AI Magazine Volume 12 Number 2 (1991) (© AAAI)

Improving Human Decision Making through Case-Based Decision Aiding[1]

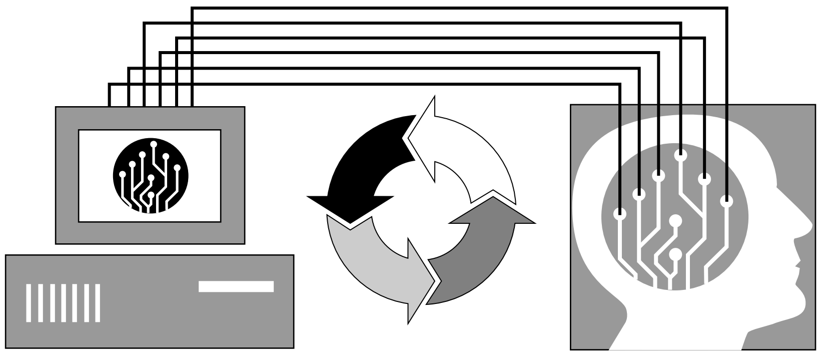
Janet L. Kolodner

Case-based reasoning provides both a methodology for building systems and a cognitive model of people. It is consistent with much that psychologists have observed in the natural problem solving people do. Psychologists have also observed, however, that people have several problems in doing analogical or case-based reasoning. Although they are good at using analogs to solve new problems, they are not always good at remembering the right ones. However, computers are good at remembering. I present case-based decision aiding as a methodology for building systems in which people and machines work together to solve problems. The case-based decision-aiding system augments the person's memory by providing cases (analogs) for a person to use in solving a problem. The person does the actual decision making using these cases as guidelines. I present an overview of case-based decision aiding, some technical details about how to implement such systems, and several examples of case-based systems.

Much of AI is sold to the world as fully automated expert systems, that is, systems based on rules that, given a problem statement, will produce a solution. Such systems have been highly successful in solving problems in many well-circumscribed domains. They have not been successful, however, in solving problems requiring creativity, broad commonsense knowledge, or esthetic judgment.

There are three ways we might deal with this problem: First, we might throw our hands up in despair, claiming that after 30 years, if AI can't solve these problems, it isn't worth continuing the endeavor. The opponents of AI have been vocal about this approach. However, this approach is the wrong way to deal with the failings of expert systems. After all, the technology is young and has been useful to many industries in solving important but narrow problems.

A second approach is to continue trying to give the computer the full capabilities people have. The majority of AI researchers are involved in this effort. Doug Lenat, for example, is attempting to put all human consensus knowledge into the machine in CYC (Guha and Lenat 1990; Lenat and Guha 1990). Others are working on a variety of necessary reasoning methods and knowledge representations. Indeed, much of the research



0738-4602/91/$4.00 ©1991 AAAI

in the AI lab at the Georgia Institute of Technology is aimed in this direction. This approach is the one that will lead us into the AI future. It will allow us to better understand what cognition is about and will allow us to eventually develop systems with improved cognitive capabilities.

The third approach is aimed toward the present. Researchers ask, "Is there a way to take what we know now about AI and create systems that can do better than current ones?" The answer to this question is, "Yes, if we can develop an appropriate symbiotic relationship between people and machines." We are then left with a set of other more substantial questions: How can we make sure that computers and people have the right interactions? What responsibilities should the computer take on? What should people be responsible for? What methodologies are available for building these symbiotic systems?

One way to approach these problems is to examine the natural reasoning people do, develop a cognitive model of this reasoning, and explore which parts of the process are easy for people and which parts are hard. At the same time, it is appropriate to ask whether the machine can provide help in those areas where people have trouble. One hopes a complement can be found. Such an approach has the advantage of producing a system that can be useful now and that can be made even better as we begin to better understand what else people need and how to make the computer perform these services.

My approach to these questions comes from experience with a reasoning methodology called case-based reasoning (Kolodner 1988; Kolodner, Simpson, and Sycara 1985; Hammond 1986, 1989). *Case-based reasoning* is an analogical reasoning method. It means reasoning from old cases or experiences in an effort to solve problems, critique solutions, explain anomalous situations, or interpret situations. Many computer programs have been written that use case-based reasoning for problem solving or interpretation. MEDIATOR (Simpson 1985; Kolodner and Simpson 1989) and PERSUADER (Sycara 1987), for example, use cases to resolve disputes. JULIA (Kolodner 1987a, 1987b; Hinrichs 1988, 1989), CLAVIER (Barletta and Hennessy 1989), and KRITIK (Goel 1989; Goel and Chandrasekaran 1989) use case-based reasoning for design. CHEF (Hammond 1986, 1989) and PLEXUS (Alterman 1988) are case-based planners. HYPO (Ashley 1988; Ashley and Rissland 1987) is a case-based legal reasoner. CASEY (Koton 1988), PROTOS (Bareiss 1989), CELIA (Redmond 1989), and MEDIC (Turner 1989) use case-based reasoning for diagnosis.

Case-based reasoning provides both a methodology for building systems and a cognitive model of people. It is consistent with much that psychologists have observed in the natural problem solving that people do (for example, Read and Cesa [1990]; Klein and Calderwood [1988]; and Ross [1986]). People tend to be comfortable using case-based reasoning for decision making.[2] In dynamically changing situations and other situations where much is unknown and when solutions are not clear cut, it seems to be the preferred method of reasoning (Klein and Calderwood 1988).

Psychologists have observed, however, that people have several problems in doing analogical or case-based reasoning. Although they are good at using analogs to solve new problems, they are not always good at remembering the right ones (Read and Cesa 1990; Holyoak 1985; Gentner 1987, 1989). However, computers are good at remembering. The idea in case-based decision aiding is that the computer augments the person's memory by providing cases (analogs) for a person to use in solving a problem. The person does the actual decision making using these cases as guidelines. In essence, computer augmentation of a person's memory allows this person to make better case-based decisions because it makes more cases (and perhaps better ones) available to the person than would be available without the machine. At the same time, the person is free to use a reasoning method that comes naturally to make these decisions.

In this article, I first present a short overview of case-based reasoning, then discuss the analogical problem solving people do and the help they need and what the computer can provide. I continue with a technical description of the *indexing problem*, the problem of making sure cases are available at appropriate times, and then discuss how cases might be chosen to put into such a system. I close with the implications of such a system for human decision making, for both novices and experts.[3]

Case-based reasoning is an analogical reasoning method.

Case-Based Reasoning: An Overview

A host is planning a meal for a set of people who include, among others, several people who eat no meat or poultry, one of whom is also allergic to milk products, several meat-and-potatoes men, and her friend Anne. Because it is tomato season, she wants to use tomatoes as a major ingredient in the meal. As she plans the meal, she remembers the following:

If we watch the way people… solve problems, we are likely to observe case-based reasoning…

> I once served tomato tart (made from mozzarella cheese, tomatoes, dijon mustard, basil, and pepper, all in a pie crust) as the main dish during the summer when I had vegetarians come for dinner. It was delicious and easy to make. But I can't serve that to Elana (the one allergic to milk).

She considers whether she can adapt this solution to suit Elana. Another case suggests an adaptation.

> I have adapted recipes for Elana before by substituting tofu products for cheese. I could do that, but I don't know how good the tomato tart will taste that way.

She decides not to serve tomato tart and continues planning. Because it is summer, she decides that grilled fish would be a good main course. However, she now remembers something else:

> Last time I tried to serve Anne grilled fish, she wouldn't eat it. I had to put hot dogs on the grill at the last minute.

This memory suggests to her that she shouldn't serve fish, but she wants to anyway. She considers whether there is a way to serve fish so that Anne will eat it.

