

Expert Critics in Engineering Design: Lessons Learned and Research Needs

Barry G. Silverman and Toufic M. Mezher

Human error is an increasingly important and addressable concern in modern-day high-technology accidents. Avoidable human errors led to many famous accidents, including Bhopal, the space shuttle Challenger, Chernobyl, the Exxon Valdez, and Three Mile Island. Many hundreds of thousands of nonfamous accidents occur each year that are equally or more avoidable. Dramatic examples make the local headlines, such as car crashes, train and plane wrecks, and military-related operations mishaps. Less dramatic consequences happen even more frequently because of millions of mundane errors that appear daily in the products we use (for example, poorly designed cars), the processes we are affected by (for example, banking or health-care institutions), and the automation that surrounds us (for example, unfriendly computers that expect us to adapt to their interfaces). We get by because humans excel at coping.

High-technology accidents occur because the human mind is ill tuned to the large vol-

Criticism should not be querulous, and wasting, all knife and root puller, but guiding, instructive, inspiring, a South wind, not an East wind.

—Ralph Waldo Emerson

An engineer who creates a design needs to determine whether the design is free of errors that can lead to high manufacturing costs, tragic accidents because of design defects, low use because of poor product quality, and a host of other downstream concerns. The domain of engineering design is much harder than other domains, and errors are more likely to arise and remain undetected until it is too late to do something about them. One way to reduce these errors is to introduce the use of expert critics into the designer's electronic support environment.

Critics are a promising approach for organizing a next-generation design support environment (DSE). Unfortunately, expert-critiquing theory offers inaccurate but widely followed guidance for helping builders create usable critic programs. Existing critics rely on batch, after-task, debiasing of experts. This form of criticism turns out to be overly limited and too often leads to user frustration. As a step toward correcting this deficiency, this article presents lessons learned from following the incorrect theory along with guidance for a more robust approach to criticism system construction. Future research needs are also identified that should help builders realize the full potential of critics in engineering DSEs.

umes of fast-changing sensory data that one needs to process to safely operate the technology. In other articles, we address the errors that the operators make and some ideas for correcting them, for example, see Silverman (1992a, 1992b). However, their mistakes are often the least serious ones in a long chain of errors, starting with those introduced in the design, development, and maintenance processes. To put it another way, the technology is designed in such a way that operators can hardly be expected to avoid accidents. When the environment is stable, the human operator can reach safe decisions. When the environment becomes unstable, there is no time for the human to pro-

cess all the variables to avoid accidents. Most technology represents accidents waiting to happen.

For example, in the Challenger explosion, the shortcomings of the O-rings had been known for several years. However, designers left them in, and managers obscured the

The design task consists of a generate-test-refine-remember process.

potential for disaster. At Three Mile Island, the faulty display of sensor-valve readings, a plethora of maintenance and management errors, and a control panel design that flashed hundreds of alarms simultaneously all contributed to the disaster. Likewise, when the British fleet was sent to defend the Falkland Islands from the Argentines, the ship design forced the British to turn off their warning radars to clear the waves so they could use the radios to receive instructions. It was at this point that the Argentines released their missile and sank an unsuspecting British ship.

Each of these accidents was an *impossible accident* (Wagenaar and Groeneweg 1988) in the mind of the operator. The operator had no inkling of the ramifications of the system designs under the current operating conditions. Also, once the accident sequence begins, the operator has virtually no way out. The remarkable thing is that accidents don't happen more often.

An entire community of professionals contributes to these accidents. Silverman addressed the role of requirement writers and system testers (1983, 1985, 1991a) and maintainers (1992a) in contributing to the technology mistakes long before they happen. In this article, we discuss the contribution of designers to accidents and the prospects for intelligent computer-aided design (ICAD) to mitigate such problems. Specifically, we examine the naval ship design domain to avoid tragedies such as befell the British. However, this problem is not just specific to the British; it is also a top priority or "red flag" for the United States Navy (Fowler 1979). Electromagnetic interference problems are also increasingly evident on civilian automobiles, airplanes, and ships that cram telephones, radios, computers, radar devices, and other electromagnetically incompatible devices into close proximity. Finally, the implications of this domain are relevant to all engineering design applications that must factor any operational (or manufacturability, sales, or other downstream) concerns into the design step. (Here, *downstream* refers to issues that arise after a product is designed.)

Modern society is on the verge of an explosion of smart, forgiving systems that will dramatically improve all aspects of the safety and quality of life. Before this explosion can happen, however, we need more knowledge of, and theories about, two things. First, we need better models of human error. Oh, there are lots of possible explanations for any given error. However, there are few, precise, computer-implemented models that can predict where errors will occur and why. Second,

we need better models of the error surfaces that will show the steepest-ascent paths toward error correction. In other words, we need better theories of explanation. What feedback strategy (for example, story telling, first-principle lecturing) will most constructively correct the human error?

There are many accounts in the intelligent tutoring system literature of novice errors and bugs, novice skill development, and novice-expert differences. However, there are no models there or in the AI literature of errors that proficient performers, or experts, make, but it is the so-called experts' errors that are of concern in each of the examples mentioned. The errors result from proficient task performers practicing in a natural environment; they are not the result of students working in a classroom. New error and critiquing models need to capture and reflect this difference. Describing results to date in deriving and applying these models in engineering design is the purpose of this article.

The Design Process and a Taxonomy of Frequent Designer Mistakes

This section examines the first goal, that of explaining the designer's error processes. We begin by examining the design process and the cognitive difficulties it poses.

The Design Process

The design task consists of a generate-test-refine-remember process. The designer uses a variety of cognitive operators to generate a design, test it under various conditions, refine it until a stopping rule is reached, and then store the design as a prototype or analog to help start a new process for the next design task. The design process is sufficiently complex that a correct and complete design simply cannot be deduced from starting conditions or simulation model results. An iterative refinement occurs in which the design is heuristically arrived at over a number of life-cycle repetitions of the generate-test-refine-remember steps. For example, these steps are attempted to create a conceptual design; a technical design; and, finally, a fully operational design. In each life-cycle stage, a design is postulated, tested, and refined. This process is sometimes referred to as the *waterfall model* of the life cycle because the iterations of the process are like several cascades that ultimately combine into a robust design.

Adding the remember step transforms the waterfall model into a memory-intensive pro-

cess that more nearly mimics one prevailing view of the designer's cognition. In this view, which is adopted here, the generate step is aided by decomposing the problem into simpler subproblems that the designer has solved before. These previously solved subproblems are analogs that need to be merged and adapted for the needs of the current design task. The analogical reasoning process proceeds opportunistically with diverse agents proposing, testing, adjusting, and retracting alternative analogs, prototypes, and ideas. This process permits varying viewpoints and different disciplines to be accommodated.

Each of the generate-test-refine-remember steps uses cases and heuristics. However, they also rely on model-based reasoning. For one thing, the designer forms a hypothesized mental model of the design. He/she struggles to express this model in terms and representations that others can use and, in so doing, improves his/her own internal model. Design details rapidly outgrow the capacity of the human brain to store them, and often, they are represented as a computer model of the design. Once this representation activity begins, much effort goes into trying to get the computer's and human's models of the design to equate. The computer offers a vital cognitive extension to the human designer's abilities. It helps him/her to express or visualize his/her design (computer-aided design [CAD]), store its myriad details (databases), and analyze the impacts of his/her model of the design (numeric simulations). The effort proceeds at several levels. For example, at the syntactic level, the designer must master the language of the computer tools; at the semantic level, the designer must guarantee that first principles and heuristics are followed; and at the pragmatic level, the designer must assure that the various components of the design fit together and work in the real world. It is design failures at the pragmatic level that left the British ship vulnerable in the Falklands.

After reaching equality of the two design models, the human can exploit the computer's visualization and testing capabilities to help his/her refinement efforts. Also, once the computer acquires the designer's mental model of the design, it can facilitate future remembering steps. Finally, because the process is semiautomated, the insertion of critics is eased.

The cognitive process just described is too poorly understood and too broad in scope to construct an expert system capable of doing the entire design task. Fortunately, critics are not expected to perform the entire task on their own. They can be inserted to help any

or all of the generate-test-refine-remember process steps, waterfall life-cycle stages, or syntactic-semantic-pragmatic model levels. They exist solely to cue the domain expert and act as an intelligent prosthetic device that makes expert system technology palatable to the human expert. Just as the spelling and grammar checkers are unable to author a memo, so are the more sophisticated engineering design critics unable to draw a design. However, both types of critics are useful adjuncts to help the expert improve his/her performance on a minimal interference basis.

Interestingly, a critic can be deployed in an application with unknown goals. For example, one can use spelling or grammar checkers to help in word processing tasks where the critics don't know the purpose of the document. However, critics cannot be deployed where they don't know the proper cues of the subtasks of interest. Thus, there could be no spelling checker in the absence of a formalized language. Likewise, there can be no design critic in the absence of engineering discipline knowledge and heuristics.

