# EXPLANATORY INTERFACE IN INTERACTIVE DESIGN ENVIRONMENTS

ASHOK GOEL, ANDRÉS GÓMEZ DE SILVA GARZA, NATHALIE GRUÉ,
J. WILLIAM MURDOCK AND MARGARET RECKER

*College of Computing*
*Georgia Institute of Technology*
*Atlanta, Georgia 30332, USA*

AND

T. GOVINDARAJ

*School of Industrial and Systems Engineering*
*Georgia Institute of Technology*
*Atlanta, Georgia 30332, USA*

Source: Fourth International Conference on Artificial Intelligence in Design, AID '96, Stanford, California, June 24 – 27, 1996. John S. Gero and Fay Sudweeks, eds. Boston: Kluwer Academic Publishers, 1996.

**Abstract.**

Explanation is an important issue in building computer-based interactive design environments in which a human designer and a knowledge system may cooperatively solve a design problem. We consider the two related problems of explaining the system's reasoning and the design generated by the system. In particular, we analyze the content of explanations of design reasoning and design solutions in the domain of physical devices. We describe two complementary languages: task-method-knowledge models for explaining design reasoning, and structure-behavior-function models for explaining device designs. INTERACTIVE KRITIK is a computer program that uses these representations to visually illustrate the system's reasoning and the result of a design episode. The explanation of design reasoning in INTERACTIVE KRITIK is in the context of the evolving design solution, and, similarly, the explanation of the design solution is in the context of the design reasoning.

## 1. Background, Motivations and Goals

Effective communication of both the design process and the design product is critical in collaborative design. Communicating the process of design and the evidence that the product satisfies its requirements can help build confidence in the design. When members of a design team work on different parts of a design problem, this kind of communication about one part of the problem can help in constraining other parts of the problem. In addition, explanation of design reasoning and its result can enable reuse of parts of the reasoning/result in subsequent design projects. Within the course of a design project, the explanation can enable reflection, support the detection of flaws, and suggest remedies for fixing them.

This is no less true of collaboration between a human designer and a knowledge system in the context of computer-based interactive design environments. When a human designer and a knowledge system are cooperatively addressing a design problem, the system must be able to explain to the designer precisely what it is doing, how and why. In addition, the system must be able to justify why the design solution it has proposed is acceptable for the given problem. Without this the user will have little confidence in the design and may be unable to detect potential flaws in it. Building usable interactive design environments thus requires both a theory of design explanations and the creation of explanatory interfaces.

The issue then becomes how may a knowledge system explain both its reasoning and the design solutions it proposes. This issue has several related but distinct facets pertaining to the content, generation, and presentation of explanations. To illustrate, let us consider the problem of explaining the design of a gyroscope. The explanation may specify how the design works, how its structure delivers its functions, how its design satisfies its requirements. Within a knowledge system, knowledge of the gyroscope's behaviors may be represented as a causal network, or generated at run-time from a representation of its design structure. To the user, the system may present the explanation in text form, or as graphics, or in some other modality such as animation. Our research on design explanations centers on the content of explanations presented to the user, and the content and representation of design knowledge and reasoning needed for generating the explanations.

The content of explanation and justification of design solutions, such as that of a gyroscope, depends both on the design phase and the design domain. For example, the explanation of the result of preliminary design is different from that of the result of configuration design: the former pertains to the function and structure of the design while the latter refers to its geometry. Similarly, the content of a justification for the design of gyroscope is different from that of an office building or a software interface. This is because the relationships between the function and the structure of the gyroscope design are fundamentally different from the function-structure relations in the design of an office building or a software interface. Our work focuses on the preliminary (conceptual, qualitative) design of physical devices such as electrical circuits, heat exchangers, and angular momentum controllers. The input to this task is a specification of the desired functions, and the output is a specification of a structure that can deliver the desired functions.

We are developing an interactive design and learning environment called INTERACTIVE KRITIK. When complete, INTERACTIVE KRITIK is intended to serve as an interactive constructive design environment. At present, when asked by a human user INTERACTIVE KRITIK can invoke a knowledge-based design system called KRITIK3 to address specific kinds of design problems. KRITIK3 evolves from KRITIK, which has been extensively described elsewhere (e.g., [Goel 1991, 1992; Goel and Chandrasekaran 1989, 1992].) INTERACTIVE KRITIK provides an explanatory interface to KRITIK3. In particular, it provides visual explanations and justifications of both KRITIK3's reasoning and the solutions it proposes. In addition, it enables the user to explore the system's design knowledge and also the design of the device generated by the system. A key feature of INTERACTIVE KRITIK is that explanation of the design reasoning is presented in the context of the evolving design solution, and, similarly, explanation of the design solution is presented in the context of the reasoning that led to it.

## 2. INTERACTIVE KRITIK

INTERACTIVE KRITIK's architecture consists of two agents: a design agent in the form of KRITIK3[1] and an interface agent[2]. Figure 1 illustrates INTERACTIVE KRITIK's architecture. The solid lines in the figure represent data flow while dotted lines represent control flow.

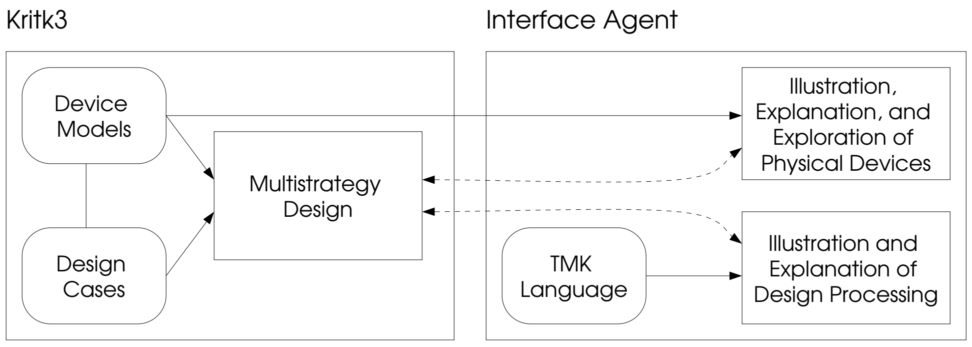

Kritk3
Interface Agent
Device Models
Design Cases
Multistrategy Design
Illustration, Explanation, and Exploration of Physical Devices
Illustration and Explanation of Design Processing
TMK Language

*Figure 1.* INTERACTIVE KRITIK's Architecture

[1]KRITIK3 runs under Common Lisp using CLOS.
[2]The interface is built using the Garnet tool [Myers and Zanden 1992].