> I remember seeing Anne eat mahi-mahi in a restaurant. I wonder what kind of fish she will eat. The fish I served her was whole fish with the head on. The fish in the restaurant was a fillet and more like steak than fish. I guess I need to serve a fish that is more like meat than fish. Perhaps swordfish will work. I wonder if Anne will eat swordfish. Swordfish is like chicken, and I know she eats chicken.

Here, she is using examples and counterexamples of a premise (Anne doesn't eat fish) to try to derive an interpretation of the premise that stands up to scrutiny.

The hypothetical host is utilizing case-based reasoning to plan a meal. In *case-based reasoning,* a reasoner remembers previous situations similar to the current one and uses them to help solve the new problem. In the previous example, remembered cases are used to suggest a means of solving the new problem (for example, to suggest a main dish), suggest a means of adapting a solution that doesn't fit (for example, substitute a tofu product for cheese), warn of possible failures (for example, Anne won't eat fish), and interpret a situation (for example, why didn't Anne eat the fish; will she eat swordfish?).

Case-based reasoning can mean adapting old solutions to meet new demands, using old cases to explain new situations, using old cases to critique new solutions, or reasoning from precedents to interpret a new situation (much like lawyers do) or create an equitable solution to a new problem (much like labor mediators do).

If we watch the way people around us solve problems, we are likely to observe case-based reasoning in constant use. Attorneys are taught to use cases as precedents for constructing and justifying arguments in new cases. Mediators and arbitrators are taught to do the same. Other professionals are not taught to use case-based reasoning but often find that it provides a way to efficiently solve problems. Consider, for example, a doctor faced with a patient who has an unusual combination of symptoms. If s/he's previously seen a patient with similar symptoms, s/he is likely to remember the old case and propose the old diagnosis as a solution to the new problem. If proposing these disorders was previously time consuming, this approach is a big time savings. Of course, the doctor can't assume the old answer is correct. S/he must still validate it for the new case in a way that doesn't prohibit considering other likely diagnoses. Nevertheless, remembering the old case allows him(her) to easily generate a plausible answer.

Similarly, a car mechanic faced with an unusual mechanical problem is likely to remember other similar problems and consider whether these solutions explain the new problem. Doctors evaluating the appropriateness of a therapeutic procedure or judging which of several are appropriate are also likely to remember instances using each procedure and make their judgments based on previous experiences. Problem instances of using a procedure are particularly helpful here; they tell the doctor what could go wrong, and when an explanation is available that explains why the old problem occurred, they focus the doctor in finding the information s/he needs to make sure the problem won't show up again. We hear cases being cited time and again by our political leaders in explaining why some action was taken or should be taken. Many management decisions are also made based on previous experience.

Case-based reasoning is also used extensively in day-to-day commonsense reasoning. The meal-planning example presented previously is typical of the reasoning we all do from day to day. When we order a meal in a restaurant, we often base decisions about what might be good on our other experiences in this restaurant and those like it. As we plan our household activities, we remember what previously worked and didn't work and use this information to create our new plans. A childcare provider mediating an argument between two children remembers what previously worked and didn't work in calming such situations and bases his(her) suggestion on this information.

In general, the second time solving some problem or doing some task is easier than the first because we remember and repeat the previous solution. We are more competent the second time because we remember our mistakes and go out of our way to avoid them.

There are two styles of case-based reasoning: problem solving and interpretive. In the *problem-solving style* of case-based reasoning, solutions to new problems are derived using old solutions as a guide. Old solutions can provide almost-right solutions to new problems, and they can provide warnings of potential mistakes or failures. In the previous example, past cases suggest tomato tart as a main dish, a method of adapting tomato tart for those who don't eat cheese, and a type of fish that Anne will eat. A case also warns of the potential for a failure—Anne won't eat certain kinds of fish. Case-based reasoning of this variety can support a variety of problem-solving tasks, including planning, diagnosis, and design.

In the *interpretive style*, new situations are evaluated in the context of old situations. A lawyer, for example, uses interpretive case-based reasoning when s/he uses a series of old cases to justify an argument in a new case. A child who says "But you let sister do it" is using a case to justify his(her) argument. Managers making strategic decisions use the interpretive style. We often use interpretive case-based reasoning to evaluate the pros and cons of a problem solution. In general, the interpretive style of case-based reasoning is useful for situation classification; the evaluation of a solution; argumentation; the justification of a solution, interpretation, or plan; and the projection of effects of a decision or plan. Interpretive case-based reasoning can also be used during problem solving, as we saw the host in our initial example do when trying to justify serving swordfish to a guest who is known to not like some kinds of fish.

Interpretive case-based reasoning is most useful when there are no computational methods available to evaluate a solution or position. Often, in these situations, there are so many unknowns that even if computational methods were available, the knowledge necessary to run them would usually be absent. A reasoner who uses cases to help evaluate and justify decisions or interpretations is making up for his(her) lack of knowledge by assuming that the world is consistent.

Both styles of case-based reasoning depend heavily on a case-retrieval mechanism that can recall useful cases at appropriate times, and also in both styles, storage of new situations in memory allows learning from experience. The problem-solving style is characterized by the substantial use of adaptation processes to generate solutions and interpretive processes to judge derived solutions. The interpretive style uses cases to provide justifications for solutions, allowing the evaluation of solutions when no clear-cut methods are available and the interpretation of situations when definitions of the situation's boundaries are open ended or fuzzy.

The major processes shared by reasoners that do case-based reasoning are case retrieval and case storage (also called memory update). To make sure that poor solutions are not repeated along with the good ones, case-based reasoners must also evaluate their solutions.

The two styles of case use, however, require that different reasoning be done once cases are retrieved. In problem-solving case-based reasoning, a ballpark solution to the new problem is *proposed* by extracting the solution from some retrieved case. This step is followed by *adaptation*, the process of fixing an old solution to fit a new situation, and *criticism*, the process of evaluating the new solution before trying it out. In interpretive case-based reasoning, a ballpark interpretation or desired result is proposed, sometimes based on retrieved cases, sometimes imposed from the outside (for example, when a lawyer's client requires a certain result). This step is followed by *justification*, the process of creating an argument for the proposed solution, which is done by comparing and contrasting the new situation to prior cases, and *criticism*, the process of debugging the argument, which is

There are two styles of case-based reasoning: problem solving and interpretive.

done by generating hypothetical situations and trying the argument out on them.

These steps are, in some sense, recursive. The criticize and adapt steps, for example, often require new cases to be retrieved. There are also several loops in the process. Criticism can lead to additional adaptation, so might evaluation. In addition, when reasoning is not progressing well using one case, the whole process might need to be restarted from the top, with a new case chosen.

Human Use of Case-Based and Analogical Reasoning

In the context of case-based and analogical reasoning, let us examine what people do well, what people do badly, and the reasons behind using case-based reasoning.

What People Do Well

Psychologists observing the problem-solving and decision-making procedures of people see them using case-based reasoning under a variety of circumstances. Ross (1986, 1989), for example, shows that people learning a new skill often refer to previous problems to refresh their memories on how to do the task. Research conducted in the lab at Georgia Tech shows that both novice and experienced car mechanics use their own experiences and those of others to help them generate hypotheses about what is wrong with a car, recognize problems (for example, a testing instrument is not working), and remember how to test for different diagnoses (Lancaster and Kolodner 1988; Redmond 1989). Other research in the lab shows that physicians extensively use previous cases to generate hypotheses about what is wrong with a patient; help them interpret test results; and select therapies when several are available, and none are well understood. Researchers also observed architects and caterers recalling, merging, and adapting old design plans to create new ones.