Although the niche just described might be a necessary set of conditions for the expert-critiquing class of programs, it is insufficient for assuring that the critic is as effective as possible. Additional conditions exist that dictate how a given critic should interact with its users. It is also vital to identify the errors that designers frequently make. To support experts, we must design systems that recognize the corners they get stuck in, the pitfalls they fall prey to, and the obstacles they confront. That is, for critics to work, it is essential to predict the errors that will arise and the repairs (criticisms) that will be needed.

Taxonomy of Errors in Design Tasks

A generic model of human error processes useful for building critics appears in Silverman (1990, 1991b, 1992a) and is briefly summarized here. Specifically, there are two major types of errors that critics can help overcome: (1) In the realm of knowledge, critics inform the user with knowledge and constraints or criticize the knowledge the user has offered. (2) In the realm of judgment, they criticize the user's reasoning, judgment, and decision processes. These two error types correspond to the distinction between "missing concept" and "misconception." They also capture the difference between "knowledge base" and "inference engine," terms used in the expert system field. The *knowledge base* holds the domain insight. The *inference engine* performs domain-independent reasoning and manipu-

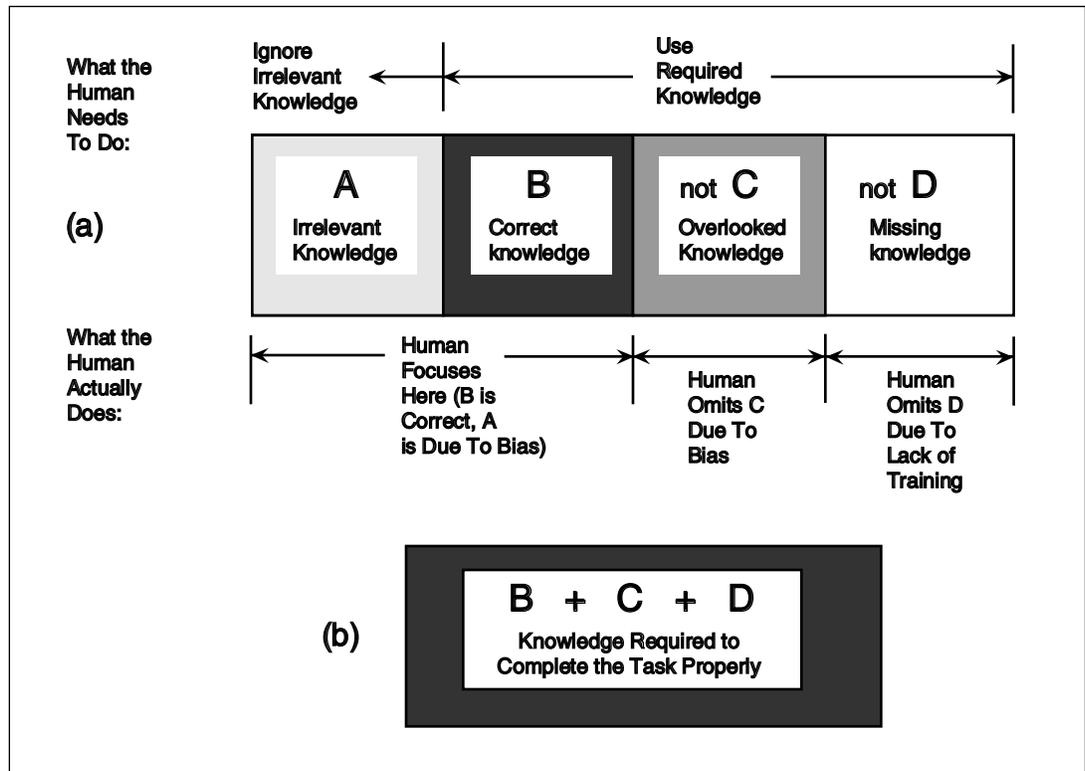


Figure 1. Categorizing a Human Expert's Knowledge. Human error is the result of misuse of various types of knowledge.

Figure 1a shows that the human needs to ignore irrelevant knowledge (A) and use required knowledge (B, C, and D). Instead, humans often focus on A and B (correct knowledge) alone. They overlook some of their normative knowledge (C) or are missing some of the other normative knowledge (D) needed to correctly complete the task. The expert critic must help humans to eliminate A and use overlooked (C) and missing (D) knowledge, as figure 1b shows.

lates the knowledge to solve problems.

To better grasp the differences between these alternative uses of critics, it is useful to visualize knowledge in the expert's mind as consisting of four principal categories. Figure 1 presents these categories. Figure 1a shows the human task practitioner to be focusing his/her attention on two categories of knowledge. He/she sees B, which is part of the normative and correct knowledge required to successfully complete the task. For various reasons of bias in judgment, the human also focuses on A, which is irrelevant to the task. In addition, the human often ignores two other categories of knowledge that are critical to successful task outcomes. The human ignores C, again for reasons of biased judgment. In D, the practitioner is unaware because of incomplete or out-of-date training.

Let us return to the analogy between the human and an expert system. Categories B, C, and D must all be in the expert's knowledge base to successfully complete the task. However, the human is only aware of knowl-

edge in categories A and B. Further, he/she has a biased judgment or inference engine. He/she omits C. Instead, he/she correctly infers B but incorrectly infers the need for A (irrelevant knowledge). He/she erroneously proceeds to match A and B to the task attributes. He/she then acts on any resulting conclusions. There is no hope that this domain expert can reach the correct task outcome no matter how well he/she reasons about A and B.

The problem of focusing on B rather than B and C is a *selective perception bias*. This bias is important. If one incorrectly defines the problem to begin with, no amount of processing or computation can compensate for the initial bias-induced error. It is critical that one consider the full set of decision dimensions or problem features. An effective problem solver misses few, if any, cues because of mental myopia or a training deficiency.

This categorization is not strictly concerned with what information the subject matter expert selects and focuses on. The categoriza-

MISCONCEPTIONS - Commissions (Category A) and Omissions (Category C)**ACCIDENTS/SLIPS/LAPSES**

Memory Lapses
Skill Slips

COGNITIVE BIASES

Availability Bias in Information Acquisition
Representativeness Bias in Information Processing
Confirmation Bias in Feedback Processing

MOTIVATIONAL BIASES

Corporate and Societal Culture
Need to Belong
Reward Systems

MISSING CONCEPTS - Category D Errors

Insufficient Training
Knowledge Decay/Half Life
Rotation to New Position/Promotion
Interdisciplinary Breadth of Engineering Domain

Table 1. Categories of Possible Designer Errors and Sample of Illustrative Causes.

There appear to be three reasons that humans add irrelevant knowledge and overlook normative cues: slips and lapses, cognitive biases, and motivational biases. Missing concept-related errors, in turn, appear to be caused by a variety of training-related reasons. Recognizing which of these causes leads to an error can help a critic do a better job of assisting humans in removing their errors. For example, missing concept-related causes suggest tutoring as a repair strategy. However, tutoring would be highly inappropriate if the error were the result of one of the other causes.

tion in figure 1 is also about how proficient humans combine and process knowledge to arrive at correctly completed task outcomes. That is, another commonly recurring weakness in practitioners is their inability to combine the more mundane information that they must routinely process.

Applying the Generic Taxonomy to Design Tasks

Just as there is no space to delineate the full model of human error processes, there is also insufficient space to explain its full instantiation in the design domain. Further, a complete model of errors in the design task requires a research undertaking that involves a significant and longitudinal descriptive field study to complete. Still, the model about to be presented offers useful guidance to critic designers in an operational sense.

At the highest level, the taxonomy of the previous subsection suggests that designers suffer from misconceptions and missing con-

cepts. Table 1 depicts these two error classes along with a sample of illustrative lower-level processes that contribute to these error classes. Misconceptions cover both the errors of commission (category A) with, and omission (category C) of, knowledge in the expert's knowledge base. Misconceptions can arise because of one of three principal reasons: accidents, cognitive biases, or motivations. *Accidents* are attentional slips or memory lapses that tend to be nonrepetitive but get by the designer "this time." As an example of an attentional slip, a ship designer who routinely specifies the cable shielding intends to double the shielding on all cables. He/she does so on the first cable and then forgets to repeat the adjustment when he/she writes out the additional cable specifications. A memory lapse example might be when a ship designer begins to specify the grounding paths for all circuits, he/she is interrupted by a computer crash. By the time he/she reboots his/her computer, he/she forgets that he/she has not completed grounding all the circuits. Slips

Cognitive biases...are tendencies the designer is likely to repeat.

and lapses occur in numerous fashions and are an important source of design errors. For a more thorough discussion of slips and lapses in general, see Reasons (1990).

