarity and discrimination criteria. They include deductive methods based on a complete domain theory (EBL/EBG, see (Mitchell et al., 1986) and (DeJong and Mooney, 1986)), methods to generate and use plausible explanations in an incomplete knowledge base (Schank et al., 1986), (Lenat and Guha, 1989), knowledge-intensive case-based learning/case-based reasoning (Hammond, 1989), (Koton, 1989), (Porter et al., 1990) and analogical reasoning and learning methods (Kedar-Cabelli, 1988), (Kodratoff, 1990).

*Apprenticeship learning.* The notion of *learning apprentice* systems was introduced in (Mitchell et al., 1985) as "interactive knowledge-based consultants that directly assimilate new knowledge by observing and analyzing the problem solving steps contributed by their users through their normal use of the system". This does not represent a particular learning method or set of methods, but a

general approach to sustained learning where new knowledge is continually acquired through observation and analysis. The approach is also well-suited to the application of semi-automatic learning methods, since an apprentice should be allowed to ask questions in order to increase its understanding (Kodratoff and Tecuci, 1987b), (Murray and Porter, 1989).

*Case-based reasoning.* Learning of specific knowledge in terms of past cases represents the major approach to sustained learning in today's machine learning research. The learning becomes a process of extracting relevant information from a problem solving experience, and indexing this case in the system's knowledge structure in a way that facilitates retrieval when a similar problem is later encountered. The case based approach to reasoning and machine learning has had a considerable growth during the last few years (Kolodner, 1993), (Richter et al., 1993), (Aamodt and Plaza, 1994). Earlier research of fundamental importance to this field includes Schank's and Kolodner's work on memory structures for learning and reasoning (Schank, 1982), (Kolodner, 1983) and the work on transformational and derivational analogy by Carbonell (Carbonell, 1983), (Carbonell, 1986).

A few attempts have been made to develop systems based on an integration of methods from all these three areas. One example is the Protos system (Bareiss, 1989), handling the problem of classifying auditory diseases. Ongoing work at the University of Trondheim also focuses on such an integration (Aamodt, 1994). While Protos emphasizes on the apprenticeship approach to case-based learning, our approach is stronger on model-driven, knowledge-intensive support to the CBR processes. Further, and unlike Protos, our approach does not advocate sustained case-based learning as an *alternative* to initial knowledge acquisition and model building, but as *complementary* to it. However, we shall see that by attacking the knowledge maintenance task with a method for sustained learning from experience, the initial knowledge modeling task is also relieved of some of its burdens.

## 2.2. Domain and task characteristics

The type of applications we address here is real-world problem solving and interactive decision support in open and weak theory domains. An *open* problem domain is characterized by incompleteness and frequent changes. Heavy interaction and feedback between a reasoning agent and its external environment is therefore required for successful decision making. A *weak theory* domain is characterized by uncertain relationships between its concepts (Porter et al., 1990). Typical open and weak theory domains are medical diagnosis, geological interpretation, investment planning, and most engineering domains (i.e. domains that involve interaction with the external world). The stronger a theory, the more certain are its relationships. Domains with strong domain theories are, for example, mathematical domains, closed technical domains, and some games, such as checkers and chess. Even some strong theory domains may incorporate problems that turn out to be open when addressed by a problem solver. Chess, for example, has a strongest possible - a complete - domain theory, but solving the problem of chess-playing by an implementation of the theory is violated by its intractability: The concept of a 'winning plan' in chess is theoretically deducible, but there is no efficient algorithm to infer it in the general case. Some closed and

in principle well-understood technical domains, such as fault-finding in VLSI-circuits, are subject to the same difficulty.

The fact that a domain is open and has a weak theory does not necessarily imply that there is little domain knowledge available. It only implies that the knowledge is theoretically uncertain and incomplete. In order to compensate for the lack of strong knowledge for which deductive, proof-oriented methods would apply, such systems therefore typically need a larger body of knowledge than systems for strong-theory domains. The knowledge should capture the domain and task reality from different perspectives which together constitute a coherent model. Such knowledge is typically used for reasoning based on *abductive* methods, i.e. methods of inference based on hypothesis generation and evaluation by producing the strongest possible explanations for hypothesis support and justification<sup>2</sup>.

A strong emphasis on explanations as an essential part of the reasoning process requires a thorough and deep type of knowledge in order to produce meaningful explanations. There are in general many different perspectives to a body of knowledge, as well as to a single concept. A car, for example, means different things to a car mechanic and to an environmental activist. The meaning of interdependencies between concepts, constraints, and other relationships is determined by the concepts' interpretation within a particular context. For a knowledge-intensive approach to problem solving and learning, knowledge has to be interpreted with respect to its purpose and intended use, within particular problem solving and learning contexts (Compton and Jansen, 1989), (Chandrasekaran, 1992).

### 3. A framework for knowledge modeling

In this section a generic knowledge modeling cycle is presented. It is based on the combination of a basically top-down driven, constructive modeling approach to initial knowledge acquisition, and the bottom-up modeling view represented by continuous learning through retaining problem solving cases. Within this context, the more specific issues of acquiring an initial knowledge model, representing knowledge models, and achieving sustained learning are discussed.

#### 3.1. General presentation

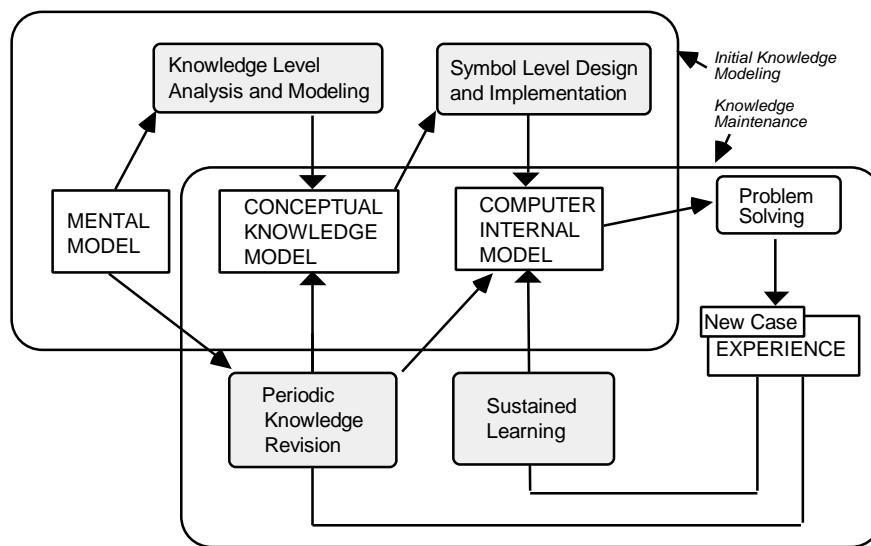
For the purpose of integrated KA and ML modeling along the lines we have drawn here, it is appropriate to split the overall knowledge modeling process, or cycle, into two top-level, successive tasks: *Initial knowledge modeling* and *knowledge maintenance*.

The objective of the *initial knowledge modeling* task is to analyze the domain and task in question, to develop the conceptual, mediating models necessary for communication within the development team, and to design and implement the initial operational and fielded version of the system. Initial knowledge modeling, in this sense, covers all phases up to the realization of a computer system according to specifications. Incomplete as this initial model may be, it constitutes the knowledge environment wherein problem solving initially takes place, and from which experiential learning starts out.

The *knowledge maintenance* task takes over where the initial knowledge modeling ends, and its objective is to ensure the refinement and updating of the

knowledge model as the system is being *regularly used*. This includes to correct errors and improve the knowledge quality, to improve performance efficiency, and to adjust system behavior according to changes in the surrounding environment, such as changing the type of users interacting with the system or the type of use made of it. The knowledge maintenance task continues throughout the entire lifetime of the system.

In figure 1 the two outer, large boxes (with rounded corners) illustrate these two top-level tasks of the knowledge modeling cycle. Within each of the two tasks, the major subtasks (round corner rectangles) and models (sharp corner rectangles) taken



**Figure 1: The knowledge modeling cycle**

as input and returned as output from these tasks are shown. The modeling subtasks are indicated by gray background. Arrows indicate the main flow of knowledge and information, and show the most important input/output dependencies between subtasks and models. As shown by the area where the two large boxes overlap, the conceptual knowledge model and the computer internal model are shared by subtasks of both initial knowledge modeling and knowledge maintenance.

A knowledge modeling cycle typically starts with a high level specification (e.g. functional specification) of the target computer system, at some level of detail. As interpreted by the *mental models* possessed by the human beings involved, this forms the basis for analyzing the relevant task types, domain models, and problem solving strategies in question. The resulting submodels are structured into a *conceptual knowledge model*. The purpose of this model is to capture and structure various types of knowledge from the perspective of the application domain and the tasks of the target computer system, without being constrained by implementational limitations. The knowledge is described at what is referred to as the *knowledge level*, where the emphasis is to capture the *goal-directed behavior* of the system, its interaction with the environment, and the real world *content* of the relevant knowledge types. The result of this intermediate stage should be a

model of domain knowledge and problem solving methods suitable for communication among the developers, and for getting a thorough understanding of the application domain and the system's tasks in relation to the external task environment. Computer-assisted analysis and modeling tools, and prototype building in order to get feedback for improving the conceptual model, are typical means of developing this model.

Once the conceptual knowledge model is in an acceptable state, it forms the basis for designing and implementing the *computer internal model*, i.e. the knowledge model of the operating target system. This model is described at a level referred to as the *symbol level*, which deals not only with intentional knowledge content, but with manipulation of *symbols* that *represent* knowledge for the computer. The type of 'transfer method' suitable for going from the conceptual model to the computer internal one, depends on how close the conceptual model is to an executable one. It may be a rather simple transfer process close to copying or minor refinement, or a full rebuilding using the conceptual model merely as a specification.

The lower, partially overlapping box illustrates the main subtasks of knowledge maintenance. It is important to note that the knowledge maintenance phase starts when a system has been put into regular operation and use. All prototyping, testing and refinement that are parts of the pre-operational development process, including knowledge validation, are covered by the initial knowledge modeling task. The knowledge maintenance task has two optional subtasks as indicated in the figure. The one we focus on here is *sustained learning*, i.e. the direct updating of the computer internal model each time a new problem has been solved. The other subtask involves a periodic and more substantial revision process, i.e. a more thorough analysis, which in this model is assumed to be made after some amount of new experience has been gathered, and not after each problem solving session. As illustrated, this revision task may lead directly to the modification of the symbol level model (computer internal model), but it may also go through an update of the knowledge level model (conceptual knowledge model) first<sup>3</sup>. The sustained learning task, on the other hand, regards each new problem solving episode, i.e. each problem just solved, as a source for immediate learning. This implies that the knowledge model (read: the knowledge base) is updated each time a problem is solved. As we shall see, case-based reasoning is a problem solving and learning approach highly suitable for this type of learning.

A crucial issue for all the modeling tasks is how to represent the knowledge in a way expressive enough to capture all relevant knowledge, efficient enough to make the knowledge available when needed, and close enough to a human interpretable language to facilitate manual inspection and refinement. The representation problem applies to the representation of a conceptual knowledge model as well as of the computer internal model, and in particular to the transfer between the two models. Hence, orthogonal to the two top-level knowledge modeling tasks, we can identify the following three core problems related to each of them:

- The knowledge acquisition problem
- The knowledge representation problem
- The learning problem

In the following three subsections, each of these problems will be elaborated - within the context of combining case-specific and general domain knowledge, and related to the integration of KA and ML methods. Each subsection is introduced by naming a development trend that has had - and still has - important impact on the problem addressed. It is natural to start with the knowledge acquisition problem, since some characteristics of the task and domain should be analyzed before the representation problem is addressed. The representation problem, in turn, has to be addressed before the learning problem. Based on recent research, we shall see that a promising way of KA and ML integration is to rely on a tool-assisted knowledge acquisition methodology for the initial knowledge modeling task<sup>4</sup>, and a machine learning method that continually learns case-specific knowledge by experience, for the knowledge maintenance task.

