The Knowledge Engineering Review, Vol. 8:4, 1993, 309–328

Engineering expert critics for cooperative systems

BARRY G. SILVERMAN AND R. GREGORY WENIG

Institute for Artificial Intelligence, George Washington University, Staughton Hall, Room 206, Washington, DC 20052, USA

Abstract

Knowledge collection systems often assume they are cooperating with an unbiased expert. They have few functions for checking and fixing the realism of the expertise transferred to the knowledge base, plan, document or other product of the interaction. The same problem arises when human knowledge engineers interview experts. The knowledge engineer may suffer from the same biases as the domain expert. Such biases remain in the knowledge base and cause difficulties for years to come.

To prevent such difficulties, this paper introduces the reader to "critic engineering", a methodology that is useful when it is necessary to doubt, trap and repair expert judgment during a knowledge collection process. With the use of this method, the human expert and knowledge-based critic form a cooperative system. Neither agent alone can complete the task as well as the two together.

The methodology suggested here offers a number of extensions to traditional knowledge engineering techniques. Traditional knowledge engineering often answers the questions delineated in generic task (GT) theory, yet GT theory fails to provide four additional sets of questions that one must answer to engineer a knowledge base, plan, design or diagnosis when the expert is prone to error. This extended methodology is called "critic engineering".

1 Introduction

As the use of computers spreads, the likelihood also grows that knowledge collection efforts will include interviews of domain personnel who are competent but not fully unbiased experts. Further, even acknowledged experts have judgment blind spots and bias tendencies. In such cases, experts exhibit improved performance during knowledge acquisition tasks when the machine is not simply a student or a passive editor. That is, they improve when the machine is a critic, able to recognize error and help identify and repair slips and inappropriate judgments.

Critic engineering attempts to advance a theory of errors and repair strategies that helps the machine fulfil its part of this collaborative role. The result of critic engineering is a criticism-based knowledge acquisition system, an expert critiquing system, or a critic that seeks to improve the knowledge being collected from the human expert. The term "knowledge acquisition" is used loosely here to include any program which assists in the collection of knowledge from a human. This could include a word processor, a computer aided design (CAD) package, a programing language editor, and so on. Critic engineering helps make this software better able to critique the knowledge it is acquiring.

1.1 Purpose and overview

Critics are difficult to build because their developers, like those of expert systems or natural language systems, must commit prodigious effort to apply them to limited domains. Yet, again like

other knowledge-based systems, the payoff is there if repeated use of critic is likely. Grammar critics are a cost-beneficial example of the criticism of manually acquired knowledge. College student vocational and curriculum plan construction might be a domain where sufficient returns to scale would warrant construction of a criticism-based knowledge acquisition system. Other domains might include, among many others, software engineering, tax return preparation, computer aided design and repetitive decision making.

The author has previously introduced real world, working and fielded expert critics (Silverman, 1990, 1991 b; and Zhou et al., 1989). While these earlier papers *present specific critics,* the goal here is to present a methodology for creating critics. The purpose is threefold: (1) It takes the first steps toward better defining an implementable version of the critic engineering methodology. This includes first principles, generic question sets, and a library of error triggers and correction strategies; (2) it presents lessons learned about the methodology drawn from specific applications and experiments in diverse domains; (3) as a not unimportant side-effect, it acquaints the reader with a large number of methodological concepts associated with how to implement criticism-based knowledge acquisition. Of particular importance is a more precise methodology of question asking for knowledge acquisition tasks. This more precise methodology takes the form of a general model of a knowledge engineer's process for acquiring and assessing knowledge. The theory behind the methodology is still being developed, and the present paper is only a status report on the progress to-date. While the methodology is usable today, sections 3 and 4 address empirical results needed to further refine our understanding of the human error and criticism processes.

1.2 What is an expert critiquing system?

Expert critiquing systems are a class of program that receive as input the statement of the problem and the user-proposed solution. They produce as output a critique of the user's judgment and knowledge in terms of what the program thinks is wrong with the user-proposed solution. Some critic programs also produce corrective repairs, while others attempt to prevent the user from committing some more common judgment errors in the first place. Critic programs may be user invoked (passive critic) or active critics. They may work in batch mode (process the entire solution after the user has finished it), or in incremental mode (interrupt the user during his problem solving task).

Simple, yet widely used, examples of critics are spell and grammar checkers. These are passive critics the user may invoke in a batch mode. The spell checkers make use of table lookup functionality to check the user's task result for a variety of syntactic errors. As such they aren't expert critics. The grammar checkers, however, are often written in a rule language to which users may add new critic rules. Yet a rule formalism alone is insufficient to qualify a system as an expert critic. Expert critics offer a range of added capabilities, a few examples of which include, among others: (1) automatically interrupt the user when they detect a legitimate error (active, incremental and accurate, rather than passive, batch, and too often wrong); (2) reason with deep principles and domain models; (3) examine the semantic content, not just the syntax, of a user's work products; (4) interactively argue a viewpoint and persuade users; (5) alter their dialogs to avoid appearing repetitive and monotonous; or (6) automatically adapt to a user's decision style and skill level. Critics that illustrate one or more of these expert-level capabilities are appearing at an ever increasing rate in medical clinician support systems (e.g., Miller, 1983, 1986; Langlotz & Shortliffe, 1983), in computer aided design environments (e.g. Spickelmeier & Newton, 1988; Zhou et al., 1989), in complex electronic environments (e.g. Fischer, 1987), and in judgment debiasing during automated knowledge acquisition (e.g., Silverman, 1991 a,b), among other applications. In these applications, users can attempt their task solutions and the (active) critic interrupts them if it finds a serious error of judgment or knowledge.

Also, these applications use a more sophisticated algorithm that takes advantage of artificial intelligence techniques. While these critics contain an expert system capability (they produce their solution to selected aspects of the problem statement), that is only a portion of their algorithm.

They also, contain (1) a difference analyser that compares the user- and machine-generated solutions, isolates any differences, decides if the difference is worth bothering the user about, and infers a way to explain the differences in terms least disruptive to the user-proposed solution; and (2) a dialog generator that attempts to put the criticism and feedback into a form of "natural language" using canned text strings.

At this point, it is worth mentioning that expert critics are different from either expert systems or expert advisory systems. Both categories of programs only accept the problem statement as input and provide their machine-generated solution as output. The expert system is intended to reach the solution, not the user, a novice. The expert advisory system, in turn, offers only suggested solutions that users may compare to their own solutions. Expert critics won't solve the problem directly, but instead examine the user-proposed solution and critique it explicitly. This difference means that expert critics offer a third type of user support.

Expert critiquing systems are ideal when users can generate their own solutions to a recurring problem. Critics help when users commit the types of errors that machines are good at eliminating. Specifically, critics can cause the user to notice and use more cues (i.e., proper problem solving steps and knowledge). Unlike an expert system, expert critics are usually unable to solve a task problem entirely on their own. They exist solely to cue the domain expert and to act as an intelligent prosthesis device that makes expert system technology palatable to the human expert. Thus as the spell/grammar checkers are unable to author a memo, so are the more sophisticated medical critics unable to interview and diagnose a patient. Still, both types of critics are useful adjutants to help experts improve their performance on a minimal interference basis.

2 Elements of critic engineering methodology: Extensions to GT theory

In this paper, we are concerned with getting expert critics to understand users' mental processes sufficiently so the critics can detect judgment errors in the users' work. This permits a knowledge engineer to write judgment critics. So, for example, in document generation, one could author critics that evaluate the content of a document, rather than just the spelling or grammar. As we will see in the example at the end of this paper, such a critic can detect erroneous passages, judgment biases, and the likely causes of users' biases. With this insight it can also engage the user in dialogues intended to debias judgment, interactively persuade, and suggest improved concepts.

