AI Magazine Volume 12 Number 3 (1991) (© AAAI)

... I focus on issues in interaction, looking at alternatives to the isolation assumptions.

AAAI-90 Presidential Address

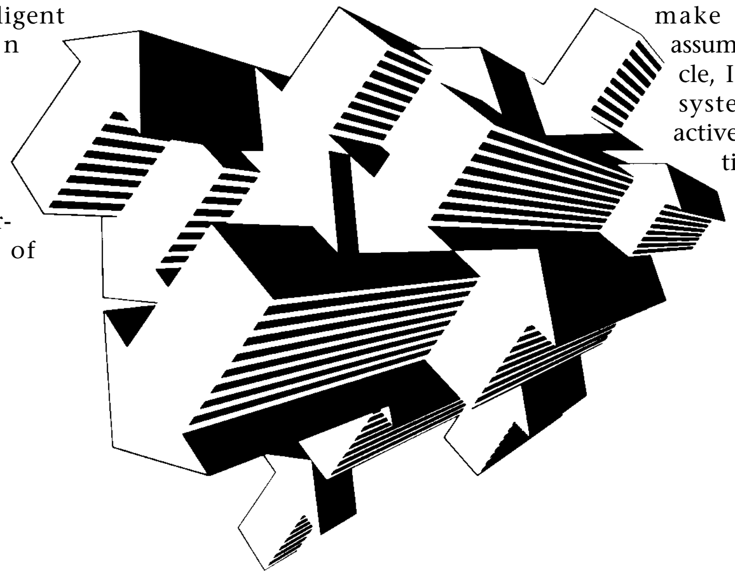
Dimensions of Interaction

Daniel G. Bobrow

Ten years ago, at the first AAAI conference, Alan Newell (1982), in his presidential address, focused on understanding the then dominant paradigm for artificial intelligence: the writing of symbolic reasoning programs for an agent that would act rationally to achieve a goal. He distinguished the "knowledge level" from the "symbol level" description of a system. The former encompasses the knowledge and conditions necessary for an agent to solve a problem. It is only when the knowledge level is reduced to a symbol level description that issues of implementation get considered.

Newell's analysis postulated a single agent, with fixed knowledge and a specified goal. The agent was disconnected from the world, with neither sensors nor effectors, and more importantly with no connection to other intelligent goal-driven agents. Research results in AI consisted primarily in the determination of the principles of construction of such intelligent, but deaf, blind, and paraplegic agents. The assumption was that a human agent would formulate the problem in a previously defined language understandable to the agent. The problem statement would include background knowledge, a description of the state of some world, operators to use in that world, and a description of a desired state (a goal). The agent would use (at the knowledge level) logical reasoning to determine a sequence of operations that could be performed to achieve the desired goal, and a more complete description of the resulting desired state.

Although this style of building AI systems is still both useful and prevalent, there has been increasing recognition of the importance of designing and analyzing systems that do not make these isolation assumptions. In this article, I want to focus on systems consisting of active agents with multiple goals, communicating among themselves, and interacting



0738-4602/91/$4.00 ©1991 AAAI

with the world. These agents can be either human or programmed machines, and can interact directly with the world. In this article, I focus on issues in interaction, looking at alternatives to the isolation assumptions. My presentation uses examples from diverse fields to illuminate issues along three dimensions of interaction: communication, coordination and integration.

The first dimension of interaction is *communication*. For communication to exist between two agents, there must be some common ground of mutual understanding. Where does this come from, and how does it develop? What techniques are used by people and systems to build and extend this base for communication? Communication between a particular pair of agents might not always be easy, or even possible. In such cases, communication can be facilitated by interposing a mediating agent.

The second dimension of interaction is *coordination*. With multiple agents with multiple active goals, progress requires agents to share resources and work toward some common goals. Various organizational structures, for example, based on markets and business hierarchies have been used in the resource-allocation process. But resources are not the only thing that must be shared. For independent agents to work together, they must be able to predict other's behavior, but not necessarily in great detail. Joint commitments to future action are a useful way of organizing this information.

The third dimension of interaction is *integration*—integration of AI systems with other programmed solutions, and integration of solutions with human work practice. Systems do not exist in isolation. Systems must solve problems that have some payoff outside the system itself. For agents that we build to be useful, they must fit in with the current work practices of both people and other computer systems. To understand how to help build such systems, it is useful to appreciate how our work relates to research efforts in such diverse fields as linguistics, psychology, computer-human interaction, ethnomethodology, and organizational dynamics. Looking outside our narrow discipline allows us to develop appropriate criteria to evaluate the power and limitations of AI, and to understand the place of AI in the broader fields of human endeavor.

This article is not intended to be a complete or scholarly overview of issues involved in each of the dimensions of interaction. I have selected some prototypical examples from current research and development, where each example is intended to provide a lesson. I want to encourage a shift of perspective that has started (Gasser 1991; Malone 1987), from systems in isolation to systems in interaction. The challenge for our field for the next decade is to build AI systems that can interact productively with each other, with humans, and with the physical world.

Communication

Communication is a critical ingredient in the interaction of agents with the world and with each other. Common ground for communication comes from common experience, leading to a common terminology for concepts. For computational systems to adapt to new communication needs, there must be mechanisms for extending terminology and to learning new ways some terminology can be used.

Some computational systems do not have the capability of improving their communication abilities. In such cases mediating agents play a crucial role. Intermediaries can provide the glue that holds systems of interacting agents together. I examine multiple roles for such mediators.

Common Ground

Communication is based on an assumption of common understanding of the language of interchange between participating agents. If something is communicated, and a receiver doesn't know the terms, or what it means to use them in certain constructions, or cannot connect them to previous experience, communication fails through lack of appropriate common ground. In traditional AI programming, meeting the requirement for common ground is usually ensured trivially by having all the computational agents use a common built-in language and vocabulary. Humans in the system are assumed to have appropriate understanding of the symbols processible by and produced by agents in the system. Some systems designers seem to believe either in their *omniscience*—they can foresee all possible uses and the background of all future interactors with the system—or their *omnipotence*—their ability to change the system to meet any contingencies. We see other examples of this omniscience/omnipotence assumption later. This subsection deals with examples of the breakdown of this assumption, and how the resulting problems are handled.

Extending Common Ground. The need for extension of common ground is often anticipated in symbolic systems. For example, in any knowledge representation system, pro-