Cognitive biases, unlike slips and lapses, are tendencies the designer is likely to repeat. Elsewhere, we explain a great many such tendencies and how critics can exploit them (Silverman 1990, 1991a, 1992a). Table 1 lists three such biases, one each for the availability, representativeness, and confirmation heuristics. These three biases are judgment heuristics that humans routinely use to save problem-solving time in the information collection, processing, and evaluation steps, respectively. Unfortunately, these heuristics are highly prone to misuse, as Kahneman, Slovic, and Tversky (1982) so aptly illustrate. As an example, a ship designer only considers design alternatives that he/she or a trusted contact has used before (the availability bias). He/she uses simple versions of Kirchoff's laws to test antenna locations for frequency incompatibilities (representativeness bias). Finally and paradoxically, he/she suffers no loss of confidence in his/her approach either because of its oversimplicity or deleterious after-action reports from the field (confirmation bias). Silverman (1985) reports an actual design case at the National Aeronautics and Space Administration, where almost this exact sequence of biases arises repeatedly in spacecraft system design processes.

The third cause of misconceptions, like cognitive biases, tends to be repetitive. Unlike cognitive biases, however, they might or might not be subconscious. Specifically, *motivational biases* tend to be goals and values the designer picks up from his/her environment and internalizes into his/her mental model of the design task. Japanese designers are not better suited biologically than Americans at designing higher-quality and more aesthetically pleasing automobiles, nor are they better able biologically to use computer-aided ship design tools. These differences result from environmental forces such as societal values and corporate culture. Motivational biases help the designer to conform, fit in, and belong. They can also cause the proficient designer to commit design errors when he/she is expected to shift mental models. For example, a Japanese car designer would tend to

design relatively unsafe cars by American standards. Likewise, a proficient FORTRAN programmer's Lisp code often looks (unsurprisingly) like FORTRAN. In the naval ship design business and many large project offices, the motivational biases imposed by management are often to always complete the task on time and on budget, even at the expense of creating tools and databases that would make the next design project quicker, cheaper, and better.

The other major error class given in table 1 is the missing concept-related error (category D). Engineering is such a multidisciplinary and broad endeavor that no individuals know how to design a complete system such as a ship. Each expert designer will always have edges to his/her expertise beyond which he/she is fallible or a non-expert. Teams of designers, each member of which has a different specialty, must cooperate to complete the task, which means that the design team manager doesn't truly know if the design is robust. Instead, he/she often develops a sense of trust in his/her designers' decisions based on whether his/her advice passes whatever testing is attempted and whether his/her designs appear to work in the field conditions that the finished systems have encountered to date. Add the five-year half-life of most engineering knowledge, and the potential grows quite high that designs will be fielded that are out of date and ill conceived. The loss of sailors' lives on the SS *Stark*, which burned for 18 to 20 hours after being struck by two Iraqi Exocet missiles, was compounded by the Navy's continued use of lightweight aluminum and Kevlar armor. This use continued despite the design lesson learned from the British Falklands experience that frigates should be built with more survivable, all-steel superstructures.

In summary, the list of classes and categories of errors in table 1 is the top level of a checklist that critic designers can use to determine what types of critics to design and deploy for a given design domain. Although this list is incomplete, it provides guidance and indicates places to look further. Over time, this list will hopefully be extended into a fully operational set of critic needs and trigger guidelines for the design domain.

The Role for Expert-Critiquing Systems in Design Support Environments

The standard tools used by today's designers include CAD packages to help generate designs, simulators for testing them, and databases and parts libraries for helping the

remember step. This collection of tools and others are referred to here as the *design support environment* (DSE). Figure 2 shows that the human gives design specifications to DSE and receives feedback in return. Both DSE and the human can adapt to each other as the process continues. For example, the human can alter his/her design because of criticism, and the critic can store a new design for later reuse.

Inside DSE are three layers: information, visualization and analysis, and synthesis and knowledge base. These layers are an attempt to make the machine's model of the design easier to specify and understand. Each layer is bisected by a dashed line to distinguish the generate-versus-test capabilities.

At the lowest or innermost DSE layer are the databases and models of the domain that permit the machine to generate, test, and remember design elements. It is a vital layer, without which the machine could not perform its share of the duties. The structure and validity of the databases and models determine what degree of support DSE offers. However, the information in this layer is almost illegible to most designers. This layer is also where the knowledge bases of the outermost layer are held.

The middle layer exists to help the designer visualize the machine's model and command its generate-and-test activities. It includes CAD and block-diagram packages to help generate the design specifications as well as graphics to display the results of the simulations. Also important here are the database and model base management systems that provide a fourth-generation language so that the designer can manipulate structures of the innermost layer, set up searches and tests, generate reports, and so on.

The outermost layer is rarely found in practice but represents the state-of-the-art ideas for further improving the design process. Included is a host of ideas for getting the machine to perform more as an intelligent design assistant. These items include ways to further insulate the designer from having to translate his/her model into the machine's innermost model, improved capabilities for testing and refining designs, and dynamic remembering abilities. No one yet has integrated all the various proposals for this layer, although many of the pieces about to be described can be seen, often in prototype form, in different vendor booths at the annual National Conference on AI. Also, interesting proposals for integration exist in Fischer (1989), Jakiela (1990), Kelly (1984), and Lemke (1989), among others.

Some of the more practical elements of the

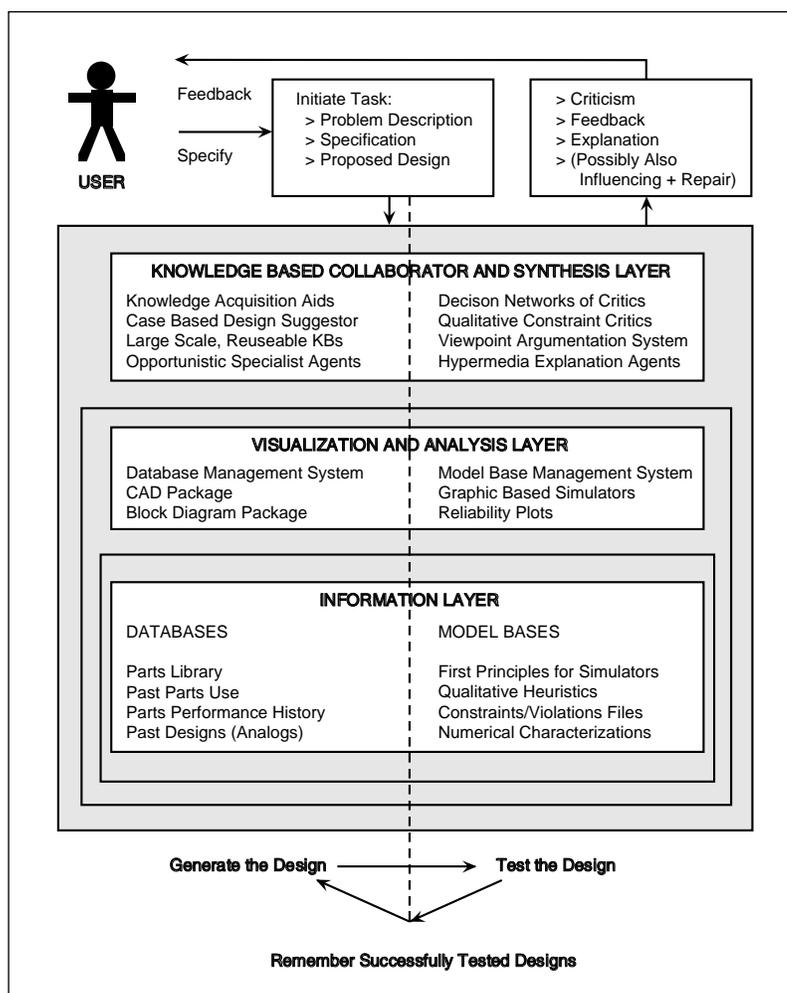


Figure 2. Layered Model of a Design Support Environment (DSE).

The standard tools used by today's designer include computer-aided design packages to help generate designs, simulators for testing them, and databases and parts libraries for helping the remember step. This collection of tools and others are referred to here as DSE. Inside DSE are three layers: information, visualization and analysis, and synthesis and knowledge base. These layers correspond to the attempt to make the machine's model of the design easier to specify and understand. Each layer is bisected by a dashed line to distinguish the generate-versus-test capabilities.

third layer are as follows: Knowledge-acquisition systems interview the designer for specifications and translate them into middle-layer entities such as block diagrams, fault and event trees, and simulation setup files (Boose 1989). This process can ease the tedium of the generation step, thus reducing some of the slips and lapses. Case-based design assistants alleviate availability biases and other remembering errors. They examine design problem characteristics and retrieve and adapt designs for similar past problems to fit the needs of the current or target problem (for example, see Murthy and Addanki [1987]). Expert systems help the designer decide which models

to use for the simulation, and they help set up, run, and display and explain the results (for example, see SACON in Hayes-Roth, Waterman, and Lenat [1983]). Large-scale, shareable, reusable engineering knowledge bases also might exist in several years that can assist in qualitative design simulations as well as provide a breadth to the machine's understanding of the designer's intentions (for example, see Forbus [1988]). Hypertext and hypermedia systems help explain both the DSE components and the results of its simulations to the curious designer. Finally, expert critics exist for specific engineering domains that check the design from heuristic viewpoints that simulators ignore or that save time in having to simulate to find errors that are already heuristically identifiable (for example, see Kelly [1985], Steele [1988], or Spickelmier and Newton [1988]).