Expert judgment critics concern the use of "normative" cues or heuristics during tasks, the biases that arise, and the strategies and factors involved in mitigating those biases. "Normative" cues are defined here as the proper set of facts to use and steps to follow to reach a correct outcome. We adopt a loose definition of "normative" here to include knowledge chunks widely believed to be proper, though not necessarily proven to be so (e.g., exceptions are allowed). For example, use the cue "i before e, except after c" to spell "receive" but not "heir". Biases, in turn, are systematic, reproducible errors (not random slips and not novice errors) committed by competent individuals. Biases are misuses of cues or heuristics that lead individuals to incorrect or suboptimal task outcomes. For example, having overconfidence in the outcome of a roulette wheel being red, based on the last 10 outcomes being black. Let us examine these definitions further in section 2.1. Once we do so we will progress to a "critic engineering" methodology for identifying cues, biases and corrective procedures (see section 2.2).

2.1 Categories of cues relevant to a task

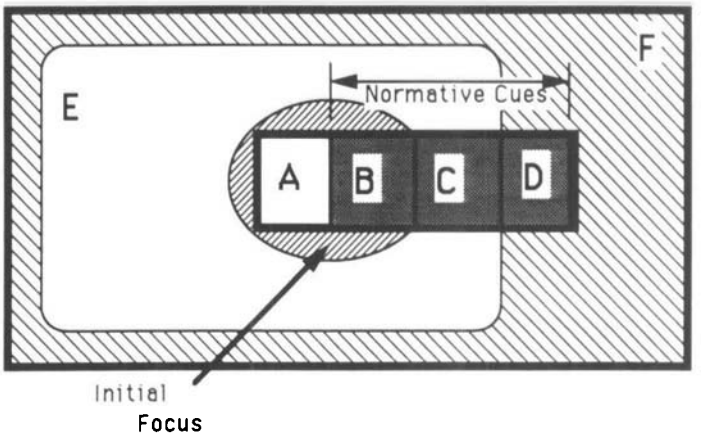
Expert task performance is more than simply a matter of adhering to normative cues. Experts often follow only high level cues. They rely on holistic, analogical reasoning from experience. This implies that ignoring some of the normative cues need not correlate with poor performance. Still, normative cues are a measurable entity and a useful point of departure for critic design. For example, no-one would argue that a Pulitzer is won just by following cues of outstanding spelling, grammar, penmanship and topic selection. Also, no-one would argue that a Pulitzer winner won't

redefine some normative cues. Still, in 99 out of 100 cases, overlooking some significant fraction of the normative cues accounts for failure to win the Pulitzer.

Thus, for each (generic) task identified in a domain, one must categorize the task cues as this section elaborates. Specifically, Figure 1 shows that for any task there is a universe of cues, F. This universe contains those cues, E, the expert can use to complete the task. There are also other cues in F that are unusable. For example, the E cues a medical doctor can use to diagnose a patient's disease are limited to those he can observe, query about or test for. Other cues may be useful from F, yet medical science still doesn't know their correlation to the disease, or it lacks a way to test for and measure them. Also included in the unusable cue portion of F are the cues labelled D that the perpendicular expert perceives as unusable but could use. "Not D" occurring in a problem is a missing knowledge error (novice error). It may be due to insufficient training or, more likely, to failing to keep up as new knowledge gets added to the field.

The region labelled A + B in Figure 1 is the cues the expert initially focuses on. Due to cognitive biases and missing knowledge errors, A + B differs from the normative set of cues, B + C + D. Yet, the correct solution to the problem requires that one use the B + C + D set. Those normative cues that are within the purview of the expert's selective attention are B. The normative cues within the expert's usable cue set, E, but outside his current focus are labelled as C.

The cues labelled C are the external manifestations of cognitive bias by the expert. For whatever internal reason—e.g., selective perception, availability, recency effects, and so on—the expert selectively focuses his attention on A + B. He neglects to factor C into his task completion effort. In a similar vein, the cues labelled D are the external manifestation of the expert's missing knowledge errors.



CUES = Task steps and domain knowledge
NORMATIVE CUES = Cues followed when doing a task properly
F = The Universe of Cues in a Task-Domain
E = The Cues That Are Usable By A Given Expert In The Task-Domain
A and B = Cues In The Focus Of Attention Of The Expert
B and C and D = Set Of Normative Cues Needed To Complete The Task Properly
A = Irrelevant, Non-Normative Cues in the Expert's Current Focus
B = Normative Cues Already Being Focused Upon By The Expert
C = Normative, Usable Cues That The Expert Is Not Focusing On (Bias)
D = Normative, Non-Usable Cues That The Expert Isn't Focusing On (Missing Knowledge)

Figure 1 Central to critic engineering is a categorization of the various types of cues in a given task-domain.

Knowing these different categories of cues is central to understanding where the expert is repeatedly going astray. Even more importantly, it is the only way to design strategies to drive his attention back from A + B to the full set of normative cues, B + C + D. For example, it is useful to know whether a given normative cue falls into C vs. D. A critic uses this to determine whether to use critiquing vs. "situated" (or on the spot) tutoring. That is, critiquing helps to mitigate bias of type "not C". Missing knowledge errors, "not D", require on the spot tutoring to inform the expert of the knowledge he needs to assimilate. As a second example, knowing precisely what cues the expert has biases for helps the designer decide what cognitive process they fall under. For example, biases can fall under the categories of knowledge acquisition, knowledge processing, intended output, and so on. This helps the designer and the critic pinpoint the type of and reason for the cognitive bias, the strategy to deploy, and the critic design to assume. As a third example, knowing the cues that appear in A gives a clue to what the expert is focusing upon and why. Knowing what is distracting the expert can help one find strategies that change his focus.

2.2 A "critic engineering" methodology for identifying cues, biases and corrective strategies

There is no existing theory that stipulates precisely what corrective approach to pursue in the presence of each given type of error. That is a subject for empirical investigation. To help such investigations, it is useful to follow a systematic methodology covering the steps to take and the variables to examine. This methodology, called critic engineering, is summarized as a multi-level question driven approach in Figure 2 and as further described in what follows. Figure 2 is a guide to the types of questions a good critic engineer should know how to ask.

At the top, one picks a domain (e.g., buying clothing, running a power plant, or software programming) that one decomposes into generic tasks (e.g., monitoring and detection, heuristic classification, planning). It is useful in the Task Level of Figure 2 to quote a set of *Generic Task* questions from Bylander and Chandrasekaran (1987). In their approach, tasks are understood via a series of questions such as listed in the top Level of Figure 2. There is an *interaction* between representation and inference technique within each task that they recommend be exploited in designing a knowledge acquisition system.

The Task Level questions define an intelligent actor's cognitive model. This is a *structural model* of the domain. It includes the inputs, outputs, representations, vocabulary, concepts, sub-tasks, work packages, interactions and acquisition system features for each generic task. For example, consider a domain of buying clothing. A GT might include obtaining the likely price information. One defines the structural model of that task by answering each of the questions of the Task Level of Figure 2.

GT analysis leads to a normative view of the domain. All the Task Level questions of Figure 2 lead to the kind of insight one gleans from *retrospective* analyses of successfully completed tasks. That is, the resulting cognitive or structural model lacks an account for errant expert behaviour. To obtain accounts of the errors, one needs descriptive analyses. One must collect these while watching individuals work (and think aloud). The goal is to uncover the reparative processes that experts use to recover from errors. (Section 2.3 addresses this more fully.)

For critic engineering purposes, the outcome of the task level analysis leads to the identification of the normative cues one must use to finish a task properly (see Cue Level of Figure 2). For example, cues might exist on where to look for price information, on how to structure that information to complete the task and solve the problem, etc. From this point, critic engineering proceeds through the steps of identifying the recurring errors, isolating the corrective strategies, and designing the actual dialogues. A critic design for a generic task requires the study of "buggy" behaviour and likely remediation strategies in addition to study of the normative task outcome.

