

*The fundamental premise . . . concerns the need for a strong model of a domain-trained knowledge engineer . . . to be embedded in a question-asking system . . .*

# Critiquing Human Judgment Using Knowledge- Acquisition Systems<sup>1</sup>

Barry G. Silverman

It is evident that when the instances on one side of a question are more likely to be remembered and recorded than those on the other, especially if there be any strong motive to preserve the memory of the first, but not of the latter, these last are likely to be overlooked, and escape the observations of the mass of mankind.

—John Stuart Mill

The fundamental premise being pursued in this article concerns the need for a strong model of a domain-trained knowledge engineer (that is, a knowledge worker) to be embedded in a question-asking system if this system is to be effective in having a dialog with and extracting usable knowledge from the user. The *user* is defined here as a nonprogramming domain expert. An *expert*, in turn, is an individual with some know-how in a given domain. Consider the following:

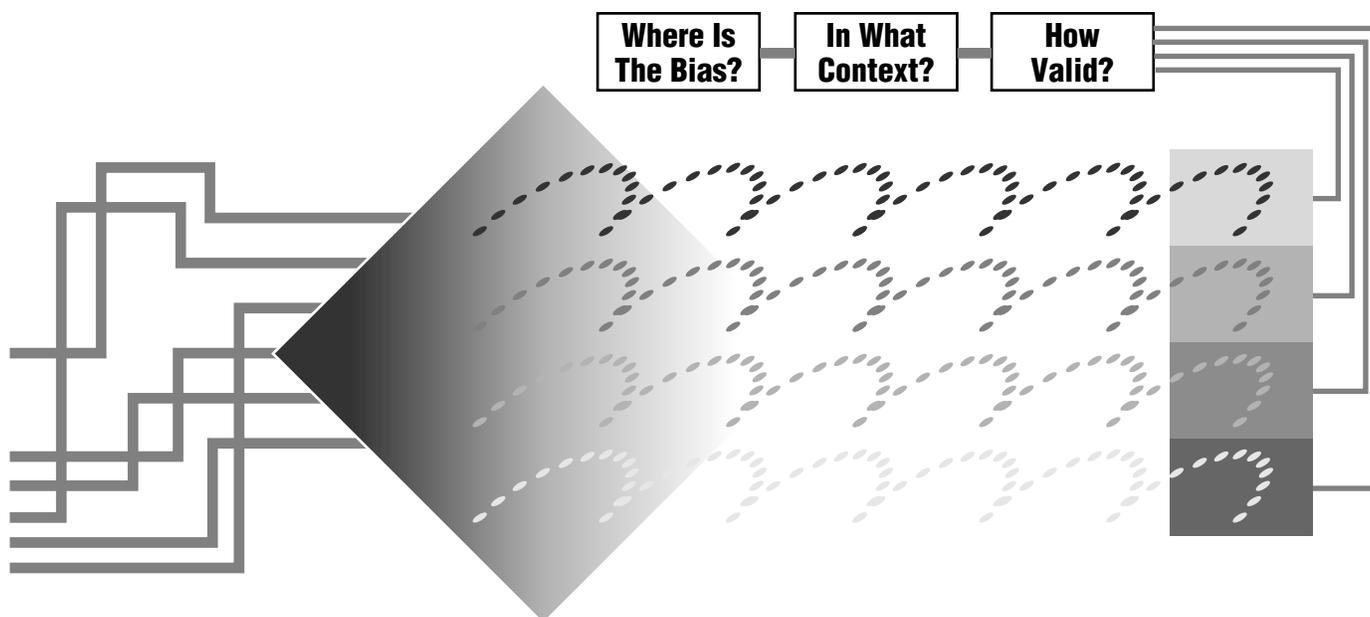
**Scene 1:** A person walks into a barbershop and asks for a haircut. What sort of haircut does the barber give? The style depends, in part, on the sex, age, and desires of the customer. How long should the sideburns be?

*Automated knowledge-acquisition systems have focused on embedding a cognitive model of a key knowledge worker in their software that allows the system to acquire a knowledge base by interviewing domain experts just as the knowledge worker would. Two sets of research questions arise: (1) What theories, strategies, and approaches will let the modeling process be facilitated; accelerated; and, possibly, automated? If automated knowledge-acquisition systems reduce the bottleneck associated with acquiring knowledge bases, how can the bottleneck of building the automated knowledge-acquisition system itself be broken? (2) If the automated knowledge-acquisition system centers on having an effective cognitive model of the key knowledge worker(s), to what extent does this model account for and attempt to influence human bias in knowledge base rule generation? That is, humans are known to be subject to errors and cognitive biases in their judgment processes. How can an automated system critique and influence such biases in a positive fashion, what common patterns exist across applications, and can models of influencing behavior be described and standardized? This article answers these research questions by presenting several prototypical scenes depicting bias and debiasing strategies.*

Should the back be tapered or squared? Should the ears be covered? Should the top be just trimmed or shortened? Most barbers only ask a few questions. Often, they assume one of a few standard models of the customer's hairstyle and pursue this model until they are given directions to the contrary. Given no direction, they still give a haircut. Even with directions, they must make numerous, perhaps hundreds, of intermediate decisions that drive the haircut toward a model in their minds. Also, far too often, they

inject a personal bias or interpretation, and although they are expert hairstylists, this bias results in a surprise—a haircut that the customer did not ask for or expect.

**Scene 2:** A student signs up for a Master's degree program and asks his adviser (1) to release the prerequisite course requirements because he had similar courses in the undergraduate curriculum, (2) to waive several required courses because of on-the-job experience in these fields, and (3) to permit several electives not normally taken by students in this curriculum. By asking a series of standard



questions, the adviser discovers the undergraduate courses aren't similar to the prerequisites. Because she has heard the on-the-job experience line before, the adviser gives a standard set of answers and explanations that convince the student he needs these courses. Finally, the adviser recalls that the electives in question aren't normally allowed because they are easy courses, such as Basket Weaving 201, and she tactfully explains this fact to the chagrin of the advisee. After some prompting for reselections by the student, a plan of courses several semesters long is eventually constructed that serves as the rules the student should follow when trying to complete his degree.

These two scenes illustrate some of the situations in which models of tasks and work are used by individuals to aid their information processing and guide their decision-making and problem-solving activities. Knowing how to elicit these models and build them into a normative judgment sequence is vital to the effectiveness of the system. As a step in this direction, this article presents (1) a human judgment theory that includes a generic model of the judgment processes (elicitation, bias identification, and corrective actions) of the knowledge-acquisition task (see Adapt the Theory and General Model of Human Bias Reduction) and that suggests that experts should be treated by the machine as fallible and (2) a methodology, called *cognitive work analysis* (CWA), for adapting the generic model knowledge to the specifics of an application (see Cognitive Work Analysis: Collect

the Domain Description and Cognitive Work Analysis: Build the Model).

Cognitive or qualitative models of work tasks are only as good as the user of the model. In the barbershop case, the barber does not have a strong enough model of how to better serve the customer, and the barber is portrayed as having the tendency to commit two human biases: the *availability bias*, in which readily available models of good haircuts are applied despite customer direction, and the *representativeness bias*, in which all customers are viewed as similar and representative of the generic customer. In the student advising case, unlike the barber who injected biases when using the models, the adviser used additional models to debias the student's program plan. The adviser used a graph of models organized under a supramodel to guide the student into an acceptable plan with no surprises or flaws. Using graphs of cognitive models in such a critiquing sequence is introduced in the next two sections.

The goal of model-based reasoning, as used here, must be kept in mind because it is easy to forget while you concentrate on the individual cognitive-modeling and -critiquing topics. This goal is to use model-directed question asking to elicit valid knowledge from humans for knowledge bases. The output of a cognitive model or set of models is a *knowledge base* that the machine can use to problem solve with. Just as the student in scene 2 wound up with a plan that he could use to guide his actions in the future, so, too, should the end result of running a cognitive