### 3.2. *The Knowledge acquisition problem*

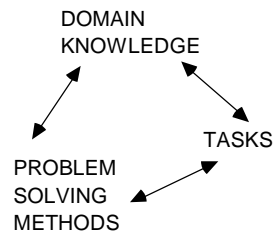
The important development trend here is the increased emphasis on *knowledge level* modeling. The adoption of the knowledge level as the most suitable descriptive level for conceptual knowledge modeling is shared by most knowledge acquisition methodologies recently developed. Well known examples are the KADS methodology (Wielinga et al., 1992), the Components of Expertise framework (Steels, 1990), the Generic Tasks (Chandrasekaran, 1992) approach, Role Limiting Methods (McDermott, 1988), and the 'method-to-task' approaches underlying the PROTEGE systems (Musen, 1989), (Puerta et al., 1991). Attempts to unify several existing viewpoints and methodologies within a knowledge-level perspective are also under way, as exemplified by the multiple perspective approach of the KREST methodology (Steels, 1993) - based on the Components of Expertise framework, and by CommonKADS (Wielinga et al., 1993).

In Newell's original paper (Newell, 1982) the knowledge level was proposed as a distinct level of description of computer systems, defined to lie above the level of data structures and programming languages. The latter was referred to as the *symbol level*. The knowledge level has knowledge (goals and means to attain them) as its medium and the principle of rationality as its basic behavioral law<sup>5</sup>. At the symbol level, the medium is symbols (data structures and programs), and the behavioral law is sequential interpretation of procedures. A system is described at the knowledge level as an intelligent agent with its own goals and with knowledge of how to achieve its goals. The principle of rationality states that an agent always will use its knowledge in a way that ensures the achievement of its goals - provided the agent has the knowledge needed.

The ways in which the knowledge level perspective is utilized within a particular knowledge acquisition methodology varies. However, it seems to be a consensus that knowledge can be grouped into three main types, or viewed from three perspectives: Tasks, domain knowledge, and problem solving methods (see figure 2). Tasks are defined by the goals that a system tries to achieve. Problem solving methods are used to accomplish tasks (i.e. to solve problems). Domain knowledge<sup>6</sup> is needed for methods to accomplish their tasks.

The original knowledge level idea has undergone some modification over the years, from Newell's highly intentional, purpose-oriented way of describing a system, to a somewhat more structured and useful type of description. This transition has also lead to modifications of the knowledge level notion itself, associated with terms such as the "knowledge use level" (Steels, 1990), a

"knowledge level architecture" (Sticklen, 1989), and the notion of "tractable rationality" (Van de Velde, 1993). The original notion of knowledge level has been extended by introducing high-level structural and methodological constraints. This makes the knowledge level more operational and useful for conceptual knowledge modeling purposes, while retaining its competence-oriented and implementation-independent aspects.



**Figure 2: Knowledge perspectives**

A number of tools have been - and are being - developed to support knowledge level modeling. They range from relatively general tool boxes, such as the KEW system developed in the Acknowledge project (Jullien et al., 1992), to strongly methodology-driven workbenches, usually including libraries of reusable modeling components. Some of these approaches are aimed at knowledge level modeling only (e.g. Wielinga et al., 1992), while others attempt to provide a bridge to a symbol-level realization as well (e.g. (Klinker et al., 1991), (Linster, 1992)).

### 3.3. The knowledge representation problem

The important development trend here is the increased focus on capturing *knowledge content* in representation systems, as opposed to a focus on formal neatness and proof-deductive properties. Discussions related to the CYC system (Guha and Lenat, 1990), and the accompanied CYCL language, have posed important questions and provided new insight into core issues of knowledge representation.

The knowledge representation problem applies for knowledge level as well as symbol level modeling. Representation languages at the knowledge level, also referred to as conceptual modeling languages, are part of modeling tools such as KREST and the CommonKADS workbench.

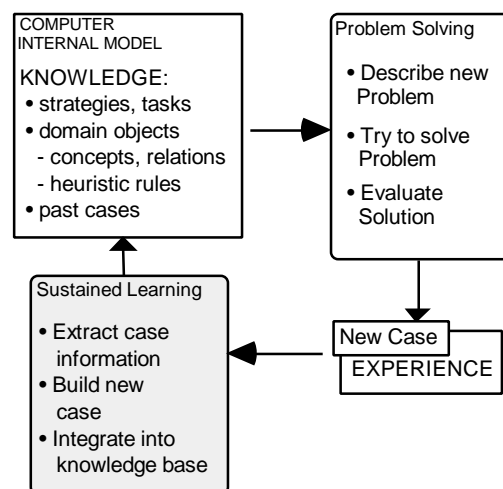
Increased understanding of the representation problem at the symbol level, and of the related reasoning issues, has also come out of work on intelligent system architectures (Van Lehn, 1990), (Weintraub, 1992), frame-based knowledge representation systems (Van Marcke, 1988), (Lenat and Guha, 1989), (Mitchell et al., 1990), and systems for integrated problem solving and learning through user-active participation (Tecuci and Kodratoff, 1990), (Althoff and Wess, 1991), (Bareiss and Slator, 1992), (Plaza and Arcos, 1993), (Aamodt, 1993). These and related results contribute to the realization of knowledge-based systems that can deal with real world, open application domains, and problems which are substantially more complex than what we could with previous methods. This fact has in turn motivated research on knowledge level modeling, since the increased complexity and depth of symbol level models has made it more important to make a thorough analysis at the conceptual level.

Given that a knowledge level model is different in purpose and scope from a symbol level model, an important question is how to ‘bridge the gap’ between the two models, so that the knowledge level model can be effectively utilized for developing the symbol level one. We will come back to this question in the next main section.

### 3.4. The learning problem

The important development trend here is the *case-based* approach to *sustained* learning by experience. The problems of actively maintaining a knowledge-based system have until now received surprisingly little attention in AI. Improving manual maintenance methods is one of the motives behind a structured modeling approach to knowledge acquisition. In manual methods, as well as some earlier automated methods for "knowledge base refinement" (e. g. the SEEK systems (Ginsberg et al., 1985)), updating and refining the knowledge is a separate, periodic effort, not a by-process of a normal problem solving session.

Figure 3 shows the basic scheme for *sustained learning*, i.e. learning as a natural subprocess of problem solving. The figure is an expansion of a part of figure 1. The upper right box illustrates the problem solving steps: First, make an internal description of the problem, i.e. try to understand the problem by integrating it into the system's internal knowledge model. Next, use whatever type of knowledge and reasoning method appropriate to suggest a solution to the



**Figure 3: Sustained learning through problem solving experience**

problem. Finally, evaluate the solution by applying it in the real world or by other means. Based on the evaluated solution, learning takes place. The learning task (lower left) retains knowledge learned from a particular experience, by extracting potentially useful information from it, and by integrating it into the knowledge base. The extracted information may be retained as a new case, or used to update the knowledge base by other means. In the next section we elaborate on these steps for a case-based learner.

A fundamental principle for sustained learning is that each problem solving experience is a powerful source of learning. For this type of learning to be successful, a mistake once made should not be repeated. Sustained learning in this

sense is a perspective to learning that has similarities to reinforcement learning (Sutton, 1992), but differs by being a symbolic, knowledge-based learning approach, whereas reinforcement learning normally uses numeric tuning and optimization methods. Further, case-based reasoning usually involves user feedback, while reinforcement learning generally is unsupervised.

In general, experiences may be stored and indexed as single, specific cases, or generalized to experience types. Recent work in machine learning on knowledge revision (including theory revision) addresses the general problem of updating a knowledge base when new information is received which contradicts or is unexplainable by the current knowledge. Although this research typically addresses the incremental learning of concepts as part of the initial knowledge modeling and testing phase (e.g. (Craw and Sleeman, 1990), (Nedellec and Rouveirol, 1993), (Mooney and Ourston, 1994), (Matwin and Plante, 1994)), these methods should also be possible candidates for sustained learning in the type of domains they address. Normally, however, knowledge revision systems address the problem of identifying and correcting an error when a conflict is detected, and not experiential learning as a continuous process.

Some problems related to sustained learning of generalizations in open domains, and suggestions on how to deal with some of them, were demonstrated by the DISCIPLE (Tecuci, 1988) and CONCLAVE (Van de Velde, 1988) systems. However, as elaborated below, the main advantages of retaining experiences as specific cases, i.e. as instances of problem solving episodes instead of as generalized rules, is that experiential learning becomes much easier - and particularly in real world domains that do not have strong domain theories. A system that combines case-based learning with rule generalization, in a closed technical domain, has recently been developed within the MOLTKE workbench (Althoff, 1992).

The learning of generalizations, in some form or another, is what machine learning traditionally has been about. Although there have been significant advancements over the recent years, the generalization problem - within a knowledge model or on a data set of some complexity - is still far from solved. Storing experiences as cases relies on methods that are able to utilize this type of knowledge when solving new problems. Such methods are provided by the subfield of *case-based reasoning*. A case-based reasoning method takes a problem description as input, looks for a similar case in the collection of past cases, uses the solution of that case to derive a solution for the new problem, and learns from the problem just solved by adding a new case or in other ways updating the case base. In effect, this is also a generalization process, but an implicit one: When solving a new problem, the matching of the problem to past cases in order to find the most similar one is always a *partial matching*, which can be viewed as a matching at a more general or abstract level. The role of general domain knowledge in this process is to guide the retrieval and matching process by providing a knowledge-rich ground<sup>7</sup> in which supporting explanations or meaningful relationships can be derived.

Some CBR methods also provide means of case generalization, for example generalizing feature values by climbing subclass hierarchies, but this is an extremely 'lazy' type of generalization compared to the more eager one performed by inductive machine learning methods. Unlike other ML methods, a case-based

system does not have to explicitly generalize its instances in order to learn. A type of generalization which is common in CBR systems, however, is the generalization of case indexes, in order to improve the similarity assessment procedure during case retrieval<sup>8</sup>.

The characteristics of a case-based learning method are therefore that 1) instance generalization is done during problem solving, not during learning, and 2) generalization is implicit in the similarity assessment made during case retrieval. In a learning-and-problem-solving perspective, part of the burden is therefore taken away from the learning part and given to the problem solving part. In this way, information (knowledge) is not “generalized away” during learning. The learning step therefore does not have to make strong assumptions about what the learned knowledge will be used for, since the generalization is postponed until an actual problem is being solved. Since the generalization process becomes a part of the matching of a given problem to a past case, all the information available about the actual problem can be used in the process. In a multifunctional knowledge base, where the knowledge is assumed to serve different functions, i.e. to be used for various type of tasks, this is clearly advantageous. The disadvantage is that very much relies upon the similarity assessment procedure performed during case retrieval. Hence, the structuring of a case-memory, the indexing of cases, and the problem of similarity assessment are all active research areas of case-based reasoning. However, even if treated differently in CBR learning than in other ML methods, the generalization problem is a major concern of case-based reasoning as well. Ongoing research on integration of case-based and inductive learning - and case-based and generalization-based problem solving - may provide new insights into this problem (e.g. (Althoff et al., 1993)).

In this chapter we have described a type of KA+ML integration that addresses a very important problem for future knowledge-based systems: How to develop and continually update a knowledge base by integrating KA and ML methods. We have argued that the main role of KA methods (possibly including inductive ML methods) is in the initial knowledge modeling phase, while the role emphasized for ML methods is in knowledge maintenance. The learning method advocated is the retaining of new experiences as concrete cases by integrating them into the existing knowledge base (containing general domain knowledge as well as a set of past cases). The rest of this paper is devoted to mechanisms of case-based reasoning (next section), and to a method for combining knowledge acquisition and machine learning within the scope just summarized (section 5).

## **4. Case-based problem solving and learning<sup>9</sup>.**

### *4.1. General presentation*

Problem solving by re-using past cases is a powerful and frequently used reasoning method for humans. This claim is supported by results from cognitive psychological research. Part of the foundation for the case-based approach is its psychological plausibility. Several studies have given empirical evidence for the dominating role of specific, previously experienced situations (what we call cases) in human problem solving (e.g. (Ross, 1989)). Schank (Schank, 1982) developed a theory of learning and reminding based on retaining of experience in a dynamic, evolving memory<sup>10</sup> structure. Studies of problem solving by analogy (e.g.

(Gentner, 1983), (Carbonell, 1986)) also show the frequent use of past experience in solving new and different problems. Case-based reasoning and analogy are sometimes used as synonyms (e.g. by Carbonell), viewing CBR *intra-domain analogy*. However, as will be discussed later, the main body of analogical research (Kedar-Cabelli, 1988), (Hall, 1989) have a different focus, namely analogies across domains (Burstein, 1989).