In Figure 2, the Error Level questions the epistemological status of the individual who is providing the answers to Task Level questions. Like a dialectical or devil's advocate, Error Level questions attempt to poke holes in the answers of Levels 1 and 2. Most of the common biases (availability, overconfidence, covariation, etc.) committed in generic tasks arise in the third level of

Task Level - Domain Task-Structural Questions

Generic Tasks: For a given domain, what are the type of tasks that occur? What is the function (input and output) of the generic task? What is the Generic Task good for?

Representation and Inference Interaction: For a given Generic Task, how do the representations and inference strategies interact with each other and with the task? How should knowledge be organized and structured to accomplish the function of the generic task? What inference strategy can be applied to the knowledge to accomplish the function of the generic task?

Concepts: What concepts are the input and output about? How is knowledge organized in terms of concepts? How does the inference strategy operate on the concepts?

Exploitation: How can the generic task-structure model (functions, interactions, concepts) be exploited by the knowledge acquisition methodology to extract and select the appropriate knowledge? Have all the specific categories of knowledge been collected and are all their interactional, functional, and/or conceptual roles understood? What structural errors (e.g., inconsistency, circularity, dead-endedness, etc.) have occurred? How can these be removed?

Cue Level - Normative Cue Utilization Questions

What is the relevant universe of cues? Which are usable? Which are normative? What normative cues must be utilized to finish this subtask properly? Are any of the cues relatively new to the task or field of interest? Are some of the new normative cues unlikely to be widely known at this point in time?

Error Level - Missing Concept and Misconception Questions:

EXTERNAL: What cues are the experts focusing on? Which of those are normative? inappropriate? What normative cues are usable by the expert that aren't now being used and why aren't they used? Which normative cues are beyond the expert's state of knowledge? How are cues judged, weighted and scaled by the expert and is that done properly? What errors of comission and omission is the expert succumbing to?

INTERNAL: What do the user's external answers tell us about his heuristic biases? missing concepts? What would I have to have had in memory to reach the user's input? What kind of heuristics (e.g., availability, overconfidence, etc.) might he have used to have reached that kind of answer? How are they biased? misused? What formal reasoning steps has he omitted or relaxed?

Strategy Level - Influencer, Debiaser, Tutor, and/or Director Questions:

POSITIVE INFLUENCER STRATEGIES: How should I better cue the user to dampen his biases? errors? What principles, examples, references should I refer him to that would alter the way he solved the problem? What successful analogs and lessons learned can be shown to him that have similar characteristics as the current one? What directors (formal reasoning aids) might he benefit from? What tutoring does he need?

NEGATIVE DEBIASER STRATEGIES: How should I repair/modify his answers so they appear more credible yet will still be acceptable to him? What alternative views of what he's authored will cause him to think about its clarity, workability, and correspondence with reality? What kinds of persuasion and viewpoint argumentation will help him overcome his biases/errors? What directors should the user be required to follow?

Design Dimension Level - Fine Tuning Questions:

How should bias and strategy information be modulated and presented so as to enhance the collaborative relationship? How should each of the critic dimensions (e.g., process, timing, mode) and collaboration factors (deep knowledge, cognitive orientation, intention sharing, control plasticity, mutual learning, dynamic memory) be set for a given user type to improve critic success and user productivity?

Figure 2 Critic engineering requires one to ask five levels of questions

Figure 2. Consider an example in the buying clothes domain. An *availability bias* occurs if a consumer thinks all shoes are expensive based on examining the only (readily *available*) catalogue he finds lying around the house. Availability is the internal cause of the bias. The external manifestation is in mistakenly thinking all shoes are expensive. The expert omitted the cues "find and compare alternatives".

The Strategy Level, in turn, exists to prevent bias and missing knowledge error from occurring. It also attempts repairs after a biased judgment occurs. The Strategy Level includes the common strategies (hint, tutor, explain, repair, etc.) defined in criticism-based knowledge acquisition (see section 2.3). This level holds prescriptive knowledge as opposed to the normative (presumed

always correct) knowledge of the Task Level. Strategies for helping the example consumer find a potentially better estimate of the cost of shoes might overcome his availability bias by preventively suggesting he looks at several catalogues, or corrective strategies might show him the information and point out a better price for the same pair of shoes.

Critic design dimensions, the lowest Level of Figure 2 provide the questions that one must answer to fine tune the strategies and make them work. For example, this asks what media and human-computer interfaces are best for pointing out the user's error. Dimensions and factors are an important part of criticism-based knowledge acquisition. They play a vital role in the success of the collaboration; however, they are beyond the focus of the current paper.

Finding specific instantiations to the questions of the Cue, Bias, Strategy and Dimension Levels for real applications is the methodological aspect of building cooperative relationships and critics. Further discussion of how to get the machine to ask these questions is in the ensuing sections.

2.3 Error identification and repair strategy principles of critic engineering

The idea captured by the Error and Strategy Levels of Figure 2 is as follows:

PRINCIPLE 1: The machine should have a library of functions that serve as error identification triggers and influencer, debiaser and director strategies.

To implement critics, it is useful to have a library of "functions" or specifications for identifying when flawed answers, biases and general cue usage failure modes arise. It includes general triggers that one can specialize for a specific domain or application. These trigger functions ease the job of the designer of the machine critic.

In the same vein, the function library should hold multiple corrective strategies. One can deploy these in a partially redundant fashion. This will "blanket" the set of likely causes of unwanted cue use and normative cue non-use. The alternative strategies include, among many others, the ones that follow:

(1) *Influencers* that help prevent a bias before it occurs. These are positive criticisms in the sense that they do not accuse the user of having erred. Some examples are:
- hinting, cueing and attention focusing to minimize potential biases;
- showing reusable defaults and analogs to provide good starting anchors;
- assisting with default and analogue adaptation heuristics;
- explaining principles, examples, and references to assist anchoring and adjusting processes (i.e., on-demand tutoring);
- presenting information about the rationale of the normative model (use hypermedia);
- repetition, often in different form, to be sure points get through; and
- visualization of the big picture to avoid selective focusing.

(2) *Debiasers* that help correct a bias after it occurs. These are negative criticisms, by definition, since they must mention the user has made a potential error. Some examples are:
- identifying Failure Modes (FM) such as missing concept or misconception;
- explaining Effect (E) and Cause (C) to try and elevate the subject's awareness;
- attempting Repair Actions (A) such as but not limited to specializing, generalizing, analogizing, substituting, adjusting and attention re-focusing to try and move the subject to a more productive pathway. This is the last step of the FMECA paradigm;
- view argumentation including two-way mutual exchanges to persuade the user that his claim, warrant or data lacks support; and
- visual feedback in the form of colourization, highlighting, etc. of imperfections in the task artifact.

(3) *Situated tutors* that help with missing concept errors. Most of the Influencers and/or debiasers can fill this role as well.

(4) *Directors* are support systems able to walk a user through the use of a tool they find too difficult to implement on their own. These include various support systems.

These are many strategies. Few users will need this much criticism. One must design triggers so redundant strategies only interrupt the user if earlier strategies fail to remove the error. This "decision network" of critics leads to a useful and conservative approach in which redundancy and repetition have a chance to operate on potentially stubborn strategies.

3 How and where critic engineering has been implemented to-date

The previous section explains a five step or five level methodology for the engineering of critics. Section 3.1 presents an overview of how and where that five step methodology has been used to engineer critic applications to date. The goal is to let the reader see that the methodology has practical value.

It would be useful for section 3.1 to present the details of each critic engineering application and experiment, yet space will not permit this. Instead, references to the various studies are provided. Still, there is a need to explain further details of the lessons learned in critic engineering to date. Section 3.2 does this.