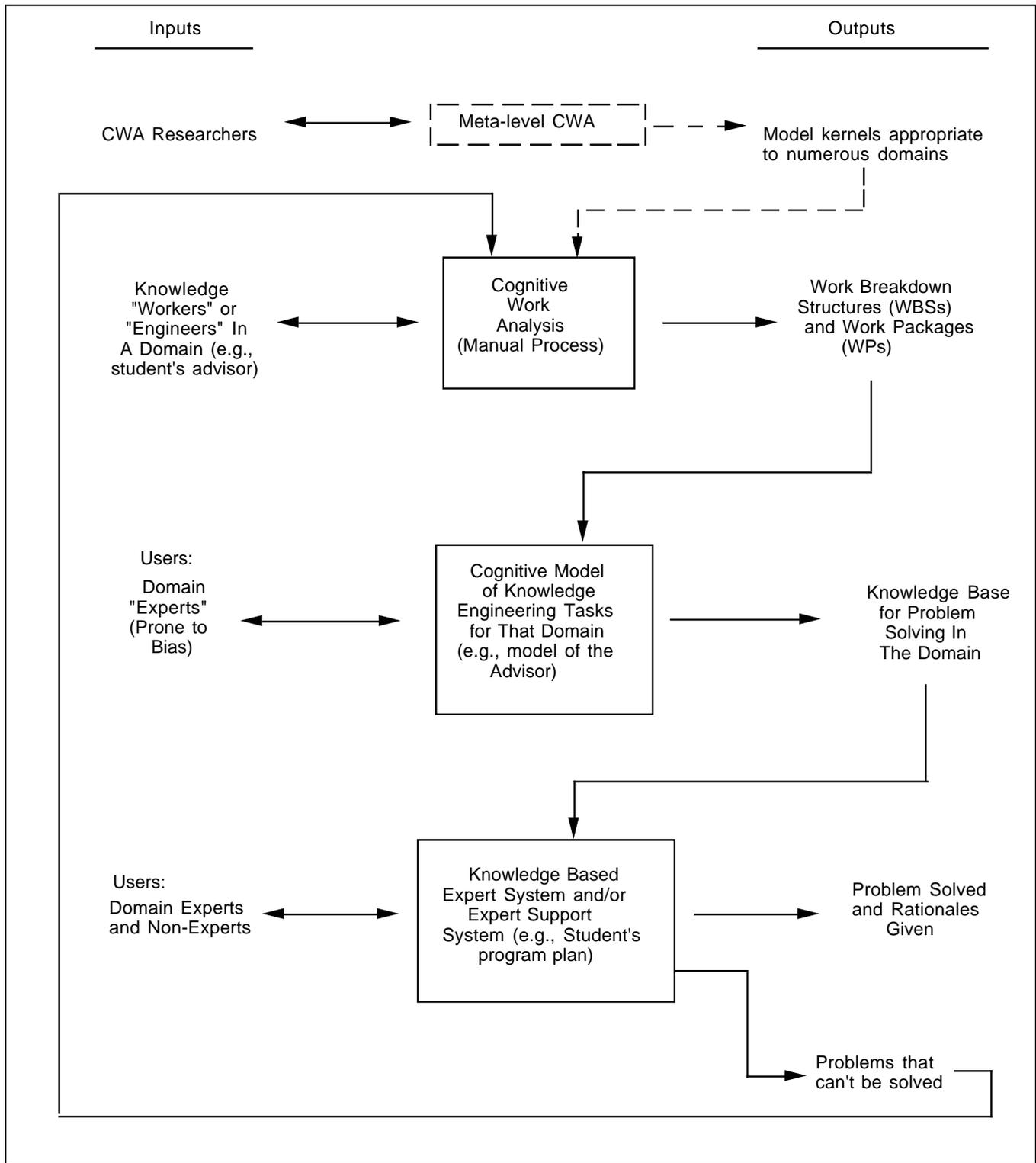


Figure 1. Inputs, Outputs, and Steps in the Cognitive Modeling Process for Knowledge Base Invention.

model lead to collecting from the human a set of rules that can be chained on by a knowledge-based expert system. This area is discussed in Cognitive Work Analysis: Evaluate the Model.

In short, as shown in figure 1, CWA is used to model the knowledge-acquisition and -critiquing work that domain-trained knowledge engineers do when collecting a knowledge base for a new domain. The term *knowledge engineers*, or knowledge workers, is used liberally here to imply any individual who engineers or constructs knowledge bases for others. Thus, the adviser is a knowledge engineer for the student, as the barber is for the customer. The input to CWA is a new domain and a set of knowledge workers in this domain, and CWA's output is a functioning cognitive model.

Running the resulting model to interview experts in this domain should lead to automated knowledge base construction on the fly. Here, again, liberal use of the term expert is useful. The student, for example, is assumed to be an expert on what he needs for his career goals. So, too, is the customer in the barbershop an expert on his needs. These experts are the users of the knowledge-acquisition system or model. The input to the model is provided in dialog with the human expert, and the model's output should be a valid knowledge base for use in problem solving (for example, in registering for courses). By using these models, the machine is, thus, adaptive in the sense that it can teach itself new rules and plans.

A research frontier is the embedding in the knowledge-acquisition system of a library of critiquing strategies that it can use for mitigating different types of expert bias and user error that it might encounter. This is the subject of this research and is indicated by the dashed lines in figure 1.

### **Cognitive Work Analysis: Collect the Domain Description**

The goal of CWA is to generate a cognitive model of a domain in terms of several specific categories of data structures and representational techniques. There are three major steps to CWA: (1) describing the domain, (2) building the appropriate model components, and (3) evaluating the final model.

Beginning with the first of these major steps, one can expect some sequence and combination of the following activities to be necessary: defining the boundaries and goals of the domain; identifying and profiling the

work tasks of a representative sample of the knowledge workers or engineers who work within these boundaries; interviewing these workers to determine the tasks and work packages that they undertake as well as the conditions under which they work; and verifying the purposes, conditions, tasks, and output of the workers. Each of these activities is discussed more fully.

#### **Define the Domain Boundaries**

Simple and seemingly small shifts in the domain boundary can cause no end of difficulty if not carefully elaborated. A furnace is a dumb, uncontrolled machine unless its thermostat is included, in which case it becomes an elegant feedback-driven system. The barber in scene 1 might have defined his job boundaries rather narrowly and when queried might respond, "I'm just doing my job." We cannot argue with success, if indeed the barber stays in business. It might be that people simply accept the barber's self-defined job boundaries with a shrug and a thought to the fact that it must be difficult to have to earn a living this way. However, if the adviser had let the student in scene 2 devise a plan that would have caused him to fail to graduate after substantial monetary investment, the student might have grounds to litigate. The stakes are higher in this situation, and the student as a customer would not simply shrug his shoulders and forget about his adviser's narrow definition of job boundaries.

The point being made here is described quite eloquently (however, from a different perspective) in the largely overlooked literature on human judgment: Slovic (1972), Hogarth (1987), Kahneman and Tversky (1973), Nisbett and Ross (1980), and Silverman (1983, 1990). These references explore the paradox that is associated with the difference between the way humans form everyday judgments and the way normatively appropriate, scientifically sound strategies would be employed. This paradox is represented in scenes 1 and 2 as the difference between the scientist (the adviser) and the lay scientist or everyday problem solver (the barber).

This is not to say that the scientist's mind is always going to form more correct judgments than the lay person's. The professional scientist, who is trained to know better, occasionally makes inferential errors and misadvises her students, and the barber's simple strategies more often than not work out for the best. Examples of professionals injecting bias in their work is documented in Silverman (1985) and Silverman and Tsolakis (1985). The point

*The goal of CWA is to generate a cognitive model of a domain in terms of several specific categories of data structures and representational techniques.*

BIAS	COMMENTS
Adjustment & Anchoring	Using of heuristics which may reduce the mental efforts required to arrive at a solution at the cost of using the full amount of information
Availability	Using of only easily available information and ignoring not easily available sources of significant information
Base Rate	Ignoring abstract information at the expense of concrete information
Conservatism	Failing to revise estimates as much as they should be revised
Data Preservation Context	Being more influenced by summarized data rather than the same data presented in detail
Data Solution	Reaching premature conclusions on the basis of too small a sample of information
'Desire for Self-Fulfilling Prophecies	Another form of selective perception
Ease of Recall	Being affected by data which can easily be recalled or assessed
Expectations	Attaching higher validity to information which confirms previously held beliefs & expectations
Fact-Value Confusion	Regarding and presenting strongly held values as facts
Fundamental Attribution Error (Success/Failure Error)	Associating success with personal inherent ability and failure with poor luck
Gamblers Fallacy	Falsely assuming that unexpected occurrence of a "run" of some events enhances the probability of occurrence of an event that has not occurred
Habit	Reutilizing the same procedure
Hindsight	Failing to think objectively upon receipt of information that an outcome has occurred & being told to ignore this information
Illusion of Control	Assuming a feeling of control over events that is not reasonable
Illusion of Correlation	Mistakenly believing that true events covary when they do not
Law of Small Numbers	Expressing greater confidence in predictions based on small samples of data with nondisconfirming evidence than in much larger samples with minor disconfirming evidence
Order Effects	Placing undue importance on the first and last pieces of information provided
Overconfidence	Ascribing more credibility to data than is warranted
Redundancy	Increasing confidence in predictions due to greater redundancy in the data
Reference Effect	Perceiving and evaluating stimuli in accordance with one's present and past experimental level for the stimuli
Regression Effects	Using the largest observed values of observations without regressing toward the mean
'Representativeness	As sample size is increased, interpreting the results of small samples to be representative of the larger population
Selective Perceptions	Seeking only information that confirms one's views and values
Spurious Cues	Accepting as commonly occurring, cues that appear only by occurrence of a low probability event
Wishful Thinking	Choosing an alternative that one would like to have associated with a desirable outcome
<i>Source: Silverman (1983)</i>	

Table 1. Common Biases in Human Judgment.