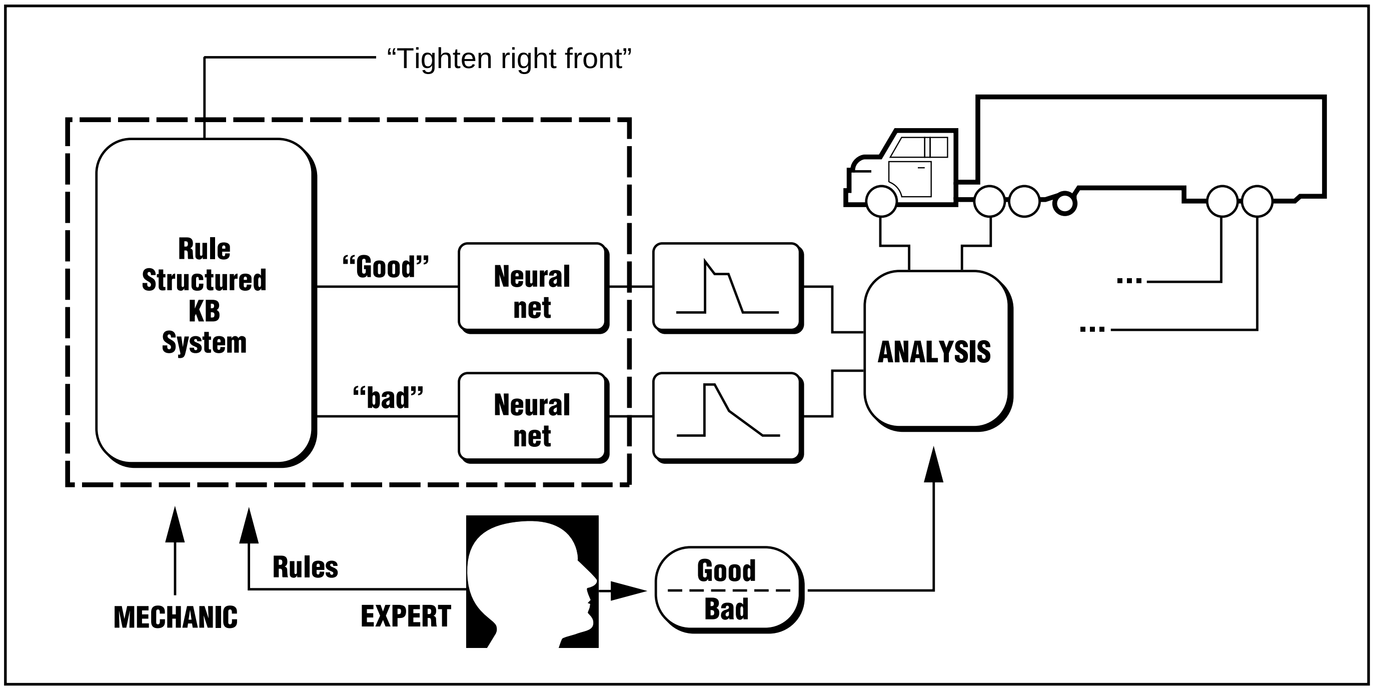


Figure 1. This Expert System Combines a Neural Network and a Rule-Based System to Support the Common Ground for Communication to the Rule-Based System.

vision is made for defining new vocabulary based on previous vocabulary. Extending vocabulary in this way is a first step in extending common ground. Another necessary part of common ground is knowledge of how a system uses the facts it has. This requires a description of system process as well as content.

Consider the problem of sharing knowledge bases for knowledge-based systems. It is very costly to build large knowledge bases. Sharing requires more than sharing terminology. To use a set of facts from another KB system, it is necessary to understand how those facts were intended to be used in a reasoning process. For example, it is useful to share the intended process of default reasoning. As a simple example, consider the problem of default reasoning. In the now classical Yale shooting problem, given the sentences "The gun is loaded" and "The gun is shot at Bill," the preferred default conclusion is "Bill is dead," rather than "The gun became unloaded." Many papers have been written about why the latter conclusion follows from some but not all theories of nonmonotonic reasoning. Communication must include descriptions of how such defaults are chosen.

Guha (1990) describes how the Cyc system describes different default reasoning processes in the language of the system. It is based on the notion of reified proofs in the logic. A *reified proof* is a representation of the proofs that makes the structure explicit and manipulable. Adding defeasible assumptions turns proofs into *arguments*. Axioms that describe argument structures and argument preferences can be represented in the logic. Reasoning at the metalevel can determine which arguments are more compelling. Such reasoning might involve the assumptions in the argument, the relative lengths of arguments, the inference rules used, or the quality of the supporting evidence.The full reasoning engine can be used to reason about the reasoning process itself. Because the axioms of the reasoning process can themselves be communicated, common ground can be extended for process as well as terminology.

Cyc is only one example of a system that extends system capabilities by using metalevel processing. The SOAR architecture (Laird, Newell, and Rosenbloom 1987) is a problem-solving system based on a problem-space model. Solving a problem is moving from an initial state representing a problem statement to a final state that meets some goal condition. Operators move the solver from one state to another. In an impasse situation, for example where there is ambiguity about which operator to apply, a new problem space is entered in which the problem to be solved is making a decision among operators.

Communication is a critical ingredient in the interaction of agents with the world and with each other.

Another example of language extension using a metalevel architecture is the Common Lisp Object System (CLOS) (Kiczales, des Rivieres, and Bobrow 1991). The interpreter for this object system is implemented in terms of objects and methods of CLOS itself. This supports introspection into the state of the object system. In addition, it allows users to experiment with different extensions to the object system itself without the danger of breaking the basic system. For example, a relatively simple and localized extension can make CLOS support persistent objects located in a database.

Lesson: System capabilities can be used for extending common ground by reifying the state and actions of the system itself.

Grounding in Experience. Expert systems are often constructed as rule-based systems that reason using information provided by a human informant. The human is expected to understand the problem as it appears in the world, and provide the system appropriate symbolic descriptions as input. The human can take the initiative in entering data that seems important, or can be requested to provide information by the rule-based system—either way acting as the system's "sensory input." But consider a problem faced by the Eaton Corporation (Smith 1991). Eaton provides a brake-balancing service for fleets of trucks. With improper balance, 18 wheelers (very large trucks) can exhibit failures as simple as excessive brake wear or as disastrous as the jack-knifing of trucks on the road.

Brake balancing is usually done by mechanics at a fleet maintenance shop. Sophisticated measurements have not been available. Eaton developed a proprietary analysis system that produces complex plots of the relationships of air pressure, temperature, braking force, and time. An expert using this system can balance the brakes much more effectively than most maintenance mechanics. The expert with a knowledge engineer was even able to program a fairly simple rule-based system for brake balancing, given the interpretations of the analyzer graphs. However, what is "obvious" to the expert in these graphs is opaque to the ordinary mechanic. Unfortunately, the time necessary to learn how to interpret the graphs is long, and requires apprenticing with the existing experts. This is an example of the breakdown of the presumption of common ground.

Eaton's solution to this problem is one that has potentially wide applicability. Eaton built a system that linked the simple rule-based system to a neural network that received the analyzer output (figure 1). An expert trained the neural network on the analyzer graphs to produce descriptions to correspond to his own. Because the data was inherently noisy and redundant, a training criterion of only 90 percent was sufficient to ensure appropriate communication. The fully trained neural net was then connected to the rule-based system. Input to the rule-based system consists of two parts: descriptions provided by the on-site mechanic, based on the usual vocabulary shared between the expert and the mechanic; and descriptions from the neural net, based on the common ground developed in the neural net through training with the expert. With this extended common ground, the combined neural net and rule-based system saves the company an estimated $100,000 per year.

Lesson: It is sometimes necessary for systems to connect directly to the world to support common ground between people; using a neural network as a front end to a symbolic system is one way to achieve this.

Negotiating Common Ground. Psychological experiments provide us with some insight into ways that humans develop and extend common ground. Clark and Wilkes-Gibbs (1986) carried out an interesting experiment in which two people were required to develop a new common vocabulary. Their task required describing pictures (geometric silhouettes) that neither had seen before. Each was given an identical deck of 12 silhouette cards in different shuffled orders. Their task was to arrange the two decks in the same order. The two people, one called the director and the other the follower, are separated by a

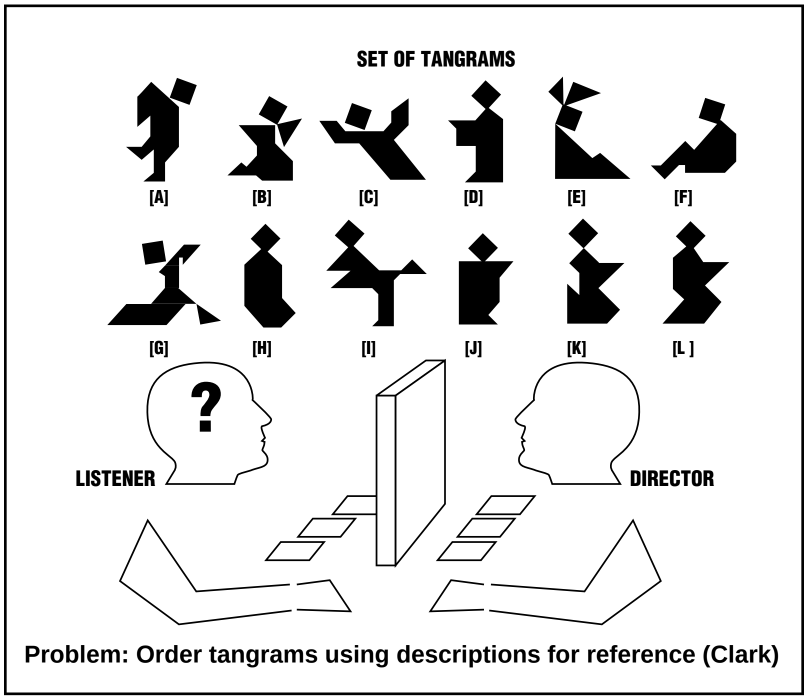


Figure 2. This Experiment Explores How People Develop and Improve Common Ground in Conversation.

screen; they cannot see each other or each other's cards (figure 2). To obtain the common arrangement, the director must communicate to the follower the order of the director's cards. In identifying cards, the director and the follower negotiate descriptions to be used to refer to the different pictures. The following dialog is adapted from one in the referenced paper:

Director:	"Do you see the one that looks like a skater?"
Follower:	"On one leg?"
Director:	"Yes."
Follower:	"Put it first, please."