## 2.1. STRUCTURE-BEHAVIOR-FUNCTION MODELS IN INTERACTIVE KRITIK

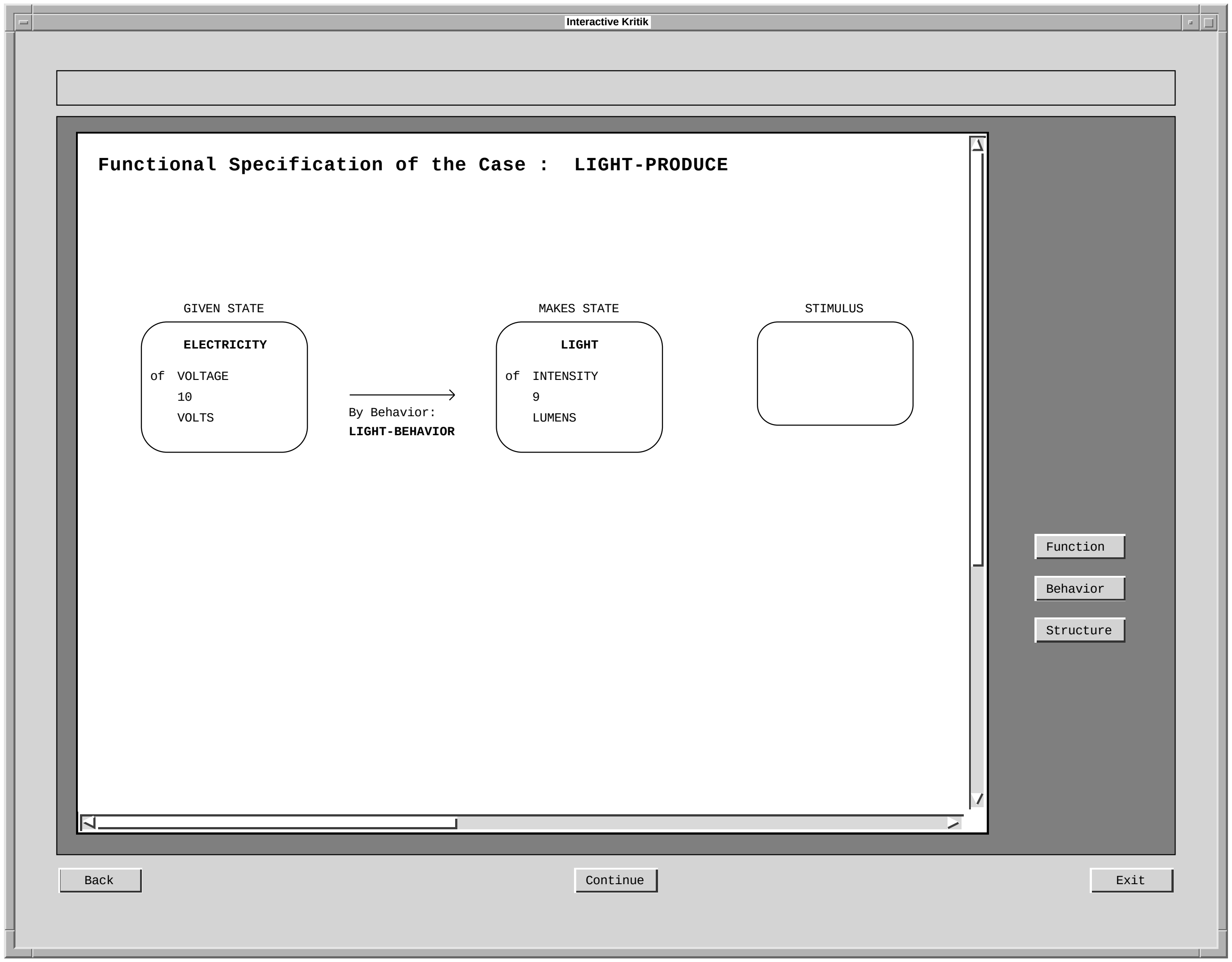
We use structure-behavior-function models (SBF models) [Chandrasekaran et al 1993; Goel 1991, 1992] for explaining and justifying designs of physical devices. The SBF model of a device provides a functional and causal explanation of how the device works, how its structure delivers its functions. This explanation makes explicit the functional and causal roles played by each structural element in the device design. Since KRITIK3 addresses the function-to-structure design task, and because the SBF model of a design created by the system explains how the proposed structure delivers the desired functions, the SBF model provides a justification for the design.

The SBF model of a device explicitly represents (i) the function(s) of the device, (ii) the structure of the device, and (iii) the internal causal behaviors of the device. The internal causal behaviors specify how the functions of the structural elements of the device are composed into the device functions. As a simple (almost trivial) example, let us consider the SBF model of an electrical circuit that produces light of intensity 9 lumens.

**Structure:** The structure of a device in the SBF language is expressed in terms of its constituent components and substances and the interactions between them. Components and substances can interact both *structurally* and *behaviorally*. For example, electricity can flow from battery to bulb only if they are structurally connected, and only if supported by the function allow electricity of switch that connects the battery and the bulb.

**Function:** The function of a device in the SBF language is represented as a schema that specifies the input behavioral state of the device, the behavioral state it produces as output, and a pointer to the internal causal behavior of the design that achieves this transformation. Both the input state and the output state are represented as *substance schemas*. The input state specifies that the substance electricity has the property *voltage* and the corresponding parameter, *10 volts*. The output state specifies the property *intensity* and the corresponding parameter, *9 lumens*, of a different substance, light. Finally, the slot by-behavior points to the causal behavior that achieves the function of producing light. Figure 2 illustrates INTERACTIVE KRITIK's visual representation of the function of the electrical circuit.