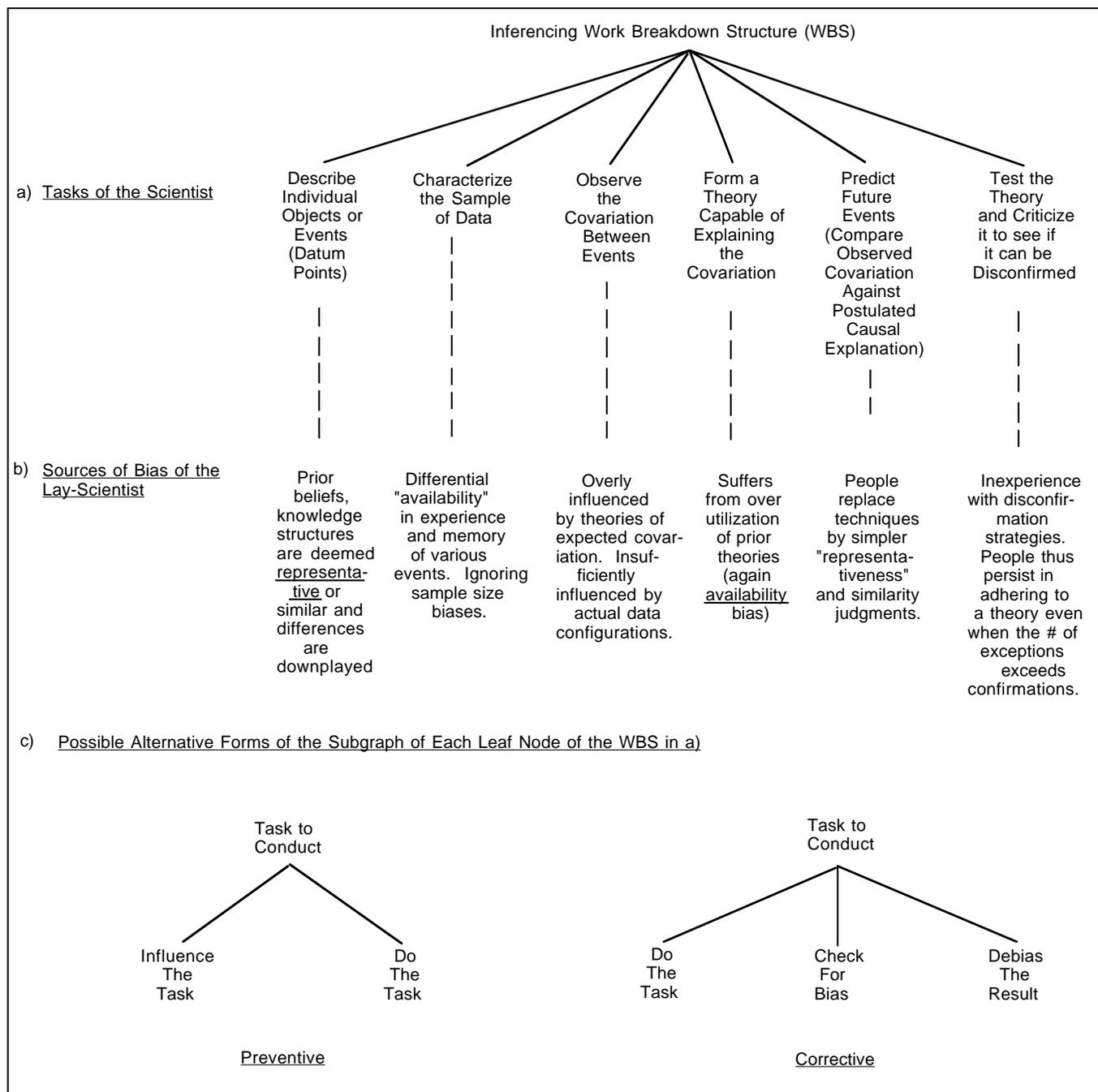


Figure 2. Inferential Thought in Human Judgment—A Cognitive Work Analysis.

is that the definition of a job's boundaries cannot be entirely entrusted to hearsay and the say-so of the knowledge worker-engineer. Verification with the consumers of the knowledge processing service is warranted. There is also no intent to cast aspersions on the barbing profession; this is an isolated example used only for the sake of argument.

Knowledge engineering practice through-

out much of the first decade of knowledge-based systems ran counter to what is presented here. The boundaries were often drawn so that the expert's say-so was assumed correct, as is so eloquently pointed out in Woods (1986). Much of the automated knowledge-acquisition literature and technique of the last half of the 1980s was also built on this presumption (Boose and Gaines 1987), and

pushing this boundary back has been, and continues to be, a major research undertaking, as shown in the following discussion. How should knowledge-acquisition systems be designed in the presence of susceptible experts? How can subjects' potential biases be detected and mitigated by the machine? If little has been done about this problem to date, what can be learned from trying different ways of achieving this goal?

### Identify Work Tasks

Not only must the boundaries be carefully elaborated but also the major categories of the work. For example, Nisbett and Ross (1980) describe the major inferential steps in forming judgments by either the scientist or the lay scientist as (1) describe individual objects or events (datum points), (2) characterize the sample of data, (3) observe the covariation between events, (4) form a theory capable of causally explaining the covariation, (5) predict future events (compare observed covariation with postulated causal explanation), and (6) test the theory (criticize it to see if it can be disconfirmed).

One can envision the adviser in scene 2 having applied such steps to her students until she began to discover all the commonly recurring errors in program plans that would lead to a failure to graduate.<sup>2</sup> In fact, these six steps are fairly universal to problem solving and can be equated to the discovery mode of thought used in machine discovery systems. Once the common and recurring student problems were discovered, one can also envision the adviser storing them in associative, episodic memory and eventually abstracting them into associative semantic memory such that they become automatic reactions to student plans.

The barber in scene 1, however, represents a prototypical lay scientist who injects bias into each of the six tasks. Kahneman and Tversky (1973) suggest two basic and universal sources of such bias arise from (1) *knowledge structure*, or a representativeness bias in which the human reduces many inferential tasks to simple similarity judgments (for example, an object, such as a customer, is assigned to one category rather than another based on overly simplistic feature similarity) and (2) *judgment heuristics*, or availability bias in which the human forms frequency, probability, or even causality assessments (for example, an object or event is judged as frequent, probable, or causally efficacious to the extent that it is readily available in memory despite the inadequacies of human memory).

Thus, the barber easily recalls only a few standard haircuts, and all customers are given one of these. Numerous other biases that might pertain to this barber but that aren't examined here are summarized in table 1. More is known about human biases than is currently known about how to debias and counter such tendencies. What systematic debiasing techniques should be used? Will knowledge-acquisition-question-asking systems need to have a library of many such routines? How will they be deployed or triggered? These and other questions have yet to be fully answered.

CWA of the Nisbett and Ross (1980) portrayal of human judgment reveals the six tasks are subject to six sources of bias related to availability and representativeness, as depicted in figure 2. The tree in figure 2a reveals a work breakdown structure (WBS) of normative scientific thought, and figure 2b suggests that the characteristic form of each leaf node must include a subgraph to correct the task process. Two alternative debiasing subgraphs are possible: (1) *preventative correction*, in which the subject is influenced before doing, for example, by helping him remember more analogs or standard (default) haircuts, or (2) *post facto correction*, in which the subject is checked using a debiaser sequence, for example, by asking whether this customer is really similar to all the others and having the barber rethink his categorization. These two options for correction are portrayed as subgraph templates, as shown in figure 2c. These options would apply during CWA to either of the knowledge workers in scenes 1 and 2, although debiasing would be needed more frequently for the lay person. The figure 2c subgraphs would apply to the experts of each domain as well.

### Interview Knowledge Workers

The point of the preceding two subsections is that one must take care when trying to analyze a domain to get the sense of the boundaries and overall tasks not just from the interviewers but also from normatively correct strategies and a general sense of what should be done in this domain. For example, the influencer and debiaser sequences are universals that must be systematically applied to virtually all tasks that humans undertake. Normatively correct strategies must serve as a metasense to be invoked when interviewing either (1) the knowledge workers or engineers to obtain their cognitive model or (2) when running this model, the experts to acquire knowledge bases (either can be biased subjects).

A number of questions have been raised

about exactly how to do it. That is, how can a machine question-asking system prevent, detect, isolate, and compensate for human biases? Further, can this system become adaptive and learn on its own how to criticize answers in a fashion that exceeds what it was originally programmed to do? Given the status of the research to date, results concerning the former question are tentatively addressed and presented here. The answers to the second question require further research, although the lessons learned in the research to date can serve as a point of departure for those interested in pursuing the matter. What follows is a summary of how existing knowledge-acquisition strategies have been combined, integrated, and extended in an attempt to address the first question. More detailed implementation issues and lessons learned from actual applications to date are explored in *Cognitive Work Analysis: Build the Model*.