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, but others attempt to prevent the user from committing some more common judgment errors in the first place. Critic programs can be user invoked (passive critic) or active critics. They can work in *batch mode* (process the entire solution after the user has finished it) or *incremental mode* (interrupt the user during his/her problem-solving task).

Expert-critiquing systems are ideal when the user can generate his/her solution 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. Thus, in engineering design applications, the user would fill in missing concepts and correct misconceptions.

CAD itself provides no feedback on the value of a given design despite the fact that commonly repeated poor designs are easy to store, match against, and feed back to the user to indicate his/her errors. Numeric simulations can help the user isolate poor design attributes. However, these simulations depend on the user specifying the precise input data that precipitate the poor design feature to show up or that cause the flaw that reveals the error. The user's input don't always cover the portion of the performance space in which a flaw appears. The simulator doesn't have the capability to heuristically notice the problem and warn the designer.

Also, simulators, design analyzers, and the like cover only small aspects of the design-testing and design-critiquing process. They leave it to the human to decide what the tool covers or omits. Thus, the design must rely on human judgment for major aspects of its coherence (completeness, consistency), clarity, and correspondence to pragmatics of real-world conditions.

Expert critics can fill these and other voids: (1) Critics compare the current design to a store of common erroneous designs. They report any errors to the user. (2) Critics check the current design for violations of good design rules, cues, and constraints that designers frequently tend to overlook or misuse. They can also run the simulation models to isolate and numerically explain analyzable errors. (3) Critics provide feedback and situated tutoring on syntactic-semantic-pragmatic specialties missing from the typical designer's training. (4) Critics have the potential to organize and give purpose to the many other ideas for the DSE knowledge-based layer. They can also decide when to invoke the other adjuvants, when to hide too much information from the user, how to explain qualitative and quantitative results, and so on.

This last point implies the critiquing process can and should be more than just another tool for the third layer of DSE. It can serve as an integrating paradigm for the deployment of this entire layer. By definition, criticism is a mutual communication and collaboration process in which both parties, not just the criticism recipient, engage in a search for truth or correctness. This collaborative search is the exact social process the designer desires from DSE.

Further, recent human factors results suggest that expert systems, unlike expert critics, are the wrong paradigm for this DSE layer. Although they are often found in support-the-expert roles, expert systems increasingly appear to have poor human factors when applied to decision-aiding settings (for example, see Langlotz and Shortliffe [1983]; Klein and Calderwood [1986]; and Roth, Bennett, and Woods [1988]). More of a joint collaborative orientation is warranted than expert system technology, alone, supplies. In fact, this concept was the motivation behind Miller's (1983) seminal work in researching a role for critics in medical decision making. Doctors, like designers, are not novices, and they interact poorly with expert systems. Expert critics are different from either expert systems or expert advisory systems. Both expert systems and expert advisory systems only accept the problem statement as input

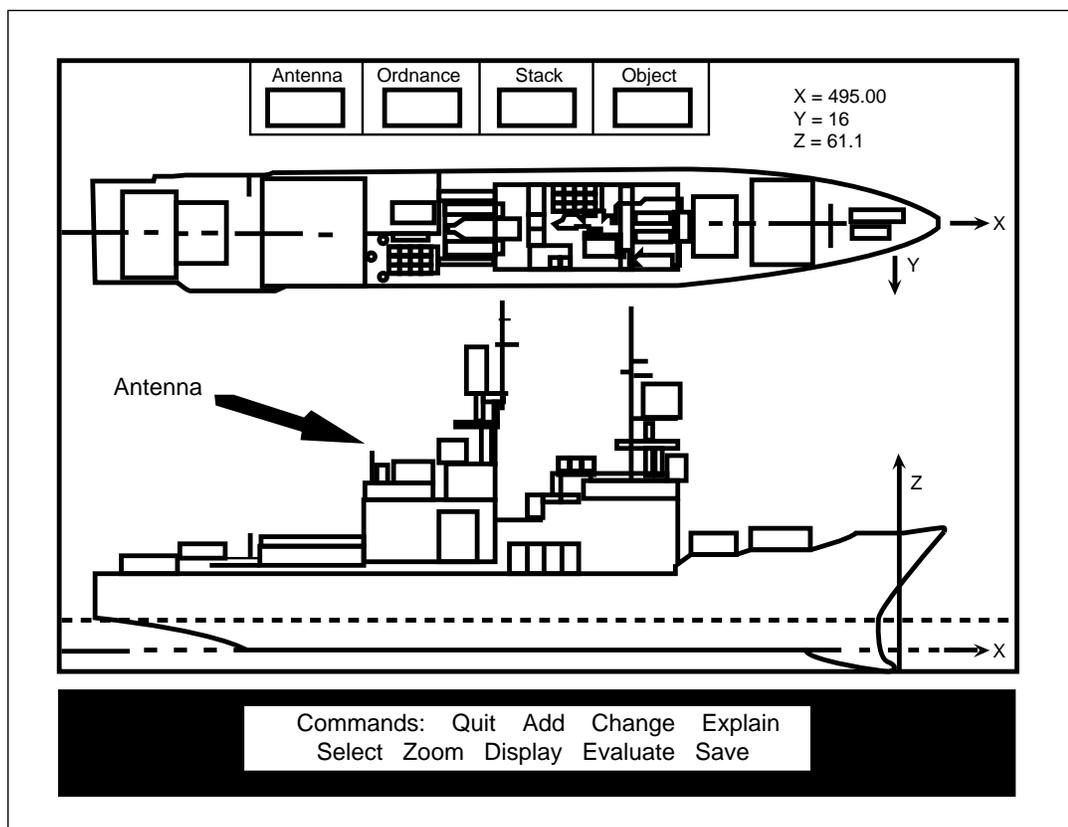


Figure 3. A Screen of the Computer-Aided Design (CAD) Software in Which the Critic Is Embedded.

The critic for the experiment was embedded in a shipboard equipment placement CAD package. The designer uses the mouse to inspect existing equipment items on the design or to zoom in and place new pieces of equipment, such as antennas, on the ship. Once he/she finalizes his/her design decisions and makes all the antenna placements, he/she toggles the “evaluate” button to invoke the critic.

and provide their machine-generated solution as output. Unlike the expert critic, expert systems and expert advisory systems do not evaluate the user-proposed solution. This difference suggests that expert critics are better suited for DSE applications.

An Example Critic to Test the Potential Benefits

With so much evidence pointing to the criticism benefits just enumerated, it was decided to conduct an investigation to see if they could be realized. However, how should the critic be constructed to reap the most benefit?

The classical critiquing systems, such as Miller (1983) and Langlotz and Shortliffe (1983), take the position that critiquing in batch, after-task mode is critical. This approach avoids interrupting, pestering, or distracting highly skilled domain experts. The publications cited here indicate that the critic should never interrupt prior to the expert committing a potential error. It's not clear to what

extent this position is critic builder judgment versus actual empirical result.

This conventional wisdom of how critiquing should work has reappeared in many critics built since those of Miller and Langlotz and Shortliffe. For example, spelling and grammar critics use few strategies and only after-task, passive debiasers. Similarly, Kelly (1984) and many subsequent engineering design critics adhere to this same wisdom (see Steele [1988], Spickelmier and Newton [1988], Jakiela [1990], Fischer [1989], and Lemke [1989]). Silverman's first foray into the critic construction process in 1988 followed this same guidance.

The example engineering design critic is a prototype that helps the ship designer who must make decisions about where to place antennas on the topsides and superstructure. Figure 3 shows that the users have a CAD package that supports a variety of typical features in the preparation of the drawing. It allows them to create objects or select from a library, drag objects around the drawing, and zoom in or change the view to work on fine

details. This CAD system is object oriented. The designer can interrogate each object on the drawing to find out its name, its location and dimension information, and so on. The designer can retrieve designs of existing ships and inspect or alter them in this package.

The problem that most designers find challenging is the electromagnetic interference that arises when they place antennas in a wrong location. Warships have many types of electronic systems (for example, radar, communication, computers) that can interfere with each other. For this reason, warships are often called *antenna farms*. The designer must add antennas to this crowded platform with care to avoid the electromagnetic interference problem. That is, he/she must avoid the invisible but real effects of electronic fields on components. For example, it is necessary to keep antennas more than 10 feet away from ordnance boxes. The antenna's field can cause ordnance to explode. They must also keep other constraints in mind that aren't outright violations but are preferred practices. For example, a good heuristic is to place all receive antennas on the stern and all transmit antennas on the bow.

CAD packages provide visualization support and permit the expression and communication of design ideas. However, they offer little or no constraint-violation information. Thus, in the example, the CAD package supports the designer in making parts placements on the ship drawing. However, it gives no feedback about poor locations when electromagnetic interference from other objects will prevent an antenna from working. As expected, CAD packages alone do little to overcome the accident prevention and other concurrent engineering problems that face most design applications.