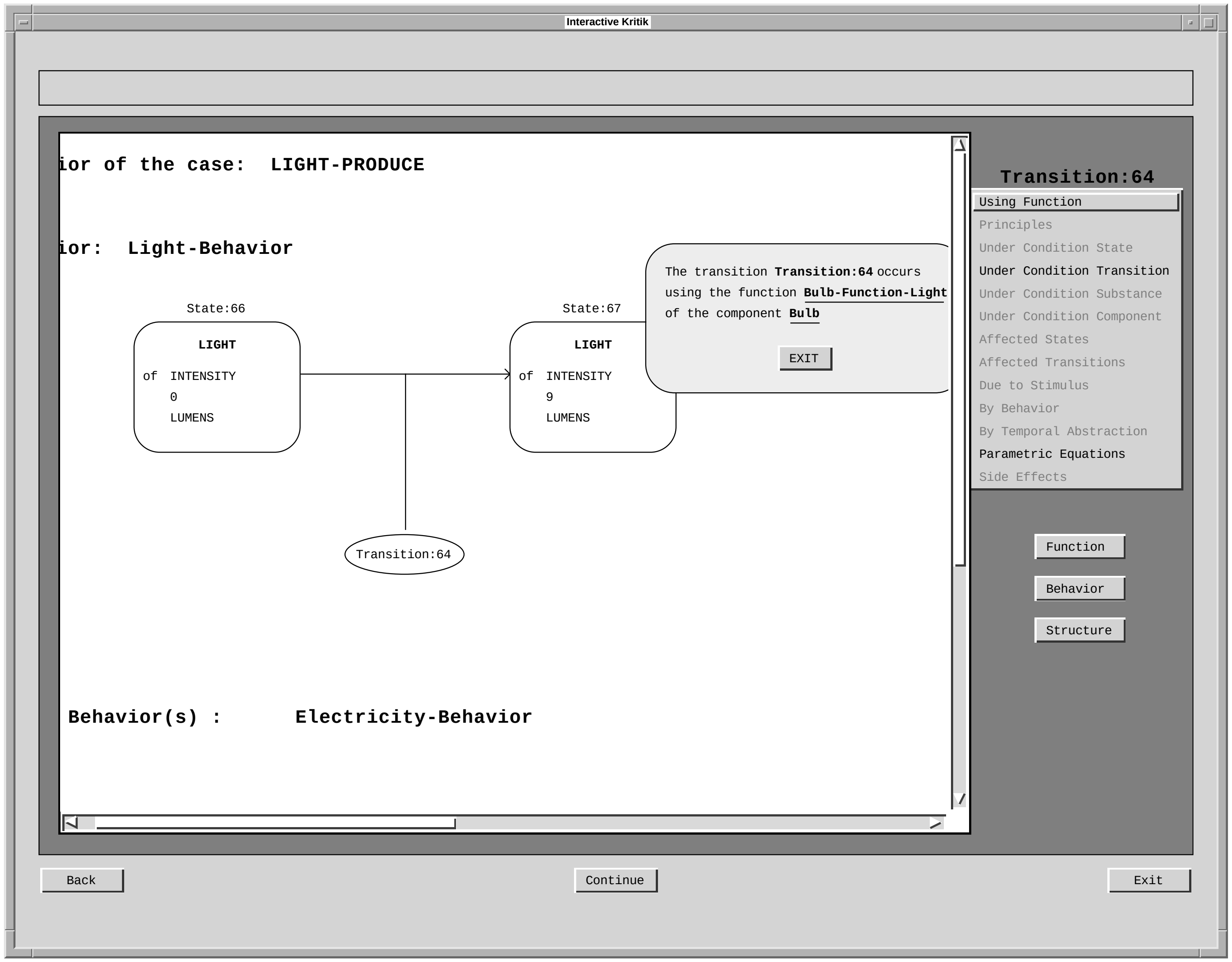
**Behavior:** The SBF model of a device also specifies the internal causal behaviors that compose the functions of device substructures into the functions of the device as a whole. In the SBF language, the internal causal behaviors of a device are represented as sequences of *transitions* between *behavioral states*. The annotations on the state transitions express the *causal*, *structural*, and *functional contexts* in which the state transitions occur and the state variables get transformed. The causal context specifies *causal relations* between the variables in preceding and succeeding states. The structural context specifies different *structural relations* among the components, the substances, and the different spatial locations of the



**Functional Specification of the Case : LIGHT-PRODUCE**

ELECTRICITY of VOLTAGE 10 VOLTS $\longrightarrow$ By Behavior: **LIGHT-BEHAVIOR** LIGHT of INTENSITY 9 LUMENS

*Figure 2.* The Function of an Electrical Circuit



ior of the case: **LIGHT-PRODUCE**

ior: **Light-Behavior**

The transition **Transition:64** occurs using the function **Bulb-Function-Light** of the component **Bulb**

Behavior(s) : **Electricity-Behavior**

*Figure 3.* A Behavioral Transition within an Electrical Circuit

device. The functional context indicates which functions of which components in the device are responsible for the transition. The behaviors are organized along the flow of specific substances through the device.

Figure 3 illustrates INTERACTIVE KRITIK's visual representation of *Light-Behavior*, the causal behavior that explains how light is generated. The state transition in this behavior has three annotations Using Function, Under Condition Transition, and Parametric Equation as indicated in the side bar on the top right of the figure. In the screen shot depicted, the description of one of these annotations, Using Function, is displayed in the pop-up dialog box in the right center of the figure. This description explains that the transition occurs due to the function *Bulb-Function-Light* component *Bulb*. Although not shown in Figure 3, the description for *Under Condition Transition* specifies that the transition is contingent on the flow of electricity through the bulb as detailed in a separate behavior labeled *Electricity-Behavior*. Similarly, the description for Parametric Equation specifies the specific equation relating the state variables.

The use of SBF models for explanation of designs is consistent with Simon's [1981] notion of functional explanations of artifacts. He has argued that explanations of artifacts pertain to, and are referenced by, the purpose of the artifact. This leads us to hypothesize that *SBF models capture the content of explanation of a device design at the "right" level of abstraction for comprehension by human designers.*

## 2.2. TASK-METHOD-KNOWLEDGE MODELS IN INTERACTIVE KRITIK

We use task-method-knowledge models (TMK models) [Chandrasekaran 1989, 1990; Goel and Chandrasekaran 1992] for explaining and justifying reasoning about a design problem. The TMK model provides a functional and strategic explanation of design reasoning in terms of the task, the methods used to accomplish the task, the subtasks spawned by the methods, and the knowledge used by the methods. Since subtasks are spawned by the methods available to the reasoner, the TMK model also provides a justification of specific tasks addressed by the reasoner in terms of the methods that spawn the tasks. Similarly, since methods serve tasks and are afforded by the available knowledge, the TMK model provides a justification of the use of specific methods by the reasoner in terms of the tasks being addressed and the knowledge that affords the methods.

The TMK model of design reasoning has three main elements. The first element, the task, is characterized by the types of information it takes as input and gives as output. KRITIK3 addresses the functions-to-structure design task in the domain of physical devices. This task takes as input a specification of the functions of the desired design. It has the goal of giving as output the specification of a structure that delivers the desired functions. The second element in the TMK model is the method. A method is characterized by (i) the type of knowledge it