3.1 Overview of selected critic engineering results to-date

This section summarizes about three-dozen critic engineers' attempts to use critic engineering methodology in five distinct domains. The five efforts and their various authors correspond to the five rows of Table 1. The columns of Table 1 summarize the details for the first four levels of critic engineering methodology. Also, the last two columns of Table 1 indicate the experiments conducted and the lessons learned about critics. The next few paragraphs discuss the entries in the body of the table. In this discussion, the reader should note that the first two rows correspond to real applications. The last three concern the use of critic engineering in experiments intended to learn more about human error, critic engineering or Principle 1.

The first row of Table 1 summarizes the largest critic application we have constructed to-date. It supports decision paper writing in the U.S. Army. Specifically, the critic supports Army personnel at 17 sites nationwide who must write hundreds of decision papers per year, one for each new piece of materiel the Army buys. For each paper, there are two principal tasks: (1) to forecast the worst mission the piece of materiel must survive; and (2) to structure the issues and criteria the piece of materiel must satisfy with respect to that worst mission. If the piece of materiel can satisfy those issues and criteria, it will be effective in the battlefield. Unfortunately, experiments show the Army personnel, with the best of intentions, tend to overly focus on technical gadgetry to the exclusion of cues about operational effectiveness and suitability. That is, the personnel focus on concrete experience they have with engineering and overlook abstract (handbook, training and intelligence) information about operation, use and support of the materiel in the field. Following Principle 1 and critic engineering methodology, the author built a critic with 1500 rules, 1500 objects, 2000 notecards and 300-odd analogs to attempt to influence, debias, tutor and direct the Army personnel to write more well-balanced decision papers. Aside from a successful implementation, several experiments were performed on both professional personnel and students. The lessons learned support Principle 1 and offer added details of how to implement the methodology, as section 3.2 explains.

The second row of Table 1 summarizes a prototype of a critic for an engineering design task. Specifically, a critic with about 3–4 dozen rules was embedded behind a ship computer aided design (CAD) package. The critic evaluates designs to help with the placement of antennas to avoid electromagnetic interference effects. Critic engineering showed the users suffered primarily from missing concept, rather than misconception, errors. Also, this was one of the first critics we built. It follows conventional critic design advice (e.g., as recommended by Miller, 1983, Langlotz & Shortliffe, 1983, or as used in grammar critics) in that it uses only batch after-task debiasing. This

Table 1 Overview of applications and experiments using critic engineering

	First 4 levels of critic engineering					
Reference	*Task level*	*Cue level*	*Error level*	*Strategy level*	*Sample*	*Lessons learned*
TIME in Silverman (1990, 1991 b; Gringrich et al., 1990)	Fore-casting and decision paper writing	Listen to HQ, take training, read handbook	Focus on technical items, not operational ones	Follows principle 1	27 experts 93 student users	Influencers successful, experts need less influencing
CLEER in Zhou et al., 1989; Silverman & Mezher, 1992)	Computer aided design of ship antennas	Follow internal mental model of task	Missing concepts about electro-magnetics	After-task, batch debiasing only	4 designer super-visors	Influencers preferred to debiasers
Silverman (1992 b), Heron et al. (1990), Bingham et al. (1990)	Probability reasoning in common sense situations	Follow formal models of prob. & Stat.	Replaces formal with heuristic models	Follows variants of principle 1	145 intermediary through expert statis-ticians	Influencers significantly more effec-tive than debiasers for all skills
Berger et al. (1991), Creevy et al. (1991), Ratte et al. (1991), Silverman (1992 c)	Hypothesis testing in large search space with feedback	Follow scient. method (use discon-firmation)	Use con-firmation, not discon-firmation	Follows variants of principle 1	70 users	Principle 1 outperforms all other alternatives dibiasers ignored
Bailley (1991), Silverman et al. (1992)	Two tasks: journalist interviews car choice	Follow internal mental model	Attention focusing biases described	Not applicable	30 subjects for each task	Descriptive results, plus over 60 bias identifi-cation rules

violates Principle 1, and experiments revealed that Principle 1 is more correct than the conventional design advice. This conclusion is based on the reactions of four supervisors of designers when they used the system. The lesson learned was that incremental, before and during task critiquing (through multiple media) is necessary to avoid frustrating users who are competent in the general task but who are missing concepts in a specific subtask. We are now building engineering critics for the Navy that benefit from these lessons (see Silverman & Mezher, 1992).

The third row of Table 1 summarizes a series of experiments two semesters of graduate students performed in a probabilistic reasoning domain. The domain consisted of situation descriptions that seemed like they could be solved via common sense heuristics, but which really required the use of formal models of probability and statistics. The students followed critic engineering methodology to identify which cues the subjects would be likely to focus on/ignore. They then used this as an "error theory" to tailor critics to alter the subjects' mental schema for solving the task. Six teams of students adopted different error theories that led them to implement different critic strategies. Each team then ran a control group and an experiment group consisting of between 15 and 49 subjects, ranging in skill from intermediate to expert. Comparing across experiments and adjusting for skill differences of the users reveals that expert statisticians use fewer before and during task influencers. Regardless of skill level, after-task debiasing is almost always unsuitable as a sole

strategy. It was found, however, that after-task debiasers become more useful as task difficulty increases. This is due to the inability of the influencers to capture the subjects' attention. From the explainability viewpoint, another lesson is that the media and wording of the criticism appears to have significant impact as well. Silverman (1992 a, b) explain these results more fully.

The fourth row of Table 1 summarizes three more experiments from graduate student critic engineers in a third semester's project. This time they chose a task of hypothesis testing in a large search space with feedback. The domain was guessing a string of numbers to induce a rule about numbers (the so-called Wason's 2 4 6 problem). The only normative cue of interest here is whether the subjects use number strings that disconfirm their hypothesis for the rule. There is a human tendency to use the confirmation heuristic, despite training in the scientific method which teaches the value of critical thinking and disconfirmation (trying to disprove a hypothesis). Individuals who use disconfirmation heuristics solve the problem. Confirmation leads to extended search and failure to reach the correct hypothesis in the time limit. The lessons learned here repeat some of those of the previous paragraphs. That is, subjects given before-task influencing, and incremental during-task critiquing, far outperform subjects with after-task debiasing alone. In fact, there was no evidence that after-task debiasing helped any subject at all. Also, critic effectiveness was highly dependent upon critic content (e.g., see Silverman, 1992 c).

The last row of Table 1 summarizes Bailley's masters thesis study in which he attempts to derive a more specific, machine implementable theory of expert errors. Bailley used only the first three levels of critic engineering methodology. He did not build any critics, but studied the cue identification processes instead. To further simplify his thesis, Bailley examined only information acquisition tasks and heuristics. His goal was to use the first three levels of methodology to generate additional rules that could assist critic engineers in identifying and categorizing specific kinds of misconception errors. He examined the heuristics of 30 subjects in each of two tasks: journalistic interviewing and car selection. By comparing their behaviour to a normative set of cues, he isolated over 60 machine implementable rules for identifying the presence of a dozen and a half common misconceptions errors. For example, he generated rules a machine (or a human engineer) could use to identify the presence of availability, overconfidence, confirmation and base rate biases, among others. This type of study is one of many steps needed to advance critic engineering methodology into a critic engineering theory. A theory would tell the engineer all the errors of each type to expect for any given task. It would also indicate precisely which criticism strategies will eliminate those errors for specific types of users and domains. Completing such a theory is a long-term goal. Many more studies such as Bailley's are needed before that goal can be reached, a possibility made more likely by the methodology presented in Silverman, Donnell and Bailley (1992). As those studies are completed, critic engineering, like knowledge engineering, methodology will become less and less of an empirical activity.