"You know what you don't know but you don't know what you know" is an easy statement to prove. What's the phone number of the president of the United States? You won't even waste memory search effort because you know you never knew it and can quickly answer to that effect. However, the question "How many degrees do you rotate the steering wheel every night when you turn your car into your driveway?" takes longer to answer but is equally unanswerable. It takes longer because you think you know the answer, and you attempt to visualize yourself doing the activity. We do not store information of this type (that is, degrees of rotation) in memory, nor are we able to consciously visualize at such a fine degree of detail, even though we know how to do the task at some motor-sensory level. Three points can be inferred from this visualization exercise, as discussed in the next three subsections.

**Direct Line of Querying versus Case-Induction Strategies.** First is the well-established principle that much of a human's expertise is situationally triggered (for example, at the edge of the driveway, the driver knows what to do) and that knowledge cannot be easily retrieved by the expert in response to direct questioning, such as "What is the trait?" "How many degrees does it rotate?" or "What is the solution?" Although an astute expert can often give direct answers to certain kinds of questions, an alternative strategy is also needed when the expert draws a blank to this line of querying. The obvious alternative is to place the expert in the situation; that is, confront the expert with an actual case (problem) and elicit the traits,

classes, mappings, and solution items that the expert pays attention to in these instances. By looping through a number of these cases, the expert's knowledge is eventually elicited. This strategy is a form of CWA widely known as *case induction*.

**Enhancing the Domain Model Specificity Strategy.** A second and related point is that perhaps the domain hadn't been "psyched out" well enough to ask the correct question. For CWA to work, questions must be posed at the level at which subjects can answer them, that is, at the knowledge level. Thus, the appropriate question to ask while you are sitting down reading this text is "In which direction and about how many rotations do you think you move your wheel as you turn into your driveway?" A second, follow-up question could be "Take a tape measure into your car, and attempt to turn into your driveway. How many degrees must you turn the wheel for this maneuver?"

**Strategies for Human Bias Reduction.** The final point inferable from the driver visualization example is the human tendency toward self-deception. Although this example was limited to the reader deceiving himself only temporarily about what he thought he knew, there are numerous documented examples of the natural tendency of normally proficient individuals toward judgment and reporting biases. However, as Slovic (1972) points out, "Simply warning a judge about a bias may prove ineffective. Like perceptual illusions, many biases do not disappear upon being identified" (p.798). The acquisition system must take stronger action to debias human or expert judgment.

A number of stronger strategies are possible, just a few of which are tested in the cases covered in this article. The ultimate objective is to have a standard set of strategies available in the shell's library that will be triggered by the various cognitive biases and user errors that arise. What this library should contain is an open research question. The current approach to developing this library is to adopt a set of strategies, try them out, and refine them as results suggest. The following five strategies are a few of those currently being explored:

First is the use of default logic to provide anchors from which adjustment will be more appropriate and cue the subject about the types of answers desired to a question. Showing defaults is an influencer strategy that extends to the reuse by others of previously successful answers to the same question posed

*... to implement the strategies, it is useful to construct a hypothesis for experimental investigation.*

for developing similar knowledge bases. These knowledge bases will improve recall biases and can affect selective perceptions.

An alternative strategy (failing in the first) involves specializing a sequence of increasingly more detailed defaults, assisting the subject in combining and abstracting these defaults into a new response (influencer), and then assuring the result is usable and measurable (debiaser). This approach can help ease the availability, conservatism, data-preservation, and data-solution categories of biases.

If the first two strategies fail, the direct line of querying might be tried but generally in a tutorial fashion that would include showing the subject principles, good and bad examples with critiques, and references that might positively influence the subject's input to try and jar base rate, conservatism, and reference effects (influencer).

Any of the first three strategies are to be followed by a case-induction loop that elicits the subject's situationally triggered expertise and that tests under what conditions earlier answers actually hold up, traps for inconsistencies, and identifies when answers need to be altered (debiaser).

All answers under these four strategies are potentially suspect and are subjected to a *credibility refinement*, or critiquing sequence of activities, that is beyond the scope of this discussion but that is fully addressed in Silverman (1990) as a formalism for debiaser strategies.

After explaining how these strategies are implemented in general, I present a case study and evaluate the results of each strategy. Hoping to reduce biases is different from actually doing it. Measuring the impact of a strategy is more difficult still. As is seen, most of the results to date only provide gross-level feedback on the value of the combined set of strategies. Also, the implementation to date of each strategy is somewhat limited. Advancing these (and other) strategy implementations and setting up controlled experiments to evaluate their impact is a fertile avenue for future research efforts.

### **Adapt the Theory and General Model of Human Bias Reduction**

In general, to implement the strategies, it is useful to construct a hypothesis for experimental investigation. What follows is a hypothesized theory cast as a general model that must be tailored and experimentally applied to each new domain. It is also only one of many possible hypotheses of how to combine and sequence the strategies. Finally,

the theory takes the form of an algorithm because in this form, it is most directly testable (the algorithm is implemented as a hierarchical graph of rules rather than as a specifically programmed source code element). The theory cast as a general model is

H1: For a given problem-solving situation, human bias can be reduced by the following sequence of steps:

1. Ask user to attempt a problem-solving sub-task.
  - a. Show defaults, analogs, or both as anchors likely to influence successful outcomes (strategy 1).
2. Check user's answer for bias (epistemological assessment).
3. If bias remains:
  - a. Tutor the user about the bias (strategy 3)
  - b. Retry step 1 at a more finely grained level of detail (strategy 2).
4. If unsure whether bias remains, then attempt a path of action designed to identify inconsistencies and, thereby, trap the user into confronting the bias (strategy 4):
  - a. Repeat steps 1 through 3, as needed, to remove the inconsistency.
5. If bias still remains, then attempt a debiasing step in which a (possibly) correct answer is injected into the user's work space (strategy 5).
6. If all redundant strategies fail, place a warning in the user's knowledge base, or load the subject's task results into the rule form of the knowledge base being acquired.

The test of these elements of H1 is threefold: (1) test of expressive richness—an attempt to express H1 in the language of the Cope system is a test of the expressiveness of this language (see the next three subsections)—(2) quantitative experimentation—real results from actual subjects are collected and statistically analyzed; a control and experimental group is used (see Conduct Quantitative Experimentation)—(3) qualitative analysis—subjects' qualitative assessments of the value of the joint cognitive system are collected and evaluated (see Collect Qualitative Reactions).

### **Cognitive Work Analysis: Build the Model**

To become effective, model building should not be an arcane discipline but should be rel-

atively easy for any computer-literate individual to do. Models should consist of relatively simple data structures that are easily edited, modified, extended, and adapted. If the computer is to begin to shoulder some of the modeling activity and elicit and infer models from modelers (rather than just act as a user-friendly model editor), then it is also important to pinpoint (1) model components that are often reused from model to model, (2) types of influencer-debiasser situations and standard combinations of model components appropriate to these types (for example, each of the five strategies in the preceding section is a commonly recurring combination), and (3) sequences of questions and steps needed to set up a recurring strategy or combination with a model.

In what follows, experiences are summarized, and lessons learned are extracted from the application of a user-friendly modeling system called Cope, available on 68020 or 286-chip machines (or higher) using the DOS or Unix operating systems. Cope is a WBS (task tree) and WBS node (frame) editor in which models can be built by computer-literate individuals (for example, in a knowledge support systems class, 12 graduate students from engineering, management, and operations research programs learned to use it and built models for a student-advising domain in a seven-week interval in May-June 1989 on 12 PS 2/80 machines). Cope, a 50,000-line C language program, includes an inference engine that runs the models and acts as a question-asking system guided by WBS that elicits knowledge bases from users (for more details, see Silverman [1990]). Although able to construct knowledge bases, Cope does not currently construct or adapt the model it has been given. By analyzing how these students built their influencer and debiasser models, I hope to pinpoint the lessons learned in terms of the three points given in the opening paragraph of this section.

To implement the various strategies and build WBSs of question sets that use the strategies to help the subject instantiate each knowledge base element—in short, the model-theoretic approach—several steps must be followed:

First, construct WBS of the overall domain and the strategies to be used in the domain. Normally, this task tree is too large to think about or develop all at once and needs to be decomposed into logical clusters of WBS nodes, that is, into subtrees that are logically interconnected but that might be worked on in isolation from each other. As sequences of tasks, trees are processed left to right in a

*To become effective, model building should . . . be relatively easy for any computer-literate individual to do.*

depth-first fashion. It is possible to designate a tree to be looped over, in which case it is processed  $N$  times with variable binding performed on each of the  $N$  cycles. Looping and jumping are powerful features, and although the modeler need only construct trees, the result is a graph of tasks for the computer to perform.

Second, for each subtree, fill in the slots of each node. *Nodes* are work packages that indicate the subtasks for the computer to perform. Subtasks at a node might include the working memory input and output to be processed; the database variables or blackboard objects to instantiate; the information to be sent to, and captured from, the user at the screen; and other intermediate trees to jump to. There is a library of precanned functions that can be used to construct the eliciting, influencing, and debiasing strategies by specifying the sequence of functions required in a given node (*node function list*).

Third, run the trees by invoking the inference engine, which backward chains across the trees, performs the functions and subtasks called for at each leaf node, propagates node statuses (met, not met, unknown) up the tree, and drives toward completion of all tasks and strategies. Because any of a number of tasks could occur at a node, the backward chainer is suspended when it reaches a node, and it is resumed when node processing is finished.