To overcome these kinds of shortcomings, most designers turn to numeric simulation—so much so that CAD package vendors routinely include simulation tools in their library of design aids. For example, Valid Engineering, a vendor of a very large-scale integrated (VLSI) circuit CAD package also distributes a range of thermal and other property simulators (Yee 1989). Likewise, in the ship design domain, one finds commercial packages such as OMNISYS, a block diagram software for specifying the waveforms that are input to each block of the design; MICROWAVE SPICE, a circuit-level simulator that characterizes the electronic behavior of each element of the design; LINE-CALC, a simulator that computes transmission and propagation losses; and LIBRA, a nonlinear frequency simulator that computes the harmonic imbalances between components. The end

result of this set of simulations is a capability to compute the topological surfaces of the electromagnetic fields and the impact of designs on these fields. That is, these simulators offer the ship designer the ability to find out that the radios work best when the radar is off.

However, simulations also fail to be fully satisfying for design-testing purposes, as already mentioned. In this domain, the electromagnetic field surrounding each spot on the design is too complex to thoroughly and accurately model. Further, the databases that drive the simulators tend to have validity problems. As a result, finalized antenna locations can often only be determined by walking the deck and superstructure of the ship. This walk through the ship is done with a hand-held oscilloscope to empirically locate the minimal interference positions while other systems are illuminated and exercised. Finally, the designer finds himself/herself undertrained in the electromagnetic specialty. He/she regularly commits missing concept errors.

The goal of a critic here is to accumulate constraint and violation heuristics that it can use to help the designer avoid having to consider and test antenna locations that are clearly unacceptable. One could extend this process to encompass many of the other roles for critics that were mentioned previously. Still, these heuristics alone should lead to critiques that improve the accuracy of early draft designs, thus speeding the design process. They should also reduce the designer's missing concept errors and avoid the potential for misconception errors.

The critic created to assist in this domain and embedded in the CAD package is CLEER (Zhou, Simko1, and Silverman 1989). The input to CLEER is a ship design showing the location of all objects, including the antenna in question. Its algorithm uses spatial reasoning and checks all objects near the antenna against rules or heuristic constraints that the designer shouldn't violate. Its output is a log file of the violations and a list of the constraint compliances it finds for this antenna. In all, the CLEER prototype contains about four dozen constraints culled from a variety of sources. These sources are readily available to the designer. However, the designer frequently overlooks these sources when racing to meet design deadlines, probably because of a confirmation bias that leads him/her to seek only evidence that confirms the design is complete and error free.

The user invokes CLEER by clicking a mouse on a command icon that says "evaluate" (see bottom of figure 3). The feedback is a scrollable screen display of the log file of criticism

(violations) and praise (compliances) of the antenna of interest. Figure 4 lists an example output for a twin-whip transmit antenna located at the position shown in figure 3. The designer wrongly places the antenna within 10 feet of an ordnance object. He/she learns why this placement is undesirable, giving the designer insight into the source of his/her error and a reason to fix it. Further, CLEER provides praise for the things the designer does right (that is, constraints that are complied with). The praise is an effort to reinforce good design practice on the part of the user.

User Evaluation of the Prototype Critic

The goal here is to evaluate a prototype to find out if after-task criticism and praise improve user satisfaction and performance. We want to explore what seems to work or not work. The method involves collecting reactions from users, specifically, from four supervisors of warship antenna-placement designers. The supervisors happened to be experts in the electromagnetic interference domain. These supervisors used CLEER and shared their reactions during four separate sessions. The supervisors realized this effort was a research investigation. They not only shared reactions but also ideas about what might or might not work for the designers in their employ. They also provided feedback about the constraints and heuristics in CLEER's knowledge bases.

Because the objective is to research features that do and don't work, the criteria for the analysis is "any single complaint that a user voices." Most of the comments concerned domain details about the constraints and heuristics. These are of little general interest, and we omit them here. Overall, the comments were positive, and the supervisors found the critic approach well suited to cuing design experts. Still, several interesting comments emerged. The following concerns were voiced by all four users:

First, because there is a missing knowledge characteristic of many users, the critic should do more before-task tutoring and explaining about why a given decision is unacceptable. Don't just tell the user he/she is wrong, or he/she will get discouraged after several tries.

Second, the after-task debiasing is like a frustrating game of 20 questions. The user shouldn't have to keep guessing different designs until he/she happens on one that is violation free. The system should have the ability to colorize the parts of the ship that are off limits. It should cue the designer through direct interrogation of colorized regions about why the part is off limits. It

The type of antenna:

Twin-whip (10-30 mHz) transmit HF antenna

Location: x = 459 y = 16 z = 61.1

Conclusion: This location is infeasible because it failed the following constraint(s)

Constraint: Placement of large, high power HF antennas on deck at distances closer than 10 feet to ordnance will affect performance of the weapon and performance of antennas.

On this ship, "mk45modo-ordnance-object" was installed at "x = 491, y = 20, z = 41"

However, this installation did pass the following constraint(s):

Constraint: Antennas should not be located closer than 1 foot from other objects.

Constraint: Transmit antennas do not function well high off the water especially if their frequencies are on the lower portions of the HF band. Their radiation patterns begin to split or scallop in the elevation plane.

Other Reasons:

- HF range communications antennas are too heavy or too large for mast mounting
- Induction of electromagnetic energy from the desired radiator into nearby structures causes them to radiate the signal and, in effect, become another part of the antenna system. At heights above 85 feet, HF antennas cannot use the hull as part of the system.

Constraint: No antenna can be placed 35 feet below the base line.

Remark: This is the lowest level of the topside of this particular ship.

Constraint: HF antennas must be 50 feet away from other receive antennas.

Reasons:

- Separation of transmit antennas from receive antennas is necessary to prevent overload of the receive antenna and generation of intermodulation products within the receive antennas.

Figure 4. Log File of Criticism and Praise from a Sample Run of the Expert Antenna Critic.

The critic spatially analyzes the designer's antenna placement decisions against a number of constraint rules that shouldn't be violated. Any violations are dumped into a log file for printing back to the screen. The log file also includes the satisfied constraints as a form of praise for the designer. The example shown here is for a twin-whip antenna placed too close to an ordnance object.

should also actively suggest good locations in a prioritized order.

Third, the typical designer is unlikely to absorb or even read praise information. Omit this information.

Fourth, the prototype analogy module would help users see how antennas are placed on similar ships. This information should be displayed before the users make a placement decision on the current ship.

Discussion of Results

The prototype critic shows that the critiquing paradigm holds some promise for DSEs, but there are still shortcomings in the way it is implemented. Many of CLEER's features are similar to the configuration of other critics and to the conventional wisdom. For example, CLEER uses passive, after-task debiasing through batch log files of feedback. It is also inattentive to the needs of the intermediate-skilled practitioner. The results of this case study raise serious questions about previously held critic design beliefs. Hindsight is twenty-twenty. We could not easily predict these design flaws without building and evaluating the CLEER prototype. Now that we know them, let's examine their implications more closely.

Next Steps and Further Research Needs

The lesson learned from the experiment described in the previous subsection is that expert-critiquing theory offers little and, sometimes, incorrect help to the critic builder in deciding the parameters of the critic relative to the task area. There is a major deficiency in the conventional wisdom, leading to critics that have less than optimal characteristics. Grammar and style critic users increasingly complain about the stupidity of their critics. Continuing to field more critics without overcoming these challenges will lead to widespread disenchantment with the critiquing paradigm.

As a step toward correcting this deficiency, Silverman (1990, 1991a, 1992a) has built expert-critiquing systems for a variety of non-design task areas with a mind toward empirically deriving a more robust theory of criticism. To date, he has derived a set of guiding principles and added lessons learned that speed the building of useful and user-acceptable critics. These guiding elements are briefly described in Strategies for Criticism, after which we discuss their implications for the DSE domain. A finished theory of criticism with true predictive value for guiding critic builders is still a topic for future research, as addressed in Further Research Needs.

Strategies for Criticism

The results of the past two years of criticism research appear in Silverman (1992a, 1992b, 1991a, 1991b, 1990). What follows is a summary of two principles and several pieces of guidance that have empirical validity. Also included here is a synthesis and integration of many important intelligent system design touchstones known to be valuable in human-computer collaboration in general. The items in this subsection constitute a different approach to the critiquing paradigm, one that is more faithful to the idea of criticism as a collaborative process.

Three reasonable categories of criticism strategy are as follows:

First, influencers work before or, possibly, during a specific subtask. They use incremental mode with heuristic reasoning and offer a positive appeal to the user's intelligence before he/she commits a common error.

Second, debiasers self-activate incrementally (that is, right after each subtask) to see if the user has committed an error and, if so, to provide negative feedback. Because they are telling the user he/she erred, it is inherently negative feedback.