3.2 Lessons learned about critic engineering methodology: Practical implications

The goal of this section is to present a few details that will be of value to readers actually trying to implement critic engineering. Section 2 offers a methodical set of questions to ask and critic strategies to implement during critic engineering. Here, we will examine some of the more mundane details of organizing oneself to ask those questions, to record the answers, and to implement the strategies. Specifically, this section illustrates (1) the use of a "cue table" to help perform the analyses needed in Levels 2 and 3; (2) the role of a cognitive task analysis diagram to direct the overall effort; (3) value of a critic language for both exploratory programing and full scale production of critics; and (4) how the library of triggers and strategies help the designer form a decision network of critics.

Rather than present these lessons learned in the abstract, the discussion illustrates them in the context of an example application in which they found repeated use. Earlier it was mentioned that in document generation, one could author critics that evaluate the content of a document, rather than just the spelling or grammar. This example shows how a critic can detect erroneous passages,

judgment biases, and the likely causes of users' biases. It also illustrates how to engage the user in dialogues intended to debias judgment, interactively persuade, and suggest improved concepts.

In particular, this discussion uses a piece of the decision problem structuring task of the first critic of Table 1. In section 3.1 we briefly described the second task of this decision paper authoring domain. That task is to get the Army decision paper writers to overcome their tendencies to focus solely on low level details of the technical gadgetry of the new piece of materiel they are trying to decide whether to purchase. Instead they need to also include a balanced set of issues and criteria (e.g., mutually exclusive, collectively exhaustive, operationally integrated, etc.). That is, the critic must help them to remember that the new piece of materiel must work in an integrated, stress-filled environment populated by soldiers (users) of possibly low-IQ. Such "operational" issues and criteria decide the ultimate usability of any new piece of materiel. Our critic will also be concerned that the issues and criteria added to the decision paper are posed in the form of a question, the answer to which is measurable.

In what follows, we will construct a cue table of the cues they should and shouldn't focus upon; follow that up with a diagram that summarizes an analysis of the cognitive tasks and cues followed by the authors (vs. what is desired); and conclude with a discussion of how we used the table and diagram to guide our critic programming efforts. In the end, the discussion will show a typical author's dialogue where his errors are corrected by the critic. The goal is to acquaint the reader with the actual critic application only to the extent necessary to clarify the points being made about the methodology. For full details about the case study, see Silverman (1991 b).

3.2.1 The role of a cue table in organizing answers to levels 1 through 3 questions

As Levels 1 through 3 of critic engineering methodology indicate, it is necessary to start with an inventory of the different categories of cues in the domain and in the subjects' behaviour. The engineer collects this information by seeking answers to the questions of Figure 2. It is useful to construct a "cue table" to help organize the answers and to show which questions still don't have answers. Lots of cue tables have been constructed for the many subtasks of the five applications and experiments of Table 1. An example drawn from this pool is the cue table on how to structure the issues and criteria a piece of materiel must satisfy with respect to the worst mission it will have to endure. This table uses the lettering scheme of Figure 1 and answers the questions of Figure 2 as follows:

F) Universe of cues

> Mutually exclusive, collectively exhaustive, integrated, operational (as well as technical), high level, and measurable are only a few of the possible attributes of issue and criteria dimensions to decision problem formulations.

E) Usable Cues

> All of the universe listed above appear usable.

A+B) Experts' Focus

O) Over-Valued Cues

> technical gadgetry alone
> viewing the materiel item in a standalone fashion
> low level details that barely affect the formulation of the decision problem

U) Under-Valued Cues

> gains in operational performance
> assessing how well the item can be integrated with other materiel plus with operating and maintenance personnel and practices
> summarize the most important issues and criteria *only* at a level that higher level management can base decisions upon.

B + C + D) Normative Cues

B) Those the Expert Already Uses

> the over-valued cues are readily used

C) Those Cues for Which the Expert is Biased
> Integration of technical concerns with field ones
> Operational and service personnel related concerns,
> High level decision maker specifications of the decision problem
> Measurability of the criteria and issue statements
> Miscellaneous other usable cues in the universe.

D) Those Cues for Which the Expert Suffers From Missing Knowledge
> newcomers appear unfamiliar with the criticisms from the high level decision makers of the domain.

Even without understanding much about this domain, the reader can see by inspection how the Army writers tend to overly focus on technical gadgetry to the exclusion of cues about operational effectiveness and suitability. That is, the cue table relates the methodology of Figures 1 and 2 to the domain. It also helps all involved—e.g., critic engineer, domain subject, and sponsor—see what errors the critic will be targeted at reducing.

3.2.2 How to use a cognitive task analysis diagram

The cognitive task analysis diagram is a visual statement of two items for any given subtask: (1) the errors that are to be attacked by the critics; and (2) a summary of what critic strategies will be used to form that attack. Having a visual statement or graph of this information helps many participants of the critic engineering process. It helps the sponsor see whether the coverage of common errors seems as broad as he desires. It helps the critic engineer organize his thoughts and plans concerning what triggers and strategies to deploy. It helps the actual critic programer see what the critic engineer intends. Finally, it helps the criticism recipient see what process he will be going through.

The term "cognitive task analysis" is used since another diagram is needed for each cognitive task in the domain. It's an "analysis" in that the domain task forms the root of the tree and the branches plot the lower levels answers of the questions of the critic engineering methodology. The nodes at each level usually hold a few keywords indicating the findings for that error, cue, bias, strategy, and so on. An example diagram appears in Figure 3. This diagram shows how the information of the cue table of the previous section forms the first few levels. For example, only the "write criteria" subtask is shown in Figure 3. For this subtask, the five cues labelled 'C' near the bottom of the cue table, appear as the five nodes of the cue and bias levels of Figure 3. The cue level shows the normative set of cues one wishes the domain personnel to follow. The bias level shows the bias tendencies of current practice with respect to those cues.

In general, the strategies for eliminating each type of error are drawn from the library (Principle 1) and follow the lessons learned of Table 1. Influencers heavily populate all cognitive task analysis diagrams although more difficult tasks use less influencers. The situation in Figure 3 shows that influencers and debiasers were selected for critiquing the "Too Low Level" error. Directors are omitted since it is felt the users can correct the error on their own. Tutors appear inside the influencers cluster and, in effect, within the debiaser cluster as well. Also, the diagram can show media choices and user sensitive dialogue if desired at the lowest level of the diagram. This level is omitted in Figure 3, since this paper omits much of the discussion of the fifth level of the critic engineering methodology.

Some critic engineers choose to expand the node of each level of their cognitive task analysis diagram. The goal is to offer a greater degree of specificity of what each node implies. For example, Herron et al. (1990) prepared their cognitive task analysis diagrams with about a paragraph of information in each node. At the cue level they include the normative models and equations of their probability and statistics domain. At the error level, they explain the weak heuristics that tend to replace the normative ones. Finally, at the strategy level, they include the actual text of each notecard the critics will show the user for each error. They printed their diagrams, using a calcomp plotter, onto poster size paper. This makes for effective briefing material. It also gives the critic programmer precise guidance.