Fourth, transform the end result from the database format into knowledge base rule form, and attach to an expert system inference engine.

Fifth, edit the resulting rule base, as desired.

Steps 3, 4, and 5 are intended to be performed by the user-domain expert rather than the model builder; however, the modeler will want to test all these steps to ensure they work properly. The steps are organized as shown in figure 3. To add more strategies and build models, one just extends the trees. If existing node functionality is insufficient, new functions are added to the function

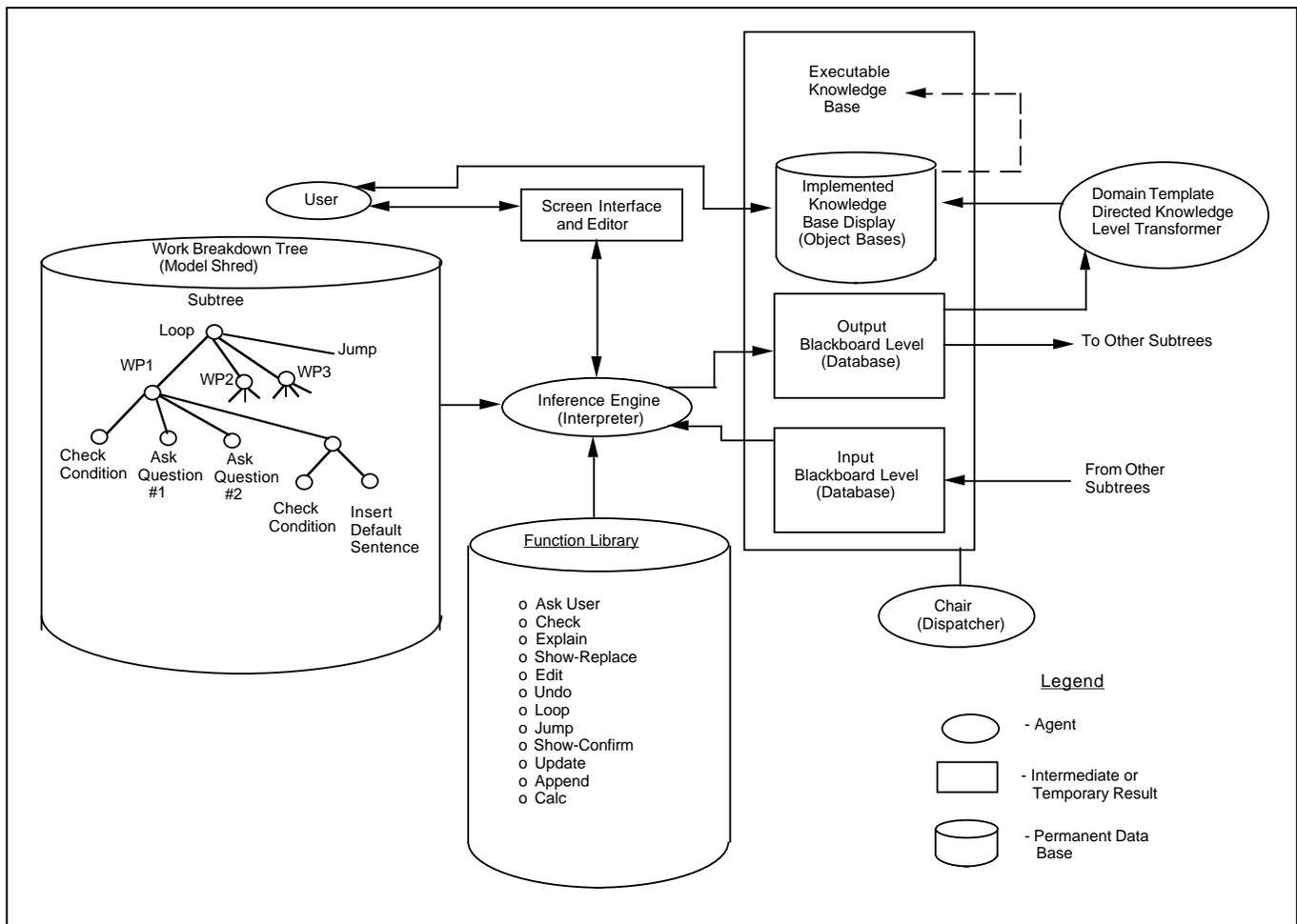


Figure 3. Overview of the Building Blocks That May Need to be Utilized in the Design of Each of the Question Sets of the Acquisition System.

library. For the time being, the transformer only converts user answers into further WBSs (decision trees) that can be run as an expert system using the same inference engine and edited using the same editor that the modelers use. To transform user answers to any other expert system shell's syntax simply requires a new transformer and a template file that indicates the data structures required by this shell.

### Construct Work Breakdown Trees and Subtrees

Work breakdown trees are effectively a programming language. The purpose of this language is to act as an electronic medium—a communication tool—that lets modelers transfer ideas, information, examples, strategies, menus, and questions to the user and that facilitates users providing responses.

As a software engineering tool, the language allows WBSs to be decomposed into subtrees or work package clusters that are more manageable to engineer and maintain (this engineering detail has less significance than the knowledge communication purposes of the language).

A *work package cluster* is essentially a goal tree that the interpreter chains over and uses to visit nodes from which it receives instructions about which functions to fire with what input and output (that is, blackboard levels, user queries and responses, and so on). As shown in figure 3, each work package cluster, or subtree, can include various combinations of as many as six types of nodes: classifying (labeled WP [work package] in figure 3), condition, question, sentence, jump, and loop.

*Classifying nodes* are nonleaf nodes that group a series of subnodes into a logical unit (that is, a rule) that can be tested as true or

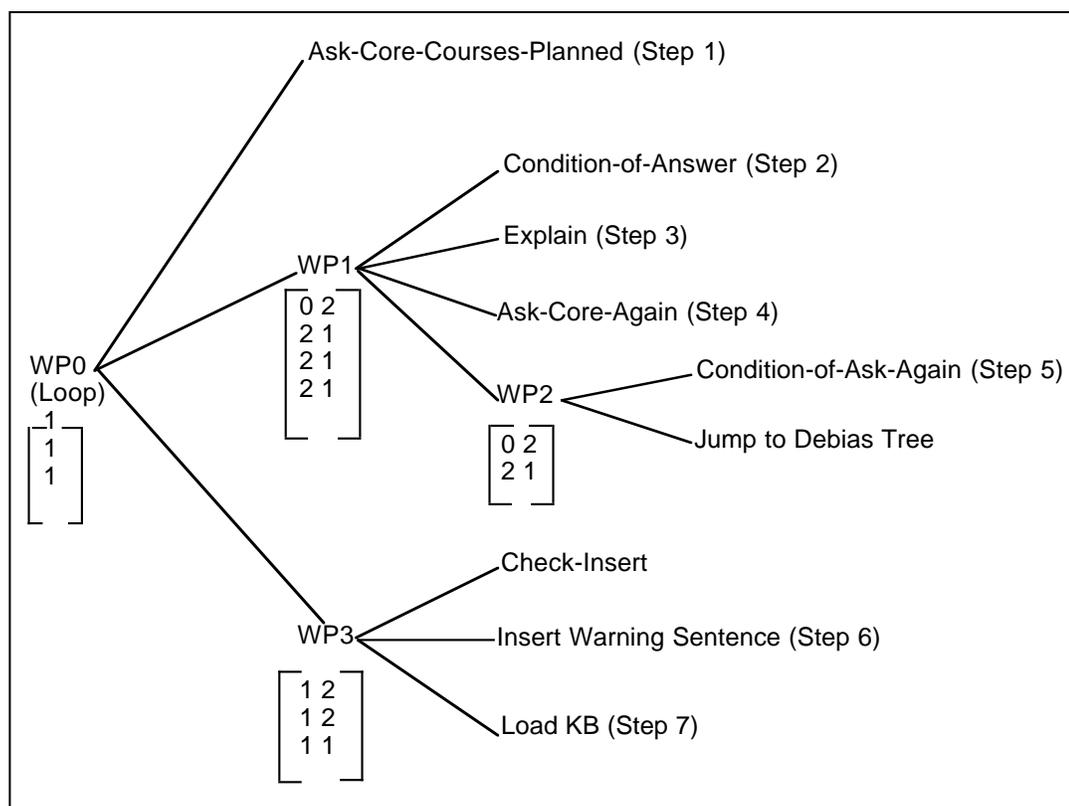


Figure 4. Influencer Example for the Curriculum Collaboration Domain.

false, as directed in the node's truth table (precise details on how node logic works are unnecessary for the purposes of this article). For instance, assuming an and-valued subtree, WP1 and its leaf nodes form a single rule for which all the subnodes must be satisfied for WP1 to be satisfied. If one of the leaf nodes is unsatisfied (for example, loop), WP1 remains unsatisfied and, hence, unfinished. Users can choose to postpone answering a question and continue to a subsequent work package or question, but eventually, the questions with unknown answers are reasked of the user; WP nodes always attempt to reach a state of completion in their children.