#### 4.2. Main types of CBR methods.

Case-based reasoning is a broad term, covering many particular type of methods. Below is a list of different CBR methods, distinguished by their different solutions to core CBR problems such as case representation, reasoning methods, and learning strategies:

- *Exemplar-based reasoning.*  
In the exemplar view, a concept is defined as the contents of its set of exemplars. CBR methods that address the learning of concept definitions for classification tasks, are sometimes referred to as exemplar-based (e.g. (Kibler and Aha, 1987), (Bareiss, 1989)). The class of the most similar past case becomes the solution to the classification problem, and there is no adaptation involved.
- *Instance-based reasoning.*  
This is a specialization of exemplar-based reasoning into a highly *syntactic* CBR-approach. The lack of guidance from general domain knowledge is compensated for by a large number of instances. This is a non-generalization approach to the concept learning problem addressed by classical, inductive machine learning methods (Aha et al., 1991).
- *Memory-based reasoning.*  
Memory organization and access within a *large memory* of cases is the focus of this type of method. The utilization of *parallel processing* techniques is a characteristic, and distinguishes this approach from the others. The access and storage methods may rely on purely syntactic criteria, as in the MBR-Talk system (Stanfill and Waltz, 1988), or they may attempt to utilize general domain knowledge, as the work done in Japan on massive parallel memories (Kitano, 1993).
- *Main stream Case-based reasoning.*  
Case-based reasoning is often used as a generic term, but also as a label to CBR methods in a more typical sense than above. For example, a typical case has a certain degree of richness of information, and a certain *complexity* with respect to its internal organization. A feature vector holding some values and an associated class is not a typical case. Further, typical CBR methods are able to *modify*, or adapt, a retrieved solution when applied in a different problem solving context. They also utilize, in some way or the other, *general domain knowledge* within the CBR processes.
- *Analogy-based reasoning.*  
Although sometimes used as a synonym to the typical case-based approach just described (e.g. (Veloso and Carbonell, 1993)), or as a more general notion (Helman, 1988), it usually characterizes methods that solve new problems based on past cases from a *different domain* (Burstein, 1989). The major focus of study has been on the *reuse* of a past case, what is called the

mapping problem, i.e. to map the solution of an identified analog (called source or base) to the present problem (called target) (Kedar-Cabelli, 1988).

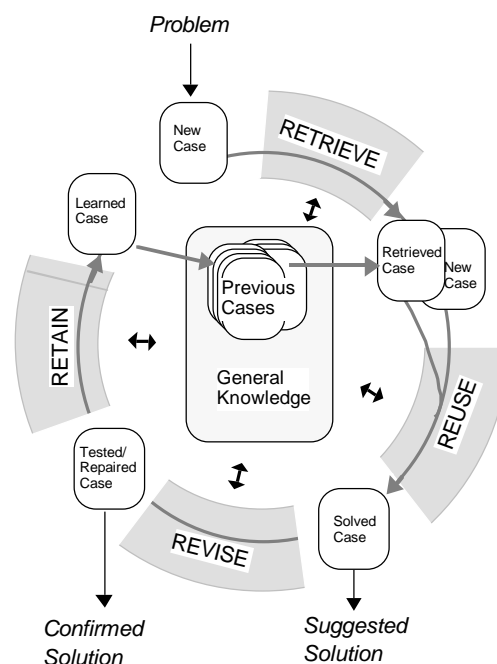
#### 4.3. The CBR cycle

At the highest level of generality, the CBR cycle may be described by the four processes<sup>11</sup>:

1. RETRIEVE the most similar case or cases
2. REUSE the information and knowledge in that case to solve the problem
3. REVISE the proposed solution
4. RETAIN the parts of this experience likely to be useful for future problem solving

A new problem is solved by *retrieving* one or more previously experienced cases, generating a solution by *reusing* the case in one way or another, *revising* the solution by checking its correctness/usefulness - updating the solution if needed, and *retaining* the new experience by incorporating it into the existing knowledge-base (case-base). The four processes each involve a number of more specific steps. In figure 4, this cycle is illustrated.

An initial description of a problem (top of figure) defines a *new case*. This new case is used to RETRIEVE a case from the collection of *previous cases*. The *retrieved case* is combined with the new case - through REUSE - into a *solved case*, i.e. a proposed solution to the initial problem. Through the REVISE process this solution is tested for success, e.g. by being applied to the real world environment or evaluated by a teacher, and repaired if failed. During RETAIN, useful experience is retained for future reuse, and the case base is updated by a new *learned case*, or by modification of some existing cases.



#### Figure 4. The CBR cycle

As indicated in the figure, general knowledge usually plays a part in this cycle, by supporting the CBR processes. This support may range from very weak (or none) to very strong, depending on the type of CBR method. As previously pointed out, general knowledge refers to general domain-dependent knowledge, as opposed to the specific knowledge embodied by cases. For example, in diagnosing a patient by retrieving and reusing the case of a previous patient, a model of anatomy together with causal relationships between pathological states may constitute the general knowledge used by a CBR system. A set of rules may have the same role.

A core part of case-based reasoning is its 'inherent' learning mechanism. The driving force behind case-based methods has to a large extent come from the machine learning community. Learning in CBR occurs as a natural by-product of problem solving. When a problem is successfully solved, the experience is retained in order to solve similar problems in the future. When an attempt to solve a problem fails, the reason for the failure is identified and remembered in order to avoid the same mistake in the future.

Case-based reasoning favors learning by experience, since it is usually easier to learn by retaining a concrete problem solving experience than to generalize from it (as elaborated at the end of section 3.4). Still, effective learning in CBR requires a well worked out set of methods in order to extract relevant knowledge from the experience, integrate a case into an existing knowledge structure, and index the case for later matching with similar cases.

In the following, the CBR cycle is explained in more detailed, focusing on case-based reasoning in the typical, main stream sense.

#### 4.4. Main stream CBR tasks.

The *RETRIEVE* task starts with a (partial) problem description, and ends when a best matching previous case has been found. Usually, an initial identification subtask comes up with a set of relevant problem descriptors, a matching subtask returns a set of cases that are sufficiently similar to the new case - given a similarity threshold of some kind, and a selection subtask then works on this set of cases and chooses the best match (or at least a first case to try out). While some case-based approaches retrieve a previous case largely based on superficial, *syntactical similarities* among problem descriptors (e.g. the CYRUS system (Kolodner, 1983), ARC (Plaza and López de Mantaras, 1990), and PATDEX-1 (Richter and Wess, 1991) systems), some approaches attempt to retrieve cases based on features that have deeper, *semantical similarities* (e.g. the Protos (Bareiss, 1989), CASEY (Koton, 1989), CREEK (Aamodt, 1991), and MMA (Plaza and Arcos, 1993) systems). Syntactic similarity assessment - sometimes referred to as a "knowledge-poor" approach - has its advantage in domains where general domain knowledge is very difficult or impossible to acquire beforehand. Semantically oriented approaches on the other hand, often referred to as "knowledge-intensive"<sup>12</sup>, are able to use the contextual meaning of a problem description in its matching, for domains where general domain knowledge is available.

*REUSE* of the retrieved case solution in the context of the new case focuses on two aspects: (a) the differences among the past and the current case and (b) what

part of a retrieved case can be transferred to the new case. In simple classification tasks the differences are abstracted away and the solution class of the retrieved case is transferred to the new case as its solution class. This is a trivial type of reuse. More typically, systems have to take into account differences in (a) and thus the reused part (b) cannot be directly transferred to the new case but requires an *adaptation* process that takes into account those differences. There are two main ways to reuse past cases<sup>13</sup>: Reuse of the past case solution (transformational reuse), and reuse of the past method that constructed the solution (derivational reuse) (Carbonell, 1986).

Case *REVISION* consists of two tasks: Evaluate the case solution generated by reuse - and if successful, learn from the success. Otherwise repair the case solution using domain-specific knowledge. The evaluation task takes the result from applying the solution in the real environment (asking a teacher or performing the task in the real world). This is usually a step outside the CBR system, since it - at least for a system in normal operation - involves the application of a suggested solution to the real problem. The results from applying the solution may take some time to appear, depending on the type of application. In a medical decision support system, the success or failure of a treatment may take from a few hours up to several months. The case may still be learned, and be available in the case base in the intermediate period, but it has to be marked as a non-evaluated case. A solution may also be applied to a simulation program that is able to generate a correct solution. This is done in CHEF (Hammond, 1989), where a solution (i.e. a cooking recipe) is applied to an internal model assumed to be strong enough to give the necessary feedback for solution repair. Case repair involves detecting the errors of the current solution and retrieving or generating explanations for them.

The *RETAIN* task takes care of the learning step. It incorporates into the existing knowledge what is useful to retain from the new problem solving episode. Learning from a successful or a failed problem solving attempt is triggered by the outcome of the revision and possible repair tasks. It involves selecting what information from the experience that should be retained, in what form it should be retained, whether a new case should be constructed, how a new case should be indexed for later retrieval, and how it should be integrated in the memory structure and knowledge base in general. A new case may be built, or the old case may be generalized or strengthened to subsume the present case as well. If the problem was solved by other methods, including asking the user, a new case is constructed. The 'indexing problem' is a central and much focused problem in case-based reasoning. It amounts to deciding what type of indexes to use for future retrieval, and how to structure the search space of indexes. Through interaction with the user, the general domain knowledge may also be updated - for example when holes or inconsistencies have been discovered during attempts to explain the new case.

#### 4.5. Two examples of *CBR* systems

Below, two example CBR systems are described. They are both well known from the literature, and they represent two different ways of combining knowledge acquisition and machine learning methods.

### *PROTOS*

Protos is a case-based<sup>14</sup> approach to concept learning and classification problem solving. A system for diagnosing hearing disorders has been developed and evaluated (Bareiss, 1989). A problem presented to Protos is described as a set of features, and the system's task is to retrieve the previous case that best matches the feature set. Cases are indexed by *reminders* from features. The category of a retrieved case is proposed as a solution to the problem, without adaptation. If the proposed solution is rejected by the user, a learning session is initiated. Protos may be asked to look for another solution or to accept a solution from the user. The user is forced to define entered terms that are unknown to Protos, by describing their relations with existing terms. In this way general domain knowledge is gradually built up.

General domain knowledge is represented as a semantic network of categories, features and relations. A category is represented by its set of member cases, and its set of links to other categories. An case is represented by its set of features, and the category to which it belongs. Each feature in a stored case is associated with a numerical importance value. This value expresses how important the feature is for classifying the case as a member of the category. For example, the feature 'backrest' is more important than 'wheels' for the category 'chairs'. A comprehensive set of relations are defined (e.g. 'part-of', 'specialization-of', 'causes', 'enables', 'suggests') where each relation has a number of explanatory strength associated. An explanation is a chain of relationships between two features or between a feature and a category. An explanation is accepted if its strength - calculated by combining strengths of each relation in the chain - is above some threshold value.

Protos always learns from a problem solving case: If a problem is successfully solved in the first attempt, no new case is constructed, but the reminders from relevant features to the case are strengthened. If a problem is successfully solved in second or later attempts, Protos tries to find the cause of the initial failure. Protos learns from the failure by weakening reminders from the features to the faulty retrieved case. If Protos is unable to suggest a solution, the case is stored as a new case. Reminders to the case, and difference links to similar cases, are installed. During the learning process, the user is asked to confirm or change suggested modifications to the case structure, and to revise explanations if needed.

Protos is a learning apprentice that relies heavily on its user. This is both a strength and a weakness. A positive effect is a quick adaptation to the real-world problem environment; the system will always be up to date with knowledge related to cases it has recently seen. The major weakness is that the knowledge model of the system eventually will represent a resource that is only partially utilized.

Protos has been thoroughly evaluated, by comparing it to senior medical students in the domain of hearing disorders. It performed at least at the level of these students (Bareiss, 1989).

### *CASEY*

CASEY (Koton, 1988) is a system that combines case-based and model-based reasoning. When a problem turns out to be unsolvable by retrieving a past case, a general domain knowledge model is used in a second attempt to solve the problem. The domain model also plays an active part in supporting the case-based reasoning and learning processes. Type of problems addressed are diagnosis of

heart diseases. The general knowledge model in CASEY is a pure *causal model*, relating features to their causal states. A problem is solved by retrieving a case, and - unlike Protos - *adapting* the past solution to the new problem. Each case contains a causal explanation that relates its features to the diagnosis. The solution to a new problem is derived by using the knowledge model to modify the explanation of the retrieved case. The reasoning method is a combination of case-based and model-based reasoning. The case based method is applied first, model-based reasoning within the causal model is performed if the case method fails to find a sufficiently similar past case. In addition to being a separate reasoning method, model-based reasoning also supports the case-based process.