Klein and Calderwood (1988) observed expert decision makers in complex, dynamically changing situations. These experts use analogs to understand situational dynamics, generate options, and predict the effects of implementing an option in several different naturalistic decision-making situations. They observed experts using cases to both suggest solutions that were then adapted and evaluate solutions and situations. In the naturalistic situations they observed, the use of analogs was far more important than the application of abstract principles, rules, or conscious deliberation about alternatives. Analogs or cases provided concrete manifestations of the rules or principles that allow them to easily be applied. Cases also allowed decision makers to be alert to causal factors operating during an incident, anticipate what might happen if a course of action was implemented, suggest options, and be reassured that an option worked and could be relied on. Their primary power, claims Klein (Klein, Whitaker, and King 1988), is that they allow the decision maker to deal with unknown and uncertain information. An analog reflects the ways variables affected solutions in the past. In the same study, Klein and Whitaker found that the case-based method is much more reliable than unstructured prediction when there are many unknowns.

Read (Read and Cesa 1990) observed people using old cases to explain anomalous occurrences and found them particularly adept at using this approach when the anomalous event reminded them of a personal experience.

The conclusion I draw from these studies is that reasoning using analogs is a natural process for people, especially when there is much uncertainty or many unknowns and during early learning. People know well how to use analogs to reason, and the use of analogs in reasoning (at least for experts) results in reliable solutions.

What People Do Badly

Despite the fact that people use cases well to reason, there are a number of pitfalls for people when using cases. Some people blindly use case-based reasoning, relying on previous experience without validating it in the new situation. A case-based reasoner might allow cases to bias him(her) too much in solving a new problem (Gilovich 1981), and often, people are not reminded of the most appropriate sets of cases when they are reasoning (Holyoak 1985; Gentner 1989). In addition, when there is much to remember, people cannot always access the right information when they need it.

Novices have a variety of other problems. They cannot do analogical reasoning well for two of reasons. First, they are missing the experiences they need to make good analogical decisions. Second, they are missing the experiences that tell them which parts of a situation are the important ones to focus on; that is, their criteria for judging the similarity of cases is deficient.

Why Case-Based Reasoning?

People use case-based or analogical reasoning in the whole variety of situations previously illustrated and discussed. One question we might want to consider is "why"? We start by considering why a doctor or anyone else trained in the practice of making logical decisions would make case-based inferences. After all, the doctor is trained to use facts and knowledge, and case-based reasoning looks like it is based on hearsay. The answer is simple. The doctor is trained to recognize disorders in isolation and to recognize common combinations of disorders. S/he also knows the etiology of disorders, that is, how they progress. However, s/he cannot be trained to recognize every combination of disorders, and the knowledge s/he has of disease processes is time consuming to use to generate plausible diagnoses. If s/he once used his(her) knowledge of the disease process to solve a hard problem, it makes sense to cache the solution in such a way that it can be reused. That is, once s/he has learned to recognize a novel combination of disorders, if s/he remembers this experience, s/he will be able to recognize it again, just as s/he recognizes more common combinations, without the difficult reasoning that was necessary the first time. The logical medical judgment comes later in deciding whether the patient does indeed have the proposed set of diseases.

Thus, case-based reasoning is useful to people who know a lot about a task and domain because it gives them a way to reuse hard reasoning that they've done in the past. It is equally useful, however, to those who know little about a task or domain. Consider, for example, a person who has never done any entertaining yet has to plan the meal specified in the introduction. His(her) own entertaining experience won't help. However, if s/he has been to dinner parties, s/he has a place to start. If s/he remembered meals s/he'd been served under circumstances similar to those s/he has to deal with, s/he could use one of these meals to get started. For example, if s/he could generate a list of large dinner parties s/he has attended, then for each one, s/he could figure out whether it was easy to make and inexpensive, and when s/he remembered one, adapt it to fit.

Case-based reasoning is also useful when knowledge is incomplete, or evidence is sparse. Logical systems have trouble dealing with either of these situations because they want to base their answers on what is well known and sound. More traditional AI systems use certainty factors and other methods of inexact reasoning to counter these problems, all of which require considerable effort on the part of the computer, and none of which seem intuitively plausible. Case-based reasoning provides another method for dealing with incomplete knowledge. A case-based reasoner makes assumptions to fill in incomplete or missing knowledge based on what his(her) experience tells him(her) and goes on from there. Solutions generated in this way won't always be optimal or even right, but if the reasoner is careful about evaluating proposed answers, the case-based methodology provides a way to easily generate answers.

Using Case-Based Reasoning to Aid Human Decision Making

We now have several facts at our disposal to use in exploring how decision making in people can be improved. First, people find it easy to use analogs in reasoning, but they often find it hard to remember the right ones. Second, the use of analogs in solving problems and making decisions has many advantages if analogs are used well: They help in dealing with uncertainty, assessing situations, deriving solutions, and so on. Third, the methodology called case-based reasoning, which has been implemented in several computer programs, provides us with computational methods for case retrieval, adaptation, and evaluation.

My proposal is to use the computational methods we have available to implement systems that help people, both novices and experts, to better do analogical reasoning. Because people have trouble remembering appropriate cases, the system will augment their memories by providing, at appropriate times, the relevant experiences of others. However, because people are better at dealing with esthetics, ethics, creative adaptation, and judgment, we leave the real decision making to people. That is, the computer will provide cases to human problem solvers at appropriate times to help them with such tasks as coming up with solutions, adapting old solutions, critiquing and evaluating solutions, and warning of potential problems.

Two hypothetical systems illustrate the division of labor. The first system is a design assistant. The second is a mediator's assistant. In both systems, the language of discourse is English, which is the easiest way to explain the interactions between a system and a person. A better interface than the one presented here, however, would probably be more graphic and allow communication through menus and mice.

Case-based reasoning is also useful when knowledge is incomplete, or evidence is sparse.

An Architect's Assistant

The hypothetical system I show here helps an architect in doing design. An architect has to deal with a number of issues: The function must be appropriate, the design must fit the site, costs must be within limits, and so on. In the example I show, the architect's problem is to design a geriatric hospital. Let us assume that the computer screen is configured with a space for notes, a space for graphic manipulations, a space for the problem specification, and a space where cases are presented. Let us further assume that each case has both a picture part and a textual part.

On the screen, we see the new problem in the problem specification space and a representation of the site, showing its contour, size, and shape, in the graphic space.[4]

Problem: Design a geriatric hospital: The site is a 4-acre wooded sloping square; the hospital will serve 150 inpatients and 50 outpatients daily; offices for 40 doctors are needed. Both long-term and short-term facilities are needed. It should be more like a home than an institution, and it should allow easy visitation by family members.

Screen: The site, its contour and shape, is shown.

The person now uses the mouse to highlight those parts of the problem specification that s/he feels need to be addressed at this point early in the design.

Person: Hospital, homeyness, provision of visitation, long-term and short-term care, and geriatric are the elements highlighted.

The computer uses these highlighted parts of the problem as preferences in finding useful cases in memory. The most useful cases will match all these features; less useful ones (that perhaps can address only some part of the problem) will match some subset of these features. The computer finds many cases and partitions them according to how well they match the highlighted criteria. It presents its ordered partitions and asks the person which s/he would like to see more detail on. At the same time, the computer does some bookkeeping (invisible right now). It tracks which cases in its lists achieve which of the requested features.

Computer: I know of many long-term facilities for the elderly, many acute-care hospitals, and one geriatric hospital: Wesley Woods in Atlanta. Of the acute-care hospitals, Piedmont Hospital in Atlanta is the most homey. Of the long-term facilities, the Jewish Home in Fairfield County, Connecticut, is closest in the type of care it provides and the ability to provide opportunities for family visitation. What would you like to see?