Third, *directors* are procedures that assist a user with the application of a cue rather than appeal positively or negatively to his/her intelligence for using it.

A critic generally includes only a few rules or knowledge chunks and, thus, is narrow and limited in scope. *Decision networks* of critics can compensate for this weakness; they are a cluster of active critics that use several influencing strategies (for example, hint, show defaults and analogs, tutor) followed by debiasers and directors. Debiasers and directors only appear if before-task critiquing fails to cause the user to apply the cue of interest. It is a decision network because there are multiple, active critics for a given major type of error, and the dialogue generation step must decide which critics to invoke when.

In addition to including one or more strategies, a decision network of critics should also exploit clarifiers. *Clarifiers* are techniques that incorporate insight about human perception and understanding to improve the delivery of a strategy. Thus, for example, when a strategy calls for presenting textual information to a user, a clarifier might include organizing the information hierarchically or through hypertext to hide what isn't initially needed but that might be probed after the initial criticism is absorbed. Another example would include the use of visualization aids such as screen icons, direct work-item manipulation, or colorization to make the intended point of

the strategy with even less textual material. The use of these and other clarifiers will improve the usability of the critics and the human factors of the strategies.

Finally, a decision network can make good use of reinforcers for those cases where the user misses the point of a given critic strategy. *Reinforcers* give a network a degree of endurance and perseverance that can be useful in instances where the first several strategies fail to remove a given bias or error from the user's behavior. Three common reinforcers are as follows: (1) *Repetition* restates the same criticism in a different way one or more times to help reach distracted or differently oriented users. (2) *Persuasion*, including cajoling by "carrot and stick" methods, can help change a recalcitrant user's perspective. Often, this change can be accomplished by presenting the cause of an error to motivate understanding of the criticism followed by the effect to show the user what he/she will encounter if an error goes unchanged. (3) *View argumentation*, in the form of textual explanations, explains why a given criticism is important. Often, the argumentation takes the form of presenting alternative viewpoints of the same problem.

In most of the critics Silverman designed, the error or missed cue is first prevented by an influencer. If this step fails, the error or missed cue is addressed by a debiaser as soon as possible, while the subtask is still fresh in the user's mind. Directors are reserved for helping the user get through overly complicated cue use sequences. Through the use of various combinations of influencers, debiasers, and directors, a reasonable critic can always be configured.

To summarize this discussion, it is vital that the critic builder not work in a vacuum. The builder must be aware of a library full of past critic results, critic theory, and relevant literature.

Principle 1: The critic builder should draw from a library of functions to create a decision network of active influencer, debiaser, and director strategies.

The function library should offer multiple corrective strategies relevant to the design domain. One can deploy these strategies in a partially redundant fashion to blanket the set of likely causes of unwanted cue use and normative cue nonuse. The following alternative strategies illustrate how the critics can be the overarching framework for DSE.

Influencer strategies help prevent errors. These strategies include hinting, cue showing, and attention focusing to minimize potential biases; showing design defaults and analogs

to provide good starting anchors; explaining engineering principles, examples, and references (that is, tutoring) to help anchoring and adjusting processes; repeating, often in different form, to be sure points get through; and visualizing the total direction that the task should take.

Debiaser strategies help correct a bias. These strategies include identifying normative cue use failure modes using any or all of the available heuristic and quantitative or qualitative modeling tools; explaining effect and cause to try to elevate the designer's awareness; attempting repair actions, such as specializing, generalizing, analogizing, substituting, adjusting, and attention refocusing to try to move the subject to a more productive pathway; using reinforcers, such as repetition, persuasion, or view argumentation; and using visual feedback, such as colorization and highlighting of imperfections in the task work artifact.

Directors are task-, cue-, and domain-specific procedures. They help users perform a task. One finds them inside the designs of reusable critics. For example, directors might include top-down assistants that perform knowledge acquisition, model setup, and other subtasks.

This listing includes many strategies, and few DSE users will need this much criticism in one session. One must craft triggers, so redundant strategies only interrupt the user if earlier strategies fail to remove the error. This approach serves to preserve the collaborative nature of the relationship, leading to a useful and conservative approach. Redundancy (and repetition) has a chance to operate on potentially stubborn errors.

Criticism Is a Two-Way Process. The effective critic needs a decision network of strategies because criticism is a process, not a one-time event. Criticism should include a true exchange of ideas and a search for truth. The recipient's response is another subjective viewpoint, not just a defense. It merits consideration by the critic.

Properly designed networks can accommodate alternative responses, corrections to criticisms, multiple reactions, and individual differences. In productive criticism, a joint cognitive relationship occurs, and collaboration processes prevail. Both the man and the machine contribute information that leads to better results, implying principle 2:

Principle 2: To promote effective criticism, there must be a mutual, two-way exchange of ideas. The critic must be flexible and adaptive.

To be flexible, the critic must back away

from its criticisms; persistently try alternative approaches; and, at appropriate times, accept that the user is correct. Decision networks are a move toward this flexibility. The critic also needs to be user oriented, able to say the right thing at the right time. The first step toward user-oriented criticism involves giving the critic simple user model techniques, including toggle switches the user can set to alter the type of criticism and simple capabilities to provide dialogues appropriate to different user types and skill levels.

To truly identify which, if any, of the errors in table 1 exist, it would be desirable to have a full-blown cognitive model of the user's intentions. Still, a simple behaviorist user model can suffice that alters its surface behavior in response to user input. For example, a good spelling checker learns new words that the user knows but that the program doesn't. Usually, the user only pushes a "learn word" button for the after-task debiaser to add a foreign word to its dictionary and go to the next spelling error. Over time, the critic becomes suited to the user, supporting the adaptivity feature of the critic.

To be adaptive, the critic must grow and learn from interacting with a given user. The more it is used, the more useful the critic should become. To achieve this goal, the critic needs a dynamic memory. It must remember new solutions the user gives it, so it won't repeat past errors. A spelling critic that learns new words or that, at least, learns to stop treating new words as errors is a trivial example of adaptiveness. A more powerful solution lies in the analogy strategy of the decision network. The analogy base grows as the user adds more successful designs. The effect of this and other adaptivity features is that the user slowly tailors the critic to his/her own needs the more he/she uses it. As this tailoring occurs, the collaboration grows.

The critic that can adapt itself in this fashion is not overly intelligent. It uses memorization rather than inductive-learning processes. However, this memorization captures the essence of the idea that criticism is a mutual-growth process. The user grows to like the debiasers even more. No proof is necessary for any reader who uses one of the better spelling critics with his/her word processor.

In this fashion, the critic can serve as the "corporate memory" for an organization. As the organization's members come, learn, perform, and leave a given job, the critic remains behind. The adaptive critic will manage the corporate memory as an asset. It will connect the current user to a string of vital experiences that he/she has yet to have personally.

A Revised Engineering Design Critic

Let us now consider how a CAD environment might exploit principles 1 and 2. This revision exploits and extends ideas pioneered by Fischer (1989). Specifically, we describe the critic's user interface, shown in figure 5. The antennas are for the specific ship shown in the two large windows toward the upper right. The designer scrolls through the upper left window to find the type of antenna to place on the ship. This window is a library of parts he/she can drag to the two ship views. He/she then uses the buttons across the base of the plan-view window to check what is already on the ship and find the location he/she desires. These three windows depict a classical style of direct-manipulation interface and capability for a CAD system. The capability of the system is primarily to support the representation of the drawing.

Critics introduce a layer of intelligence into the design environment because they cue the user to the violations and preferred-practice errors he/she commits. The goal with these cues is to be as nonintrusive and as helpful as possible. The direct-manipulation-style interface is an attractive criticism-delivery medium. Less skilled designers need textual criticisms as well. That is, principles 1 and 2 apply. The two types of influencers in figure 5 capture these concerns, as explained in the figure caption.

The colorized locations and analog pictures serve to subtly cue the more well-skilled designers. The analog influencers appear to the left side because English-speaking people read left to right. Users tend to see these influencers before they begin the task in the two windows to the right. The "show analogs" button allows the user to scroll the library holdings. The user clicks "update analogs" after a successful session to add the result to the library of analogs. The less skilled designers can inspect explanatory note cards by clicking the "check equipment" and "analog ?" buttons. Also, well-skilled designers click the "analog ?" button to read why a given location was chosen, helping them decide if the location should be reused in the current ship. Novices, in turn, won't know how to transform the analogous location to the current ship, but the debiasers can help ease them through the process.