Cases are stored in a dynamic memory structure as described in (Schank, 1982) and (Kolodner, 1983). The structure is a discrimination network, where the top node contains common properties of all cases in the structure. Downwards in the memory structure cases are indexed according to their differences with other cases. The cases themselves are leaves in the tangled tree-structure. An intermediate node represents a generalized description of the cases indexed under the particular node. A feature is regarded more general than another if it is contained within more cases than the other. Concerning expressiveness, the only relation for deep modeling is 'causes'. The only moderator of the causal relation is a numeric probability - or likelihood - measure. This measure does not capture the underlying reasons for one cause being more plausible than another. Expressiveness is also limited by features and states being flat (property-name property-value) pairs, with no structuring of properties.

CASEY always learns from a problem solving case: If a problem is successfully solved by case-based reasoning, CASEY stores the new case if it has significant features different from the previous case. If the new case is identical to the previous one, information about the importance of a feature is updated. If case-based reasoning fails, and the causal model solves the problem, a new case is created and indexed in the memory structure. The learning in CASEY does not involve user interaction. The system is designed to improve performance efficiency of model-based reasoning within the deep heart failure model. CASEY learns associational, compiled knowledge by extending or modifying<sup>15</sup> its case base.

Unlike Protos, CASEY does not interact with the user in its problem solving and learning phases. In a sense, it is a counter-example of the type of interactive decision-support system addressed in this paper. However, CASEY is interesting because of its tight coupling between case-based and generalization-based reasoning, based on its integration of general domain knowledge developed by normal knowledge acquisition means, and specific knowledge learned by the case-based method.

CASEY was evaluated on the basis of improvement of performance speed compared to the problem solving within the causal system itself. Its improvement increases with the number of cases seen, of course, and tests at several states of development all reported highly significant improvements (Koton, 1989).

#### 4.6. Fielded CBR applications

Even if still is a young field, there are some fielded CBR systems that been in use long enough to have become evaluated. We briefly summarize two of these systems, to illustrate how CBR methods can successfully realize knowledge-based interactive decision support systems.

The first fielded CBR system was developed at Lockheed, Palo Alto. The problem domain is optimization of autoclave loading for heat treatment of composite materials (Hennesy and Hinkle, 1991). An autoclave is a large convection oven, where airplane parts are treated in order to get the right properties. Different material types need different heating and cooling procedures, and the task is to load the autoclave for optimized throughput, i.e. to select the parts that can be treated together, and distribute them in the oven so that their required heating profiles are taken care of. There are always more parts to be cured than the autoclave can take in one load. The knowledge needed to perform this task reasonably well used to reside in the head of a just a few experienced people. There is no theory and very few generally applicable schemes for doing this job, so to build up experience in the form of previously successful and unsuccessful situations is important. The motivation for developing this application was to be able to remember the relevant earlier situations. Further, a decision support system would enable other people than the experts to do the job, and to help training new personnel. The development of the system started in 1987, and it has been in regular use since the fall 1990. The results so far are very positive. The current system handles the configuration of one loading operation in isolation, and an extended system to handle the sequencing of several loads is under testing. The development strategy of the application has been to hold a low-risk profile, and to include more advanced functionalities and solutions as experience with the system has been gained over some time.

The second application was developed at General Dynamics, Electric Boat Division (Brown and Lewis, 1991). During construction of ships, a frequently re-occurring problem is the selection of the most appropriate mechanical equipment, and to fit it to its use. Most of these problems can be handled by fairly standard procedures, but some problems are harder and occur less frequently. These type of problems - referred to as "non-conformances" - also repeat over time, and because regular procedures are missing, they consume a lot of resources to get solved . General Dynamics wanted to see whether a knowledge-based decision support tool could reduce the cost of these problems. The application domain chosen was the selection and adjustment of valves for on-board pipeline systems. The development of the first system started in 1986, using a rule-based systems approach. The testing of the system on real problems initially gave positive results, but problems of brittleness and knowledge maintenance soon became apparent. In 1988 a feasibility study was made of the use of case-based reasoning methods instead of rules, and a prototype CBR system was developed. The tests gave optimistic results, and an operational system was fielded in 1990. The rule-base was taken advantage of in structuring the case knowledge and filling the initial case base. In the fall of 1991 the system was continually used in three out of four departments involved with mechanical construction. A quantitative estimate of cost reductions has been made: The rule-based system took 5 man-years to develop, and the same for the CBR system (2 man-years of studies and experimental development and 3 man-years for the prototype and operational system). This amounts to \$750.000 in total costs. In the period December 90 - September 91 20.000 non-conformances were handled. The reported cost reduction, compared to previous costs of manual procedures, was about 10%, which amounts to a saving of \$240.000 in less than one year.

## 5. KA+ML integration in an iterative modeling cycle

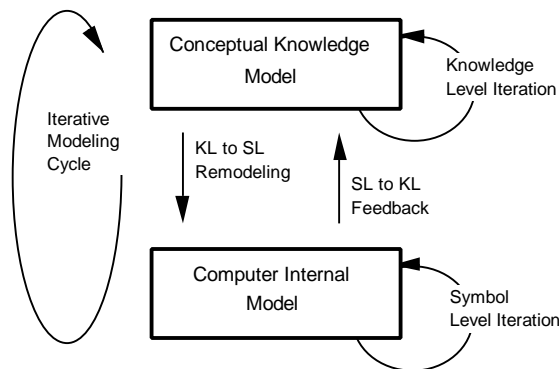
### 5.1. General presentation

We have previously described the knowledge level account to modeling as a means to construct a conceptual model of the application domain, its tasks and possible methods. We have further contrasted knowledge level modeling with symbol level model realization, resulting in an implemented and operational system. In order to take advantage of a thoroughly worked out knowledge level model in constructing the model at the symbol level - and to maintain that model over time - there has to be a means for going from one model to the other. This is, generally, far from a straightforward matter, and the bridging of the two description levels is currently a problem of great concern within the knowledge acquisition community (Klinker et al., 1991), (Steels, 1993), (Duursma and Van de Velde, 1992), (Greboval and Kassel, 1992), (Linster, 1992). Our thesis is that the gap between the knowledge and symbol levels should be shrunk by adopting a more *iterative*, less top-down driven modeling process. The potential of realizing sustained learning by case-based techniques is our motivation for relaxing the focus on initial knowledge modeling in favor of knowledge maintenance and iterative system development.

Top-down modeling is characterized by being a type of *refinement* process, where important decisions are taken at an early stage, and further specialized, or refined, in course of development. From a general development perspective, this is of course a cyclic, and also iterative process - for example iterating between suggested specializations at the knowledge level, their symbol level implementation, testing, and feedback to the knowledge level model. The term iterative, however, also has a stronger interpretation in the context of system development, which involves a more balanced contribution from bottom-up and top-down modeling methods. This type of iteration typically involves frequent *backtracking* of decisions earlier made, which in principle is different from a (iterative) refinement process.

Ideally, a top-down refinement process is what we would want, and our methods should support the kind of thorough analysis and comprehensive structuring that minimize backtracking. However, the world is not ideal, and backtracking in model development will always be necessary. This is why it has to be an inherent part of any knowledge acquisition and learning strategy, no matter how good analysis methods and tools we are able to develop. And further, given that there will always be backtracking, it may be worthwhile to pull it into the iterative design process in the first place, to see if it can be turned into a positive contributor to model building, instead of necessarily something to avoid. This is the rationale behind the strongly iterative approach suggested in the following, and also the reason for putting a stronger emphasis on bottom-up learning within what is referred to as the *iterative modeling cycle*.

Figure 5 gives a top-level view of the iterative modeling cycle (KL = knowledge level, SL = symbol level).



**Figure 5: Iterative knowledge level and symbol level modeling**

It illustrates iterative modeling composed of modeling at the knowledge and symbol levels. It covers initial conceptual modeling at the knowledge level, initial symbol level realization, and intra- and inter-model feedback loops to both of the model types. The arrows represent different type of modeling activities. They may all involve manually driven KA methods as well as ML techniques. It should be noted that the cycle is *not a total system development cycle*. It describes only the part related to *knowledge modeling*. At this level of abstraction, the modeling cycle is, in principle, open to be specialized into almost any knowledge modeling method, and may therefore serve as a basis for discussing different approaches.

### 5.2. Specifying the general framework

The first step toward a specialization of the iterative modeling cycle may be found in the figure's naming of the KL-to-SL arrow. It is important that this process is regarded as a *remodeling* process, to indicate that the transition is more complicated than a mere transformation. This is a central issue in our framework, since the bridging of the KL-SL gap is a major motivation for our research, and since our proposal differs from methodologies that attempt to make this transition an automatic one:

Current approaches to an automatic bridging of this gap may be split into two types: Those based on an *automatic operationalization* of knowledge level models into executable, symbol level ones, and those based on a set of *pre-defined links* between types of knowledge level constructs and existing symbol level program components. Unfortunately, both methods impose severe constraints on the expressive power and flexibility of modeling choices at one or both of the levels. In the first approach (e.g. (Vanwelkenhuysen and Rademakers, 1990), (Linster, 1992)), where a common representation language is used at both levels, a well worked-out KL model is more or less automatically made operational at the symbol level. The *degree of automated transfer* between the two levels is the major, and significant, difference between this approach and our remodeling approach. This difference reflects different views on the role of top-down vs. bottom-up modeling, since the weakening of a strong, definitional role for top-down modeling necessarily leads to a reduced reliance on the strength and power of the knowledge level model.

The second approach (e.g. (Klinker et al., 1991), (Steels, 1993)) emphasizes the reuse of symbol level components, and some of these methods have a less developed notion of the knowledge level. This represents a rather programming-

driven approach, leaning toward a software-reuse tool, and hence some distance away from a knowledge modeling approach in the sense we have used the term in this paper. Further, it assumes a rather well-described domain (since the linking is pre-defined and rather straightforward), and rather simple, well-known inference and reasoning methods (part of the reusable library). It represents an oversimplification of the tedious and difficult task of developing competent and robust knowledge-based systems tailored to particular problem domains and environments. In a longer time perspective, however, when more experience has been gained in how to develop useful, competent and reliable knowledge-based systems, this approach seems a very promising one<sup>16</sup>.

The first approach is top-down driven, while the second to a large extent is driven by bottom-up programming. By giving a stronger role to iterative modeling through knowledge maintenance, made possible by case-based reasoning methods, a new approach to combined KL and SL modeling can be defined, based on the following characteristics:

- A balanced integration of top-down, knowledge-level driven, and bottom-up, symbol-level driven modeling.
- The development of an initial knowledge level model is always the first modeling step, and here is where KA methods play their major role. Modeling should be based on recent insight into knowledge level modeling, and a systematic knowledge analysis should be made by studying the relationships between goals/tasks, methods to solve tasks, and the domain specific knowledge and information needed by these methods (Aamodt, 1990a).
- Continuous evolution of models by sustained learning from experience. Given the problems with generalization based learning methods for real world domains, case-based reasoning should be the core ML method, at the symbol level. Application problem solving should combine case-based and generalization-based (model-based) reasoning methods.
- Active user involvement in problem solving and learning will identify errors or holes in the general domain knowledge, which should feed into KA methods that update the knowledge-level model (the SL-KL feedback, see figure 5). This may, in turn, lead to a new traverse of the KL-SL link since the integration of new/updated knowledge at the KL in turn is elaborated to produce new knowledge.
- A modeling language is needed, at both levels, which is expressive and flexible, and this should be based on the object-oriented representation paradigm. It may be the same language syntax, at the two levels, but the semantics will have to be different. The semantics at each level should be specified from the needs of modeling at that level. User transparency, as well as a procedural but clearly defined semantics, are important properties of such a language.

The result should be a coherent model of general knowledge (domain models) and situation-specific knowledge (cases) at the two levels, and a system at the symbol level that reasons and learns by combining case-based and generalization-based methods. As an example of a system architecture for realizing systems according to the above characteristics, and which is also motivated by an effective integration of KA and ML methods, the CREEK system is summarized in the following.