The person chooses to see Wesley Woods, the only geriatric hospital with both long-term and short-term facilities. The computer shows a picture of the layout of Wesley Woods and provides a commentary.

Person: Show Wesley Woods.

Screen: Picture and layout of Wesley Woods are shown.

Computer: The goal in designing Wesley Woods was to create a homey atmosphere. To achieve this goal, they used residential, rather than institutional, furnishings, forms, and materials and made sure it was cheery and colorful. They made all the corridors short, so people would feel like they were in a home-sized place. The functional units of the hospital were hidden from normal sight, so the patients didn't feel like they were in a hospital. The design did a very good job of providing homeyness. However, there was one big problem. It is hard for patients and staff to find their way around because the views give no distinguishing landmarks or orienting features, and it is not possible to see one's destination until nearly there. It looks so much like a home that patients and visitors are not sure what to do once they arrive there.

The person makes some notes based on the Wesley Woods successes and failures (make sure corridors have views; make sure corridors have distinguishing features), decides that there are large portions of this design s/he can borrow, and begins to manipulate this design (using icons) to make it fit the new site. It won't fit on the site because of the slope. S/he adapts its design by adding a lower floor, subtracting part of the lower floor of Wesley Woods, and putting the two main entries on different levels.

The computer, which is responsible for bookkeeping, now notices that the building has entries on two levels. It knows this configuration to be a difficult design problem and, therefore, attempts to retrieve cases that can help the designer. It retrieves several cases that show how to use two entrances. It interrupts the designer and makes a suggestion. The architect uses this information.

> **Computer:** I have several examples of hospital buildings with entrances on two levels. Georgia Baptist Hospital has lobbies on two levels, one for inpatients, one for outpatients. Brighams and Womens in Boston has entries on two levels with a two-floor lobby that combines inpatient and outpatient functions. The lobbies are connected by a ceremonial escalator.
>
> **Person:** I'll put the outpatient lobby on the bottom level and the inpatient lobby on the second level.

The computer has tracked the goals of the designer and how they were achieved in other cases that the designer might not have seen yet. It has a mechanism for noticing when design goals are violated by design decisions. In this case, because the inpatient entrance is above usable ground, and the Jewish Home in Fairfield had provided for easy visitation by providing easy access to the usable outdoors, the program notices a violation of design goals. It is also able to retrieve another case, where even though the ground sloped, access to the outside was easily provided. It interrupts the architect to warn of the potential problem and suggest a solution.

> **Computer:** One of the nice features of the Jewish Home in Fairfield is the access to the outside that patients have. That's a potential problem if the inpatient lobby is on the upper floor.
>
> I know of another institution where access was important and was provided by putting patient rooms and gathering places against the same slope, so people could walk right outside. In another, the two functions were divided between two different buildings, each on a different part of the slope, so everyone had access to the outside.

I do not continue with the example here but stop to point out the responsibilities of the machine and the person. The computer has two responsibilities: bookkeeping and retrieval. The bookkeeping it does is of two varieties. First, it tracks the good points of designs it remembers. The Jewish Home, for example, had the advantage of providing patients with easy access to the outside. This configuration made visitation by family members pleasant. In addition, the computer makes easy inferences to make sure, to the best of its knowledge, that good points of designs it knows of are not violated and that bad points are not repeated.

The case retrieval done by the system has three purposes: First, cases suggest solutions to problems or methods for computing solutions. Second, cases provide warnings of potential problems. Third, cases help in critiquing.

Everything the computer does is in service of the responsibilities of the person using the system. Thus, the human user is responsible for hard adaptation, the choice of features to consider, the choice of cases to consider, the evaluation of suggestions, the evaluation of warnings, and all decision making.

A Mediator's Assistant

In the architect's assistant, the computer looked over the person's shoulder and provided warnings and suggestions as necessary. In this example, also hypothetical, we see the user requesting cases from the system to help him(her) in answering several necessary questions to assess the situation and come up with a solution. This example is based on the reasoning done by an automated case-based mediation system called PERSUADER (Sycara 1987). PERSUADER's task was to mediate labor-management disputes.

In this example, we see the mediator presented with a problem. The union wants a large salary increase, but the company cannot afford it. To figure out what to do, the mediator must collect some information. S/he wants to know why the company is losing money. The system helps him(her) by providing several cases of companies losing money, each of which suggests a different explanation. Note again that I use English interaction because it is easy to show. The person might actually interact with the system in some other way.

> **Problem:** Southern Airlines presents its employees with the ultimatum that if they don't take wage cuts of 8 percent, the company, which has become noncompetitive, will go bankrupt.
>
> **Mediator:** Get me cases to suggest why the company is losing money although the industry is prosperous.
>
> **Computer:**
>
> **Great Railroad Company:** When it was losing money in prosperous times, it was because the union was insisting on featherbedding, an expensive and unnecessary practice.
>
> **Northern Airlines Shuttle:** When it was losing money in prosperous times,

it was because the management was not treating the employees well. The employees, in turn, were not as productive as they should have been.

Western Trucking: When it was losing money in prosperous times, it was because of mismanagement by the executive board. Board members were using the resources of Northern Airlines for purposes unrelated to the company.

The mediator considers each alternative, collecting the information s/he needs and decides that this situation is a case of mismanagement. S/he now needs to find a solution. S/he adds to the representation of the problem that there is mismanagement in the company and asks the computer to suggest solutions. The computer responds with several cases.

Computer:

Western Trucking: Management was forced to resign, and the employees took a pay cut until things were back to normal. It took time to get back on the road, but the consensus is that the new management did better than the old management could have.

Foobaz Airlines: Employees took a pay cut to avoid bankruptcy, but management was forced to put employees on the board and give them a share in the company. The president took things under control, firing those managers who had been involved in mismanagement. S/he made it clear that employees on the board were to be taken seriously. Because employees felt responsible for their share of the company and also that they were listened to, productivity went up immediately, and the company began making more money. Two years later, the employees not only regained their old salaries but began making more than their counterparts in other airlines. This approach could not have been taken without the willingness of both parties to cooperate in the agreement.

In this hypothetical example, the aiding system is much more passive than in the previous example. Even so, cases that are recalled by the system provide the user with much important data. We can think of this system as a smart interface for a database system. It is able to recall partially matching cases that a decision maker can base analysis and decisions on. It is more than a database because it does its retrieval based on the specifics of a situation and finds partially matching cases that can be used to answer the specific questions of the user. It does not require the full matching of features, nor does it require a database administrator to formulate queries. Rather, it allows the decision maker to ask the questions and be close to the data decisions are based on.

Implementing Case-Based Decision-Aiding Systems

The examples show us that case-based decision-aiding systems can be passive or active. They can be used to aid novices or experts, and they can be used to help a wide variety of decision-making activities. Some function of such a system depends on the use it will be put to, but much of the function of any such system depends on the cases that are in the system and the means of retrieving them. Several issues come up in discussing implementation. We could consider how to build the systems, what retrieval algorithms to use, what memory update algorithms to use, and so on, or we can consider some conceptual issues that are independent of any particular implementation. We choose to do the latter, making the assumption that a smart programmer can program a retrieval algorithm or that case-based reasoning shells will be available and used. In this section, therefore, I first consider the representation of cases; then the issue of assigning indexes to cases such that they can be retrieved at appropriate times; and, finally, the choice of cases to seed a case-based decision aider.

Representing Cases

Representations of cases can be in any of several forms, including predicate representations, frame representations, or representations resembling database entries. What is important to this discussion is the content that must be represented.

There are three major parts to any case, although for any particular case, they might not all be filled in: First is the *problem-situation description,* the state of the world at the time the case was happening and, if appropriate, what problem needed solving at this time. Second is the *solution*, the stated or derived solution to the problem specified in the problem description. Some case-based reasoners also store traces of how the problem was solved. Third is the *outcome*, the resulting state of the world when the solution was carried out.