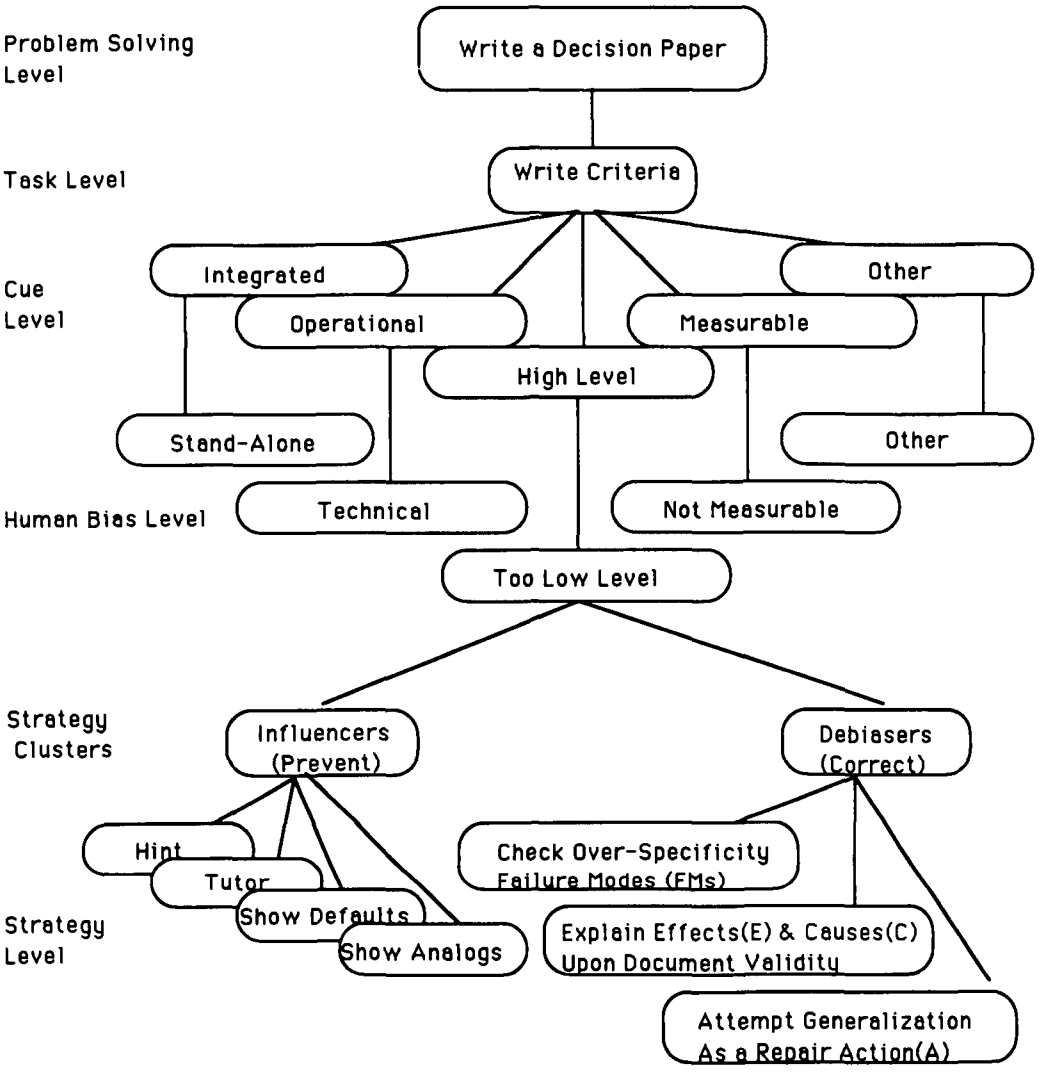


Figure 3 Cognitive task analysis diagram showing an actual set of biases and critiquing strategies for the criteria writing subtask

3.2.3 The value of critic programing language

The architecture of a critic was stated earlier to include a differential analyser, expertise module, dialogue generator, text files and interface. One would like to have a critiquing language that offers all these features. If the reader's application fits the features the language supports, then critic construction will be relatively easy. Rather than constructing a new critic program from scratch, the engineer only needs to learn a shell that already exists. One also may attempt extensions to the architecture of the language as a given application requires.

With these goals in mind, the author created a high level, knowledge based, frame and tree representation, critic language called COPE. COPE also supports graphs, procedural code, variable binding and iteration. It eases prototyping of critics, experimenting with critic features, and implementation of production-grade critics. Written entirely in the C language, the basic COPE language kernel runs on several platforms and is embeddable. It can also be used in standalone mode as a criticism based knowledge acquisition system. The windows system in most modules of the standalone version of COPE make use of a third party package that runs on most DOS and UNIX operating systems.

The critic engineer programs three separate tools when creating a critic in COPE.

(1) There is a question-asking and critiquing system in which one instantiates the influencer, debiaser, tutoring, and director strategies for a given domain. This tool lets the programmer use hypertext and rule tree/graph formalisms to communicate knowledge to the user.
(2) There is a graphical knowledge level editor that can show the user a visual tree (graphs and drawings are being added) of the knowledge he is transferring. The programmer edits this module to colour cue the expert directly manipulate the work artifact being created in the domain.
(3) There is an analogical reasoning module that learns from experience and that suggests similar, past cut-and-paste solutions for new problems.

The programmer must prepare a dictionary and interface software for each new domain to which the analogical reasoning module will apply. In what follows, discussion illustrates the types of interactions found in using tools (1) and (3) only. Silverman (1990) describes Tools 1 and 2 more fully, while Tool 3 appears in Staff (1989, 1990).

Tool 2 has some strict limitations that are worth explaining before going on to the next section. In particular, the user is constrained primarily to menu or push button selections when answering questions. This permits the avoidance of natural language dialogues, yet, it implies substantial knowledge engineering work for the critic engineer. That is, every question the critic asks the user must have canned answers. Every canned answer must branch to a potential error trigger(s). Every trigger, in turn, may fire strategies that ask still more questions. So, the cycle repeats until the task is completed. Tool 2 does permit free text answers, usually only after the critic has exhausted what it has to say in a given segment of the dialogue. To help the programmer process free text answers, there is a bank of string match functions. He can program these functions into a rule tree to check for evidence of undesirable words or ideas in the free text answers.

To summarize, a critic-tailored computer language can help the programmer do some things that would otherwise be prohibitive in terms of effort and cost. It also constrains the critic engineer to identify the full dialogue that will result between his critic and the potential users. Future researchers may overcome this obstacle, but for now there is no simple solution. Using another programing language will not help either.

3.2.4 How the library of triggers and strategies form a decision network of critics

Describing strategies in the abstract leaves the critic engineer with too many unanswered implementation questions. On the one hand, some designers might be asking how to instantiate the library of triggers and strategies for a given domain. On the other hand, other readers might see too many instantiation possibilities. They might think that a prodigious amount of work is needed and no one could afford to program such a system. In fact, neither view is correct.

This section attempts to put such questions to rest by explaining the programing implications of the trigger and strategy selections. The former group of readers will see some concrete examples from an actual working critic. They can map these into their own domain to help pinpoint ways to tackle errors. The latter group of readers will see the "shortcuts" and limits of what the critics can do. Critics are readily implemented and many successful applications exist to-date, e.g., see section 1.2. Simplifying assumptions and strict limits on their intelligence are an important element of their success to date.

What follows is an explanation of an actual user-computer dialog for one of the biases found when the user writes "criteria". The other Figure 3 biases are subjected to a similar set of strategies. However, space restrictions as well as readability concerns dictate that explaining one illustrative strategy suffice. Also due to space restrictions, the ensuing dialogue omits the interface, window, and hypertext features that help convey textual information to users.

The strategies of the lowest level of Figure 3 implement two top level strategies of Principle 1: (1) influence the user before she finalizes her answer (with hints, defaults, analogs, principles, examples, and/or repetition); (2) debias and direct the user after she gives an answer (with

FMECA critiquing or "credibility refinement" techniques). Discussion now proceeds through each of these strategies and the resulting types of dialogues possible.

- *Question asking as user hinting, cuing and attention focusing*—This is the simplest form of acquisition strategy and is prototypical of virtually all interviewing systems. For example, Boose's NeoETS asks "Would you like to add another trait". COPE asks questions relevant to the domain such as "would you like to add another criterion?" (see Figure 4 a). While easy to take for granted, question asking cues the users and helps focus their thoughts. Over the course

(a) *Question asking example: The prototypical knowledge acquisition system approach*

Would you like to specify a criterion measure for the SATCOM communication/ transmit category?