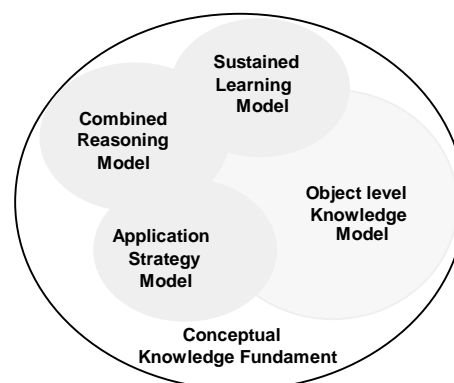
### 5.3. CREEK - iterative modeling through knowledge-intensive case-based reasoning and learning

The CREEK system (Aamodt, 1991), (Aamodt, 1994) is an architecture for knowledge-intensive case-based problem solving and learning, with the goal of studying the impact of sustained learning on the overall knowledge modeling cycle.

CREEK contains, at the top level, four modules integrated within a common conceptual basis as illustrated in figure 6.

Each module represents a particular sub-model of knowledge. The four modules are an object-level domain knowledge model, a strategy level model (for example a model of diagnostic problem solving), and two internal meta-level models - one for combining case-based and other types of reasoning, and one for sustained learning. CREEK integrates problem solving and learning into one functional architecture. The user is able to - and assumed to - interact with the system in all its phases of problem solving and learning.

Previously experienced cases and general knowledge of domain relationships are held in the object level knowledge model. The other models contain general knowledge in the form of concept models and/or rules. It is important to note that all the concepts are 'glued together' into a single, interconnected knowledge model. Diagnosis task concepts, for example, such as "symptom" and "diagnostic-hypothesis" (part of the application strategy model), and learning task concepts, such as "case-indexing" and "failure-generalization" (part of the sustained learning model), are defined within the same representation structure as general domain concepts like "appendicitis" and "fever", and case-related domains terms as "Patient#123456" and "current-radiation-dosage" (which all are part of the object level knowledge model).



**Figure 6: The knowledge modules in CREEK**

#### 5.3.1. Representation

All types of knowledge and information are captured in a frame-based representation language called CreekL, an extension of SFL (Aakvik et al., 1990), (Aakvik et al., 1991). It is a flexible and extendible language, with a procedural semantics and similarities to FRL (Roberts and Goldstein, 1977) and RLL (Greiner and Lenat, 1980).

A knowledge model represented in CreekL is viewed as a semantic network, where each node and each link in the network is explicitly defined in its own frame. Each node in the network corresponds to a concept in the knowledge model, and each link corresponds to a relation between concepts. A concept may be a general definitional or prototypical concept, a case, or a heuristic rule, and describe knowledge of domain objects as well as problem solving methods and strategies. A frame represents a node in the network, i.e. a concept in the knowledge model. Each concept is defined by its relations to other concepts, represented by the list of slots in the concept's frame definition. Main characteristics of the representation are:

- Concepts are represented as a network of frames
- Concepts are either entities or relations
- Entities are physical or abstract objects of a domain
- Relations are named associations (links) between concepts

A CreekL frame is basically a four-level structure of frame, slots, facets, and value-expressions, where a value-expression is a list containing the actual value itself and annotations such as the justification for a value, a time stamp, etc. A part of the frame structure in BNF form is:

```

<knowledgebase>      :=      {<frame>}
<frame>              :=      <framename {<slot>}>
<slot>               :=      <slotname {<facet>}>
<facet>              :=      <facetname {<value-expression>}>
<value-expression> := <proper-value value-justification value-source
                        value-time stamp value-miscellaneous>

```

Below, two example frames are shown, the first is a general domain concept and the second a case:

car		
has-subclass	value	family-car sports-car limousine van
subclass-of	value	vehicle means-of-transportation sporting-gear
has-part	value	wheel fuel-system engine electrical-system
has-number-of-wheels	default	4
has-colour	value-class	colour
has-fault	value-class	car-fault
has-age	value-dimension	years
	if-needed	(time-difference *current-year*
		self.has-production-year)

```

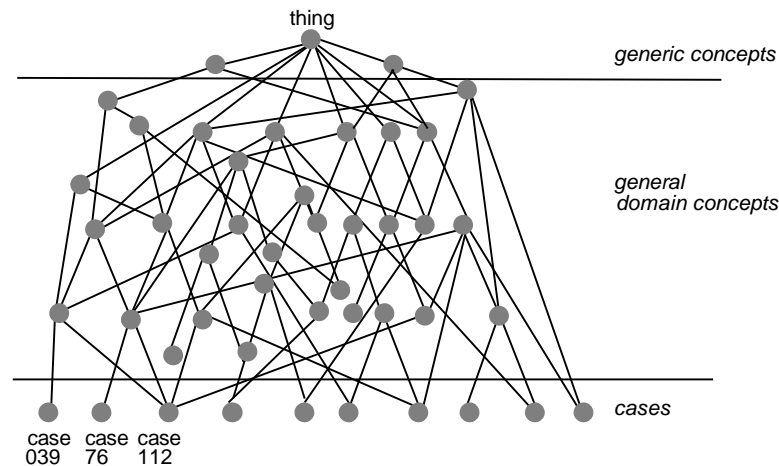
case#461
instance-of      solved-case car-starting-case
of-car          N-VD-123456
has-process-status solution-proposed
has-input-time  22/07/93 12:10
has-relevant-finding
                has-electrical-status electrical-system-ok
                has-engine-status (engine-turns engine-does-not fire)
                has-ignition-status (spark-plugs-ok distributor-ok)
                has-weather-condition (low-temperature moisty normal-air-pressure)
has-location-of-problem inside-garage
has-driving-history normal-driving
has-solution    (water-in-gas-tank
                 (0.92
                  ((engine-does-not-fire caused-by water-in-gas-mixture)
                   (water-in-gas-mixture caused-by water-in-gas-tank))
                  (carburettor-fault has-status false)
                  (((low-temperature combined-with moist) leads-to
                    condensation-in-gas-tank)
                   (condensation-in-gas-tank causes water-in-gas-tank))
                  (water-in-gas-tank does-not-violate
                   (electrical-system-ok spark-plugs-ok distributor-ok
                    engine-turns normal-air-pressure)))
                 )
same-solution-in case#06 case#88 case#388 case#401
difference-case  case#128

```

Facets are shown for the car concept, but left out for case#461 since they are all value facets. A case solution (has-solution slot of a case) also contains a justification or support for the solution, in the form of an explanation with a computed explanation strength (0.92 in the above case)<sup>17</sup>.

Each slot in a CreekL frame corresponds to a relation in the knowledge network, and each symbolic term is represented as a concept. Figure 7 illustrates how cases and general knowledge are integrated into the general knowledge structure. The nodes in the network are concepts and the links are relations. For drawing clarity, only a very small knowledge network, and only a very small number of relations to/from each node, is shown. Generic concepts are contained in a fixed part of the knowledge base, and includes top level 'world' concepts (e.g. thing), as well as representational primitives (e.g. frame, slot).

To enable a thorough representation of knowledge, CreekL facilitates explicit definitions of relations as well as of symbolic values<sup>18</sup>. For example, if the user wants to introduce a new slot, called *has-color*, on the frame *car*, the system will automatically create the frame *has-color* and give it a slot called *used-to-describe* with the value *car*. The user may enter additional slots on the *has-color* frame in order to better describe what it means to have color. The system also automatically creates a frame for each symbolic value entered into a slot. The inference methods that operate within this semantic network of frames are typical frame language methods like property inheritance (default and forced inheritance), frame matching (concept recognition), and constraint enforcement (constraint propagation). Access methods exist to retrieve and update any information item at any of the five levels of a frame structure.

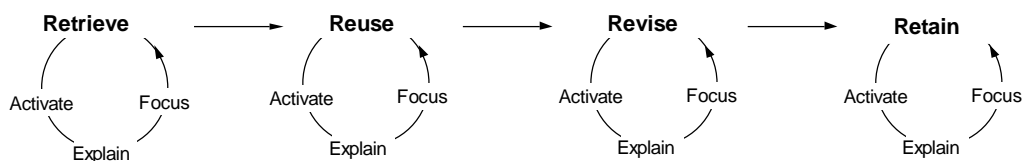


**Figure 7: Integrating cases and general knowledge**

Since the language has an operational (procedural) semantics, it enables the expression of informal and vague models (at the knowledge level) as well as specialized and detailed models (at the symbol level). Hence the remodeling involved in moving between the levels may be performed within the same basic object structure, and based upon a common representational syntax.

### 5.3.2. Problem solving

The case-based interpreter in CREEK contains a three-step process of 1) activating relevant parts of the semantic network 2) generating and explaining derived information within the activated knowledge structure, and 3) focusing towards and selecting a conclusion that conforms with the goal. This *activate-explain-focus* cycle, referred to as an 'explanation engine' (Aamodt, 1994), is a general mechanism that has been specialized for each of the four CBR tasks described in section 4 (see figure 8).



**Figure 8. The CBR process and the explanation engine**

The extensive, explanation-driven way of utilizing general domain knowledge in the CBR subtasks is a feature that distinguishes CREEK from other CBR systems.

The process of remembering (retrieving) and reusing previous cases may be used extensively in all three steps, but a more typical pattern is to use general knowledge in the activation step, cases for generating hypotheses, and a limited number of cases together with general knowledge for the focusing step.

The case-based method of Creek thus relies heavily on an extensive body of general domain knowledge in its problem understanding, similarity assessment, case adaptation, and learning. The general domain knowledge is assumed to be extensive enough to provide a back-up capability of problem solving on its own: If reasoning from case-specific knowledge fails, for example when no similar

situation has previously occurred, the system may use its general knowledge to generate a suggested solution. This knowledge is typically built up by rather 'deep' relationships - for example a combination of causal, structural, and functional relations (depending on what is most appropriate within the application domain). It may also contain more shallow associations, in the form of if-then rules. Figure 9 shows the top level combined reasoning algorithm.

```

1 Input representational-primitives KB;
2 Input general-domain-knowledge KB;
3 Input case KB;
4 Interconnect all concepts in the three KBs;
  <result is a semantic network of general concepts and cases>
5 While general-domain-knowledge does not need off-line revision
  For each problem do <case-based reasoning>
  Begin
6   Input problem case ;
7   Determine relevant features;
8   If strong reminding to a case
9     Retrieve the case most strongly reminded of; <case retrieval>
10    Assess relevance of retrieved case for current problem case
        and justify the validity of its solution;
11    If solution of retrieved case is acceptable
12      Learn from the experience; <to learning>
13    Else
14      If case-based-reasoning failed <cbt termination test>
15        Attempt generalization-based-reasoning;
16        Learn from the experience; <to learning>
17        If generalization-based-reasoning fails
18          Ask user for solution and justification;
19          Learn from the experience ; <to learning>
20      Else
21        Attempt to modify solution of retrieved case ; <case adaptation>
22        If case adaptation successful
23          Go to 11 ;
24        If sufficiently strong reminding to another case
25          Go to 9 ;
26        Attempt to update problem case in the light of new
                information;
27          Go to 8 ;; <final CBR attempt>
28    Else
29      If case-based-reasoning failed <cbt termination test>
30        Go to 15;
31      Else
32        Elaborate to get more information about problem case ;
33        Go to 7 ;;
34  End;;;

```

**Figure 9: Combined reasoning in Creek**

The process of selecting the initial reasoning paradigm starts after relevant features has been extracted from the input case, and halts when either a reasoning method has been chosen, or a plausible solution to the problem has been found. If the problem features give a reminding to a previous case that is above a certain threshold, case based problem solving is tried (figure 9, line 8-9), otherwise generalization-based reasoning is activated (line 14-15). Generalization-based reasoning includes rules-based as well as deeper model-based reasoning. The value of the threshold level depends upon the relative size and contents of the case base. It is set manually to begin with, and automatically adjusted over time according to how well the chosen reasoning paradigm performs. If the system is unable to suggest a solution, the user has to solve the problem by other means (line 17-18).