Depending on what is included in a case, the case can be used for a variety of purposes. For example, cases that include a problem and solution can be used in deriving solutions to new problems. Those cases with a situation description and outcome can be used in evaluating new situations. If the case also has a specified solution, it can be used in evaluating proposed solutions and anticipating potential problems before they occur.

In addition, a case is as useful in later reasoning as the information it holds. Several items might be represented in an outcome, for example. The baseline is *execution feedback,* that is, what the state of the world was after (and while) the solution was carried out. With this information, a case-based reasoner can anticipate potential problems and project the outcome of new solutions to aid in evaluation. If the case also includes an explanation of why an outcome came about (that is, the causal connections between the initial situation, the solution, and the outcome) and the way in which it was repaired, the case can also be used for guidance in repairing a similar failure in the future.

Similarly, if the solution part of a case designates only a solution, it can be used to help in proposing a solution to a new case. If the solution also includes a store of how it was derived, then the old solution method can be attempted in cases where the old solution is inapplicable. If it includes connections between the problem description, the situation, and the solution, they can be used to help in guiding adaptation.

The Indexing Problem

Perhaps the biggest issue in case-based reasoning and the design of case-based decision-making aids is the retrieval of appropriate cases. How can we make the computer find the right ones at the right times? As I stated earlier, I call this problem the indexing problem. Essentially, the problem is assigning appropriate labels to cases at the time they are entered into the case memory to ensure that they can be retrieved at appropriate times. In general, these labels designate under what circumstances the case might appropriately be retrieved. They are used at retrieval time to judge the appropriateness of an old case to a new situation.

The analysis of some remindings (something that serves as a reminder of something else) collected from people, coupled with experience in building case-based reasoning systems, has led the case-based reasoning community to propose several guidelines for index (label) selection: (1) indexes should be predictive, (2) indexes should be abstract enough to make a case useful in a variety of future situations, (3) indexes should be concrete enough to be recognizable in future cases, and (4) predictions that can be made should be useful.

Perhaps the biggest issue in case-based reasoning… is the retrieval of appropriate cases.

Predictive Features I begin by explaining what predictive means. As previously stated, at the most basic level, a case is a description of a problem, its solution, and the outcome of carrying out the solution. Different combinations of problem descriptors are taken into account in coming up with the solution and are responsible for choices made about the solution. Other combinations of problem and solution descriptors, coupled with descriptors of the world, are responsible for parts of the outcome. Any descriptor combination that is responsible for some piece of the solution or its outcome is said to be *predictive* of the part of the solution or outcome that it influenced.

Some examples illustrate. First, consider a meal that was a failure because some guest, a vegetarian, could not eat the main dish, which was meat. The combination of descriptors, “guest was a vegetarian” and “meat was an ingredient in the main dish,” was responsible for the failure. If we see this combination again in a meal we plan, we can predict the same failure (the vegetarian won’t be able to eat the main dish). These descriptors are predictive of a particular outcome.

Predictive descriptors can also predict better-than-expected outcomes. Consider a cook who decided to try a new recipe that included a combination of novel ingredients, for example, peanut butter, ginger, and eggplant. She might have been leery of the result but willing to take a chance. The dish turned out to be good; eggplant, peanut butter, and ginger complemented each other well. This combination of descriptors—”peanut butter is an ingredient,” “ginger is an ingredient,” and “eggplant is an ingredient”—was responsible for a successful outcome, the tasty dish. If a dish with this combination of descriptors is considered again, this case can be used to predict that it will be tasty. Alternatively, if these ingredients are available again, this case can be used to suggest a dish that combines them.

Two more examples complete the illustration. Consider now a doctor whose patient had a novel set of symptoms. She considered many different diagnoses and tried many different treatments before finally figuring out what the combination of disorders was and what treatment was effective. The combination of symptoms, which is responsible for the difficulty in reasoning and predicts a diagnosis and treatment, is a good index. Finally, consider a legal decision that was determined by a loophole. Those features of the case that enabled the loophole are the predictive ones. They allowed the loophole to be used in this case and, if seen again, predict that the loophole can be used again.

Abstractness of Indexes Although cases are specific, indexes to cases need to be chosen so that the case can be used in as broad a selection of situations as appropriate. Often, this approach means indexes should be more abstract than the details of a particular case. Consider, for example, a case from CHEF (Hammond 1986, 1989). CHEF just created a recipe for beef and broccoli, a stir-fried dish. When it first created the recipe and tried it out, it found that the broccoli got soggy. It fixed the order of the steps in the recipe so that the broccoli remained crisp. This case could be indexed in several ways: (1) dish is prepared by stir frying, dish includes beef, and dish includes broccoli and (2) dish is prepared by stir frying, dish includes meat, and dish includes a crisp vegetable.

The first set allows this case to be recalled whenever beef and broccoli are to be stir fried together. This index, however, would not allow recall of this case, for example, when chicken and snow peas are to be stir fried. However, the order of the steps probably has to be the same as for beef and broccoli—snow peas are also a crisp vegetable that should remain crisp. Indexing by the second set of descriptors makes this case more generally applicable.

Concreteness of Indexes The danger of abstract indexes is that they can be so abstract that the reasoner would never realize that a new situation had these descriptors except through extensive inference. Thus, although indexes need to be generally applicable, they need to be concrete enough so that they can be recognized with little inference. Consider another example from CHEF to illustrate this point. CHEF just created a new recipe for a strawberry souffle. It created this dish by adapting a recipe for vanilla souffle. When it first made the souffle, it fell. CHEF figured out that the problem was that the liquids and leavening were not balanced: There was too much liquid for the amount of leavening in the recipe. It also figured out that the extra liquid was because of the juice in the strawberries. It solved the problem by increasing the leavening to counter the effect of the liquid in the strawberries. This case could be indexed in several ways: (1) dish is of type souffle, and liquids and leavening are not balanced; (2) dish is of type souffle, and dish includes strawberries; (3) dish is of type souffle, and dish includes fruit; and (4) dish is of type souffle, and dish has a lot of liquid.

The last three indexes are clearly better than the first because their features are directly recognizable without inference. The fourth is okay but probably not as good as 2 and 3 because it is hard to tell the difference between too much liquid and the right amount. Indexes 2 and 3 are the most concrete and recognizable, and of them, the third, which mentions fruit rather than strawberries, is more generally applicable.

Usefulness A final consideration in choosing indexes is the criterion of usefulness. Indexes should be chosen to make the kinds of predictions that will be useful in later reasoning. In general, any issue that came up in solving one problem could come up again in another one. All combinations of descriptors that predict how to deal with reasoning issues or predict the outcome are, thus, useful. In practice, however, a particular case-based decision aider will be responsible for aiding some subset of decisions that must be made. Guidelines for the kinds of indexes that are useful for retrieving cases to aid with different reasoning tasks are as follows:

> To use cases to help generate solutions to problems, index on combinations of descriptors responsible for the choice of a particular solution, solution component, or solution method.

For example, if a reasoner must choose a means of achieving goals, it should index cases by goal, constraint, or feature combinations that were responsible for solving a problem in a particular way.

> Cases recalled based on combinations of descriptors that were responsible for failures are useful for a number of reasoning tasks: anticipating potential problems, explaining reasoning errors and failures, and recovering from reasoning errors and failures
>
> To use cases for evaluating proposed solutions, index on combinations of case descriptors that were responsible for each case's outcome and on combinations of descriptors that describe outcomes.