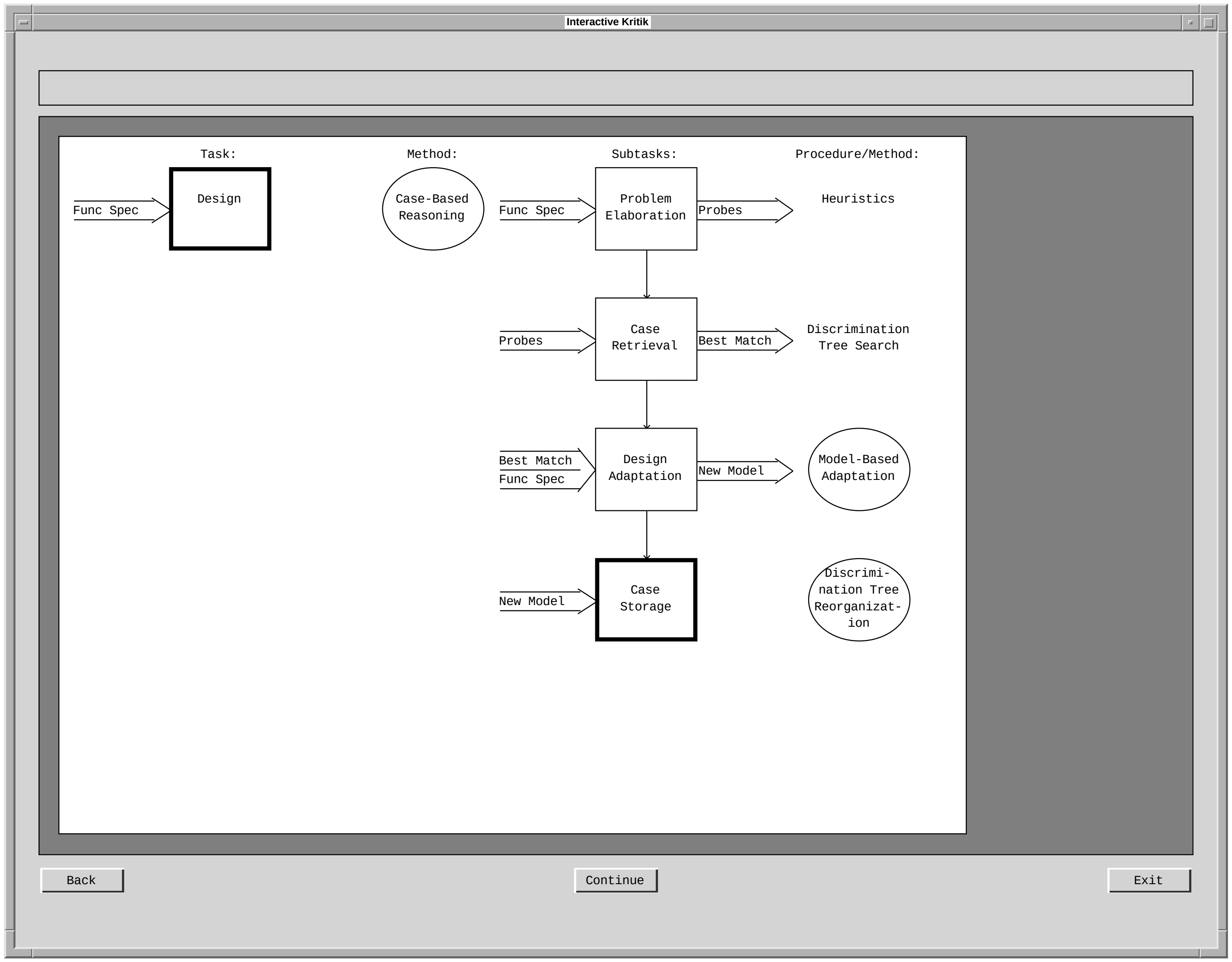


*Figure 4.* The Overall Design Task

uses, (ii) the subtasks (if any) it sets up, and (iii) the control it exercises over the processing of subtasks. KRITIK3 uses the method of case-based reasoning for addressing the function-to-structure design task. Figure 4 illustrates INTERACTIVE KRITIK's visual representation of this method. The figure shows that the method sets up the subtasks of problem elaboration, case retrieval, design adaptation, and case storage. It also shows the order in which these tasks are executed. In addition, it shows the input-output specification of these tasks. For example, the task of design adaptation takes as input the specification of the desired functions and the best matching case retrieved from the case memory. It gives as output an SBF model for a candidate design as indicated in Figure 4.

The third element in the TMK model is knowledge. A specific type of domain knowledge is characterized by its content, by its form of representation, and by its organization. Consider the example of diagnostic knowledge. In some domains, heuristic associations that directly map signs and symptoms into fault categories may be available. In a knowledge system, this associative knowledge might be represented in the form of production rules and organized as an unordered list. KRITIK3 contains two kinds of domain knowledge: past design cases and case-specific SBF models. We already have briefly described the representation and organization of the SBF models. Design cases are indexed by the functions delivered by the stored designs, and organized as leaf nodes of a discrimination tree.

The design of the KRITIK family of systems embodies a TMK model of of function-to-structure design of common physical devices [Goel and Chandrasekaran 1992]. We derived this model by analysis of the above task domain using the following methodology [Chandrasekaran 1989, 1990]:

**Task Identification:** First, the task is specified in terms of the generic types of information it takes as input and the generic types of information desired as its output.

**Knowledge Identification:** Next, the domain is analyzed in terms of the kinds of knowledge available in it.

**Method Identification:** Then, the different methods afforded by the different kinds of available knowledge are identified. This step also involves the identification of the subtasks that each method may set up.

**Method Selection:** Next, since more than one method may be feasible, the criteria for selecting a specific method is specified. These criteria may include factors such as properties of the desired solution and computational properties of the methods.

**Recursive Task-Domain Analysis:** Finally, the above steps are repeated for each of the subtasks that the selected method sets up.

This recursive decomposition of the given task continues up to an "elementary" level at which the domain affords knowledge that can "directly" map the input to the (sub)task into its desired output. At this level, no method is needed; instead, a procedure directly applies the relevant knowledge to solve the task. The recursive