### 5.3.3. Sustained learning

CREEK learns from every problem solving experience. If a successful solution was directly copied from, or just slightly modified on the basis of a previous case, the reminding to that case from each relevant feature is strengthened. No new case is stored. If a solution was derived by significantly modifying a previous solution, a new case is stored and difference links between the two cases are established. A new case is also created after a problem has been solved from rules or from the deeper knowledge model alone.

```

1  If solution for problem case was derived from case-based-reasoning
2    If adaptation of solution from retrieved case was needed
3      Construct new case based on retrieved case and justification;
4      Weaken reminders to retrieved case ;
5    Else
6      If retrieved case and problem case are mergable
7        Modify retrieved case ;
8        Strengthen reminders to retrieved case ;
9      Else
10     Construct new case based on retrieved case ;;
11  Else
12    If solution for problem case derived from
      generalization-based-reasoning
13     Construct new case based on solution and justification;
14    Else
15     Construct new case based on solution and justification
      from user ;;
16  Check new case by re-running the initial problem;
17  If new case is not retrieved
18    Update reminding strengths to new case ;
19  Update case base;

```

**Figure 10: Learning in Creek**

The main target for the learning process in CREEK is thus the case base. But a system may also update its general knowledge through interaction with the user during problem solving. Since heuristic rules are represented within the conceptual model, they are available for the same tasks as the conceptual model in general. A rule is viewed as a shallow type of relation, and may also be used to support case-based problem solving as well as learning. Even if the explanatory strength of a shallow relation in general is low, it may add to other explanations for the same hypothesis and, thus, contribute to a justification of an hypothesis. Figure 10 illustrates the learning algorithm. The user is assumed to actively take part in both the problem solving and learning processes.

This approach to integrated problem solving and learning also has an impact on the *validation* issue, since what will be validated is not only application problem solving behavior, but also learning behavior. On the one hand, this increases the complexity and difficulty of validation. On the other, explicit validation in the traditional sense becomes less important, *since the system continually improves through the solving of problems*. The shift from thinking about systems in the traditional sense of development followed by a periodic maintenance scheme, to adaptive and continually evolving systems, is important and has many possible implications.

#### 5.3.4. An example of case-based reasoning in CREEK.

The example case previously shown, case#461, is a case that has been solved and learned. Below we will briefly describe the main steps from an initial problem description to a solved and learned case, by referring to the four-task reasoning model of figure 8, in which each subtask is described by the Activate-Explain-Focus cycle. For a more extensive description of this example, see (Aamodt, 1994).

The system assists the diagnosis of car starting problems. Let us assume that the following input case description is given. (All facets are 'value' facets, and facet names are dropped for the sake of simplicity):

case#0		
instance-of	initial-state-case	car-starting-case
of-car	N-VD-123456	
has-task	find-car-starting-fault	
has-electrical-status	electrical-system-ok	
has-engine-status	engine-turns	engine-does-not fire
has-ignition-status	spark-plugs-ok	distributor-ok
has-weather-condition	low-temperature	moisty normal-air-pressure
has-location-of-problem	inside-garage	

**Retrieve** starts by activating the problem solving context. We assume a large, multifunctional knowledge base, in this case of knowledge about cars and other vehicles, and in order to produce focused explanations later on, we want to activate just those parts and perspectives of this knowledge that is potentially relevant for the current task. The context is determined by a method of goal-focused spreading activation as previously described. The spreading activation method activates all input findings and goal relevant concepts. In our example the spreading starts out from the top-level application task, i.e. find-car-starting-fault, and the input features (has-electrical-status, etc.). Spreading-relations include general taxonomic ones (has-subclass, subclass-of, has-instance, instance-of), causal relations (causes, caused-by), associational relations (occurs-together-with, leads-to), and application-specific relations (tested-by, test-for). The final job of Activate is to use the findings as indexes to the case base and retrieve a set of cases whose matching strength is above a certain threshold. A finding concept has a slot relevant-finding-in which holds a list of cases and a computed relevance factor for the finding with respect to each case. For example:

spark-plugs-ok	
subclass-of	ignition-system-finding
relevant-finding-in	(case#19 0.7) (case#85 0.6) (case#366 0.95) (case#123 0.6)
....	

Combined reminders from the findings to the cases they point to are computed, and a set of matching cases are retrieved. The Explain step then evaluates the match between the problem case and the cases retrieved. This is necessary since the initial case retrieval was based on a rather superficial matching process. The relevance to the problem has to be justified for findings that match well, and mismatched findings have to be explained as not important. The system tries to 'explain away' findings that mismatch, and attempts to construct an explanation that shows the similarities of findings with syntactically different values in the two cases. For example, a finding in one of the retrieved cases that is missing in the input case is that recent driving condition is hard-driving. In the knowledge model, hard driving and broken carburetor membrane are connected:

hard-driving *always-leads-to* extreme-engine-load *may-lead-to* abnormally-high-carburetor-pressure *causes* broken-carburetor-membrane

Based on this explanation, *has-recent-driving-history* is regarded an important finding. The system therefore asks the user about the value of this finding in the input case. The user replies that the recent driving has been normal. The degree of match of the case in question is then reduced.

The Focus step of Retrieve makes the final selection of the best case, or rejects all of them. In our example case#123 being returned as the outcome of the Retrieval task:

case#123	
instance-of	solved-case car-starting-case
has-task	find-car-starting-fault
of-car	N-CC-345678
has-solution	carburettor-valve-stuck
has-electrical-status	electrical-system-ok
has-engine-status	engine-turns engine-does-not fire
has-ignition-status	spark-plugs-ok distributor-ok
has-weather-condition	high-temperature moisty low-air-pressure
has-location-of-problem	inside-garage
.....	

**Reuse** uses the solution of the selected case in solving a new problem, often involving some kind of modification of the past solution. The Activate step starts out from the solution of the best matching case, and spreads activation to concepts representing all expected findings given the solution of the retrieved case. The spreading relations used for this typically include causal and functional relations, as well as direct associations (e.g. *implies* and *co-occurs-with* relations). Values of expected findings that are not known, is requested when needed, i.e. during the Explain step.

Explain has two main jobs to do. One is to evaluate the solution proposed by the retrieved case. Expected findings have to be confirmed, or explained as irrelevant for the present problem. An attempt is made to infer expected findings before asking the user. If all relevant expectations are covered for, control is given to the Focus step. If not, the second Explain step, modification of the solution, is triggered. An attempt is made to produce an explanation structure that justifies a replacement or tweaking of the solution. For example: The solution in the retrieved case was stuck carburetor valve. This is suggested to the user, but after inspection the carburetor turns out to be OK. The main explanation path from this solution to the findings is

carburetor-valve-stuck *causes* too-rich-gas-mixture-in-cylinder *causes* no-chamber-ignition  
*causes* engine-does-not-fire

See figure 11. Before looking for a better matching case, the system tries to modify its explanation of the findings. By searching backwards along the explanation path (lower part of figure 11) for other explanations of its states, it finds that engine-does-not-fire may also be caused by water-in-gas-mixture, in turn caused by water-in-gas-tank. The fault water-in-gas-tank is also supported by the findings moisty and low-temperature, via their relations to condensation-in-gas-tank (not shown in the figure), and is therefore the solution suggested by Explain.

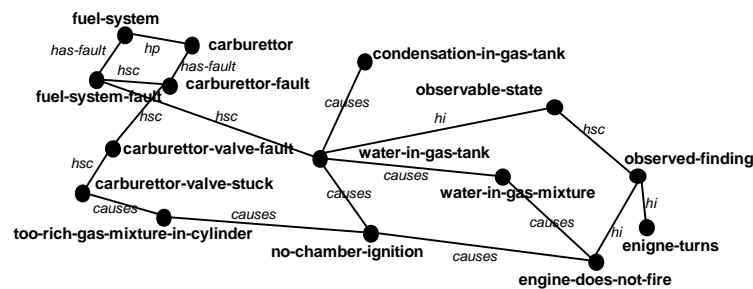


Figure 11. Partial knowledge structure in the car domain

Focus checks whether the proposed solution confirms with external requirements, and proposes the suggested solution to the user. The condensation-in-gas-tank suggestion is checked and confirmed by the user. Focus also extracts a minimal, sufficient explanation for the case, based on the explanation structures produced by the Explain step. The result of this is given as input to the learning task (i.e. Retain).

**Revise** evaluates the suggested solution - condensation in gas tank - by applying it to the actual problem. In this example the fault is confirmed, and needs no revision.

**Retain** is the learning task that captures the experience from the problem just solved. The Activate step here works on the structure returned from Retrieve and extracts potential concepts and structures for learning, i.e. the possible contents of a new case, and explanation structures that support the solution. New or modified concepts that may have been introduced by the user is also marked for the learning process.

The Explain step perform three tasks: First, it justifies whether a new case needs to be constructed or not. This is needed if no similar past case was found, if a retrieved solution needed significant modification, or if the set of relevant problem findings are sufficiently different in the two cases. Explanations has to be produced in order to assess the significance of a modification or of feature differences. In our example a new case is added to the case base. The second subtask is to determine what structures to retain from a problem solving experience. Among these are the relevant findings with respect to the final solution. For each finding currently in the case, its relevance is explained. The strongest explanation from the solution to the findings, i.e. a structure of explanatory relationships, is retained in the learned case. If this does not constitute a sufficiently strong explanation for the solution (threshold value), additional 'parallel' explanation paths are added. Learning of generalizations does not play a strong role in our method, but a lazy generalization of values for findings is done if justified by an explanation or by the user. Given that a new case is to be stored, the third subtask is to compute the importance (degree of necessity) and predictive strength (degree of sufficiency) of case findings, in order to determine their relevance factors.

The Focus step puts together the new structures into a case frame, and actually stores the new or modified case and its indexes. This also includes updating the relevant-finding-in slot of relevant findings. A finding may be relevant even if it is not an explicit part of an explanation path. The status of the electrical and ignition systems, for example, support the fact that a solution is found within the fuel

system. In our example, the explanation structure being stored contains four paths that together explain the relevant findings. This leads to retaining the new case#461, with the content as previously shown.

Following the learning task, a test is run by entering the initial problem once again. It is checked whether the case just learned is retrieved, or - if no new case is learned - whether the problem solving procedure is otherwise improved.

## 6. Conclusion and future trends

The insight which changed the focus of AI research from general problem solving to explicit knowledge modeling was an important one. During the last ten to fifteen years the notion of an explicit knowledge model has been substantially extended and refined. Starting from a view of explicit knowledge as a collection of If-Then rules, it has been realized that the knowledge models which are needed to develop competent and robust real world systems, is substantially more complex. Methods are needed that are able to capture and utilize the various types of knowledge available: Heuristic rules as well as deeper, principled domain knowledge, generalized knowledge as well as specialized knowledge related to problem cases, and object level knowledge as well as knowledge to control the problem solving and learning processes.

Any attempt to also include learning methods into such a complex environment may seem to reach beyond a realistic level of ambition. We will argue that the contrary is the case, for two reasons:

First, the kind of systems we are aiming at may turn out to be too complex to be fully developed before they are put into operation. Therefore, methods that are able to capture a system's experience as it is used, and improve the system's knowledge and behavior as more problems are being solved, may be what is required in order to facilitate a practical realization of such systems. It is a reasonable assumption that the users of intelligent decision support systems are more likely to accept a system with some weaknesses, as long as the system is able to improve over time. And particularly so, if the system is able to interact with the user in an intelligible way. So, incorporating methods for sustained learning into future knowledge-based systems, will help the development and user acceptance of the systems, rather than impair it.

Second, in order to examine the potential of machine learning methods for gradually improving a system's behavior during normal operation, the methods need to address learning in a highly knowledge-intensive environment. Our learning methods should be able to take advantage of the various types of existing knowledge in their learning methods. The development of such systems should be based on a system architecture that enables the expert and knowledge engineer to explicitly express the various types of knowledge relevant for a particular application. This architecture should also contain problem solving and learning methods which are able to effectively utilize a continually improving body of knowledge.

The integrated approach described here is a step in this direction. A framework for knowledge-intensive learning and problem solving by integration of KA and ML methods has been described, and some existing integration approaches has been discussed within this framework. Further, the CREEK system has been used

to illustrate a system design based on the integration framework. CREEK suggests an improved approach to competent knowledge-based systems that continually learn from experience, but extending results achieved within knowledge-intensive learning methods, learning apprentice methods, and the case-based reasoning approach to problem solving and learning. At the university of Trondheim we are currently interested in exploring the impacts of case-based reasoning on other methods for knowledge acquisition, problem solving and learning. We are working together with the research institute SINTEF to develop industrial applications in this area, currently directed toward a case-based decision support system for drilling operations (Nordbø et al., 1992). We are also moving into medical diagnosis and treatment, another domain ideal for research on methods for open and weak theory domains.