*Condition nodes* are tests and checks of what exists in the database or what the user has answered thus far. Condition nodes often act as gates for the inference engine to decide whether it should execute the rest of a work package. *Question nodes* send menus and questions to the screen for user querying, and *sentence nodes* insert requisite information (boilerplate information) or warnings into the working memory database (and, ultimately, into the users knowledge base once trans-

formed). *Jump nodes* cause the inference engine to suspend the current tree from completing and process an intermediate tree before completing the assessment of the current tree. *Loop nodes* cause the current tree to be cycled over N times based on the value of a cycle variable. The following case study should help clarify the process of tree design.

#### Case Study: Student Plan Knowledge Base Generation.

For the student-advising domain in scene 2, suppose every incoming student is required to include three core courses in his plan of study—Fundamentals of Engineering Administration I (EAD211) and II (EAD212) plus Management Decision Making (EAD269). However, there is a tendency for students to omit some of these courses. The foreign students who don't know English well attempt to omit EAD211 and EAD212 because these are management courses that require students to read cases and write essays, skills they haven't mastered yet. The American students, however, tend to omit EAD269, which is a math-intensive course with a reputation for being quite ardu-

ous. In both cases, the students often try to substitute courses that are known to be easy.

From a bias viewpoint, the students can be thought of as exhibiting a number of the biases listed in table 1, such as (1) *selective perception*, creating plans from a limited viewpoint even though it will cause them educational deficiencies in the long run; (2) *availability bias*, suffering from a lack of analogs on how to successfully cope with the difficulties that the tough courses present; (3) *conservatism*, trying to maximize chances of short-term success (at the expense of their long-term good); and (4) a variety of other biases that are self-explanatory, such as wishful thinking, illusion of control, and habit.

An influencer sequence might try to prevent students from constructing course plans that omitted 211, 212, or 269 by use of hinting, direct query with explanations (situated tutoring), debiasing, the showing of analogous students' successful plans, and plan-failure handling (other strategies might also be possible, but only these are illustrated here). The precise sequence might involve the following tailoring of the H1 algorithm: (1) show required courses, and ask which of these the student will include in his plan; (2) check his answer for missing courses; (3) explain the requirement for these courses more fully; (4) ask him if he would like to reanswer the question in step 1; (5) if 4 is "no," jump to a debias tree, but if 4 is "yes," loop back to 1 and repeat; (6) if all else has failed, insert a warning sentence in the student's plan; and (7) load the plan knowledge base with rules of the following form: If 'core course' is planned for = true, THEN core course requirement will be satisfied ELSE 'warning' (here, core course and warning are instantiated with their actual values from the results of the session).

This sequence of tasks is converted into a work breakdown subtree, as illustrated in figure 4. The WP# nodes are classifiers that hold truth tables or *trade-off matrixes*, which indicate the pathways the inference engine should take through the tree. The rows of a matrix correspond to the children of this WP# node, and the columns are different pathways through these children. A 1 means the child node must be tested and must be met or true, a 0 means the child must be not met or false, and a 2 means the child's truth value no longer matters. Thus, the truth table of WP0 indicates the three children are in an and configuration, and all must be true to complete this work breakdown subtree. The table for WP1, however, is disjunctive and

indicates that if the condition node is false, then skip the rest of the work package, or do all the rest of the work package. In this fashion, the inference engine is, thus, told how to chain through the nodes of the tree until a jump node is reached (for example, after step 5 in figure 4), whereupon a path is designated, and processing of this intermediate tree is performed to compute the truth value of the calling (jump) node. More details of the question, condition, sentence, and loop nodes are offered in the ensuing subsection.

### Fill in the Slots of Each Node

Tree nodes follow the same object-facet-attribute (or frame-type check-slot) formalism regardless of which of the six kinds they are. Specifically, each node consists of the following slots:

[Keyword: (name of the node)

Input BB Level: (what variables to retrieve from what database)

Phrase-List:

Format: Phase-1@Phrase-2@

Text: (string)

Principle: (notecard)

Example: (notecard)

Reference: (notecard)

Function List:

Format: Function-1@Function-2@ . . .

Output BB Level: (where to place function output)

Delete-List:

Format: Variable-1@Variable-2@ . . .

Add-List:

Format: Variable-1(value)@Variable-2(value)@ . . .

Jump Path: (message to chair to try another subtree)

Status: (met, not met, or unknown)

Evaluation Order: Do, Jump, and/or Loop. ]

Several of these slots are almost self-explanatory, such as keyword; input and output BB levels that indicate where to get shared-variable values and where to make additions or deletions based on user responses or function results; and jump, which is a path name and a request to the engine to process another tree prior to completing this one. Other slots are not as clear and require further elaboration, particularly because they will be used differently depending on the kind of leaf node the modeler wants to add.

Showing all the slots for each of the objects of figure 4 isn't necessary for the reader to capture the gist of the technique or the strategies. Instead, the first three nodes from the tree are elaborated.

Keyword: Ask-Core-Courses-Planned  
 Phrase-list: EAD 211@ EAD 212@ EAD 269@  
 Text: Which of the core courses listed in the menu do you plan to include in your program plan?  
 Function-list: Ask-user-mult@  
 Output.db: Courses.db  
 Add-list: Core-Selected!Answer-list@  
 Eval-order: Do@

Keyword: Condition-of-Answer  
 Input-db: Courses.db  
 Phrase-list: EAD 211@ EAD 212@ EAD 269@  
 Text: |'Core-Selected|  
 Function-list: Check-all@  
 Output.db: Courses.db  
 Add-list: Student-Bias!Missing-phrases@  
 Eval-Order: Do

Keyword: Explain  
 Input-db: Courses.db  
 Phrase-list: Continue After Reading@  
 Text: You have omitted the following required courses from your plan:  
       |Student-Bias|. You must select and read PRINCIPLE, EXAMPLE, and REFERENCE and then select "Continue After Reading" when you are done. Otherwise you will be unable to proceed.  
 PRINCIPLE: All students who wish to obtain the Masters degree must take the core courses including EAD 211, 212, and 269 for 9 credit hours.  
 EXAMPLE: A good course plan includes all three core courses as follows:  
           EAD 211, EAD 212, EAD 269. Any example other than this is a poor one and will require counseling with your advisor.  
 REFERENCE: Engineering School Bulletin, p. 286

*Figure 5. Detail of Slot Values for Three Illustrative Influencer Nodes of the Curriculum Collaboration Work Breakdown Tree.*

### **Student Curriculum Planning Case Study Continuation.**

The first node elaborated in figure 5 has an eval-order slot set to "do" (rather than loop, chain, or jump), which signals the inference engine to process the function in the function-list. Ask-user-mult is invoked from the function library, which displays the phrase-list as a screen menu from which multiple selections can be made. It also displays text at the screen as a question to accompany the menu. Finally, it takes the user's menu selections, stuffs them into a temporary variable called answer-list that the output.db and add-list slots tell it to bind to the core-selected variable to be added to the

courses database.

The first step taken when a do is encountered in the eval-order, as it is again in the second node of figure 5, is to instantiate any variables enclosed in bars. Thus, l'Core-selectedl of the text slot is instantiated to a list of the selections made in the preceding node. The check-all function in the function-list then compares this list with the list contained in the phrase-list slot and sorts the items in the phrase-list into found phrases and missing phrases. If missing phrases is empty, there is no student bias, and the node succeeds; otherwise, it's set to "not met."

Finally, node 3 of figure 5 is another ask-

a) Influencing via Hinting and Querying

Which of the core courses listed in the menu do you plan to take?

1. EAD 211
2. EAD 212
3. EAD 269

>3  
Do you wish to select another (y/n)?>n

b) Situation Sensitive Explaining

You have omitted the following required courses from your plan: EAD 211.  
You must and read PRINCIPLE, EXAMPLE, and REFERENCE and then select "Continue After Reading" when you are done. Otherwise you will be unable to proceed.

Continue After Reading  
PRINCIPLE  
EXAMPLE  
REFERENCE

PRINCIPLE: All students who wish to obtain the Masters degree must take the core courses including EAD 211, 212, and 269 for 9 credit hours.  
EXAMPLE: A good course plan includes all three core courses as follows: EAD 211, EAD 212, EAD 269. Any example other than this will require counselling with your advisor.  
REFERENCE: Engineering School Bulletin, p. 286

Which of the core courses listed in the menu do you plan to take?

1. EAD 211
2. EAD 212
3. EAD 269

c) Debiasing

You have omitted 211 and 212 from your program plan despite earlier instructions and warnings. Since your nationality is foreign and your EFL scores are low, it seems you might be concerned about verbal obstacles. Please select one of the following to inspect plans for taking English prep in tandem with engineering electives and for postponing EAD 211 & 212 until late in your program:

1. Inspect a default plan for your situation
2. Examine analogous students' plans
3. Obtain further background

Figure 6. Dialog Listing Resulting from the Trees That Elicit a Plan.

user node that reveals a few more aspects of the ask-user functionality. In particular, the ask-user function also displays a menu of three background notecards at the screen: principle, example, and reference. These notecards can be filled with information intended to influence the user, such as school policy regarding core courses, and the user can be prompted to read these carefully before making his final decision, as is done in node 3.