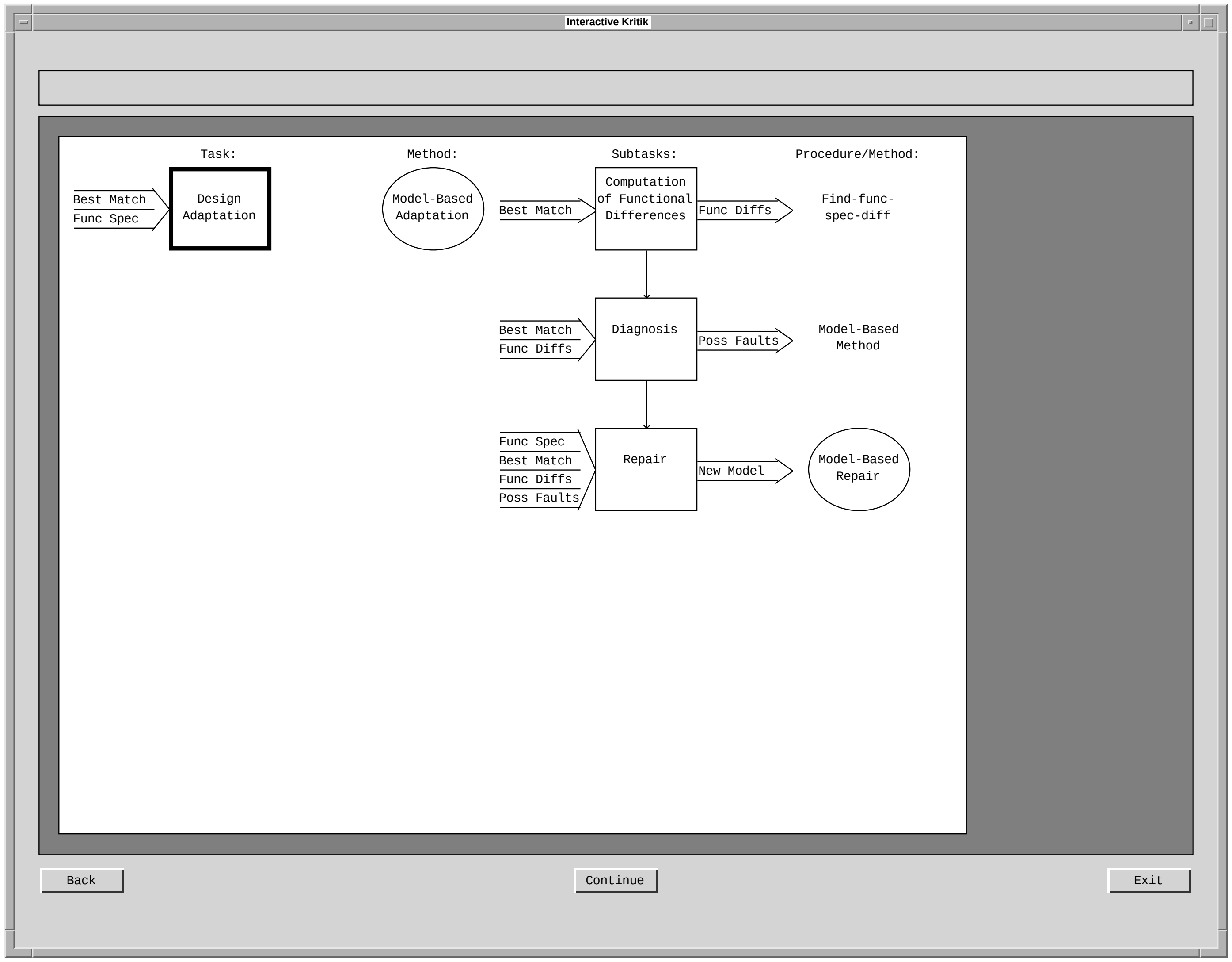


*Figure 5.* The Design Adaptation Task

task decomposition results in a task-method-subtask tree. For example, design adaptation is a subtask of the design task set up by the case-based method as illustrated in Figure 4. KRITIK3 uses a model-based method for addressing the task of adaptation as Figure 5 illustrates. The model-based method sets up its own subtasks of the design adaptation task. The first of these subtasks is the computation of differences between the desired function and the function delivered by the design retrieved from the case memory. KRITIK3 uses a simple pattern matching procedure for this task.

The TMK language for describing a knowledge system's reasoning is consistent with Marr's [1977] task-level and Newell's [1982] knowledge-level analyses of intelligent agents. Marr proposed that the reasoning of an intelligent agent can be analyzed at three levels. At the highest level is a specification of the tasks addressed and the mechanisms used by the agent. At the next level are the specific algorithms and data structures that the mechanism uses. At the lowest level is the architecture (or language) of implementation. Similarly, Newell proposed several levels of analysis of intelligent agents. The highest level in his scheme pertains to the agent's goals and the knowledge that enables the accomplishment of the goals. The next level concerns the symbolic structures that implement the mechanisms of the higher level. The next lower level specifies the physical devices that implement the symbolic structures, and so on. Marr suggested that the highest level in his scheme, the task-level, constituted the computational theory of the agent. Similarly, Newell suggested that the highest level in his scheme, the knowledge level, constituted the computational theory of the agent. This leads us to the hypothesis that *TMK models capture the content of explanation of design reasoning at the "right" level of abstraction for communication with human designers.*

## 3. The Explanatory Interface in INTERACTIVE KRITIK

The explanatory interface in INTERACTIVE KRITIK not only explains and justifies design reasoning and device designs, but also enables the user to explore the device designs and to reflect on the reasoning.

### 3.1. DESIGN EXPLANATION IN INTERACTIVE KRITIK

The interface agent in INTERACTIVE KRITIK has access to all the knowledge of KRITIK3 including its design cases and device models. It uses KRITIK3's SBF models of physical devices to graphically illustrate and explain the functioning of the devices to the users. It also graphically illustrates and explains the reasoning of the system in generating a new design. Within the context of a design episode, INTERACTIVE KRITIK provides graphical representations of both the designs retrieved from the case memory and the new designs created. Thus it provides representations of intermediate designs in addition to the final designs. The different design versions are presented as the design reasoning unfolds, i.e., in the context of the design subtask at hand.

The working of a device is illustrated to the user on several interrelated screens. One screen represents the device function; Figure 2 is an example of INTERACTIVE KRITIK's screen illustrating the function of an electrical circuit. The means by which the function of a device is achieved is explained by the internal causal behaviors in the SBF device model. Figure 3 shows an illustration by INTERACTIVE KRITIK of the main behavior, *Light-Behavior*, of the electrical circuit that produces light. A different screen shows the secondary behavior, *Electricity-Behavior*, of this device: the behavior of the electricity in this circuit.

KRITIK3's reasoning is illustrated on multiple screens identifying the tasks that the system performs while solving a problem and the methods it uses, as indicated in Figures 4 and 5. For each (sub)task, INTERACTIVE KRITIK illustrates the *reasoning state* both before and after the accomplishment of the (sub)task. The reasoning state specifies the task context and the method context. In addition, when appropriate, INTERACTIVE KRITIK illustrates the design knowledge available to KRITIK3. For example, in explaining the task of case retrieval, it graphically illustrates the case memory.

## 3.2. DESIGN EXPLORATION IN INTERACTIVE KRITIK

INTERACTIVE KRITIK enables the user to browse through different facets of a device design. This includes not only the final design proposed by KRITIK3 but also the intermediate designs it may have generated, for example, the design retrieved from the case memory. Exploration of a given design through browsing is enabled by the SBF model for the design.

As we explained in Section 2.1, different parts of an SBF model are closely interrelated. For example, the specification of a function in the SBF model acts as an index to the causal behaviors that accomplish the function. Also, the specifications of the state transitions in a causal behavior act as indices into the functional specifications of the structural components of the device. In addition, the description of a device component contains a specification of its functions, and points to the causal behaviors of the device in which the component plays a functional role. This indexing scheme enables the user to browse through the SBF model of the design.