There are many issues we need to address in order to obtain real synergy from KA+ML integration in a total knowledge modeling context. This paper has been an attempt to structure this discussion by raising some important questions, providing a framework for discussion, and suggesting a new direction of focus. There are two main messages from the work reported here. The first is that KA+ML integration should in its current state be conducted within the context of a total, iterative knowledge modeling cycle, rather than at the detailed level of specific algorithms. Otherwise it will be difficult to move machine learning into real world, user-interactive decision support systems. The second is that, since it is regarded as a must for future knowledge-based systems to be able to learn from their problem solving experience, we have to focus much more on modeling development through sustained learning, and to weaken our reliance on top-down modeling and on maintenance schemes based on manual, periodic updates.

---

## Notes

<sup>1</sup>Note that the term “problem solving” has a very general interpretation in this paper. Problem solving is not necessarily the finding of a concrete solution to an application problem, it may be any problem put up by an application task. For example, to justify or criticize a solution proposed by the user, to interpret a problem situation, to generate a set of possible solutions, or to generate expectations from observable data are examples of problem solving.

<sup>2</sup>Abductive inference is often referred to as “inference to the best explanation” (e.g. (Thagard, 1988)). While its original meaning, as defined by C.S. Pierce (Ayer, 1968), was as hypothesis formation only, its common interpretation in AI (e.g. (Josephson and Josephson, 1994), (Leake, 1993)) is as a non-deductive alternative to both hypothesis formation and evaluation.

<sup>3</sup>Note that the framework is a *descriptive*, not a *prescriptive* one. Hence it does not assume that any particular approach is more preferable than another. We will later use the framework to discuss such issues.

<sup>4</sup>This may include inductive learning methods to form concept definitions or classification rules, but that type of KA+ML integration is not the topic of this paper (our focus being on learning for knowledge maintenance).

<sup>5</sup>The medium is what is being processed, and the behavioral law is the basic principle describing how what the medium expresses leads to a system's behavior.

<sup>6</sup>Actually, domain knowledge is not a very good term, since task- and method knowledge often is domain specific as well. It is hard to find a better term to indicate this type of knowledge,

---

however, although 'object knowledge' 'application knowledge' and just 'models' have been proposed. We will stick to domain knowledge, but bear in mind that the other knowledge types are not necessarily domain independent.

<sup>7</sup>Sometimes referred to as 'background knowledge'. This is a suitable term to describe some knowledge-support to a algorithmic-driven learning method, but it easily leads to a wrong perspective on knowledge-intensive learning methods, where the emphasis is on learning *within* the existing knowledge, not by using it as a background.

<sup>8</sup>A simple inductive learning of case indexes is also incorporated in some commercial CBR shells (Harmon, 1992).

<sup>9</sup>This section is based on a compression of parts of (Aamodt and Plaza, 1994), modified to fit to the context of this paper.

<sup>10</sup>The term 'memory' is often used to refer to the storage structure that holds the existing cases, i.e. to the case base. A memory, thus, refers to what is remembered from previous experiences. Correspondingly, a reminding is a pointer structure to some part of memory.

<sup>11</sup>As a mnemonic, try "the four REs".

<sup>12</sup>Note that syntactic oriented methods may also contain a lot of general domain knowledge, implicit in their matching methods. The distinction between knowledge-poor and knowledge-intensive is therefore related to *explicitly represented* domain knowledge. Further, it refers to *generalized* domain knowledge, since cases also contain explicit knowledge, but this is *specialized* (specific) domain knowledge.

<sup>13</sup>We here adapt the distinction between transformational and derivational analogy, put forth in (Carbonell, 1986).

<sup>14</sup>Actually, Protos is called an *exemplar-based* system, emphasizing that all cases are stored exclusively as concrete, non-generalized exemplars. A concept definition is viewed extensionally, as a category, defined by the collection of cases (exemplars) that belong to the category.

<sup>15</sup>Modification of feature importances, measured by the number of times a feature is seen in case, and the number of times it is used in a causal explanation.

<sup>16</sup>Interestingly, this approach has the potential of being the basis for a case-based knowledge modeling tool that learns by retaining its modeling experiences and assists knowledge engineers by reusing these experiences.

<sup>17</sup>the most detailed description of CreekL is given in (Aamodt, 1991). An account of the explanation-driven reasoning and learning processes, including an example that involves the above two frames, is given in (Aamodt, 1994).

<sup>18</sup>A symbolic value is a proper value that is not a number, a text string or a lisp-function.

## References

Aakvik G., Mengshoel O. J., Nordbø I., and Vestli M. (1990). SFL - The Sintef Frame Language. The Acknowledge Project, ACK-STF-T3.1-WP-003-A. Trondheim.

Aakvik G., Aamodt A., and Nordbø I. (1991). A knowledge Representation Framework Supporting Knowledge Modeling. *Proceedings EKAW-91*, Fifth European Knowledge Acquisition for Knowledge-based Systems Workshop, Crieff, Scotland, May 1991.

Aamodt A. (1990a). A computational model of knowledge-intensive problem solving and learning. In: Bob Wielinga, et al. (eds.). (1990): *Current trends in knowledge acquisition*. IOS Press, Amsterdam, June 1990.

- 
- Aamodt A. (1990b). Knowledge-intensive case-based reasoning and learning. *Proceedings of ECAI-90*, Ninth European Conference on Artificial Intelligence, Stockholm, August 1990. pp. 1-20.
- Aamodt A. (1991). *A knowledge-intensive, integrated approach to problem solving and sustained learning*. Ph.D. Dissertation, University of Trondheim, Norwegian Institute of Technology, Department of Computer Systems and Telematics. May 1991. Also as University Microfilms International, Dissertation 9208460.
- Aamodt A. (1993). A case-based answer to some problems of knowledge-based systems. To appear in *Proceedings of SCAI, 1993* , *Fourth Scandinavian Conference on Artificial Intelligence*, Stockholm, May 1993. IOS Press.
- Aamodt A. and Plaza E. (1994). Case-Based Reasoning: Foundational issues, current state, and future trends, *AI Communications* , Vol 7, no.1, pp. 39-59.
- Aamodt A. (1994). Explanation-driven case-based reasoning: In S.Wess, K.Althoff, M.Richter (eds.), *Topics in case-based reasoning*. Springer. pp 274-288.
- Aha D., Kibler D., Albert M. K. (1991), Instance-Based Learning Algorithms. *Machine Learning*, vol.6 (1).
- Althoff K-D. and Wess S. (1991). Case-based knowledge acquisition, learning and problem solving for real world tasks. *Proceedings EKAW-91, European Knowledge Acquisition Workshop*.
- Althoff, K-D. (1992). *Eine fallbasierte Lernkomponente als Bestandteil der MOLTKE-Werkbank zur Diagnose technischer Systeme*. (University of Kaiserslautern, Doctoral dissertation). Infix Verlag.
- Althoff, K.-D., Bergmann, R., Maurer, F., Wess, S., Manago, M., Auriol, E., Conruyt, N., Traphoener, R., Braeuer, M. & Dittrich, S. (1993). Integrating Inductive and Case-Based Technologies for Classification and Diagnostic Reasoning. In: E. Plaza (ed.), Proc. ECML, 1993) Workshop on "Integrated Learning Architectures"
- Anderson J.R. (1990). *Cognitive psychology and its implication, 3rd edition*. W.H. Freeman and Co.
- Ayer A.J. (1968). *The origins of pragmatism; studies in the philosophy of Charles Sanders Pierce and William James*. Book I, Charles Sanders Pierce. Freeman, Cooper & Co.. (Chapter 3, pp. 63-100)
- Bareiss R. (1989). *Exemplar-based knowledge acquisition; A unified approach to concept representation, classification, and learning*. Academic Press.
- Bareiss R. and Slator B.M. (1992). *From Protos to ORCA: Reflections on a unified approach to knowledge representation, categorization, and learning*. Northwestern University, Institute for the Learning Sciences, Technical Report # 20.
- Boy G. (1991). *Intelligent assistant systems*, Academic Press.
- Boose J. (1986). *Expert transfer for expert systems design*. Elsevier, New York.
- Brown, B., Lewis, L. (1991). A case-based reasoning solution to the problem of redundant resolutions of non-conformances in large scale manufacturing. In: R. Smith, C. Scott (eds.): *Innovative Applications for Artificial Intelligence 3*. MIT Press.
- Burstein, M.H. (1989) Analogy vs. CBR; The purpose of mapping. Proceedings from the Case-Based Reasoning Workshop, Pensacola Beach, Florida, May-June 1989. Sponsored by DARPA. Morgan Kaufmann. pp 133-136.
- CACM (1994). *Communication of the ACM*, March 1994, Vol.37, No.3. (Special issue on commercial applications of AI).
- Carbonell J. (1983). Learning by analogy - formulating and generalizing plans from past experience. In R.S. Michalski, J.G. Carbonell, T.M. Mitchell (eds.): *Machine Learning - An artificial Intelligence Approach*, Vol.1. Morgan Kaufmann. pp137-161.
- Carbonell J. (1986). Derivational analogy. In R.S. Michalski, J.G. Carbonell, T.M. Mitchell (eds.): *Machine Learning - An artificial Intelligence Approach*, Vol.2. Morgan Kaufmann. pp 371-392.

- 
- Chandrasekaran B. (1992). Task-structure analysis for knowledge modeling. *Communications of the ACM*, Vol. 35, no. 9, September (Special issue on modeling), pp. 124-137.
- Clancey W. (1989). Viewing knowledge bases as qualitative models. *IEEE Expert*, Vol.4, no.2, Summer. pp 9-23.
- Clancey W. (1993). The knowledge level reinterpreted: Modeling socio-technical systems. *International Journal of Intelligent Systems*, Vol.8, no.1, pp. 33- 50.
- Compton P. and Jansen R. (1989). A philosophical basis for knowledge acquisition. *EKAW 89, Third European Workshop on Knowledge Acquisition for Knowledge-Based systems*. Ed. by J. Boose, B. Gaines, J.G Ganascia. ECCAI/AFCET/ARC, Paris. pp.75-89.
- Craw S. and Sleeman D. (1990). Automating the refinement of knowledge-based systems. *Proceedings of ECAI-90*, Ninth European Conference on Artificial Intelligence, Stockholm, August. pp 167-172.
- David J-M., Krivine J-P. and Simmons R. (eds.) (1993). *Second generation expert systems*, Springer Verlag.
- DeJong G. and Mooney R. (1986). Explanation based learning - an alternative view. *Machine Learning (journal)*, Vol.1. pp 145-176.
- DeJong G. (1988). An introduction to explanation based learning. In: *Exploring artificial intelligence*; papers from the 6th and 7th National Conferences on Artificial Intelligence. Morgan Kaufmann Publ. pp 45-81.
- Duursma C. and Van de Velde W. (eds.) (1992). *Operationalising knowledge models*. KADS-II Project Report KADS-II/T1.2/VUB/RR/004/1.0, Free University of Brussels - VUB, October.
- Ford K. , Bradshaw J., Adams-Webber J. and Agnew N. (1993). Knowledge acquisition as a constructive modeling activity. *International Journal of Intelligent Systems*, Vol.8, no.1, pp. 9-31.
- Gaines B. (1993). Modeling practical reasoning. *International Journal of Intelligent Systems*, Vol.8, no.1, pp. 51-70.
- Ginsberg A., Weiss S., and Politakis P. (1985). SEEK2 - A generalized approach to automatic knowledge base refinement. *Proceedings IJCAI-85*. Morgan Kaufmann. pp 367-374.
- Greboval C. and Kassel G. (1992). An approach to operationalize conceptual models; The shell Aide. In T. Wetter, K.-D. Althoff, J. Boose, B.R. Gaines, M. Linster, F. Schmalhofer, eds.: *Current Developments in Knowledge Acquisition: Proc. of the 6th European Acquisition Workshop EKAW'92*, Springer Verlag. pp. 37-55.
- Greiner R. and Lenat G. (1980). A representation language language. *Proceedings AAAI-80*, Morgan Kaufmann. pp. 165-169.
- Gentner D. (1983). Structure mapping - a theoretical framework for analogy. *Cognitive Science*, Vol.7. s.155-170.
- Guha R. and Lenat D. (1990). Cyc; A midterm report. *AI Magazine*, Fall, pp. 33-59.
- Hall R. P. (1989). Computational approaches to analogical reasoning; A comparative analysis. *Artificial Intelligence*, Vol. 39, no. 1. pp 39-120.
- Hammond K. (1989). *Case-based planning*. Academic Press.
- Harmon P., Maus R. and Morrissey W. (1988). *Expert systems tools and applications*. John Wiley & Sons.
- Harmon, P (1992). Case-based reasoning III, *Intelligent Software Strategies*, VIII (1).
- Helman D.H. (ed.) (1988). Analogical reasoning. Kluwer Academic. pp 65-103.
- Hennessy, D. and Hinkle, D. (1992). Applying case-based reasoning to autoclave loading. *IEEE Expert* 7(5), pp. 21-26.
- Josephson J. and Josephson S. (1994), *Abductive inference, computation, philosophy, technology*. Cambridge University Press.
- Jullien C., Shadbolt N. and Wielinga B. (eds.) (1992). *Acknowledge Project Final Report*. ACK-CSI-WM-DL-007, Cap Gemini Innovation.