### Run the Model and Related Steps

Running the models just created leads to the screen dialogs shown in figure 6a (Note: There is a windows package in Cope that produces the text of the dialog in a better human interface, but because of space restrictions, a simple listing of the dialog is printed instead). This dialog listing only collects a small part of a single rule for the domain: Earlier trees collect background on the student (for example, foreign versus American, English as a foreign language scores, math aptitudes), and latter trees collect other rules. Specifically, the dialog of figure 6a is rather positive and preventative in character: It acts before the student has a chance to firm up his bias. That is, it first gives a strong hint that "you should take the core courses," although it allows latitude at this point because there might be legitimate reasons for not doing so. There are numerous follow-up dialogs possible for further influencing his answer and determining why the student omitted EAD269 (for example, further hinting, isolating the reason and touting the virtues of the core courses, showing examples of good plans), only one of which is shown in figure 6b. The entire rule being worked on is eventually constructed and transformed into the following format. In this case, the student persisted with his bias despite all influencing attempts.

IF: (1) Warning—EAD 211 not in plan  
(2) Warning—EAD 212 not in plan  
(3) EAD 269 is taken

THEN: Warning—Core course requirement will not be satisfied.

At this point, the user (or student) might want to further edit the rule. For example, the student and adviser will want to discuss the warning and make an adjustment. The editor used in Cope to elicit user changes is graphic and displays the rules as trees with the if conditions as children nodes and the then clause as the parent node. Graphic tree editors are particularly useful for visualizing, and user-friendly accessing of, large collections of entries, as was established by the recent semantic data model literature on intelligent databases (Potter and Trueblood 1988).

Before the user reaches this point, however, the Cope tree designed in figure 4 calls a debiasing tree (step 5 of the algorithm of  $H_0$ ) that will attempt one more strategy for critiquing the user's knowledge base. Specifically, a negative, corrective strategy is attempted (the definition of debiasing). As shown in the dialog in figure 6c, initially, this strategy mildly berates the user, then sympathizes with his perceived dilemma and indicates a default plan for coping with the difficulties it thinks the user's poor judgment is attempting to circumvent. Further dialog branches are offered that can buttress the user's confidence in the concept behind the default plan, or alternatively, the user can opt for immediately inspecting and tailoring the default. The default and analog plans are offered to the user, again not as requirements but as suggestions because it is the user that ultimately must tailor the final plan and schedule into a workable one. That is, the default and analog plans suggest how to take quantitative electives in tandem with English preparatory courses and postpone the highly verbal courses such as EAD211 and EAD212 until later in the program; however, conflicts with actual schedules and user desires must be resolved by the user.

### Cognitive Work Analysis: Evaluate the Model

The model of cognition in knowledge acquisition, as conducted thus far, is merely a hypothesized model of cognition. To verify that the model is more than just conjecture requires empirical evidence of both the quantitative and qualitative varieties. Standard procedures are used for these purposes in each of the two ensuing subsections, respectively.

### Conduct Quantitative Experimentation

The full hypothesis was delineated earlier (see Adapt the Theory and General Model of Human Bias Reduction), and only a summary of it is tested here. Also, the test is, at best, only a partial test of the full  $H_1$  because only a subset of the full algorithm was implemented in the student case study (although this test included an essential subset). Finally, it is easier to test for the absence of user improvement. For all these reasons, the null hypothesis to be tested is

$$H_0: m_2 - m_1 = 0,$$

where  $m_2$  is the mean performance level of the sample of measurements drawn from the students who did use Cope to complete their curriculum plans, and  $m_1$  is the mean for the

sample of students who did not use it. For the sake of simplicity, performance level is measured as either unbiased (that is, no errors) or biased and scored as 1 or 0, respectively.

Because the experiment is to determine whether the computer collaborator reduces the bias, the alternative hypothesis is

$$H_a: m_2 - m_1 > 0 .$$

Assumptions of the experiment are that (1) the difference in bias of the two populations of samples is the result of individual differences, and the variability for the two populations of measurements is equal and (2) because small samples are involved in both sets of measurements, the student's *t* test is used in a one-tailed evaluation at the  $\alpha = .05$  significance level.

For the student case study, 11 students were asked to prepare curriculum plans without Cope, of which 4 were biased, for a value of  $m_1 = .64$ . Eight additional students were given the same task but were given Cope as a collaborator. This time, the results included only a single bias, for a value of  $m_2 = .875$ . Applying the *t* test for 17 degrees of freedom leads to a rejection of  $H_0$  if  $t > 1.74$ ; however, the computed *t* value is only 1.13. As a result, the null hypothesis cannot be rejected for the student example; similar proportions of student plan improvements would have to be observed over large sample sizes to reject  $H_0$  and accept  $H_1$ .

In a separate case study involving document authoring in the U.S. Army, measurements were both easier to obtain and harder to evaluate. They were easier to obtain in the sense that no unbiased documents could be found in the sample of about two dozen examined; the mean for the control group was zero ( $m_1 = 0$ ). Figuring out a simple measurement scale for the experimental group, however, remains problematic. In a workshop in which 10 Army experts attempted the authoring task with the assistance of a 1000-node work breakdown tree, all results were also biased (according to three knowledge workers who served as judges). The judges' subjective evaluation was that the users' results were only slightly biased and that a vast improvement could be observed.<sup>3</sup> This result implies a different scale, such as the number of errors committed on the first attempt, should be used. Unfortunately, no such data were logged for the control sample: There were only several knowledge workers' statements (and numerous anecdotes) about high error occurrences. Several measures are now being worked out, however, and future experiments are being planned that will help to test the hypothesis.

## Collect Qualitative Reactions

Although from a statistical viewpoint, it is premature to definitively claim that the man-machine collaboration approach is superior to an unassisted approach, the qualitative reactions of subjects both with and without the machine collaborator are highly informative. Comments from unassisted experts include "I wish that someone would explain what they want," "The decision makers don't know what they want," "The rules keep changing," and "I just keep rewriting until they see something they like."

Comments from experts working with the machine collaborator include "I wanted all the information and assistance it gave me." "It helped me see some flaws in my plan," "Now I understand some of the reasons for this document," "It needs to do even more influencing and debiasing than it did," "The user's guide needs to be improved so more can use it," and "When are you going to build more of these (collaborators) for the other tasks I do?"

Early studies collected numerous qualitative reactions at this level of detail. Currently, ongoing studies include the collection of in-depth protocol results both during and after the subject's use of Cope, with special attention devoted to isolating what cues the remaining biased individuals missed and how the theory, strategies, or implementation need to be adjusted in light of this information.

## Discussion of Results

This article theorized that a generic cognitive model of how to criticize, influence, and debias human judgment could be a suitable machine collaborator for human experts engaged in knowledge-acquisition endeavors. The theory includes a hypothesized generic model (H1) and a methodology (CWA) for adapting this model to specific instances. Five major strategies included in the theoretical model were discussed (default logic, specializing-abstracting, tutorial, case induction, and credibility refinement) and sorted into two major clusters—influencer and debiaser—with this article focusing primarily on the former.

In this article, an attempt was made to see how influencer strategies should be formed to affect the selective perception, availability, conservatism, wishful thinking, illusion of control, and habit types of bias. Both general theoretical hypotheses and domain-specific hypotheses were presented and tested for the proper sequence and content of the influencer strategies they entailed.

The domain-specific model of cognition of the knowledge-acquisition task illustrated here includes the use of critics that employ hinting and direct query with explanation, situated tutoring, default logic, and analogs. Because the general model is unclear about exactly how to organize these critiquing strategies for a specific application, it is necessary to experiment with different variations of this model: Only a few of the overall set of strategies, sequences, and other variables have been attempted. The next four paragraphs outline the goal of the variations attempted thus far.

For direct query with explanation, it is believed that specific cues, rather than generic, are more effective in prompting the user. For example, instead of saying simply, "Do you want to add another item (for example, course, trait, measure) to the knowledge base?" the system asks specific questions about individual items about to be added, such as about course EAD211 or the measure for the receiver function in the communication subdiscipline.

For situated tutoring, in the literature on human judgment, Tversky's and Kahneman's (1973) point about the effects of a cognitive anchor is widely demonstrated and accepted. Once subjects have made a first pass at the problem, the initial judgment might prove remarkably resistant to further information, alternative models of reasoning, and even logical or evidential challenges. Thus, at the time of the first pass, it is vital to prime the user with principles, good and bad examples, warnings, and so on, before the user's answer has time to gel. Priming the user in this manner requires a situated tutor or, more precisely, a partial tutor capable of indicating generally correct directions but incapable of holding a store of all possible user errors and corrective approaches.