Yes
No
> **Yes**

NOTE: The user had earlier indicated the "communication/transmit" function as being critical. Now he is indicating he would like to author a criterion for that function.

(b) *Default logic is a cut-and-paste concept that also influences "well-formed" phrases*

Select the one default you would like to work with from the list of currently available defaults. You must select on this screen but you may reject it on the next screen if you wish. If no defaults appear below or none are satisfactory, inspect PRINCIPLE for guidance on what type of MOE is desired under the Communication: category.

Receive:
Receptions successfully completed
Time to detect jamming
Messages intercepted
Transmission rate

Transmit:
Time to transmit message
Messages effectively jammed
Message backlog

Survivability:
Probability of detecting a physical signature
Probability of detecting an electronic signature

ECCM:
Probability of countering enemy ECM
> **PRINCIPLE**
NOTE: The user types "principle".

(c) *Tutorial mode is an important way of influencing answers*

PRINCIPLE: A MOE generally is a quantitative expression of the degree to which a system meets its objectives, hence an analytical standard of comparison. It has been stated that in producing a MOE you (1) use your understanding of the Army to obtain an approximate measure of the system's effectiveness. (2) then adjusts this measure so that it becomes possible to relate it to the system's elements. (3) readjust the measure until it is satisfactory to the decision-maker, and (4) re-readjust the measure until the projected study does not exceed the time and effect deadlines. In order to develop a good approximate measure you need to know, among other things, (1) the Army or at least your school or center, (2) know the system that is under study, and (3) understand the decision maker(s).

NOTE: The user types the following:

> Successful transmissions in a dirty battlefield

Figure 4 Examples of selected critiquing strategies and user responses

(d) *Analog reasoning for display of reusable cases*
Would you like to inspect successful past communication related criteria for analogous effectiveness issues before editing a new one?

Yes
No

> **Yes**

Please select one analog that you would like to adapt as a criterion for the following issue: Does the SATCOM perform its mission in its intended environment? You will be given a chance to select another one after you adapt this one.

The SCOTT must demonstrate an Operational Availability (Ao) of at least .96 for communications segment in both peacetime and wartime operations.

Using standard I/O devices, the SCOTT must demonstrate a 90 percent probability that it can successfully handle four User Control Interface Devices in use simultaneously operating at data rates up to 2.4 kbs.

You have selected the following criterion as an analog. Please type in the version of it you would like to use with your issue: Does the SATCOM perform its mission in its intended environment? Criterion: Using standard I/O devices, the SCOTT must demonstrate a 90 percent probability that it can successfully handle four User Control Interface Devices in use simultaneously operating at data rates up to 2.4 kbs.

> Using standard I/O devices, SATCOM must demonstrate a 90 percent probability that it can successfully handle three User Control Interface Devices simultaneously operating at data rates up to 4 kbs.

Please select "other" and type in a unique 2 to 3 word name under which your analog criterion may be saved and displayed on the dendrite.

> UCID handling

(e) *Debiaser showing machine criticism and repair of user's error*
The idea of Critical Operational Issues and Criteria (COIC) is to summarize for decision makers at headquarters. The measures you've selected thus far are:

- Using standard I/O devices. SATCOM must demonstrate a 90 percent probability that it can successfully handle three User Control Interface Devices in use simultaneously operating at data rates up to 4 kbs.
- SATCOM will communicate through the atmospheric conditions produced by a high-altitude nuclear burst.
- Receptions successfully completed.
- Successful transmissions sent in a dirty battlefield.

The following are inappropriate:

- Receptions successfully completed
- Successful transmissions sent in a dirty battlefield.

for the following reason.

These MOEs are often too specific for a COI. A higher level composite MOE which incorporates these MOEs is preferable.
A more appropriate version is:

- Communications successfully completed.

Do you:

1. Accept as is
2. Edit further
3. Reject

> 2

Communications successfully completed in a dirty battlefield.

What 2 to 3 word name would you like this to be saved under and displayed on the dendrite?

> Receive/Transmit

Figure 4 *continued*

of the task, the cues given to the user are a vital part of the invalidity avoidance and robustness enhancement activity. As Figure 4 (a) shows, the user selects one of the legal answers.

- *Default logic*—Having said "yes" to the question about specifying a measure, the user is next shown a set of default measures that might be useful in this setting. Providing the user with sample default answers such as "number of transmissions successfully completed" (see Figure 4 b) actually serves two purposes. Most of us find it easier to copy and edit than to write from scratch. What we copy serves as an "anchor" that affects our final result. The second purpose of providing defaults is thus to bias the user with a positive starting point or anchor. Hundreds of defaults are stored in the system's memory. These are culled from the literature, interviews and other products of the critic engineering process. They are organized as lists in a database, and the critic binds a variable to them at the proper juncture in the dialogue. We are currently considering alternative ways to make these lists dynamic, that is, they should grow as users add new defaults via the free text option. Yet, the addition of new defaults needs to be constrained to only those acceptable to upper management.
- *Tutoring with principles, examples and references notecards*—Not seeing a default he likes or not understanding the thinking behind them, the user can inspect tutorial style information hierarchically stored in layers of PRINCIPLE, EXAMPLE and REFERENCE notecards. These notes are also invokable by the system for automated tutoring. The dominant strategy found in the computer assisted instruction literature is to explain background principles buttressed by examples of good and bad answers, along with critiques and references for further reading. This is a rather direct appeal to the user's ability to understand and learn what is being offered. In traditional computer assisted instruction, if the user keeps giving the wrong answer he is given hints and, ultimately, the correct answer. In critiquing, the machine doesn't know the precise answer but it can make use of tutoring in both passive (user inspect option) and active (machine hinting) modes. Principles, examples and references give the critic the ability to explain what it is looking for, and to help steer the user should that user need or want it, as is the case of the user of the dialogue in Figure 4 c. An area for future research is to alter the sequence of notecards shown to a user based on user skill level and history of what criticism that user has already seen. At present, the same tutoring sequence is shown to all users who trigger it.
- *Re-use of similar cases*—If the user wishes still further insight as to what the machine is looking for, he might next answer "yes" to the machine's offer to show him analogous information. As the user strays over a region of the problem space, similar past cases may have covered the same territory. COPE examines the current context, retrieves similar cases from episodic memory and suggests successful past decision paper shreds with keywords appropriate to the current problem dubbed in. As a strategy, analogy serves the same two purposes as defaults: cut-and-paste plus anchoring. Figure 4 (d) shows the terse mode of interaction in which the critic suggests a list of analogs for consideration. Using Tool 3, the programmers also created a verbose mode of interaction. In verbose mode, the user can browse through and alter the search indices, similarity weights, and analog suggestion list.
- *Debiasing via the failure mode, effects, causes and actions (FMECA) strategy set*—While the machine may not know the correct answer, in some cases it is able to detect what is clearly incorrect. For the Army example, the detailed defaults are stimuli for encouraging user thought but are not to be collectively included in the decision paper. The machine considers too large a default list selection as an over-specificity error (it notices this by string matching against the answer list rather than by checking the number of "menu" items selected since selection can lead to editing). Figure 4 (e) shows the user committed an over-specificity error of selecting both "receiving" and "transmitting" measures rather than of selecting a combined "communicating" measure. This is an example of how they get caught up in low level technical minutia. In attempting a corrective action, the critic first cues the user to combine his selections after which it suggests its own version as a default. Tutoring accompanies the machine suggestion (i.e., a principle notecard is printed as the "cause and effect" for the error) as does an acceptance confirmation and further editing screen. Other errors automatically "corrected" in the Army

example pertain to suboptimalities, infeasibilities, contradictions, ignoring important operational measures, and other deficiencies in that domain. The critic engineer must expend significant effort to think through all the possible errors likely to arise in this dialogue. The more creativity the designer invests here, the more debiasers there will be to program. There is a snowball effect here. That is, the first few debiasers tend to be hard to think of. Yet once you get in the swing of it, a great many other debiasers become apparent.