- 
- Kedar-Cabelli S. (1988). Analogy - from a unified perspective. In: D.H. Helman (ed.), *Analogical reasoning*. Kluwer Academic. pp 65-103.
- Kibler D. and Aha D. (1987). Learning representative exemplars of concepts; An initial study. *Proceedings of the fourth international workshop on Machine Learning*, UC-Irvine, June. pp 24-29.
- Kitano H. (1993). Challenges for massive parallelism. *IJCAI, 1993* , *Proceedings of the Thirteenth International Conference on Artificial Intelligence*, Chambery, France. Morgan Kaufmann. pp. 813-834.
- Klinker G., Bohla C., Dallemagne G., Marques D. and McDermott J. (1991). Usable and reusable programming constructs. *Knowledge Acquisition*, 3, pp. 117-136.
- Kodratoff Y. and Tecuci G. (1987a). What is an explanation in DISCIPLINE. *Proceedings of the Fourth International Workshop on Machine Learning*, Irvine, California. July. Morgan Kaufmann Publ. pp 160-166.
- Kodratoff Y. and Tecuci G. (1987b). Techniques of design and DISCIPLINE learning apprentice. *International Journal of Expert Systems*. Vol 1, no. 1. pp 39-69.
- Kodratoff Y. (1990). Combining similarity and causality in creative analogy. *Proceedings of ECAI-90*, Ninth European Conference on Artificial Intelligence, Stockholm, August. pp 398-403.
- Kolodner J. (1983). Maintaining organization in a dynamic long-term memory. Reconstructive memory, a computer model. 2 papers in *Cognitive Science*, Vol.7, pp. 243-280, and pp 281-328.
- Kolodner J. (1993). *Case-based reasoning*, Morgan Kaufmann.
- Koton P. (1989). Using experience in learning and problem solving. Massachusetts Institute of Technology, Laboratory of Computer Science (Ph.D. diss, October 1988). MIT/LCS/TR-441.
- Koton P. (1989). Evaluating case-based problem solving. In: *Proceedings from the Case-Based Reasoning Workshop*, Pensacola Beach, Florida, May-June 1989. Sponsored by DARPA. Morgan Kaufmann. pp 173-175.
- Leake D. (1993). Focusing construction and selection of abductive hypotheses. In *IJCAI, 1993* , pp. 24-31. Morgan Kaufmann.
- Lenat D. and Guha R. (1989). *Building Large Knowledge-Based Systems; Representation and Inference in the CYC Project*. Addison-Wesley.
- Linster M. (1992). Linking models to make sense to modeling to implement systems in an operational modeling environment. In T. Wetter, K.-D. Althoff, J. Boose, B.R. Gaines, M. Linster, F. Schmalhofer, eds.: *Current Developments in Knowledge Acquisition: Proc. of the 6th European Acquisition Workshop EKAW'92*, Springer Verlag. pp. 55-74.
- Matwin M. and Plante B. (1994). Theory revision by analyzing explanations and prototypes. In R. Michalski and G. Tecuci (eds.) *Machine learning, a multistrategy approach, Volume iV*. Morgan Kaufmann. pp 217-238.
- McDermott J. (1988). Preliminary steps toward a taxonomy of problem solving methods. In S. Marcus (ed.): *Automating knowledge acquisition for expert systems*. Kluwer Academic Publ.. pp 226-266.
- Michalski, R. and Tecuci, G. Eds. (1990). *Machine learning: A multistrategy approach*, vol. IV, Morgan Kaufmann, San Mateo.
- Mitchell T., Mahadevan S. and Steinberg L. (1985). LEAP; a learning apprentice for VLSI design. *Proceeding of IJCAI-85*. Morgan Kaufmann. pp 573-580.
- Mitchell T., Keller R., Kedar-Cabelli S. (1986). Explanation based generalization - a unifying view. *Machine Learning (journal)*, Vol.1 s.47-80.
- Mitchell T. , Allen J. , Chalasani P. , Cheng J. , Etzoni O., Ringuette M. and Schlimmer J.C. (1990). Theo: A framework for self-improving systems, In K. VanLehn (ed.), *Architectures for Intelligence*, Lawrence Erlbaum. pp 323-356.
- Mooney R. and Ourston D. (1994). A multistrategy approach to theory refinement. In R. Michalski and G. Tecuci (eds.) *Machine learning, a multistrategy approach, Volume iV*. Morgan Kaufmann. pp 141-164.

- 
- Morik K. (1990). Integrating manual and automatic knowledge acquisition - BLIP. In K.L. McGraw, C.R Westphal (eds.): *Readings in KNOWLEDGE Acquisition*, pp 213-232. Ellis Horwood.
- Murray K. and Porter B. (1989). Controlling search for the consequences of new information during knowledge integration. In: *Proceedings of the Sixth International Workshop on Machine Learning*, Cornell University, June 26-27. Morgan Kaufmann. pp 290-296.
- Musen M.A. (1989). Conceptual models of interactive knowledge acquisition tools. *Artificial Intelligence* 1(1). pp 73-88.
- Nedellec C. and Rouveirol C. (1993). Hypothesis selection biases for incremental learning. *AAAI, 1993) Special workshop on Training Issues in Incremental Learning*.
- Newell A. (1982). The knowledge level, *Artificial Intelligence*, 18. 87-127.
- Nordbø I., Skalle P., Sveen J., Aakvik G., Aamodt A. (1992). *Reuse of experience in drilling - Phase 1 Report*. SINTEF DELAB and NTH, Div. of Petroleum Engineering. STF 40 RA92050 and IPT 12/92/PS/JS. Trondheim.
- Plaza E. and López de Mántaras R. (1990). A case-based apprentice that learns from fuzzy examples. *Proceedings, ISMIS, Knoxville, Tennessee*. pp 420-427.
- Plaza E. and Arcos J. (1993). *Reflection, memory, and learning*. Institut d'Investigacio en Intelligencia Artificial, CSIC/CEAB, Report de Recerca IIIA 93/2. Also to appear in *Proceeding from MSL, 1993) Workshop on Multistrategy Learning*.
- Porter B., Bareiss R. and Holte R. (1990). Concept learning and heuristic classification in weak theory domains. *Artificial Intelligence*, vol. 45, no. 1-2, September. pp 229-263.
- Puerta A., Egar J., Tu S. and Musen M. (1991). A multiple-method knowledge-acquisition shell for the automatic generation of knowledge-acquisition tools. In *KAW-91, Proceedings of the 6th Banff Knowledge Acquisition Workshop*. AAI/University of Calgary.
- Richter M. and Wess S. (1991), Similarity, uncertainty and case-based reasoning in PATDEX. In R.S. Boyer (ed.): *Automated reasoning, essays in honor of Woody Bledsoe*. Kluwer, 1991, pp. 249-265.
- M. Richter, S. Wess, K.D. Althoff, F. Maurer (1993), *EWCBR, 1993) : First European Workshop on Case-Based Reasoning*. University of Kaiserslautern SEKI Report SR, 1993) -12 (SFB 314) Kaiserslautern, Germany, November 1993.
- Riesbeck C. and Schank R. (1989). *Inside Case-based reasoning*. Lawrence Erlbaum.
- Roberts R. and Goldstein I. (1977), *The FRL manual*. MIT AI Laboratory Memo 409, Cambridge.
- Ross B.H. (1989). Some psychological results on case-based reasoning. *Case-Based Reasoning Workshop, DARPA 1989*. Pensacola Beach. Morgan Kaufmann. pp. 144-147
- Schank R. (1982). *Dynamic memory; a theory of reminding and learning in computers and people*. Cambridge University Press.
- Schank R., Collins G. and Hunter L. (1986). Transcending inductive category formation in learning. *Behavioral and Brain Sciences*, Vol.9. pp. 639- 650. Pages 651-686 contains commentaries by several authors.
- Simon H., Kaplan C.A. (1989). Foundations of cognitive science. In M.I Posner (ed.): *Foundations in cognitive science*, MIT Press (A Bradford Book). pp. 1-47.
- Stanfill C., Waltz D. (1988), The memory based reasoning paradigm. In: *Case based reasoning. Proceedings from a workshop*, Clearwater Beach, Florida, May 1988. Morgan Kaufmann Publ. pp.414-424.
- Steels L. (1990). Components of expertise. *AI Magazine*, 11(2), Summer. pp. 29-49.
- Steels L. (1993). The componential framework and its role in reusability, In J-M. David, J-P. Krivine, R. Simmons (eds.), *Second generation expert systems*, Springer Verlag. pp 273-298.
- Sticklen J. (1989). Problem solving architecture at the knowledge level, *Journal of Experimental and Theoretical Artificial Intelligence* 1 (4) 233-271.

- 
- Strube G. (1991). The role of cognitive science in knowledge engineering, In: F. Schmalhofer, G. Strube (eds.), *Contemporary knowledge engineering and cognition: First joint workshop, proceedings*, Springer. pp 161-174.
- Sutton R.S. (ed.) (1992). *Machine Learning*, Vol.8, nos 3/4, May. Special issue on reinforcement learning.
- Tecuci G. (1988). DISCIPLÉ; a theory, methodology and system for learning expert knowledge. University de Paris-Sud, Centre d'Orsay. These presente pour obyenir le titre de Docteur en Science.
- Tecuci G. and Kodratoff Y. (1990). Apprenticeship learning in imperfect domain theories. In Kodratoff and Michalski: *Machine Learning: An artificial intelligence approach*, Vol III, Morgan Kaufmann. pp. 514-551.
- Thagard P. (1988). *Computational Philosophy of Science*, MIT Press/Bradford Books.
- Tulving, E. (1972). Episodic and semantic memory. In E. Tulving and W. Donaldson: *Organization of memory*, Academic Press, 1972. pp. 381-403.
- Turban E. (1992). *Expert systems and artificial intelligence*. Macmillan Publishing Co.
- Van de Velde W. (1988). *Learning from experience*. Ph.D. Dissertation, Free University of Brussels - VUB.
- Van de Velde W. (1993). Issues in knowledge level modeling, In J-M. David, J-P. Krivine, R. Simmons (eds.), *Second generation expert systems*, Springer. pp 211-231.
- Van Lehn K. (ed.) (1990). *Architectures for Intelligence*. Lawrence Erlbaum.
- Van Marcke K. (1988). The use and implementation of the representation language KRS. Ph.D. Dissertation, Free university of Brussels - VUB. VUB AI Lab Technical Report 88\_2.
- Vanwelkenhuysen J. and Rademakers P. (1990). Mapping a knowledge level analysis onto a computational framework. *Proceedings of ECAI-90*, Ninth European Conference on Artificial Intelligence, Stockholm, August. pp. 661-66.
- Veloso, M.M., Carbonell, J. (1993). Derivational analogy in PRODIGY. In *Machine Learning* 10(3), pp. 249-278.
- Weintraub M. (ed.) (1992). *Proceedings of the ML92 Workshop on Computational Architectures for Machine Learning and Knowledge Acquisition*. Aberdeen, Scotland.
- Wess S., Althoff K-D. and Richter M.M. (eds.) (1994). *Topics in Case-Based Reasoning; Selected Papers from the First European Workshop on Case-Based Reasoning - EWCBR, 1993* . Springer Verlag.
- Wielinga B.J., Schreiber A.Th. and Breuker J.A. (1992). KADS: A modeling approach to knowledge engineering. *Knowledge Acquisition*, 4(1).
- Wielinga B., Van de Velde W., Schreiber G. and Akkermans H. (1993). Towards a unification of knowledge modeling approaches, In J-M. David, J-P. Krivine, R. Simmons (eds.), *Second generation expert systems*, Springer Verlag. pp 299-335.