The debiasers appear in the lowest window of the screen. This window displays during- and after-placement criticisms. These criticisms concern what is either a violation or just a suboptimization (preferred practice ignored). It is not essential for the user to

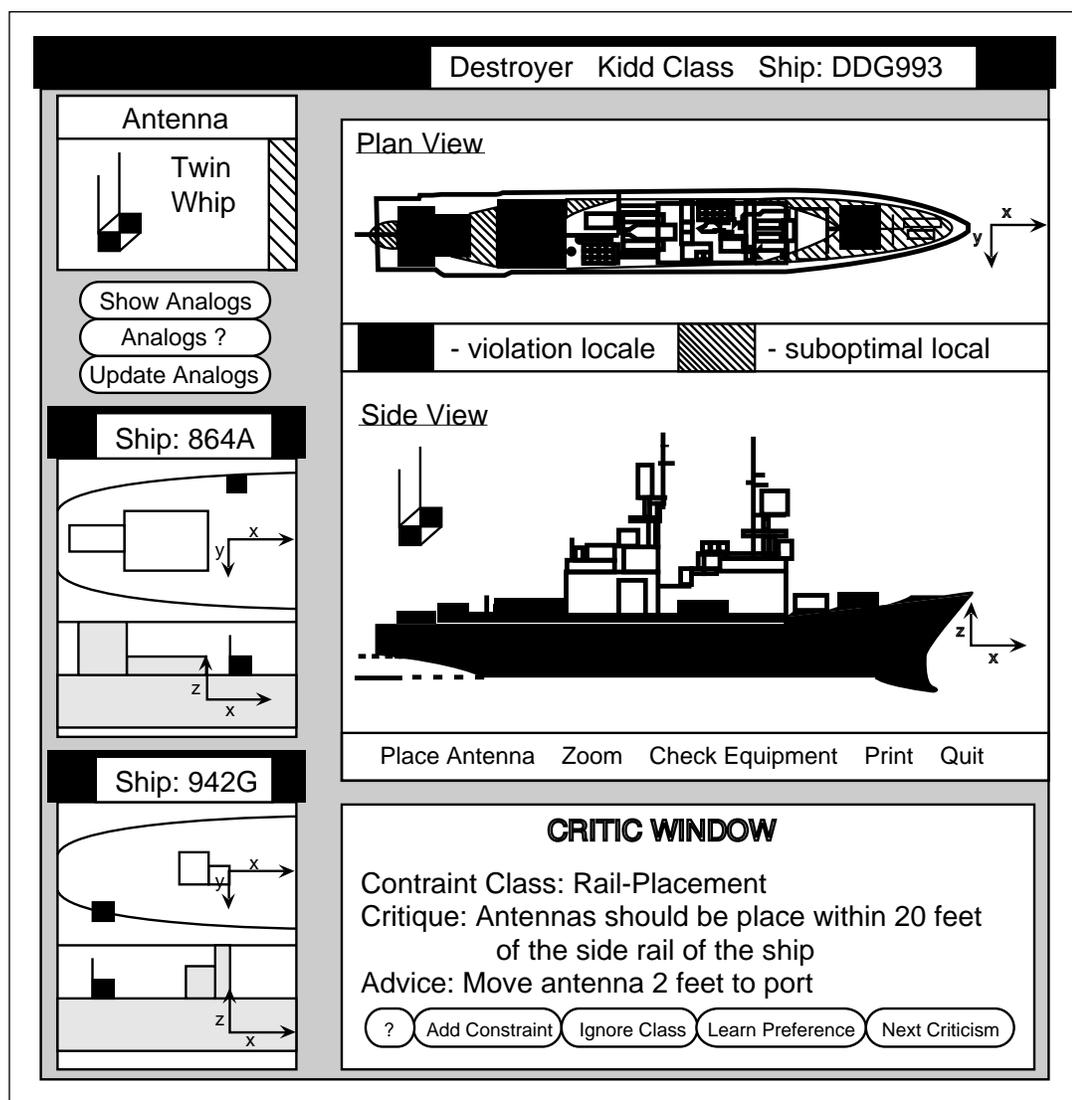


Figure 5. Multistrategy, Mixed-Initiative Critiquing Interface for a Computer-Aided Design Environment.

The designer selects an antenna by scrolling in the upper-left window. He/she drags the antenna of interest to the two ship windows. With the aid of the commands across the base of the plan-view window, he/she places the antenna on the ship. Influencers, in the form of colorizations of the ship, show constraint violations and suboptimal antenna placements before the user selects these locations. Another influencer is the catalog of analogs the user can inspect. By clicking "analog?" the user can see a note card indicating the precise location and the rationale for this choice. A stack of debiaser note cards, in turn, appears at the bottom of the screen when the user attempts to place the antenna in a colored area. The user can tell the critic to "ignore class" of error for the duration of this session or "learn preference" of designer for violating this cue in all sessions. Also, the user can obtain help by clicking on the "?" button or add a constraint the critic is unaware of for this ship. Finally, when the user is done, he/she can click on "update analogs" to add the new case to the library of analogs.

incorporate these latter cues. The user can adapt the debiasers to his/her perspective in four ways. The user can click "next criticism" to advance past a given problem. The user can click "ignore class" so that this problem class won't be brought up again during the rest of this session. The user can click "learn preference," and the user model updates the

file of what types of critics to show to this user. Finally, the designer might know something about this ship, this class of ships, or the current captain of this ship that can lead to an added violation or practice concern. The user can click "add constraint" to update the critic's knowledge base.

By now, the reader should see more fully

how to add visual criticism, decision networks, user flexibility, and adaptive memory processes to a critic program. Features such as these make the user more comfortable with the software the more he/she uses it. Also, the more the software is used, the smarter and more personalized it gets. This ability can be dangerous if an ineffective user adds erroneous designs to the analog base or incorrect constraints to the critic's knowledge base. However, judicious use of passwords to, and supervision of, the master knowledge and analog bases can minimize such difficulties. Overall, principle 2 leads to usability enhancement.

Further Research Needs

The revised critic described in the previous subsection is only a hypothesized improvement. Silverman has yet to conduct empirical studies of user reactions to, and productivity improvements from, its use. Still, based on valid principles of the criticism paradigm, this critic appears to be more promising than that described earlier. Future research is needed to bear this fact out and integrate the critic and the three DSE layers. For example, the revised critic does not currently access the numeric simulators or databases of the domain.

In engineering design, many disciplines and perspectives need critics. In naval ship design, the electromagnetic interference problem is only one of many that critics could help with. Electromagnetic interference is also probably the most important problem. Lives can be lost in battle if this problem is neglected. Once the electromagnetic interference critics are complete, one can start work on the others. In the long term, it is likely that whole divisions of various design firms will exist only to build and maintain a broad spectrum of ship design critics. The same future is likely for VLSI design firms, parts design firms, and so on.

Several aspects of conventional wisdom in building critics lead to much productive results in first-generation critic applications. To further advance the state of the art and practice, some of this conventional wisdom must be reconsidered, as this article has begun to do.

Don't interrupt the expert until he/she has erred: An important piece of conventional wisdom from the classical critic systems is to let the expert solve the problem on his/her own until he/she errs because interruptions are bothersome. Classical critic systems use batch, passive, after-task critiquing. The support for this belief is usually the builder's statement that his/her users prefer this approach. In contrast, the user results and reactions reported in Silverman's research,

plus principles 1 and 2, reflect a different situation. In Silverman (1991a), incremental, before-task knowledge critics allow novices to almost double their performance. Further, incremental, multiple-strategy (before- and after-task) critic networks lead competent users to near-perfect results. Trace files indicate only the before-task knowledge that critics activate. User reactions suggest they welcome the "interference" of the before-task critics. Also, in the engineering design domain, waiting until after an error to inject criticism leads to frustration and poor acceptance by the user community (see Discussion of Results). The point appears to be that before-task knowledge critics, if done as hierarchical text and in a visual fashion, do not bother experts. In addition, they assist experts by reducing their misconceptions. However, an expanded role also appears feasible. Critics that can assist a broad range of skill levels also seem prudent for the design domain where expertise is nonuniformly distributed and where missing concept errors also arise.

Criticism should be textual and visual: Direct-manipulation interfaces are potentially powerful and attractive. Critic builders should use them to deliver criticism in lieu of text wherever possible. However, this approach is not used by today's engineering design critics. Engineering design drawings are conducive to direct manipulation and visual criticism. For example, in the initial version of the ship design critic, criticism was textual and referred to the ship remotely. The revised critic used direct manipulation of the design to unobtrusively communicate criticism in the work artifact. Colorization of the ship can reveal violations without the need for text. Also, the display of analogs is another nontextual mode of providing before-task advice. In short, engineering applications have no excuse for perpetuating antiquated, text-only critiquing media. At a minimum, a mix of metaphor and textual initiatives usually leads to the most effective results.

User models are beneficial: At a minimum, a critic should include a behaviorist user model with a series of databases. These databases cover the user behavior history and preferences. This model also holds several simplistic dialogue and network toggle switches or triggers that shift the interaction from novice to intermediate to expert mode and back depending on what seems to work for a particular user. There might be a short inventory of demographic and skill-level questions that new users should answer. However, most of the user model's inferences depend on what seems to work over time

with each type of user: For example, what reusable strategies work in which generic tasks for what class of user? Which bias triggers seem most accurate? The machine can observe its users to collect much of these data. Builders can analyze such data to learn more about user differences. As they learn more, they can add more toggle switches to the critic design. In this fashion, the critic will gradually become more user sensitive, and we learn more about collaboration theory. Behaviorist user models are not hard to add to the critic, as our case study shows. They significantly improve the critic's acceptance. However, no critics prior to 1990 include user model features. It is time to change this practice.