4 Discussion of results and conclusions

This paper introduces a five step critic engineering methodology. It covers a specific theory of question asking, a cognitive task analysis procedure for implementing the theory, a principle to guide application of the procedure, and a discussion of several implementation lessons learned. The remainder offers some final points about critic engineering.

4.1 Value and limits of the critic engineering methodology

The design methodology presented here can lead the critic engineer to improvements in the designs of critics in the following ways:

(1) Most existing critics use superficial rules about user errors. This article's methodology exploits a systematic framework for identifying user errors. This may lead designers to more well thought out critics.
(2) Most existing expert critics are uni-strategy oriented (i.e., they use after task, batch debiasing). One can use this methodology to design such critics. More often it will encourage the addition of a wider set of strategies. Decision networks can better handle user confusion than can single strategy critics.
(3) The library of critics includes adaptive strategies not used in today's critics. An example is the analogy and dynamic memory capability that causes the critic to grow more useful the more one uses it. Another example is the overlaying of strategies in the network that different types of users will trigger.

In summary, the methodology and principles of this paper can lead designers to improvements in the current state of the practice.

It is important that the guidance can lead the critic engineer to successful outcomes much of the time. Equally important is to know what the methodology can't lead to. A good designer should understand the limits of the critiquing paradigm.

A principal source of limitation lies in cue coverage. Real world domains are semi-structured and broad ranging. It is difficult to discover and capture all the cues and rules of even small portions of such domains. In general, the automated critic will only be a small part of the needed critic.

One must set realistic goals. Only in the narrowest of well-structured domains should one expect to eliminate the need for human critics. More often, the goal will be to eliminate those errors that are most repetitious, harmful, costly and/or easy to remove. This will free the human critics to concentrate on what errors remain.

For example, most grammar critics eliminate only about 25% of the grammar errors. Similarly, the Army critic removes first draft errors. It leaves the subtle errors to the Army chain of command to remove. One should never expect machine critics to completely eliminate the need for human critics. Yet, both are beneficial tools.

If priorities are poorly set, critics can be guilty of following the 80:20 rule. The newer designers in our own projects frequently place their effort on eliminating the 80% of errors that the average user most often commits. The last 20% of the errors tend to be the most costly to eliminate. Unfortunately, that 20% can be the difference in some users' opinion of the system. It can also reduce the tendency of critics to behave like blind automatons in various situations. There is no

simple fix for this design problem other than to invest more time and effort on the last 20%. It is this extra effort that will decide the ultimate usability of the system.

4.2 Critic engineering

The current generation of knowledge-based systems pursued the definition and use of models of specific task types. The same holds true for the design of critics. Critics use a library of bias identification triggers and standard critiquing strategies appropriate to specific task types. Applying that library, however, involves a "critic engineering" process that is markedly different from traditional knowledge engineering. The steps of critic engineering extend knowledge engineering in the directions needed to model the error commission and repair processes of domain experts. This paper offers extensions to generic task theory for the case where critic, rather than expert, systems are the design goal.

Acknowledgements

The author gratefully acknowledges the financial support of the U.S. Army/TRADOC/TED and U.S. Navy/SPAWAR, but points out that all opinions, data, and conclusions are not official government positions. Also, the domain data in the case study has been purposely falsified for security reasons. Finally, thanks go to Dr Michael L Donnell who originally suggested Figure 1 to our critic research group.

References

Bailley, D, 1991. *Developing a Rule Base for Identifying Human Judgment Bias in Information Acquisition Tasks*, Masters Thesis, Engineering Management Department, George Washington University, DC, April.

Berger, M, Coleman, L, Muilenburg, B and Rohrer, R, 1991. *Influencer Effects on Confirmation Bias*, Experiment Report, Institute for Artificial Intelligence, George Washington University, DC, April.

Bingham, WP, Datko, L, Jackson, G and Oh, S-H, 1990. *Establishing and Approving Quality Critical Operational Issues: An Application Using the COPE Shell*, Experiment Report, Institute for Artificial Intelligence, George Washington University, April.

Boose, J, 1984. "Personal construct theory and the transfer of human expertise". In: *Proceedings of the National Conference on AI*, Morgan Kaufman.

Bylander, T and Chandrasekaran, B, 1987. "Generic tasks for knowledge based reasoning: The right level of abstraction for knowledge acquisition" *Int. J. Man–Machine Stud.* **26** 231–243.

Creevy, L, Neal, R, Reichard, E and Turrentine, B, 1991. *Comparison of a Dynamic Critic to a Static Critic Using the Wason's 2 4 6 Problem*, Experiment Report, Institute for Artificial Intelligence, George Washington University, April.

Fischer, G, 1987. "A critic for Lisp". In: *Proceedings 10th International Joint Conference on Artificial Intelligence*, pp 177–184, Morgan Kaufman.

Gingrich, S, Lehner, G, Lepich, C, Powell, D and Vautier, J, 1990. *The Mission: A Group One Final Report*, Experiment Report, Institute for Artificial Intelligence, George Washington University, April.

Herron, T, Mackoy, R, Mohan, et al., 1990. *Critics for Expert Statisticians*, Experiment Report, Institute for Artificial Intelligence, George Washington University, December.

Langlotz, CP and Shortliffe, EH, 1983. "Adapting a consultation system to critique user plans" *Int. J. Man–Machine Stud.* **19** 479–496.

Miller PL, 1983. "ATTENDING: Critiquing a physician's management plan" *IEEE Trans. PAMI* **5** (5) September, 449–461.

Ratte, D, Crowe, C, Skarpness, P and Allen, T, 1991. *Watson's 2 4 6 Confirmation Bias*, Experiment Report, Institute for Artificial Intelligence, George Washington University, April.

Silverman, BG, 1990. "Critiquing expert judgment via knowledge acquisition systems" *AI Magazine* **11** (3) Fall, 60–79.

Silverman, BG, 1991a. "Expert critics: Operationalizing the judgment/decisionmaking literature as a theory of 'bugs' and repair strategies" *Knowledge Acquisition* **3** (2), June, 175–214.

Silverman, BG, 1991b. "Criticism based knowledge acquisition for document generation" *Innovative Applic. Artif. Intell.* pp 291–319, AAAI Press.

Silverman, BG, 1992a. *Critiquing Human Error: A Knowledge Based Human Computer Collaboration Approach*, Academic Press.

Silverman, BG, 1992b. "Human–computer collaboration" *Human–Computer Interaction* **7** (2) Summer, 165–196.

Silverman, BG, 1992c. "Modeling and critiquing the confirmation bias in human reasoning" *IEEE Trans. Systems. Man and Cybernetics*, September/October 973–983.

Silverman, BG, Donnell, ML and Bialley, D, 1992. *Toward the Implementation of Cognitive Bias Theory: A Methodology and a Rule Base*, Institute for AI Technology Report, Washington, DC.

Silverman, BG and Mezher, T, 1992. "Expert critics for engineering design applications" *AI Magazine* **13** (1) April, 45–62.

Spickelmier, RL and Newton, AR. 1988. "Critic: A knowledge-based program for critiquing circuit designs" *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp 324–327, IEEE Computer Society Press.

Staff, 1989. *Knowledge Engineers Guide to COPE*, Potomac: IntelliTek.

Staff, 1990. *COPE Users Guide*, Potomac: IntelliTek.

Zhou, H, Simkol, J and Silverman, BG, 1989. "Configuration assessment logics for electromagnetic effects reduction (CLEER) of equipment on naval ships" *Naval Engineers J.* **101** (3) May, 127–137.