Note that any case can have several indexes associated with it. Consider again the strawberry souffle example. Analyzing it again based on the criterion of usefulness, the first descriptor feature set (including "liquids and leavening are not balanced") would be a good index in a system that helps a person to determine how to recover from its failures and knows how to assign blame for failures when they occur. If the same system helps with the creation of solutions, the third descriptor set (including "dish includes fruit") is also a good index.

A Method for Selecting Indexes for Cases The method for identifying appropriate indexes for chosen cases has the following steps: First, determine what the case could be useful for. Second, determine under what circumstances the case would be useful for each of these tasks. Third, massage the circumstances to make them as recognizable and generally applicable as possible.

To illustrate, consider the following case:

> **Problem:** Twenty people were coming to dinner; it was summer, and tomatoes were in season; we wanted a vegetarian meal; and one person was allergic to milk products.
>
> **Solution:** We served tomato tart (a cheese-and-tomato pie). To accommodate the person allergic to milk, we used tofu cheese substitute instead of cheese in one of the tarts.

The first step is to determine what the case could be useful for. There are two possible uses for this case: (1) it provides guidelines for choosing a vegetarian main dish with tomatoes and (2) it provides guidelines for accommodating a person allergic to milk when a main dish with cheese is being served.

The next step is to determine under what circumstances this case would be useful for each of these purposes. For the first purpose, it would be useful in two circumstances; for the second, it would be useful in one:

> **First Purpose:** (1) Goal is to choose a main dish, dish is to be vegetarian, and dish is to include tomatoes. (2) Goal is to chose a main dish, dish is vegetarian, and time is summer.
>
> **Second Purpose:** Main dish has cheese as an ingredient, one or a few guests are allergic to milk products, and goal is to accommodate these guests.

These three circumstances provide the general framework for the indexes to this case. The next step is to massage them to make them as recognizable and generally applicable as possible. There is no massaging necessary for the first two sets of features. For the last index set, we need to change "goal is to accommodate these guests" to something more informative for a case-based reasoner. We change the descriptor to "goal is to adapt the main dish."

Choosing Cases and Choosing Indexes

The cardinal rule in choosing cases for a case-based aiding system is that cases must be chosen according to the needs of the users. That is, an analysis of the reasoning goals of system users must be done before choosing cases. This analysis is similar to a task analysis. A user might have several kinds of reasoning goals. Some systems will help with the derivation of a solution, some with critiquing, and some with both.

When a system is to help with the derivation of solutions, those reasoning subgoals that people have in deriving solutions must be discovered. Car mechanics, for example, need to come up with hypotheses about what's wrong with the car, test their hypotheses, select repairs for a car's "disorder," and carry out the repairs. Mechanics have particular hypotheses about what's wrong with cars, particular tests that are done for each, and particular repairs. A system to help a car mechanic needs cases that suggest hypotheses about what's wrong with a car under particular conditions, ways of testing hypotheses, and repairs for particular problems.

In addition, if a system is to help with the anticipation of problems before they arise, it must have cases that point out potential problems, and each case must be indexed by those features that were responsible for its failure.

Systems that help in evaluating or critiquing solutions must store solutions with both good and poor outcomes. These cases must be indexed by those features of the problem and solution that predict outcome. A system whose job is to help explain the reasons for poor solutions also needs to index these cases by descriptions of their outcomes.

In general then, cases must cover the range of problems that will come up in the course of reasoning. They should also cover the range of mistakes that are already well known. At the same time, however, system builders must remember that collecting cases is incremental. A system can start incomplete and be augmented with use. In fact, system builders should think of a training phase for case-based systems. The system is first seeded with a variety of problems, then trained with another set of problems to make sure the range of subgoals is covered. This approach results in additional cases being added to the case library. In addition, in domains where there are many unknowns, one should count on adding new cases as they are encountered in the normal course of using the case-based decision aider.

Implementations to Date

Several case-based decision-aiding systems have been built to date. All resemble the hypothetical mediation system that was previously shown. That is, the computer acts to augment human memory by retrieving cases

Scenario Situation: Soviet invasion of Europe, a U.S. Division at Fulda Gap, facing a salient (bulge) in the Soviet line, with a hill behind U.S. troops.

	Attacker	Defender
Nationality	Soviets	U.S
Troop Strength	3700	1100
Heavy Tanks	54	34
Light Tanks	30	30
Morale	tired	fresh
Initiative	-	+
Terrain	Rugged, mixed	
Mission	Seize hill	Hold territory
Method	Frontal Assault	Static defensive line

Retrieved Cases: 9 cases from WWII, all attacker wins
- In one battle, rapid assault $\longrightarrow$ major victory
- In two other battles, delaying actions $\longrightarrow$ successful second defense

Comparative Analysis: Significant factors generated by retriever from its clustering:

These factors favor Attacker Win:
Defender lacks reserves
Defender lacks depth

New Mission and Method:

	Attacker	Defender
Nationality	Soviets	U.S.
Mission	Seize hill	Delay
Method	Frontal Assault	Defend in depth

Retrieved Cases: 18 cases, all defender wins

Figure 1. A Sample Session with Battle Planner.

but takes a fairly passive role in doing this retrieval: It retrieves what it is asked to retrieve. Some systems also have analysis capabilities built in that draw generalizations based on the retrieved cases. In these systems, both the cases and the generalizations drawn from them are available to the user.

Cognitive System's Battle Planner (Goodman 1989) was the first major case-based decision-aiding system to be built. It is being used in some classes at West Point to help teach battle planning. It holds approximately 600 cases, all of them battles, primarily from World War II. The system helps users to analyze and repair their doctrine-based solutions. The user inputs a description of the battle situation and his(her) solution. Because the armed services provide doctrine about how to deal with different kinds of battle situations, a user who knows the doctrine finds it fairly easy to create these doctrine-based solutions. The problem, of course, is that the doctrine does not account for the subtle factors of a situation.

It is these subtle factors that Battle Planner helps with. The system recalls cases with similar situations and similar solutions and presents their outcomes to the user. It also attempts to analyze outcomes to provide an accounting of why, in general, the proposed type of solution succeeded or failed in these kinds of situations. The user uses the analysis, plus the individual cases, to modify the original solution and begins the process again, this time attempting to debug the new solution. The process continues until the user is satisfied with his(her) solution.

Figure 1 illustrates: The person enters the description of a battle situation and his(her) solution. Here, the mission and method fields describe the solution; the other fields describe the battle situation. The user is planning for the American (defender) side. Battle Planner retrieves nine World War II cases, all in which the attacker wins. It provides commentary about what options and variations were tried in prior cases (and their outcomes) and then supplies a comparative analysis. Armed with this information, the user reformulates his (her) solution. This time, the cases retrieved tell him(her) that the solution is satisfactory.

Another case-based decision-aiding system, LADIES (Duncan 1989), is being used by Bell Canada to aid the development of *depreciation studies* (predictions of how long an item will last). The system helps users specify factors that are important in predicting depreciation and then helps the user predict depreciation.

Many other such systems are currently under development. At Georgia Tech, one system, called ARCHIE, will help architects design a building. Another version of it will help facility managers lay out the office space and furniture for an organization. Another system under development at Georgia Tech, ED, will help elementary school science teachers plan hands-on science lessons. In addition to lesson planning, it will help teachers anticipate the questions students will ask and will help them understand the circumstances in which they should be interacting with the students during hands-on exploratory activities.

Such systems can also be used (with some additions) for training. Several systems of this sort are under development in industry and universities. Trouble shooting seems to be an area where there is particular interest in developing this kind of system.

Implications for Human Decision Making

Case-based reasoning seems to be a natural reasoning methodology in people. The thesis of the researchers at Georgia Tech is that if we view case-based reasoning as a cognitive model, it can inform the design of decision-aiding systems. In particular, because people are good at using cases but not as good at recalling the right ones, useful systems could be built that augment human memory by providing people with cases that might help them to reason but allowing all the complex reasoning and decision making to be done by the person.