The initial view of an SBF model is a representation of the device's functional specification, as in Figure 2. From here the user can push interface buttons to move among the functional, behavioral, and structural representations of the device. Additionally, the user can click on the name of the behavior by which the function is achieved (e.g., *Light-Behavior* in Figure 2) and "jump" directly to that behavior. Figure 3 illustrates the *Light-Behavior* screen. This screen presents *Light-Behavior* and labels all other behaviors (in this case, just the *Electricity-Behavior*) which the user can select to jump to a different behavior. When a user clicks on a particular transition a menu pops up allowing the user access to a variety of options relating to that transition, as indicated in Figure 3. This allows direct access to structural and behavioral information relating to that transition. For example, if the transition selected is dependent on another behavior, the user can jump directly to that behavior. The structure screen provides similar capabilities for looking at the components of a device and the connections between them.

## 3.3. DESIGN REFLECTION IN INTERACTIVE KRITIK

The explicit SBF representation of a design enables the user to inspect each element and aspect of the device design. Similarly, the explicit TMK representation of the trace of design reasoning enables the user to inspect each task, method, knowledge source, and reasoning state. This enables the user to reflect on the design reasoning. For example, the user can examine the TMK reasoning trace and detect potential flaws in it.

As we mentioned in Section 2.1, the SBF model of a device design not only explains how the device works but also justifies the design by showing how its structure delivers the desired functions. And as we mentioned in Section 2.2., the TMK model not only explains the reasoning of KRITIK3 but also justifies the tasks it sets up and the methods it uses. In addition, the user can also ask INTERACTIVE KRITIK for a justification for specific reasoning choices. As an example, consider the situation in which KRITIK3 retrieves a design case from its case memory. The TMK trace shows the user the probe KRITIK3 had prepared to retrieve a case and the case the system actually retrieved from its case memory. The user can now ask why did KRITIK3 retrieve this particular design case. Since the reasoning trace explicitly specifies the probe prepared by KRITIK3, and how the system's retrieval method probed the case memory - the branches it followed, the matches it made, and their results - the trace provides a justification for why the particular case best matches the given problem.

## 3.4. CRITIQUE

There is still a great deal of work to be done on INTERACTIVE KRITIK's user interface. Some issues which would need to be addressed before the system could be used as a practical tool include the improved display of the structure of a device, the building of better graphical representations, and provision of additional interaction capabilities. More importantly, INTERACTIVE KRITIK needs to be formally evaluated in a real world setting. But this kind of evaluation also requires additional work on the user interface.

## 4. Discussion

This research builds on earlier work on three topics at the intersection of AI and Design: design methods and process models, design knowledge and device models, and interactive design environments.

**Design Methods and Process Models:** A major goal of AI research on design has been to develop computational methods and process models for design. This has led to the development of several computational methods for design; examples include heuristic search [Stallman and Sussman 1977], heuristic association [McDermott 1982], and plan instantiation and expansion [Brown and Chandrasekaran 1989, Mittal, Dym and Morjaria 1986]. Recent research on case-based design (e.g., [Goel and Chandrasekaran 1992, Maher, Balachandran and Zhang 1995, Navinchandra 1991]) has led to the development of multi-strategy process models for design. KRITIK3 is a multi-strategy process model of design in two senses. First, while the high-level design process in KRITIK3 is case-based, the reasoning about individual subtasks in the case-based process is model-based. For example, KRITIK3 uses SBF device models for adapting a past design and for evaluating a candidate design. Second, design adaptation in KRITIK3 involves multiple modification methods. While all modification methods make use of SBF device models, different methods are applicable to different kinds of adaptation tasks.

A closely related research direction concerns the language for specifying the computational methods and process models for design. McDermott [1982] describes R1's method for configuration design in the language of constraints of a design problem, components available in the design domain, heuristic associations pertaining to the constraints and the components, and selection and activation of the associations. But this language is much too specific to R1's method. This method-specificness of the language becomes a major problem for describing and explaining multi-strategy process models such as KRITIK3.

Task-level [Marr 1977] (or, equivalently, knowledge-level [Newell 1982]) accounts make a clearer separation between knowledge-based reasoning and its implementation in a knowledge system. In the mid-eighties, Chandrasekaran [1988] proposed the language of Generic Tasks for analyzing and modeling knowledge-based problem solving, and showed that this language enables more perspicuous explanations [Chandrasekaran, Tanner, and Josephson 1989]. In the late eighties, Chandrasekaran [1990] related Generic Tasks with task structures: [Chandrasekaran 1989] describes a high-level task structure for design; [Goel and Chandrasekaran 1992] describe a fine-grained task structure for case-based design. In their work on the elevator design project called VT, McDermott and his colleagues [McDermott 1988, Marcus et al 1988] described a similar task-oriented language for analyzing knowledge-based design.

Our TMK models represent a generalization of task structures based on Generic Tasks. Also, our hypothesis that TMK models provide the "right" level of abstraction for explaining knowledge-based reasoning is based in part on earlier work on explanation in the Generic Task framework. But TMK models make the specific role played by a particular type of knowledge more explicit than earlier models. Consider, for example, the functional role of an SBF model of a past design in KRITIK3. Since the SBF model is associated with the past case, it affords a method for adapting the past design. The TMK model makes this affordance explicit. Thus, while task structures are useful for explaining the control of reasoning in terms of task-method interactions, TMK models are also useful for explaining knowledge-method interactions. In particular, they enable the explanation of the organization and indexing of different kinds of knowledge, the kinds of knowledge available for addressing a task, and the methods that become feasible because of the available knowledge.

**Design Knowledge and Device Models:** Explanation of physical devices has been a major topic of research not only in AI and in Design but also in Cognitive Engineering. AI research on device modeling and explanation can be traced as far back as Hayes [1979] work on "naive physics" in which he described a component-substance ontology. At about the same time, de Kleer developed the method of qualitative simulation for diagnosing electrical circuits [de Kleer 1984]. This work led to the no-function-in-structure principle [de Kleer and Brown 1984] which states that the behaviors of each structural component must be represented in a manner independent of their functional contexts.

In contrast, in the early eighties, Chandrasekaran and his colleagues developed the Functional Representation (FR) scheme [Sembugamorthy and Chandrasekaran 1986, Chandrasekaran et al 1993] in which the functions are not only represented explicitly, but also used to reference the causal behaviors responsible for their accomplishment. The causal behaviors in turn reference the functions of the device substructures. Since the function of a substructure refers to the causal behaviors that result in it, this gives rise to a hierarchical organization of the device model. Also in the mid-eighties, Bylander proposed a taxonomy of primitive behaviors [Bylander 1991] based in part on Hayes' component-substance ontology. He also described a method of composing the primitive behaviors into more complex behaviors. Our SBF models evolve from Chandrasekaran's Functional Representation scheme and Bylander's ontology of behaviors. In particular, they use FR's organizational scheme in which the device functions act as indices to the causal behaviors and the causal behaviors index the functions of device substructures. The specification of the functions, behaviors and structure in SBF models, however, is based on Bylander's well-defined behavioral ontology.