Analogy for positive criticism and adaptivity: Analogy to prior concrete experience is one of the most commonly encountered forms of expert intuition in engineering design. Engineers analogize almost as often as they breathe. However, the critiquing field has overlooked the role that analogy can play as both influencer and adaptivity aid.

Analogical-based reasoning systems are now popular and successful. Many successful, stand-alone analogical, or case-based, reasoning systems will ultimately be built. Critic builders should not overlook the fruitful role that analogical reasoning can play when integrated as part of multistrategy, criticism-based systems. For example, in the criticism-based, problem-solving approach, the showing of analogs and defaults (a form of analogical reasoning comparable to extracting a model of norms from a cluster of analogs) extends the capabilities and robustness of the decision networks of critics. It also dynamically remembers the successful designs from past sessions. The analogical reasoning module can use this knowledge to offer the user a critic that becomes more domain knowledgeable the more that it is used. The critics, in turn, act to reduce analogical transfer and transformation errors by the human reasoner. The analogical and criticism-based approaches have much to offer each other. The same is true of criticism and other knowledge-acquisition techniques.

Concluding Remarks

First-generation expert-critiquing systems were powerful in their ability to flexibly react to a user-proposed solution in a batch, passive, after-task mode. As more results become available and as more precise evaluations occur, it becomes clear that alternative critic approaches have an equally important role.

Based on the idea that critics should help the expert to use more cues, some of these alternative approaches are beginning to focus on the following points: First, critics can help the expert before he/she psychologically commits to an erroneous solution. For example, one can include before-task influencers and situated tutors in a decision network that also contains batch, after-task critics. This approach is particularly important in engineering design applications where many of the errors result from missing knowledge (non-expert-style errors). Second, critics in the design domain already have a pictorial representation—the CAD drawing—that they can and must exploit. Delivering criticism visually and by direct manipulation avoids the problems of textual criticism and remote reference to the artifact. Third, there is the prospect of tailoring the critic to various characteristics of the human partner by integrating limited degrees of user modeling into the critic's architecture and giving the critic a dynamic memory that becomes more useful the more that it is used.

Engineering design enterprises are currently seeking smart extensions for CAD environments. Knowledge-rich critics are a starting place for filling this need. They reduce human error and increase the productivity of competent designers. Critics contribute to the development of better products. By building critics properly, they will become a successful contributor to smart CAD environments. They might even serve as the integrating paradigm for the entire DSE knowledge-based layer.

Acknowledgments

The authors want to thank Joel Simkol of the U.S. Navy and Thomas W. Reader of White Sands Missile Range for their support of this engineering design critic research. All findings, opinions, and conclusions are those of the authors alone. They do not represent the official position of any organization or group.

References

- Boose, J. 1989. A Survey of Knowledge-Acquisition Techniques and Tools. *Knowledge Acquisition* 1(4): 3–38.
- Fischer, G. 1989. Creativity Enhancing Design Environments. Presented at the First Conference on Modeling Creativity and Knowledge-Based Creative Design, October, Heron Island, Australia.
- Forbus, K. 1988. Intelligent Computer-Aided Engineering. *AI Magazine* 9(3): 23–36.
- Fowler, E., 1979. Navy Electromagnetic Interference Problems. Technical Memorandum for the Office of the Chief of Naval Operations, U.S. Navy, Crystal City, Virginia.

- Hayes-Roth, F.; Waterman, D.; and Lenat, D., eds. 1983. *Building Expert Systems*. Reading, Mass.: Addison-Wesley.
- Jakiela, M. 1990. Intelligent Suggestive CAD Systems. Ph.D. diss., Dept. of Mechanical Engineering, Univ. of Michigan.
- Kahneman, D.; Slovic, P.; and Tversky, A. 1982. *Judgment under Uncertainty: Heuristics and Biases*. Cambridge: Cambridge University Press.
- Kelly, V. 1984. The critter System: Automated Critiquing of Digital Circuit Designs. In Proceedings of the Twenty-First Design Automation Conference, 419–425. Washington D.C.: IEEE Computer Society.
- Klein, G., and Calderwood, R. 1986. Human Factors Considerations for Expert Systems. In Proceedings of the National Aerospace and Electronics Conference, volume 3, 921–925. Washington, D.C.: IEEE Computer Society.
- Langlotz, C., and Shortliffe, E. 1983. Adapting a Consultation System to Critique User Plans. *International Journal of Man-Machine Studies* 19:479–496.
- Lemke, A. 1989. Design Environments for High-Functionality Computer Systems, Ph.D. diss., Dept. of Computer Science, Univ. of Michigan.
- Miller, P. 1986. *Expert-Critiquing Systems: Practice-Based Medical Consultation by Computer*. New York: Springer-Verlag.
- Miller, P. 1983. Attending: Critiquing a Physician's Management Plan. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-5: 449–461.
- Murthy, S., and Addanki, S. 1987. PROMPT: An Innovative Design Tool. In Proceedings of the Sixth National Conference on Artificial Intelligence, 637–642. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Reasons, J. 1990. *Human Error*. New York: Cambridge University Press.
- Roth, E.; Bennett, K.; and Woods, D. 1988. Human Interaction with an Intelligent Machine. In *Cognitive Engineering in Complex Dynamic Worlds*, ed. E. Hollnagel, G. Mancini, and D. D. Woods, 23–70. New York: Academic.
- Silverman, B. G. 1992a. *Critiquing Human Error: A Knowledge-Based Human-Computer Collaboration Approach*. London: Academic. Forthcoming.
- Silverman, B. G. 1992b. Human-Computer Collaboration. *Human Computer Interaction* 7(2). Forthcoming.
- Silverman, B. G. 1991a. Criticism-Based Knowledge Acquisition for Document Generation. In *Innovative Applications of Artificial Intelligence*, volume 3, eds. R. Smith and C. Scott, 291–319. Menlo Park, Calif.: AAAI Press.
- Silverman, B. G. 1991b. Expert Critics: Operationalizing the Judgment/Decision-Making Literature as a Theory of Bugs and Repair Strategies. *Knowledge Acquisition* 3:175–214.
- Silverman, B. G. 1990. Critiquing Expert Judgment via Knowledge-Acquisition Systems. *AI Magazine* 11(3): 60–79.
- Silverman, B. G. 1985. Potential Software Cost and *Computer* 18(5): 86–96.
- Silverman, B. G. 1983. Analogy in Systems Management: A Theoretical Inquiry. *IEEE Transactions on Systems, Man, and Cybernetics* 13(6): 1049–1075.
- Spickelmier, R., and Newton, A. 1988. CRITIC: A Knowledge-Based Program for Critiquing Circuit Designs. In Proceedings of the 1988 IEEE International Conference on Computer Design: VLSI in Computers and Processors, 324–327. Washington D.C.: IEEE Computer Society.
- Steele, R. 1988. Cell-Based VLSI Design Advice Using Default Reasoning. In Proceedings of the Third Annual Rocky Mountain Conference on Artificial Intelligence, 66–74. Denver, Colo.: Rocky Mountain Society for Artificial Intelligence.
- Wagenaar, W., and Groeneweg, J. 1988. Accidents at Sea: Multiple Causes and Impossible Consequences. In *Cognitive Engineering in Complex, Dynamic Worlds*, eds. E. Hollnagel, G. Mancini, and D. D. Woods, 133–144. London: Academic.
- Yee, J. 1989. Managing Stress to Design Reliability. *Reliability Engineering and System Safety* 23:257–268.
- Zhou, H.; Simkol, J.; and Silverman, B. G. 1989. Configuration Assessment Logics for Electromagnetic Effects Reduction (CLEER) of Equipment on Naval Ships. *Naval Engineers Journal* 101(3): 12–137.



Barry G. Silverman is director of the Institute for Artificial Intelligence and professor of engineering management at George Washington University. He was a recipient of the 1991 Innovative Application of AI award from the American Association for Artificial Intelligence for his work on critiquing systems. Also, he has written on related topics for over 50 journal articles, 12 books or proceedings (including *Expert Systems for Business* and *Critiquing Human Error*), and 100 technical reports. He received a B.S. (1975) and an M.S. (1977) in engineering and a Ph.D. (also 1977) from the University of Pennsylvania, where he was a Sloan Foundation scholar. Silverman is a fellow of the Washington Academy of Science, is a fellow of the Institute of Electrical and Electronic Engineers, is a recipient of the Leo Schubert Award for excellence in college teaching, and serves as associate editor of *IEEE Expert* and AI area editor of *Interfaces* of the Operations Research Society and the Institute of Management Science.



Toufic M. Mezher is a Ph.D. candidate at George Washington University (GWU). He received his B.S. in civil engineering from the University of Florida in 1982 and an M.S. in engineering management from GWU in 1987.