There are a variety of decision-making situations in which case-based decision aids could usefully be deployed. Case-based decision aids could help with such problem-solving tasks as diagnosis, design, planning, scheduling, and explanation. Evaluation tasks that cases can help with include criticism (evaluating the goodness of a solution or interpretation), justification, interpretation (classification), and projection (given a solution, project outcome or effect when carried out).

There are, of course, advantages and disadvantages in all decision-making methods. The major disadvantage of the case-based method is that the solution space is not fully explored, and as a result, there is no guarantee of an optimal solution. In addition, it requires the collection of hundreds or thousands of concrete cases. However, there seems to be a pretty good match between what people naturally do and what case-based systems can do. The hope is that the pitfalls of the analogical reasoning that people do can be ameliorated by using these kinds of tools.

It is important to note here that the case-based reasoning approach to analogical reasoning provides some new pragmatic ways of dealing with problems that the analogical reasoning community has been grappling with for many years. One issue that has been a focus of research in analogical reasoning is judging similarity. Case-based reasoning focuses instead on usefulness. A case is useful if it can help achieve the goals of the reasoner. Similarity becomes an issue only at the point where two cases look equally useful—when this situation arises, a more similar case might win out. Rather than attempting to come up with algorithms and heuristics for judging similarity, the goal at Georgia Tech has been to come up with a way of marking cases for their usefulness. Indexes are this means. They designate which of several case descriptors are the more important ones to consider in judging how useful a case might be.

Of course, one cannot predict all the situations in which a case can be useful. When information about usefulness is unavailable, a retriever must be able to retrieve and match cases based only on similarity. The preferred and more usual mode, however, is to select cases based on usefulness, falling back on strict similarity judgments only when absolutely necessary. Methods for dynamically judging usefulness that can fall back on similarity judgments are being developed (for example, Kolodner [1989]).

Related to the issue of judging similarity is the issue of what features to use for analog retrieval. Although the analogical community

Case-based reasoning seems to be a natural reasoning methodology in people.

has been debating whether people use surface or abstract features, individual features or combinations, or descriptors or relationships, the case-based reasoning community concentrates on the content of useful indexes. Sometimes surface features are the right ones to index on, sometimes abstract features, sometimes individual features, sometimes combinations of features, sometimes relationships between features, and sometimes relationships between relationships. It depends what reasoning tasks the reasoner is responsible for and what descriptors of an old case were responsible for its solution or outcome.

This concentration on pragmatics allows us to propose systems that can help people to do a better job of analogical reasoning. Although people tend to use surface features for retrieval when they are unfamiliar with a task or domain, a case-based system that is retrieving based on pragmatics can provide the person with cases that s/he would have been unable to retrieve from his(her) own memory. This ability could be a great boon to novices. Although people tend to discount solutions that are inconsistent with what they want a solution to be, a case-based system can present cases with failed solutions, along with explanations of why they better watch out, giving them less reason to discount negative results and helping them to make use of these negative results to create informed solutions.

More concretely, there are a variety of potential benefits to using case-based aiding systems for novices, experts, and corporations. For novices, such a system can provide a range of experience they haven't had. Rather than solving problems from scratch, the wisdom of many experts is available. There are several areas where novices should be able to perform better using such a system. First, with more cases available, they will be able to recognize more situations and the solutions or evaluations that go with these cases. Second, if cases that are available include failure cases, novices will be able to benefit from the failures of others. With failed cases available and presented to the novice by the system, the novice will be able to recognize potential problems and work to avoid them. This skill is one that novices rarely have. Third, novices will have available to them the unanticipated successes—and, therefore, the tricks—of experts that they wouldn't otherwise have. Fourth, retrieved cases will allow novices to better recognize what is important in a new situation. Cases indexed by experts and retrieved on the basis of a description of a new situation will be those that experts would recall and will show the novice ways of looking at a problem that s/he might not have the expertise for without the system. Fifth, the ability to recognize what is important will allow for better critiquing of solutions and situations. Additionally, novices will have access to obscure cases that they otherwise would not be able to make use of. These obscure cases can help with any of the tasks previously listed.

Using these systems during a training period also provides students with a model of the way decision making ought to be done, for example, what things ought to be considered, and provides them with concrete examples on which to hang their more abstract knowledge. Much of the expert decision-making skill people have comes from observing experts and discussing with experts why they solved problems in certain ways. A case-based aiding system can provide at least some of this experience.

The benefits of these systems are not just for novices. In some domains, there is much to remember. For tasks where there is much to remember, case-based aiding systems can augment the memories of even expert decision makers. In addition, as previously discussed, both experts and novices tend to focus on too few possibilities when reasoning analogically or to focus on the wrong cases. Case-based aiding systems can help to alleviate these problems.

Finally, consider the potential benefits of such systems for corporations. An extension to the notion of a case-based aiding system is the notion of *corporate memory*, a means of maintaining the knowledge and wisdom of corporate employees in a corporate database.

Such systems would allow corporations to have the knowledge of its employees even after they leave the corporation, would alleviate the bottleneck that arises when one person owns the expertise that many need, and could facilitate communication between different branches of the corporation. It is this last function that I concentrate on here.

A case-based system provides the potential for feedback from one part of an organization to be considered by other parts of the organization. Such a system would work as follows: All employees of the organization working on some project would record their feedback and decisions in the system. All work on one project would be gathered into one case. The case would be indexed in the case library in ways that would aid the decision making of all employees. Each employee using the system would have available the feedback, solutions, and rationale of all the other employees working on the project. Those working on design, for example, would have available the feedback from those in manufacturing who assembled the design, those in testing who verified the artifact, those in marketing who had to sell it, and those users who used it. As the case library was expanded, feedback from different divisions of the corporation could be used to inform decision makers in another division. Designers, for example, could take manufacturability, testability, and usability into account as they created the design, resulting in better design decisions. Of course, there are many new problems that need to be addressed before such systems can be built, not the least of which is organizing and accessing the huge amounts of data within one case representation. Case-based technology provides a platform to begin thinking about such future projects.

Acknowledgments

This research was supported in part by the National Science Foundation under grant IST-8608362 and in part by the Defense Advanced Research Projects Agency under contract F49620-88-C-0058, which is monitored by the Air Force Office of Scientific Research. Thanks to Elaine Rich for helpful comments on the first draft.

Notes

1. See the article by Ashok Goel in this issue.

2. Some psychologists call it analogical reasoning (for example, Gentner 1987; Holyoak 1985; Ross 1986), and others call it comparison-based prediction (Klein 1982; Klein, Whitaker, and King 1988).

3. Readers wanting additional information about case-based reasoning should read the article by Steven Slade (1991) in the spring 1991 issue of this magazine. For more technical detail, see Riesbeck and Schank (1989) and Kolodner (1988).

4. I thank Craig Zimring for this example.

References

Alterman, R. 1988. Adaptive Planning. *Cognitive Science* 12:393–422.

Ashley, K. D. 1988. Modelling Legal Argument: Reasoning with Cases and Hypotheticals. Ph.D. diss., Dept. of Computer and Information Science, Univ. of Massachusetts at Amherst.

Ashley, K. D., and Rissland, E. L. 1987. Compare and Contrast: A Test of Expertise. In Proceedings of the Sixth National Conference on Artificial Intelligence, 273–278. Menlo Park, Calif.: American Association for Artificial Intelligence.

Bareiss, E. R. 1989. *Exemplar-Based Knowledge Acquisition: A Unified Approach to Concept Representation, Classification, and Learning*. Boston: Academic.