Interchanges like this have a very different character than someone following simple written instructions. In the latter, a "literary mode" of communication, the author bears full responsibility for creating phrases and sentences that a reader can comprehend. The author must make assumptions about what is known previously, and ensure that references in the text are unambiguous. In contrast, in a conversation both participants bear the responsibility of assuring that what has been said has been heard and understood before the conversation goes on. Conversation consists of turn taking, where each contribution both specifies some content and grounds it. Negotiation about what is perceived can be used as the basis for reference. For example, one phrase "the one that looks like a skater" can be responded to with a question, such as "On one leg?" or an acknowledgment "OK, I got it." Later references to the same card might use different phrases. A new phrase can be a shortened form, for example, "the skater," or can refer to the conversation "the figure I called the skater last time." The subjects perform the ordering task six times, alternately playing the role of director and follower. Clark and Wilkes-Gibbs found that subjects got much more efficient in their communication over time, using fewer words for each reference, and needing much less negotiation.

Lesson: Humans use many skills in conversation that should be designed into systems with computational agents. These include sharing of communication responsibility and improvement of performance through practice.

Using Intermediaries

It is sometimes impossible for two agents to communicate together directly, in conflict with the simple assumption of designer omniscience/omnipotence. But we don't design all systems that we must interact with. Some existing systems use only one fixed language: Database servers understand SQL; robot arms understand commands to change joint positions. Even if two agents can use the same language, there can be semantic incongruence of the terms; one database on hotels might have "Price" mean before tax, and another after and with breakfast. Finally, even with common language and semantics, one system might not be able to keep up with the communication rates of another, and an intermediary might be needed for buffering.

A *mediator* is an agent who helps bridge these differences between two or more agents, like a transformer that reduces the very high voltage the power company uses to minimize transmission power loss to the lower safer voltage used in our homes. The neural network system used in the Eaton brake-balancing analyzer can be seen as a mediator between a symbolic system and the Eaton instrumentation. I look at examples of mediators that bridge a wide range of mismatches.

Supervisory Control. Sheridan (1984) of MIT used a mediator to solve a language mismatch problem that arose in getting a person to control a remote manipulator/vehicle that I call "Robby." The command language understood by Robby is very low level (change-joint-angle-by x) and unnatural for humans to generate. Robby was intended to be deployed remotely, for example, on the moon or on the sea bottom, making it impractical to use a simple direct manipulation interface based on visual and force feedback. The time delays for feedback signals were too great.

Sheridan introduced a third agent to aid in supervisory control, telling Robby what to do without telling it how (figure 3). The mediating agent sat between the human and Robby, translating goal-oriented language statements by the person (for example, move-arm-to-object or close-hand) into appropriate low-level commands. It also took local feedback from Robby and constructed a schematic display of Robby's current state. The human did the overall task of planning and monitoring. The mediator, using a rule-based system, transforms goals into plans and commands for direct action for Robby, and interprets sensory feedback from Robby as it does the actions.

Lesson: Mediating agents cannot only bridge mismatches, but also allow more effective partitioning of tasks in a human/machine system.

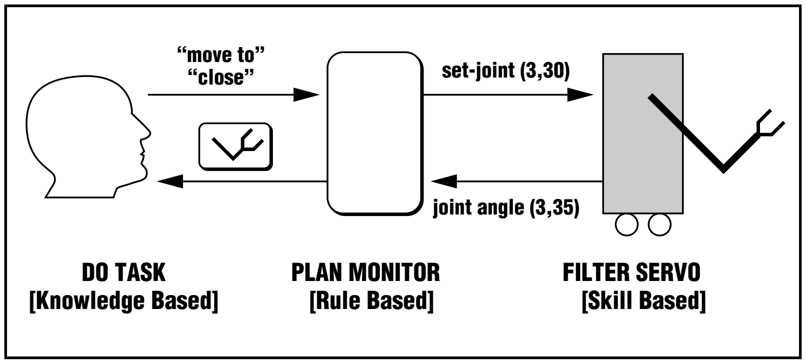


Figure 3. This Mediating Agent Supports Language Translation and Task Focusing.

Cognitive Coprocessor. Humans process different sensory inputs at different rates. We can read at 600 words per minute or more, but only process speech at best at 200 words per minute. Humans do cognitive processing slowly (seconds) compared to perceptual processing (tenths of seconds). Robertson, Card, and Mackinlay (1989) developed a general architecture for interactive user interfaces whose basic premise is that it is much easier for people to process slowly changing visual images (such as you get in a movie) than to make cognitive shifts. They use a mediator, which they call a cognitive coprocessor, to bridge mismatches between humans and machine agents with respect to processing rates, immediate memory capabilities, and representational forms.

Their application domain is what they call *information visualization*. It uses two- and three-dimensional animated objects to represent both structural relationships and the content of an artificial environment (for example, a file system or a database), and a set of artificial active agents that can be sent to retrieve information from the different stores. For example, consider the problem of understanding where certain people have their offices. Pictures of the people are retrieved from an image store, organizational information from another database, plans of the building from a third, and office assignments of people from yet another (figure 4). The cognitive coprocessor constructs a composite picture, with portraits of the individuals, a three-dimensional image of the building, and lines leading from the labeled pictures to the offices. Controls in the interface let you look at the building image from different positions, with different colorings and labelings, giving both overall views and detailed images in context to connect a person with a location. This eliminates time-consuming cognitive steps, such as finding a person's room number, using it as a key to search a map, and then understanding where it is in the context of the actual building structure. The cognitive coprocessor also uses visual animation to support easier changes of focus in an artificial world; moving to viewing another part of an organizational structure made by a smooth rotation of nested cones.

Lesson: Three-dimensional visual processing, which most people are very good at, can sometimes be used to replace cognitive processing, which seems harder and is more error prone.

Distributed Knowledge Base Systems. Gio Wiederhold (1991) proposes mediators as a crucial part of the architecture of future distributed information systems. He foresees many databases on a network distributed across the country and around the world. Each database builds in certain assumptions about the meanings of terms and the expected uses of the data. Mediators are explicit active modules between the user's application and the data resources. Because of the distributed nature of the system, some mediators are used to bring together data from multiple sources; some locate desired data; others bridge semantic mismatches between needs and stored information. Wiederhold provides an interesting classification of mismatches that can benefit from mediator intervention.