In Cognitive Engineering, Rasmussen [1985] proposed a hierarchical organization for presenting device knowledge to human users. His device models also specify the structure, the behaviors, and the functions at each level in the hierarchy. Our hypothesis that SBF models provide the "right" level of abstraction for explaining the working of a device to a human user is supported by

Rasmussen's empirical work. Govindaraj [1987] has used similar hierarchical organization schemes for enabling engineering students to explore the design of complex devices containing hundreds of components. Following his device models, the causal behaviors in our SBF models too are organized along the flow of specific substances in the device.

In Design research, [Gero et al 1991] and [Umeda et al 1990] have also described FBS models (for function-behavior-structure). While the details of the representation schemes differ, in both their FBS models and in our SBF models, behavior mediates between function and structure. Indeed, a major theme of our work on the KRITIK family of systems has been that while the design task takes a functional specification as input and gives a structural specification as output, much of the design reasoning is at the intermediate behavioral level.

**Interactive Design Environments:** A core issue in interactive design environments is how human designers and knowledge systems may share design responsibilities. AI research on interactive design environments covers a broad range of human/system responsibility sharing. At one extreme, the system acts as a knowledge source but leaves almost all reasoning to the human designer. Traditionally, knowledge bases for design have contained knowledge of design components and materials. But recent work on design knowledge bases has focused on providing human designers with access to libraries of design cases; examples include CADET [Sycara et al 1991], CADRE [Hua and Faltings 1992], CASECAD [Maher, Balachandran and Zhang 1995], FABEL [Voss et al 1994], Archie [Pearce et al 1992], AskJef [Barber et al 1992], and ArchieTutor [Goel et al 1993]. At the other extreme of this spectrum are autonomous knowledge systems that perform almost all design reasoning by themselves. Human interaction with these systems is limited to formulating design problems, supplying the problems to the system, and receiving the solutions generated by the system; examples include R1, AIR-CYL, and the original KRITIK system.

In between these two extremes lies a large range of potential sharing of responsibility between the system and the user. An important goal of design environments in the middle of this spectrum is to enable humans to construct new designs. Fischer et al's [1992] JANUS and Steinberg [1987] VEXED are two examples of constructive design environments. The goal of the INTERACTIVE KRITIK project is also the building of a constructive design environment. We will not describe here how, when completed, INTERACTIVE KRITIK may enable a human to construct new designs (but see [Grué 1994]). Instead, we focus the rest of this discussion on the issue of explanatory interface in the current version of INTERACTIVE KRITIK since this is already operational.

Mostow [1989] has argued that when a knowledge system in an interactive design environment proposes a solution to a design problem, then the system should also provide the human designer with an explanation of the reasoning that led to the solution. His BOGART system uses derivational analogy [Carbonell et al 1989] for generating solutions to design problems. Following the theory of derivational analogy, BOGART provides the human designer with an explanation of its reasoning in the form of a derivational record. The derivational record contains a trace of the system's reasoning in the language of design goals, operators, and heuristics for goal decomposition and operator selection.

We share the premise that in any interactive design environment, the knowledge system must be able to explain its reasoning. However, we believe that the language of goals, operators and heuristics is too low level to be accessible and comprehensible to human designers, especially novice designers. Instead, we hypothesize that the TMK language is at the "right" level of abstraction. More importantly, we believe that in addition to explaining its reasoning, the knowledge system must also be able to justify the design solution it proposes. INTERACTIVE KRITIK uses SBF models for justifying its design solutions.

Further, we believe that it is critical that the explanation of design reasoning should be grounded in the context of the evolving design solution, and, similarly, the explanation of the evolving design should be grounded in the context of the design reasoning that led to it. The advantages of situating design explanations in this way are two fold. First, situating the explanation of design reasoning in the context of the evolving design solution makes the explanation more meaningful. This is because the explanatory terms can now get their meaning from the specific parts of the design to which they refer. Second, situating the explanation of the design solution in the context of the design reasoning makes the explanation more complete because of the availability of previous versions in the evolution of the design solution.

## 4.1. CONCLUSIONS

Interactive design environments typically contain knowledge systems as major components. A human designer may use the interactive environment for design construction and experimentation. The knowledge systems may help automate specific and selected portions of this process, leading to human-system cooperative design. This raises the issues of usability and learnability of the knowledge systems. Human designers are unlikely to work with these systems if they cannot easily use them and also easily learn how to use them. Designers are more likely to use these systems if they can form a mental model of how the system works, how it reasons about problems, and if they can develop some confidence in the solutions generated by the system.

So the issue becomes how might a knowledge system enable the user to form a mental model of its reasoning, how might it explain its reasoning and justify its answers. Our work on INTERACTIVE KRITIK is based on three related ideas:

**1.** Explanations of a knowledge system need to capture the functional and strategic content of reasoning in addition to its knowledge content. Task-method-knowledge models enable this kind of task-level and knowledge-level explanation, which facilitates effective communication between the system and the user.

**2.** Explanations of physical systems need to capture the functionality and causality of the systems in addition to their structure. Structure-behavior-function models enable this kind of explanation at a level of abstraction that facilitates effective communication between the system and the user.

**3.** Explanation of design reasoning needs to be grounded in the context of the evolving design solution, and, similarly, the explanation of the evolving design needs to be grounded in the context of the design reasoning that led to it.

INTERACTIVE KRITIK demonstrates the computational feasibility of these ideas.

ACKNOWLEDGMENTS

Much of this research was done during 1993-94 when all the authors were with Georgia Institute of Technology in Atlanta, Georgia, USA. Andrés Gómez is now with the Key Centre of Design Computing, University of Sydney, Sydney, Australia; Nathalie Grué is now with the Institute for Learning Sciences, Northwestern University, Evanston, Illinois, USA; and Margaret Recker is now with Victoria University, Wellington, New Zealand. This work has benefited from contributions by Sambasiva Bhatta, Michael Donahoo, Vinay Pandey, and Eleni Stroulia. It has been funded in part by a grant from the Advanced Research Projects Agency and partly by internal seed grants from Georgia Tech's Educational Technology Institute, College of Computing, Cognitive Science Program, and Graphics, Visualization and Usability Center.

## References

J. Barber, M. Jacobson, L. Penberthy, R. Simpson, S. Bhatta, A. Goel, M. Pearce, M. Shankar, and E. Stroulia. Integrating Artificial Intelligence and Multimedia Technologies for Interface Design Advising. *NCR Journal of Research and Development*,6(1):75-85, October 1992.

D. Brown and B. Chandrasekaran. *Design Problem Solving: Knowledge Structures and Control Strategies*, Pitman, London, UK, 1989.