Barletta, R., and Hennessy, D. 1989. Case Adaptation in Autoclave Layout Design. In *Proceedings of the DARPA Workshop on Case-Based Reasoning,* volume 2, ed. K. Hammond, 203–207. San Mateo, Calif.: Morgan Kaufmann.

Duncan, N. M. 1989. Case-Based Reasoning Applied to Decision Support Systems. Master's thesis, Queen's Univ., Kingston, Ontario, Canada.

Gentner, D. 1989. Finding the Needle: Accessing and Reasoning from Prior Cases. In *Proceedings of the DARPA Workshop on Case-Based Reasoning,* volume 2, ed. K. Hammond, 137–143. San Mateo, Calif.: Morgan Kaufmann.

Gentner, D. 1987. The Mechanisms of Analogical Learning. In *Similarity and Analogical Reasoning,* eds. S. Vosniadou and A. Ortony. New York: Cambridge University Press.

Gilovich, T. 1981. Seeing the Past in the Present: The Effect of Associations to Familiar Events on Judgments and Decisions. *Journal of Personality and Social Psychology* 40(5): 797–808.

Goel, A. 1989. Integration of Case-Based Reasoning and Model-Based Reasoning for Adaptive Design Problem Solving. Ph.D. diss., Dept. of Computer and Information Science, The Ohio State Univ.

Goel, A., and Chandrasekaran, B. 1989. Use of Device Models in Adaptation of Design Cases. In *Proceedings of the DARPA Workshop on Case-Based Reasoning,* volume 2, ed. K. Hammond, 100–109. San Mateo, Calif.: Morgan Kaufmann.

Goodman, M. 1989. CBR in Battle Planning. In *Proceedings of the DARPA Workshop on Case-Based Reasoning,* volume 2, ed. K. Hammond, 246–269. San Mateo, Calif.: Morgan Kaufmann.

Guha, R. V., and Lenat, D. 1990. CYC: A Mid-Term Report. *AI Magazine* 11(3): 33–59.

Hammond, K. J. 1989. *Case-Based Planning: Viewing Planning as a Memory Task*. Boston: Academic.

Hammond, K. 1986. CHEF: A Model of Case-Based Planning. In Proceedings of the Fifth National Conference on Artificial Intelligence, 65–95. Menlo Park, Calif.: American Association for Artificial Intelligence.

Hinrichs, T. R. 1989. Strategies for Adaptation and Recovery in a Design Problem Solver. In *Proceedings of the DARPA Workshop on Case-Based Reasoning,* volume 2, ed. K. Hammond, 115–118. San Mateo, Calif.: Morgan Kaufmann.

Hinrichs, T. R. 1988. Toward an Architecture for Open-World Problem Solving. In *Proceedings of the DARPA Workshop on Case-Based Reasoning,* volume 1, ed J. Kolodner, 182–189. San Mateo, Calif.: Morgan Kaufmann.

Holyoak, K. J. 1985. The Pragmatics of Analogical Transfer. In *The Psychology of Learning and Motivation*, ed. G. Bower, 59–88. New York: Academic.

Klein, G. 1982. The Use of Comparison Cases. In IEEE Proceedings of the International Conference on Cybernetics and Society, 88–91. Washington, D.C.: IEEE Computer Society.

Klein, G., and Calderwood, R. 1988. How Do People Use Analogues to Make Decisions? In *Proceedings of the DARPA Workshop on Case-Based Reasoning,* volume 1, ed J. Kolodner, 209–223. San Mateo, Calif.: Morgan Kaufmann.

Klein, G.; Whitaker, L.; and King, J. 1988. Using Analogues to Predict and Plan. In *Proceedings of the DARPA Workshop on Case-Based Reasoning,* volume 1, ed J. Kolodner, 224–232. San Mateo, Calif.: Morgan Kaufmann.

Kolodner, J. L. 1989. Selecting the Best Case for a Case-Based Reasoner. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society,* 155–162. Hillsdale, N.J.: Lawrence Erlbaum.

Kolodner, J. L., ed. 1988. *Proceedings of the DARPA Case-Based Reasoning Workshop,* volume 1. San Mateo, Calif.: Morgan Kaufmann.

Kolodner, J. L. 1987a. Capitalizing on Failure through Case-Based Inference. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, 715–726. Hillsdale, N.J.: Lawrence Erlbaum.

Kolodner, J. L. 1987b. Extending Problem-Solving Capabilities through Case-Based Inference. In *Proceedings of the Fourth International Machine Learning Workshop*, 167–178. San Mateo, Calif.: Morgan Kaufmann.

Kolodner, J. L., and Simpson, R. L. 1989. The MEDIATOR: Analysis of an Early Case-Based Problem Solver. *Cognitive Science* 13(4): 507–549.

Kolodner, J. L.; Simpson, R. L.; and Sycara, K. 1985. A Process Model of Case-Based Reasoning in Problem Solving. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 284–290. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Koton, P. 1988. Reasoning about Evidence in Causal Explanation. In Proceedings of the Seventh National Conference on Artificial Intelligence, 256–261. Menlo Park, Calif.: American Association for Artificial Intelligence.

Lancaster, J. S., and Kolodner, J. L. 1988. Varieties of Learning from Problem-Solving Experience. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society.* Hillsdale, N.J.: Lawrence Erlbaum.

Lenat, D., and Guha, R. 1990. *Building Large Knowledge-Based Systems*. Reading, Mass.: Addison-Wesley.

Read, S., and Cesa, I. 1990. This Reminds Me of the Time When . . . : Expectation Failures in Reminding and Explanation. *Journal of Experimental Social Psychology* 26.

Redmond, M. 1989. Combining Case-Based Reasoning, Explanation-Based Learning, and Learning from Instruction. In *Proceedings of the Sixth International Workshop on Machine Learning*, ed. A. Segre. San Mateo, Calif.: Morgan Kaufmann.

Riesbeck, C., and Schank, R. 1989. *Inside Case-Based Reasoning*. Hillsdale, N.J.: Lawrence Erlbaum.

Ross, B. H. 1989. Some Psychological Results on Case-Based Reasoning. In *Proceedings of the DARPA Workshop on Case-Based Reasoning,* volume 2, ed. K. Hammond, 144–147. San Mateo, Calif.: Morgan Kaufmann.

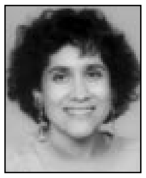
Ross, B. H. 1986. Remindings in Learning: Objects and Tools. In *Similarity and Analogical Reasoning*, eds. S. Vosniadou and A. Ortony. New York: Cambridge University Press.

Simpson, R. L. 1985. A Computer Model of Case-Based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation. Ph.D. diss., School of Information and Computer Science, Georgia Institute of Technology.

Slade, S. 1991. Case-Based Reasoning: A Research Paradigm. *AI Magazine* 12(1): 42–55.

Sycara, E. P. 1987. Resolving Adversarial Conflicts: An Approach to Integrating Case-Based and Analytic Methods. Ph.D. diss., School of Information and Computer Science, Georgia Institute of Technology.

Turner, R. M. 1989. A Schema-Based Model of Adaptive Problem Solving. Ph.D. diss., School of Information and Computer Science, Georgia Institute of Technology.



Janet L. Kolodner is a professor in the College of Computing at the Georgia Institute of Technology. She received her Ph.D. in computer science from Yale University in 1980. Her research investigates issues in learning, memory, and problem solving. As part of these investigations, she pioneered a reasoning method called case-based reasoning. Kolodner wrote the book *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model* and edited *Memory, Experience, and Reasoning. Proceedings: Case-Based Reasoning Workshop* is a collection of papers describing the state of case-based reasoning in 1988. She is currently working on a case-based reasoning textbook and has authored dozens of technical papers.