A reference mismatch occurs when the key to access a database is not the one naturally

A mediator is an agent who helps bridge these differences between two or more agents…

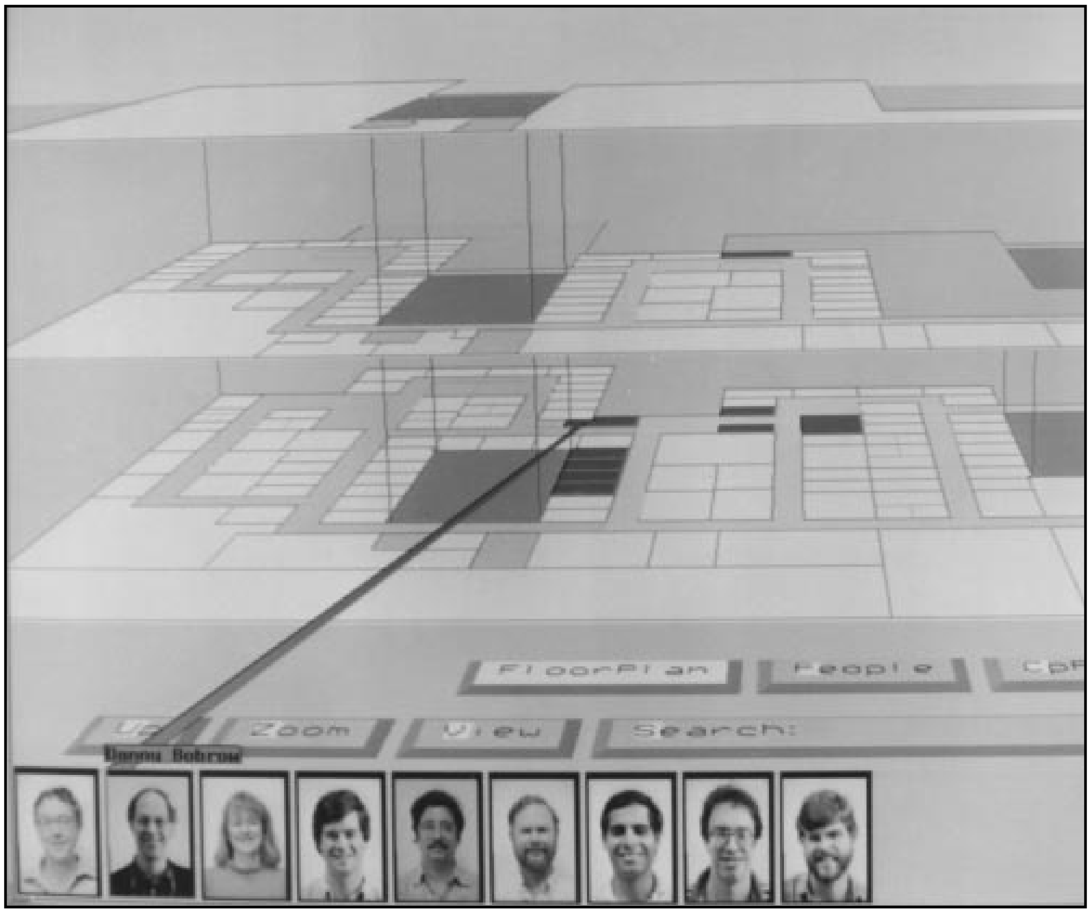


Figure 4. This Three-Dimensional View Allows Perceptual Processing to Replace More Difficult Cognitive Processing.

available to the application. For example, the application might have only the name of a person, and another database might use that person's social security number as a key. *Scope mismatch* is exemplified by a product sales database organized by sales person, when for a specified task what is wanted are sales organized by product types. A *temporal mismatch* is where data is stored by calendar year rather than financial year. It is an *abstraction mismatch* when income is associated only with an individual rather than being family based. A *domain semantics mismatch* occurs where a database is organized by postal codes, when a task requires sorting by town (there can be several towns in one postal code zone, and several postal codes in a single town).

Even when mediators cannot do appropriate representation transformation, they might be able to flag mismatches. For example, consider the interpretation of the phrase "excessive pay." For a personnel organization evaluating performance this means one thing, and it means something quite different for the Internal Revenue Service, which does not allow untaxed payments of expenses to exceed a certain amount. Wiederhold calls this a problem of context-dependent value semantics.

Lesson: Mediators can play a key role in distributed information systems by providing task-dependent representation transformations, and ensuring that assumptions on the use of data are met or flagged.

Real-Time Mediator. The Consolidated Edison Company of New York operates an electric utility system that serves a 593 square mile area. The operations control center logs about 150–200 events a minute when everything is going well. When there are problems, this can increase to roughly 6000 alarms per minute. Human operators must process these incoming signals and decide what to do. The problem is compounded by there being redundant alarms, alarms of different priorities, and alarms in 35 different categories; and different operators specialize in handling particular types of alarms. Critical data can be lost in the flurry of alarms.

A real-time expert system, SAA (Silverman 1991), was installed in the center. SAA filters out nuisance events (for example, multiple reports or false togglings due to telephone line noise). It also analyzes the system state to determine which components are failing. SAA integrates information from three models of the system (physical, functional, and temporal) to do the analysis and provide recommended operator actions using a rule-based system based on the ConEdison operating procedures. Use of this real-time system mediating between incoming alarms and human operators has provided significant benefits to ConEdison's operators: The sheer number of alarms seen in an emergency situation has been reduced by a factor of 10; this has increased the operators' confidence that they know what is happening and what might be useful to do. It has also helped in the standardization of the responses.

Lesson: Real-time mediation can transform a flood of data into a stream of information more useful to human agents.

Coordination

The computational world of the near future will contain multiple computational agents. An agent might be tied to a particular task, or might have several active goals. A computational agent might move from one machine to another to perform a job and/or different machines might be assigned as a resource for different agents. To coordinate a set of agents that share common ground for communication still requires policies and procedures for resource management and goal coordination. Resource management requires making effective use of a limited set of resources; coordinating goals among agents requires that agents have knowledge of what others are committed to do in the future.

Resource Management

The simplest model for resource management is the central planning model that has been used in most AI systems. This is another version of the omniscience/omnipotence assumption in system design. The design postulates that a problem-solving system can be based on a controller that creates agents directed toward specific subproblems. This controller provides each agent with appropriate resources. Conflicts among agents are settled by the central planner. In both world politics and large-scale computational systems this assumption has a tendency to break down. As communication costs become dominant, the central planner becomes an information and action bottleneck. The system fails if it does not remain connected with the controller in constant communication with all the subagents. Recently, there has been work on developing other models for resource allocation that seem to be more robust. One is based on a market model and the other on a hierarchical organizational model.

Resources are more than just the availability of computational cycles. Cycles can be used more effectively using knowledge of the world to reduce the computational load. Animate vision systems, for example, can be significantly simpler if they can move in the world instead of just observing it from a fixed position. Another mechanism for making effective use of resources is specialization of function to adapt to changing demands of the world. Adaptive specialization is a feature of both natural and computational systems.

A Market Model. In a market system, the value of a resource is dependent on the demand for that resource. There is no centralized control. Sellers hawk their wares to the highest bidder, and buyers make bids based on the worth of the wares to them and their current resources. This model has been adapted for controlling computational resources. SPAWN (Waldspurger et al. 1989) is a computational control system that uses this "microeconomic" approach to resource management. The sellers are users who wish to sell otherwise unused processing time on their workstations on a computer network. The buyers are users who wish to purchase computational cycles to accomplish some task.

Whenever a computer is going to have some "free" time, it broadcasts this fact to all potential buyers. An auction is held for the next available time slice. Communication costs need to be minimized. Therefore SPAWN does not use a multi-round auction where all bids are heard and can be responded to by all potential buyers. Instead, SPAWN uses a "sealed bid second-price" (Dutch) auction in which no competitor can see a bid, and the price paid by the highest bidder is that bid by the second highest bidder. This bidding mechanism has been shown to get approximately the same prices at much lower communication cost. Note also that if there is no competition for a resource, the resource is "free." Experimental results show this mechanism leads to excellent use of available resources,