For default reasoning, in general, the easier it is to think of examples of a category or proposition, the more likely the correct answer will be forthcoming. Human memory is often distorted by vividness, recency, representativeness, and other criteria. The idea of providing a wide array of default answers to choose from is an attempt to mitigate memory biases at the point of first decision and provide good anchors that will bias users in the correct direction. Also, it is useful to suggest a default plan when users persist in their bias, as shown in figure 6b.

For analogs, a list of successful analogs also extends the user's memory and increases the number of anchors that can be chosen and adjusted from. Successful past analogs have

*Human memory is often distorted by vividness, recency, representativeness, and other criteria.*

the advantage over defaults in that they worked before but the disadvantage in that they bind the user to repeat the past. Defaults plus analogs offer an interesting counterbalance.

A test of the generic model is also, indirectly, a test of the methodology by which this model is adapted to a domain. Properly refining the CWA methodology is of particular importance because it is concerned with patterns that would allow the knowledge-acquisition-system (rather than the knowledge base) modeling process to be better understood and, ultimately, automated. The state of the methodology was discussed in this article, including the following steps:

1. Collect the domain description.
  - 1.1. Define domain boundaries.
  - 1.2. Identify work tasks.
  - 1.3. Interview knowledge workers.
  - 1.4. Adapt theory-model of human bias reduction.
2. Build the model.
  - 2.1 Construct work breakdown trees.
  - 2.2. Fill in the slots.
  - 2.3. Run the model.
  - 2.4. Transform the knowledge base.
  - 2.5. Edit the results.
3. Evaluate the model.
  - 3.1. Conduct quantitative experiments with experts.
  - 3.2. Collect experts' qualitative reactions.
  - 3.3. Analyze the results.

Testing of both this methodology and the general model was accomplished using a pair of actual case studies in which average experts constructed biased knowledge bases that a knowledge worker had to manually criticize, influence, and debias. This knowledge worker was modeled; that is, the influencer strategies were described using the methodology and emulated using an adaptation of the model. In both cases, only a portion of the resulting strategy (model adaptation) was actually displayed in this article: This strategy involved a direct line of querying followed by a reinforc-

*The bottleneck of the 1990s will clearly be the model intensiveness of automated knowledge-acquisition systems.*

ing tutorial mode (principle and example being offered), after which the question to be answered was returned to, and a warning was placed in the knowledge base if a biased answer was retained.

As a test of the computational richness of the Cope language and the abilities of the methodology, much attention was devoted to computational issues and to how precisely to build such strategies into cognitive models. Although most of the details of the modeling language were omitted, enough were included to give the reader a sense of realization that each of the four variations, plus others, could be readily modeled both in general and for specific domains. Various types of chain-loop-jump trees, node slots, function library capabilities, and other details were offered to illustrate the generality of a specific modeling language and uncover recurring patterns that would pinpoint further areas of generality not yet exploited for automated model-building purposes.

An empirical set of tests of the implemented models was attempted that led to relatively promising results, although the task of obtaining sufficient quantitative results to statistically validate the promise of the joint cognitive approach is still unfinished. Specifically, both individual users' qualitative remarks and quantitative performance improvement results to date overwhelmingly favor the theory and the idea that human-computer collaboration (rather than strictly human-to-machine transfer) is an important model of the knowledge-acquisition task. However, lest we fall prey to a judgment bias ourselves, it is worth noting that the sample sizes used thus far do not permit the statistics to bear out the trend. The final answer to the validity of the human judgment theory presented here can only be found by conducting further tests with increased sample size.

Although the influencer strategy variations presented are both intuitive and apparently

consistent with the literature on human judgment, questions for future research must also include the following: Is each strategy implemented correctly, in the sense of maximizing its impact on bias mitigation? What alternative ways of implementing each strategy exist, and which ways fare better in controlled experiments? How do the four strategies interact? Is there an ordering and combining sequence preferable to those suggested here? Are there other important influencer strategies? How many strategies should be used in tandem? When do users get turned off or information overloaded? Are user skill levels and other individual differences important variables and in what ways? Are the intents of the various strategies being realized in the way they are implemented and perceived by various types of users? In short, the progress made to date has helped to establish a theoretical and computational beachhead, but this work is literally only a beginning.

The results presented here are only a beginning from another direction as well. This direction, as mentioned at the outset of this section, concerns the automation of the cognitive-modeling process itself. Clearly, a certain degree of automation has already been achieved. For example, the function library holds the capability for crafting the various strategies, and the tree nodes and node slots are elicited by the machine in either simple question-asking style (for example, "Would you like to add another child?" or "What is the output.db of this node?"), or they can be input by more experienced modelers through the graphic editor. However, the creation of the strategies themselves, their subtrees, and the balancing of these tree truth tables is largely a manual, time-consuming activity. For example, could there be a meta-work breakdown tree that is cognizant of the available influencer and debiaser strategies that queries the user for which one to insert next or, better yet, that somehow knows when each strategy should be deployed (that is, it has the benefit of the answers to the questions posed in the prior paragraph)? How could this metatree or metamodel best walk the modeler through the process of building a knowledge-acquisition system for a specific domain? How would it account for differences between domains or even within different parts of the same domain? These and a host of other longer-term research questions must be answered if large-scale knowledge bases are eventually to be built. Quickly creating, editing, and altering the knowledge-acquisition system seems to be a prerequisite to the widespread development of cost-effec-

tive knowledge-acquisition systems. Early feedback on maintaining large-scale knowledge bases at a few pioneering sites such as the U.S. Army or the National Aeronautics and Space Administration indicates that knowledge base maintenance of larger systems cannot be done reliably by humans but depends instead on automated rule generators tracking what they invented and why so that they can recall what needs to be altered as systems evolve. The same points apply to the automated knowledge-acquisition systems.

In the 1950s and 1960s, AI research emphasized general problem-solving techniques that were search intensive. In the 1970s, the paradigm shifted to knowledge-based problem solving using domain-specific rules to minimize or eliminate search. As larger rule-based systems emerged in the 1980s, knowledge acquisition replaced search as the key bottleneck. The advent of automated knowledge-acquisition systems has demonstrated that this bottleneck can be broken. The bottleneck of the 1990s will clearly be the model intensiveness of the automated knowledge-acquisition systems. Only further research such as that begun here will erode this next bottleneck.

### Acknowledgements

This research has been sponsored over the years by a consortia of government agencies and private-sector organizations to whom grateful acknowledgment is made and for whom a number of theoretical contributions, working test beds, or applications have been developed.

### Notes

1. This article is an abridged version of material from a forthcoming book entitled *Building Expert Critics* (Silverman 1990).
2. Of course, a short cut to such an extended discovery process—and a sign of an intelligent entity—would be for her to ask questions; that is, question-asking systems save time if they can locate knowledge sources (for example, older professors) to learn from. They must still pursue the six steps to verify what was learned from question asking and to handle “outlier” data points, but the overall theory-formation process is accelerated.
3. One judge’s feeling that a marked improvement was observed was sufficiently strong to cause him to recommend the project for continuation.

### References

- Boose, J., Gaines, B. 1987. Five Special Issues on Knowledge-Acquisition Systems. *International Journal of Man-Machine Studies* 26–27:1–313.
- Hogarth, R. M. 1987. *Judgment and Choice: The Psychology of Decisions*. New York: Wiley.
- Kahneman, D., and Tversky, A. 1973. On the Psychology of Predictions. *Psychology Review* 80:237–351.
- Nisbett, R., and Ross, L. 1980. *Human Inference: Strategies and Shortcomings of Social Judgment*. Englewood Cliffs, N.J.: Prentice-Hall.
- Potter, W. D., and Trueblood, R. P. 1988. Traditional, Semantic, and Hyper-Semantic Approaches to Data Modeling. *Computer* 21:55–63.
- Silverman, B. G. 1990. *The Role of Human Judgment Theory in Knowledge-Based Systems*. Forthcoming.
- Silverman, B. G. 1985. Potential Software Cost and Productivity Improvements: An Analogical View. *Computer* 18(5): 86–95.
- Silverman, B. G. 1983. Analogy in Systems Management: A Theoretical Inquiry. *IEEE Transactions in Systems, Man, and Cybernetics* 13:1049–1075.
- Silverman, B. G., and Tsolakis, A. 1985. Expert Fault Diagnosis under Human Reporting Bias. *IEEE Transactions on Reliability* R-34:336–373.
- Slovic, P. 1972. Psychological Study of Human Judgment: Implications for Investment Decision Making. *Journal of Finance* 27: 779–799.
- Woods, D. D. 1986. Cognitive Technologies: The Design of Joint Human-Machine Cognitive Systems. *AI Magazine* 6:86–92.



**Barry G. Silverman** is a professor and director of the Institute for Artificial Intelligence in the Engineering Administration Department at George Washington University. He has published numerous papers on the nature of expert bias and the use of human judgment theory to guide explorations into collaborative roles for knowledge-based systems, his long-standing research interest. Silverman is also associate editor of *IEEE Expert*; co-organizer of the annual AI Systems in Government Conference; and author of over 40 articles, 10 books or proceedings, and four software products.