T. Bylander. A Theory of Consolidation for Reasoning about Devices. *Man-Machine Studies.* 35:467-489, 1991.

J. Carbonell, C. Knoblock and S. Minton. PRODIGY: An Integrated Architecture for Planning and Learning. *Architectures for Intelligence*, Van Lehn (editor), Lawrence Erlbau, 1989.

B. Chandrasekaran. Generic Tasks as Building Blocks for Knowledge-Based Systems: The Diagnosis and Routine Design Examples. *Knowledge Engineering Review*, 3(3):183-219, 1988.

B. Chandrasekaran. Task Structures, Knowledge Acquisition and Machine Learning. *Machine Learning*, 4:341-347.

B. Chandrasekaran. Design Problem Solving: A Task Analysis. *AI Magazine*, pp. 59-71, Winter 1990.

B. Chandrasekaran, M. Tanner, and J. Josephson. Explaining control strategies in problem solving. *IEEE Expert.* 4(1):9-24, 1989.

B. Chandrasekaran, A. Goel, and Y. Iwasaki. Functional Representation as Design Rationale. *IEEE Computer*, 48-56, January 1993.

J. de Kleer. How Circuits Work. *Artificial Intelligence*, 24:205-280, 1984.

J. de Kleer and J. Brown. A Qualitative Physics Based on Confluences. *Artificial Intelligence*, 24:7-83, 1984.

G. Fischer, J. Grudin, A. Lemke, R. McCall, J. Ostwald, B. Reeves and F. Shipman. Supporting Indirect Collaborative Design with Integrated Knowledge-Based Design Environment. *Human-*

*Computer Interactions*, 7(3):281-314, 1992.

J. Gero, H. Lee and K. Tham. Behavior: A Link Between Function and Structure in Design. *Proc. IFIP WG 5.2 Working Conference on Intelligent CAD*, Columbus, Ohio, pp. 201-230, September 1991.

A. Goel. A Model-based Approach to Case Adaptation. *Proc. Thirteenth Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, pp. 143-148, August 1991.

A. Goel. Representation of Design Functions in Experience-Based Design. *Intelligent Computer Aided Design*, D. Brown, M. Waldron and H. Yoshikawa (editors), North-Holland, pp. 283-308, 1992.

A. Goel and B. Chandrasekaran. Functional Representation of Designs and Redesign Problem Solving. *Proc. Eleventh International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, pp. 1388-1394, 1989.

A. Goel and B. Chandrasekaran. Case-Based Design: A Task Analysis. In *Artificial Intelligence Approaches to Engineering Design, Volume II: Innovative Design*, Tong and D. Sriram (editors), Academic Press, pp. 165-184, 1992.

A. Goel, M. Pearce, A. Malkawi and K. Liu. A Cross-Domain Experiment in Case-Based Design Support: ARCHIETUTOR. *Proc. AAAI Workshop on Case-Based Reasoning*, pp. 111-117, 1993.

T. Govindaraj. Qualitative Approximation Methodology for Modeling and Simulation of Large Dynamic Systems: Applications to a Marine Power Plant. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-17 No. 6, pp. 937-955, 1987.

N. Grué. Illustration, Explanation and Navigation of Physical Devices and Design Processes. M.S. Thesis, College of Computing, Georgia Institute of Technology, June 1994.

P. Hayes. Naive Physics Manifesto. *Expert Systems in the Microelectronics Age*, Edinburgh University Press, Edinbugh, UK, pp. 242-270, 1979.

K. Hua and B. Faltings. Exploring Case-Based Building Design - CADRE. *AI(EDAM)*, 7(2):135-143, 1993.

J. McDermott. R1: A Rule-Based Configurer of Computer Systems. *Artificial Intelligence*, 19:39-88, 1982.

J. McDermott. Preliminary Steps Towards a Taxonomy of Problem Solving Methods. *Automating Knowledge Acquisition for Expert Systems*, S. Marcus (editor), Kluwer, Boston, MA, 1988.

M.L. Maher, M.B. Balachandran, and D. Zhang. *Case-Based Reasoning in Design*, Erlbaum, Hillsdale, NJ, 1995.

S. Marcus, J. Stout, and J. McDermott. VT: An Expert Elevator Designer that Uses Knowledge-Based Backtracking. *AI Magazine*, 9(1):95-112, 1988.

D.Marr. Artificial Intelligence — A Personal View. *Artificial Intelligence*, 9(1), 1977.

S. Mittal, C. Dym and M. Morjaria. PRIDE: An Expert System for the Design of Paper Handling Systems. *Computer*, 19(7):102-114, 1986.

J. Mostow. Design by Derivational Analogy: Issues in the Automated Replay of Design Plans. *Artificial Intelligence*, 1989.

B. Myers and B. Zanden. Environment for rapidly creating interactive design tools. *Visual Computer*, 8:94-116, 1992.

D. Navinchandra. *Exploration and Innovation in Design: Towards a Computational Model*, Springer-Verlag, New York, 1991.

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

M. Pearce, A. Goel, J. Kolodner, C. Zimring, L. Sentosa and R. Billington. Case-Based Design Support: A Case Study in Architectural Design. *IEEE Expert*. 7(5):14-20, 1992.

J. Rasmussen. The Role of Hierarchical Knowledge Representation in Decision Making and System Management. *IEEE Trans. Systems, Man and Cybernetics*, 15:234-243, 1985.

V. Sembugamoorthy and B. Chandrasekaran. Functional representation of devices and Compilation of Diagnostic Problem Solving Systems. *Experience, Memory and Reasoning*, J. Kolodner and C. Riesbeck (editors), Erlbaum, Hillsdale, NJ, pp. 47-73, 1986.

H. Simon. *The Sciences of the Artificial (2nd ed.)*, MIT Press, 1981.

R. Stallman and G. Sussman. Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis. *Artificial Intelligence*, 9:135-196, 1977.

L. Steinberg. Design as Refinement Plus Constraint Propagation: the VEXED Experience. *Proc.*

*Sixth National Conference on Artificial Intelligence*, pp. 830-835, 1987.

K. Sycara, D. Navinchandra, R. Guttal, J. Koning, and S. Narsimhan. CADET: A Case-Based Synthesis Tool for Engineering Design. *Expert Systems*, 4(2):157-188, 1991

Y. Umeda, H. Takeda, T. Tomiyama, and H. Yoshikawa. Function, Behavior and Structure. In *Proc. Fifth International Conference on Applications of AI in Engineering*, 1:177-193, 1990.

A. Voss, C-H Coulon, W. Grather, B. Linowski, J. Schaaf, B. Barstsch-Sporl, K. Borner, E. Tammer, H. Durscke, and M. Knauff. Retrieval of Similar Layouts - About a Very Hybrid Approach in FABEL. *Proc. Third International Conference on AI in Design*, Lausanne, pp 625-640, August 1994.