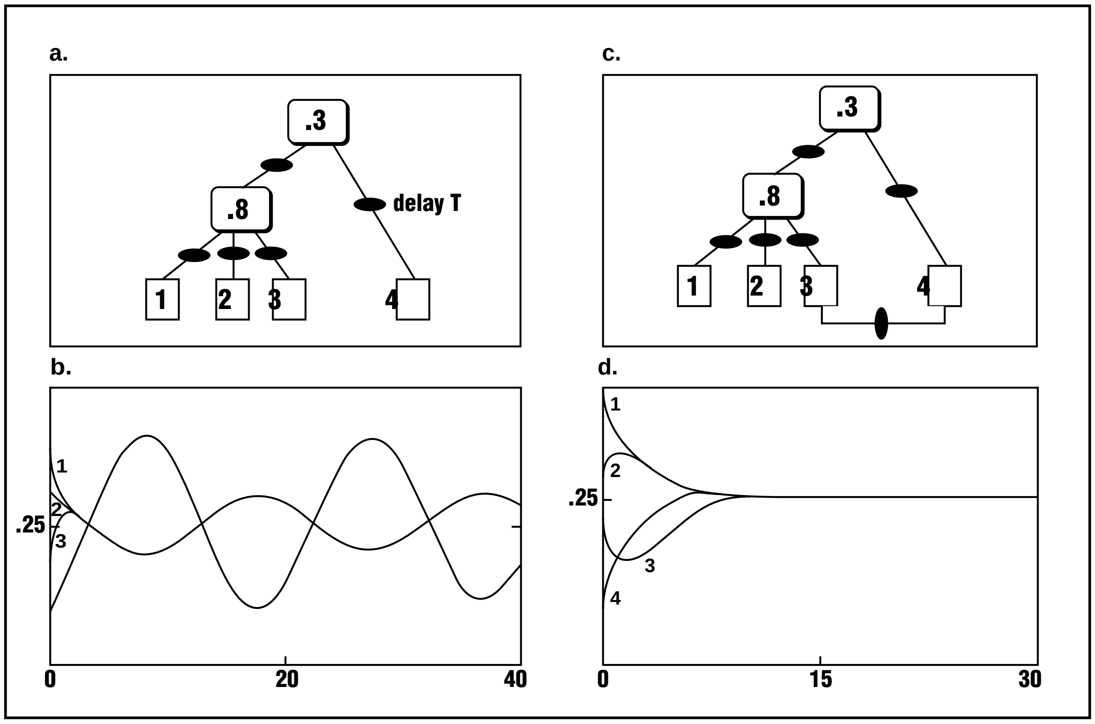


Figure 5. Hierarchical Processing with Delays Can Be Improved by the Addition of Informal Links.
(a) Simple hierarchical communication model with delays. (b) Load distribution over time in the simple hierarchical model. (c) Addition of an informal link to the hierarchical model. (d) Stability of load balancing with informal link.

with natural controls for supporting high-priority tasks and heterogenous computer resources. It also provides guidance for individual task managers about whether it pays to spawn parallel processes or focus their resources because extra computational cycles are expensive at the moment.

Lesson: Centralized control is not necessary to ensure good use of resources; a market-based system using a broadcast medium works well for resource distribution in small- to medium-size groups.

Hierarchical Communication Model. The advantage of using a completely open market approach is that there is no centralized control; buyers and sellers can appear and disappear; and there is efficient use of resources. However, even with the Dutch auction described above, communication costs and interference grow proportionally to the number of potential buyers and sellers, and delays in information propagation can destabilize the system. Experiments have shown that fruitful use of resources grows linearly as the number of buyers and sellers increases, until it reaches a certain threshold. At this boundary (roughly 20 processors in the SPAWN experiments) effective utilization starts to fall off dramatically because processors are spending so much time listening or responding to market messages from the larger number of participants.

Lumer and Huberman (1990) show that a hierarchical communication model has much better utilization characteristics (figure 5). In this hierarchical system, a processor communicates frequently to a small peer group. For a wider group, the communication rate is reduced by a significant factor, and by another reduction for the next outer group. This is comparable to a hierarchical business structure in which people talk mostly to their co-workers and their boss. The boss passes on some small fraction of what they say, and reports a small fraction of what comes from the boss's peers and boss. With this communication model, balancing of resources can be done for a much larger organization, though with significantly longer time delay for stabilization and load balancing when there are radical changes in processing demands or processor availability.

Suppose the boss introduces significant delays in the communications. This can introduce serious oscillations in resource utilization when the delays exceed a critical value. An interesting finding in these experiments was that these oscillations in resource use can be stabilized by the establishment of a few informal links in the organization, that is, communication links across disparate parts of the organizational hierarchy. Lumer and Huberman (1990) draw parallels between these results and social organizations, indicating the trade-offs between reducing the information-processing burden through the installation of hierarchy and an increase in performance by the use of direct lateral connections between participants of interdependent subsystems. Of course, adding all possible lateral links brings the system back to the full market organization and its problems of scale.

Lesson: Hierarchical communication structures reduce communication load for a market-based resource-allocation model. A few extra non-hierarchical links reduce instabilities and increase the system's ability to rebalance loads quickly under changing demand.

The World as a Resource. Reasoning about the world is a difficult task, but it can be made close to impossible by insisting on the assumption that a system have a complete model of the world to get the job done. If the world is reasonably stable, just using this stability without representing it can make some reasoning tasks much easier. For example, in remembering a path one takes through the woods, one need not remember the shape of every tree or bush. One need only remember critical features at choice points. Manipulations of the world are also an alternative to massive computation. If there is no easily distinguishing feature at a choice point, one can change the world (break a twig) to ease later decision processing.

Integrating a model of how one can and will interact with the world can lead to more efficient processing. When we put a paper down on the desk, we never remember its exact coordinates. Based on knowledge of our own perceptual abilities, we believe that we will be able to locate the article later, given a rough description of where it is and what it looks like. Of course, there are things that can happen in the world that can make this untrue (as happens too often for papers on my desk). But, by and large, it is an effective way of dealing with what would otherwise be an impossible representational task.

The ability to be active and mobile in the world is an important resource, particularly for perceptual processing. Ballard (1991) demonstrates this in his analysis for what he calls an animate vision system. Ballard rejects the idea of a vision system as a passive device whose job is to construct a detailed representation of a scene from a position. Rather, the animate vision system has positional and gaze control mechanisms that support active seeking of information. By moving the point of vision while staying focused on a particular spot, it becomes much easier to isolate a "picture" of an individual object. All and just those spots that don't have relative motion with respect to the point of focus are part of the same object (roughly speaking). In this way, a high virtual resolution can be achieved with a low-resolution device if the system can control the direction of gaze.

The simplest model for resource management is the central planning model…

Lesson: Knowledge about and the ability to interact with the world can be traded off against computational resources.

Adaptive Agent Specialization. Systems are often made up of agents with different capabilities. A crucial issue in making good use of resources is the adaptability of the mix of agent types. Nature makes good use of this. For example, in a honey bee hive, a variety of worker bees serve in specific roles (Wilson 1971). Some forage for nectar, some unload the nectar from the foragers, some build the hive, some take out the garbage, and so on. Although it looks like a genetic adaptation with fixed numbers of bees in different genetic castes, this is not the case. When a queen bee goes off to start a new hive with a number of workers, all the workers do all the tasks. It is only gradually that workers take fixed roles. As the season changes, and foragers have to go farther afield to find flowers, they take longer to get back. If an unloader has to wait too long for a forager to return, the unloader shifts task and becomes a forager, thereby shifting the balance between foragers and unloaders. Several similar simple adaptive mechanisms help to maintain the overall worker balance.

A similar adaptation was a characteristic of the Enterprise system (Malone et al. 1988). Enterprise is a system similar to SPAWN. However, sellers of services rather than buyers did the bidding. A Lisp processor needing a service, such as compiling or printing a file,

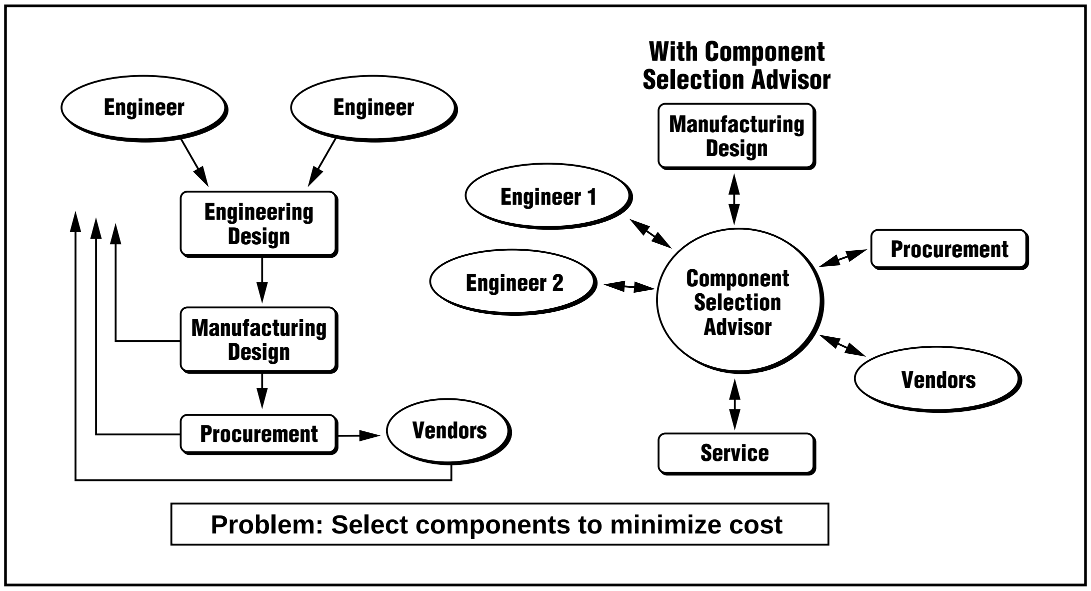


Figure 6. Design and Manufacturing Cost Modeler (DMCM).
A sequential design process is often used. DMCM supports communication, breaking the sequential nature of design and development.

broadcast this need. Servers on the network bid to fulfill the service requests. There were different speed servers, and communication could take different amounts of time depending on network connections. A bid consisted of an expected time till completion of the requested job. A service requester waited to get a set of bids and then chose the server promising earliest completion. Queue balancing was achieved without servers having to communicate because servers with full queues would not bid. In addition, there was an increased ability for a server to do a job fast if it had already been doing it. Software to do a task was loaded on demand (sometimes from files, sometimes just by paging in the working set); so having done a task recently reduced the overhead for doing the same kind of task again soon. Some processors became specialized in printing after they had all the fonts loaded, and so on. And like the bees, when the environment changed, the balance of processors tuned to particular tasks could shift. Adaptation is a crucial issue in maintaining good resource use.

As the Internet evolves from a communication resource to a knowledge resource, with many useful agents available for certain tasks, it will probably start to look like a knowledge market (Stefik 1988). Integrators will put together services composed of agents found on the net, and offer these combinations for sale. Stefik points out that a useful addition to the network as knowledge medium will be agents specializing in *metaservices*. By "meta" here, I mean those that deal with the finding and manipulating of knowledge agents rather than providing knowledge for some task in the world. For example, one might find certifiers and testers of knowledge services (the National Bureau of Standards or the *Consumer Reports* of the knowledge market). It will be necessary to have agents that fill a role played by current publishers: filtering, formatting, and distributing knowledge. There will be agents that serve as contract negotiators and bankers will deal with issues of making appropriate payments for use of individual packages. These metaresources amplify the effects of being in a highly connected, knowledge-rich world.

Lesson: Specialization of agents, by design or adaptation, can increase the effective use of resources.

Goal Coordination

A simple computational model of multi-agent problem solving postulates a single agent in control, directed toward a single goal. This central planner creates or initializes other agents, and gives them their individual subtasks and evaluation criteria for determining successful completion. Agents work on their assigned subtask until they are finished (they might fail); agents are then reassigned by the central planner. If we think of a business organization as the paradigmatic multi-agent system, we see this simple model is violated in a number of ways. First, each agent has multiple goals and will work to satisfy a number of them simultaneously. Some standing goals are related to the agent's role in the organization, some to his(her) nature as a human being, and others to particular task assignments. Second, varying goal set and agent types can lead to different trade-offs in performing a task and determining its successful completion. Finally, there is no necessary guarantee that an agent will work directly and completely on one given task.

For a system of agents of this type to achieve a common goal, agreement on future action must be reached. This requires communication among the participants to consider trade-offs that span multiple agents and their goals, and developing ongoing joint commitments to action. Knowledge-based systems can facilitate this person-to-person communication.

Component Adviser. In Xerox, like many companies, the traditional process of design starts with a specification given to engineers. Engineers then try to design a device with the specified functionality and with minimal cost. But engineers often take a very narrow view of relevant costs, usually restricting themselves to the unit manufacturing cost of the device. Every penny counts. After the engineers complete their design, it is evaluated and possibly modified by manufacturing engineers who want to minimize manufacturing costs, especially tool setup costs, which the design engineer usually hasn't considered (figure 6). Finally the design can end up being modified in the field when service problems arise, although this should be avoided because of the high cost. This waterfall model of design (one stage following only after all considerations from a previous stage have been accounted for) leads to long time-to-market because of the serial dependencies. It also often results in overly expensive systems cost if one counts both manufacturing and service costs.

To improve the design process within Xerox, the Knowledge Based System Competency Center (KBSCC) has built a system to help coordinate many different trade-offs in the early stages of copier part design (Hatfield and Crowfoot 1990). The Design and Manufacturing Cost Modeler (DMCM) is a rule-based system linked to company databases that describe standard parts, material costs, and so on. The system is usable by an engineer as soon as a preliminary design is completed. The engineer wants to use DMCM because it does automatic cost estimating of a part. Cost estimating can be painful because it is both time consuming and detailed, and might require not easily accessible current knowledge of Xerox processes and material procurements.

DMCM supports communication of information from one part of the organization to another and makes the design engineer's job easier. It allows an engineer to make cost comparisons of two partially completed designs. It provides advice to make a part more manufacturable, allowing the engineer to avoid the delays inherent in getting a finished design back for redesign because it cannot be manufactured easily. Further, DMCM can estimate service costs for particular purchased parts (for example, a 24 volt motor versus an 18 volt motor), feeding back analyses to the engineer, and feeding forward advance information to the purchasing department for long lead-time items.

Lesson: A system representing goals and trade-offs of different kinds of agents can be an effective tool for communication and coordination.

Conversation-Based Coordination. The Coordinator (Winograd 1988) illustrates how even a partial formalization of knowledge of human processes can be helpful in coordination; based on the premise that people act through language, it uses a formalization of speech acts to structure communication. The Coordinator is an augmentation of a standard electronic mail system. Electronic messages in a business setting can be categorized as moves in a conversation. For example, one message can be a request, and another commits a person to a future action. Most speech acts can be thought of as a kind of commitment: An *assertion* commits a person to backing the truth of what is said; a *promise* commits a person to some future course of action. By viewing messages as moves in a conversation, one can put constraints on responses. Certain moves set up the expectation for others in return. For example, a request is usually followed by a promise, a refusal, or a counter offer.

AI… must connect with the real tasks that give rise to the intellectual puzzles on which we spend much of our time.

The Coordinator is a semiformal system—one that has a formal representation of only part of the information available in the electronic mail. The Coordinator knows about the structure of conversations, and the categories of speech acts. Dates for promises, the identities of the senders and recipients of messages, and the state of each conversation are understood and manipulated. The English text contained in the body of the message is interpreted only by the human users. The Coordinator organizes and reports the progress of conversations in which a user is involved; it highlights such things as requests that have not been responded to, commitments the user has that are pending, and due dates for each commitment.

In addition to such conversations for action, Winograd's language/action perspective distinguishes three other kinds of conversations. Characterized by different moods and purposes of the participants, there are conversations for orientation (creating shared background), clarification (checking background assumptions), and the exploration of possibilities (speculating on alternatives). Each is supported by a formalized theory of the moves in the language game. Partial formalizations such as these, combined with appropriate viewing tools, can support work groups in coordinating their multiple goals and activities.

Missing is a formal model of what it means to have a commitment. Cohen and Levesque (Cohen and Levesque 1990; Levesque, Cohen, and Nunes 1990) have studied the commonsense relationships that exist among terms such as belief, commitment, goal, plan, and action. They developed a formalization that captures many of our intuitions about these terms and how they relate to each other. Although not yet used as the basis for a cooperative system, their analysis is an important step in understanding the requirements for such a system, and understanding the relation of communication requirements and joint action.

Lesson: Formalization need not be complete to be useful in aiding human coordination.

Integration with Work Practice

If we look at the papers in AI conferences and journals, very few make explicit reference to real-world problems. For AI to be successful as a science and in engineering, it must connect with the real tasks that give rise to the intellectual puzzles on which we spend much of our time. When we ignore this grounding, we construct for ourselves two traps: the *NP-hard trap*—the need to do well on problems that never arise in practice—and the *AI-hard trap*—the need to solve all problems from scratch using the tools and techniques from artificial intelligence programming practice.

NP-Hard Trap

A common and often useful technique in science is to find an abstraction of a problem, and work to find a general solution to that class of problems. The problem-space difficulty is often thought to be characterized by the complexity of the hardest problem in the space. Sometimes people realize that this might be an untypical measure, and move to the average time to solve problems over the entire space. Both of these can be inappropriate measures though.

Let us look at a simple example where the fallacy of the worst case can be easily seen. Consider the problem of factoring a number. To guarantee that you can find a factor for any number is thought to be NP-hard, and this difficulty is the basis for the use of this problem in various cryptographic techniques. But consider the problem in practice. Half of the numbers you might pick at random are divisible by 2; a third by 3; and so on. In fact almost all numbers are easily factorable. Thus in ordinary practice, finding a factor for a number is easy, and the only hard problem is finding a number for which simple algorithms don't work. Worrying about the worst-case analysis can sometimes lead to pretty mathematics, but poor science and engineering.

But what about using average case results. Suppose N is the measure of problem size. Suppose a small fraction of the problems of any size are really hard, with difficulty growing exponentially, say as 2^N. But suppose the fraction of hard problems goes down with the size of the space—say only $1/N$ of them are difficult. Suppose the remaining problems (most of them) can be solved in some constant time. The average time for solution for this "almost always easy" problem is dominated by $2^N/N$, which is exponential. On the other hand, if you asked for the proportion of problems solved in a time proportional to N, it would be getting closer and closer to 1. So worst-case and average times are not necessarily good measures of how well algorithms can do. Unfortunately, almost all the analyses in AI papers are of this form.

AI-Hard Trap

A great deal of interesting software technology has been driven by needs of the AI community. Automatic storage management systems, symbolic computation, backtracking and dependency-directed backtracking, and reflective systems and even time sharing were inventions of AI researchers. AI languages are very well developed; Common Lisp is one of the world's great program libraries. This success as we see it has sometimes led to ignoring systems that can be made up of pieces built in different languages and in different styles. Simplifications are often ignored, especially when choosing the right part of the problem for the machine, and the right part for a person. Given that tasks are part of a wider human context, it is useful to remember that we need not live by AI alone.

… it is useful to remember that we need not live by AI alone.

Programming System Integration

An interesting exception to the rule that AI systems are written completely in an AI programming language is the Ford Motor Company ESCAPE system (Di Santas and Smith 1990). A central computer is connected to all the Ford dealers within the United States and Canada. Each night a large Cobol program collects information and requests from all these dealers. Some requests are for refunds for services performed under warranty. Warranty rules are complicated, particularly over the broad range of vehicles under consideration, and change over time. Whenever a change was made in the set of warranty rules, it was necessary to implement the change as a modification to the Cobol program that checked compliance of requests with the rules. This led to a couple of serious problems. First, the translation of a rule to a Cobol program to check that rule took significant time to get right. So the system's rules were usually not up to date. Second, there was no way for the warranty administrators to check whether their intentions were accurately translated because they did not read Cobol. This sometimes led to disagreements with dealers when there was a discrepancy between what was implemented and the rules the dealers were told.

The solution adopted by Ford was to build a rule-based program to check the warranties' regulations. However, rather than having to replace the entire Cobol program, this module, written in ART-IM, was embedded in the Cobol program. By defining suitable interfaces, this standard expert system technology could be used in the context of the working Cobol program that embodied knowledge about communication, data storage and retrieval, and so on. The results were remarkable from Ford's point of view. Although there was a small performance penalty for using the rule-based system, the time for updating shrank by an order of magnitude, and their confidence in having the correct translation rose dramatically. They reported that the Warranty Administrator now was willing to participate in the checking and formalization of the rules as written in the rule language.

Lesson: AI technology can provide a bridge between programming and standard work practice, and play a key role in the overall system architecture.

Business Practice Integration

A somewhat different notion of integration of AI into work practice has been adopted by the Mrs. Fields Cookie Company. Founded in the heart of Silicon Valley, Mrs. Fields has been a user of sophisticated computing from the opening of their first store. A repertoire of computer-based tools ranging from electronic mail to online databases is provided to their more than 200 franchises. Some tools use AI technology, and integrate information from a number of sources. For example, there is a rule-based adviser for planning the amount of cookies to be baked tomorrow. It can use sales information from the database for last year at this time, for the past days and weeks, as well as local information such as the current weather, holidays, and scheduled special events.

A real-time adviser using sales information makes recommendations about whether to give away free samples. It considers how many cookies of what kinds have sold, how long it will be before the next batch will be out, and how long different cookies have been around; Mrs. Fields cookies are always served fresh—within two hours or so after baking. Another program provides guidance for office managers in interviewing potential employees, suggesting questions to ask and evaluation criteria. All these expert systems programs can operate as a "glass box" rather than just giving directions about what to do. This means that as people use the programs, they can "see" what is going on inside if they wish, and hence learn why things are done instead of just what to do.

Lesson: AI technology helps integrate traditional programs with expert advisers. These can improve people's skills and adaptability in the context of the company work practice.

Information Integration

Information is only useful when presented at the right time. The JANUS etc. system (Fischer, Lemke, and McCall 1990) is an interesting integration of two systems that didn't work as well as expected separately, but work much better together. Both systems are intended as models for teaching and information-providing services. Their example domain is designed to teach people about architectural design principles for kitchens.

The JANUS-VIEWPOINTS system provided a hypertext-based system, allowing a user to explore the space of alternatives and their trade-offs in kitchen design. Users participating in an experiment took guided tours of the hypertext, or did free exploration that allowed them to focus on areas of particular personal interest. After they felt comfortable, they were asked to design a kitchen for themselves, using a computer-based construction kit that simplified making the drawings for the design. There was a palette with drawings of tables, stoves, refrigerators, and so on. These could be picked up and placed in the diagram. The construction kit program was on the same computer as the hypertext and could be easily consulted. However, when doing the design, subjects seldom used the hypertext to check what they had learned (and hence often made suboptimal decisions). When asked, they said it took too long to find what they wanted in the hypertext, and they were unsure of where to look or what issues were at stake.

As an alternative approach, Fischer and his colleagues built an expert adviser that would monitor work with the construction kit. Any time the user added or moved something in the kitchen, the JANUS-CRACK rule-based critics evaluated the changed situation. When it detected a problem, a critic immediately pointed this out to the user with a brief comment. For example, if the stove were put right next to a door, the critic might comment: "this stove placement could be a fire and burn hazard to passers-by (such as small children)." Users had the option of responding to the advice by changing the design. In changing their design in response to the comment, users might make another mistake. For example, they might place the stove too far away, hence making the work triangle larger than the recommended size. In this experiment, the users learned only from trial and error, and had nothing but the pointed comments of the critic to guide them; hence, it was quite understandable that they might make a series of mistakes placing the same appliance.

When CRACK and VIEWPOINTS were connected, users exhibited interestingly different behavior. When a critic commented on a user action, a user could click in the visual interface to get directly to the related argument structures in the VIEWPOINT hypertext. They did this often and at that point they were more disposed to read about trade-offs and heuristics for the appliance they were working with. They learned more than what their mistake was, and produced better resulting designs. This program illustrates providing appropriately contextualized information for proper learning.

Lesson: Integrating action and information worlds makes learning more effective. AI technology can provide the bridge between these.

Multidiscipline/Work Practice Integration

As a final example, I want to look at a system that is putting it all together. LMS (Sullivan, Fordyce, and LMS Team 1989) is a real-time, transaction-based, integrated knowledge-based decision support system. It is installed in an IBM semiconductor manufacturing facility where it is helping to improve the tool utilization and cycle time in the plant by supporting short-term and tactical decision making. Before the system was put in place there was a complex of other systems: Some of these were ultrareliable data systems that recorded transactions on particular machines; others were paper-based systems that contained information such as process specifications, machine and operator evaluations, and lot locations. Each individual system was state of the art at the time it was installed.

The challenge for AI for the decade is to deal with the issues of interaction.

Because they were purchased separately, there was independent ownership by different groups. These factors presented several social and technical barriers to linking up the different systems. The total system was running around 240,000 transactions a day. Without the tools to integrate this flow, the IBM MIS group felt that they were "data rich and information poor."

IBM did the implementation of the integrated system in stages. First, they made real-time connections to the existing technology, and provided various users and managers personalized views and selections from the ongoing stream. Then they empowered them to make online decisions. With buy-in from these users, rules for proactive interventions by a knowledge-based system could be added to the transaction recording and viewing system. Scheduling, decision making, and reporting tools employed AI technology, but only as one of a wide variety of resources, both computational and human. The end results were quite astonishing. Not only did IBM save lots of money with improved throughput at the plant, they improved their human resources as well. "We started with some knowledge engineers who in the course of this work became expert in our systems operations; we also have expert users who now do their own knowledge engineering. There is no single expert who can do what is done by LMS."

Lesson: AI can be the glue to empower people close to the work to make good real-time decisions. Systems must embrace many technologies, and be brought in with a sense of ownership at all levels of the work force.

The Challenge

The challenge for AI for the decade is to deal with the issues of interaction. The interaction between agents requires establishing, maintaining, and extending common ground. Common ground between human and computational systems can be aided by mediators that link to the capabilities and natural properties of task agents, people, and the world we live in. We need systems that are open and can grow; that allocate resources fairly to get jobs done without requiring either centralized planning or excessive communication. With agents with multiple goals, coordination of actions must proceed through mutual commitments. Finally, we must realize that not all problems of interest are AI problems, nor are results relevant to our research always to be found just in journals of our fields.

In working with intelligent agents, it is useful to explore what is happening in such fields as cognitive psychology, anthropology, economics, neural modeling, and in the rest of computer science. The natural tendency of any science is to isolate itself as a community, and develop its own practices. But AI must cross the boundaries; to be intelligent is to be intelligent about something; success must come in the context of tasks from outside the field. People who are part of the systems must be moved as high as possible in the intellectual food chain, so that participation in the system helps them to be better in ways that they care about. We must not just look to build intelligent systems to solve specific problems; we need to learn how to help corporations and nations build intelligent organizations that support human values, and enhance the world we live in.

Acknowledgments

This article and the lecture it was based on greatly benefited from suggestions by Johan de Kleer, Jim des Rivieres, Brian Falkenhainer, Olivier Raiman, Mark Shirley, Mark Stefik, and Brian Williams.

References

Ballard, D. 1991. Animate Vision. *Artificial Intelligence* 48(1): 57–86.

Clark, H., and Wilkes-Gibbs, D. 1986. Referring as a Collaborative Process. *Cognition* 22:1–39.

Cohen, P., and Levesque, H.1990. Intention Is Choice with Commitment. *Artificial Intelligence* 42(3): 213–262.

Di Santas, J, and Smith, R. 1990. The Ford ESCAPE system for Claims Authorization Processing. In *Innovative Applications of Artificial Intelligence 1,* eds. A. Rappaport and R. Smith, 203–216. Menlo Park, Calif.: AAAI Press.

Fischer, G.; Lemke, A.; and McCall, R. 1990. Towards a System Architecture Supporting Contextualized Learning. In Proceedings of the Eighth National Conference on Artificial Intelligence, 420–425. Menlo Park, Calif.: American Association for Artificial Intelligence.

Gasser, L. 1991. Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems Semantics. *Artificial Intelligence* 47(1–3): 107–138.

Guha, R. 1990. The Representation of Defaults in Cyc. In Proceedings of the Eighth National Conference on Artificial Intelligence, 608–614. Menlo Park, Calif.: American Association for Artificial Intelligence.

Hatfield, S., and Crowfoot, N. 1990. Knowledge-Based Systems in Cost Estimating and Design. *Autofact `90*, November 12–15, 21-1–21-10.

Huberman, B., ed. 1988. *The Ecology of Computation.* Amsterdam: North Holland.

Kiczales, G.; des Rivieres, J.; and Bobrow, D. G. 1991. T*he Art of the Metaobject Protocol.* Cambridge, Mass.: MIT Press.

Laird, J. E.; Newell, A.; and Rosenbloom, P. S. 1987. SOAR: An Architecture for General Intelligence. *Artificial Intelligence* 33(1): 1–64.

Levesque, H.; Cohen, P.; and Nunes, J. 1990. On Acting Together. In Proceedings of the Eighth National Conference on Artificial Intelligence, 94–99. Menlo Park, Calif.: American Association for Artificial Intelligence.

Lumer. E., and Huberman, B. A. 1990. Dynamics of Resource Allocation in Distributed Systems, SSL-90-05, Xerox PARC, Palo Alto, California.

Malone, T. 1987. Modeling Coordination in Organizations and Markets. *Management Science* 33(1): 1–19.

Malone, T.; Fikes, R.; Grant, K.; and Howard, M. 1988. Enterprise: A Market-Like Task Scheduler for Distributed Computing Environments. In *The Ecology of Computation*, ed. B. Huberman, 177–205. Amsterdam: North Holland.

Newell, A. 1982. The Knowledge Level. *Artificial Intelligence* 18(1): 87–127.

Rappaport, A., and Smith, R., eds. 1991. *Innovative Applications of Artificial Intelligence 2.* Menlo Park, Calif.: AAAI Press.

Robertson, G.; Card, S.; and Mackinlay, J. 1989. The Cognitive Coprocessor Architecture for Interactive User Interfaces. In Proceedings of UIST, 10-18. New York: Association of Computing Machinery.

Sheridan, T. B. 1984. Supervisory Control of Remote Manipulators, Vehicles, and Dynamic Processes: Experiments in Command and Display Aiding. In *Advances in Man-Machine System Research 1*, 49–137. Greenwich, Conn.: JAI Press.

Silverman, S.; Dixon, J.; Fink, T.; Kotas, P.; Shoop, A.; Ramesh, B.; Klahr, P.; and Abche, A. 1991. A Real-Time Alarm Analysis Adviser. In *Innovative Applications of Artificial Intelligence 2*, eds. A. Rappaport and R. Smith, 101–116. Menlo Park, Calif.: AAAI Press.

Smith, M. L. 1991. Cooperating Artificial Neural and Knowledge-Based Systems in a Truck Fleet Brake-Balance Application. In *Innovative Applications of Artificial Intelligence 2*, eds. A. Rappaport and R. Smith, 264–280. Menlo Park, Calif.: AAAI Press.

Stefik, M. 1988. The Next Knowledge Medium. In *The Ecology of Computation*, ed. B. Huberman, 315–342. Amsterdam: North Holland.

Sullivan, G.; Fordyce, K.; and LMS Team. 1989. Logistics Management System: Implementing the Technology of Logistics with Knowledge-Based Expert Systems. In *Innovative Applications of Artificial Intelligence 1*, eds. H. Schorr and A. Rappaport, 183–200. Menlo Park, Calif.: AAAI Press.

Waldspurger, C.; Hogg, T.; Huberman, B.; Kephart, J.; and Stornetta, S. 1989. SPAWN: A Distributed Computational Economy, SSL-89-18, Xerox PARC, Palo Alto, California.

Wiederhold, G. 1991. Mediators in the Architecture of Future Information Systems. Washington, D.C.: IEEE Computer Society. Forthcoming.

Wilson, E. O. 1971. *The Insect Societies.* Cambridge, Mass.: Harvard University Press.

Winograd, T. 1988. A Language Perspective on the Design of Cooperative Work. In *Computer-Supported Cooperative Work: A Book of Readings*, ed. I. Greif, 623–653. San Mateo, Calif.: Morgan Kaufmann.

Daniel Bobrow is a research fellow at Xerox Palo Alto Research Center in the scientific and engineering reasoning area in the Systems Sciences Laboratory. He is a past president of the American Association for Artificial Intelligence and a past chair of the Cognitive Science Society. He has done research in AI, particularly knowledge representation and expert systems, cognitive science, programming languages, and operating systems and is interested in the integration of these ideas to improve our knowledge of human thinking and make better the human condition.