Knowledge Systems Laboratory **July 1991**
Report No. KSL 91-47

Design Rationale Capture as Knowledge Acquisition Tradeoffs in the Design of Interactive Tools

by

Thomas Gruber
Catherine Baudin
John Boose
Jay Weber

KNOWLEDGE SYSTEMS LABORATORY
Department of Computer Science
Stanford University
Stanford, California 94305

Appears in Birnbaum, L., and Collins, G. (eds.) *Machine Learning: Proceedings of the Eighth International Workshop.* San Mateo, CA: Morgan Kaufmann, 1991, pp. 3-12.

Design Rationale Capture as Knowledge Acquisition: Tradeoffs in the Design of Interactive Tools

Thomas Gruber
Knowledge Systems Lab, Stanford University
701 Welch Road, Building C
Palo Alto, CA 94304
gruber@sumex-aim.stanford.edu

Catherine Baudin*
Artificial Intelligence Branch
NASA Ames Research Center - M.S. 244-17
Moffett Field, California 94035
baudin@pluto.arc.nasa.gov

John Boose
Boeing Advanced Technology Center, 7L-64
Boeing Computer Services, P.O. Box 24346
Seattle, WA 98124
john@atc.boeing.com

Jay Weber
Lockheed Artificial Intelligence Center
3251 Hanover Street, O/96-20, B/254F
Palo Alto, CA 94304-1191
jay@laic.lockheed.com

Abstract

This paper introduces a panel to be held at the Knowledge Acquisition Track of the Machine Learning Workshop (ML91). This panel will focus on the problem of acquiring design rationale knowledge from humans for later reuse. The design of tools for design rationale capture reveals several fundamental issues for knowledge acquisition, such as the relationships among formality and expressiveness of representations, and kinds of automated support for elicitation and analysis of knowledge. This paper sets the background for discussion by identifying dimensions of a design space for design rationale tools, and then includes position statements from each panelist arguing for various positions in this space.

1 THE PROBLEM OF DESIGN RATIONALE CAPTURE

For many engineering tasks, including redesign, verification, and diagnosis, it is important to know why an artifact was designed the way it was. *Design rationale* is a general term referring to the knowledge or reasoning underlying a design. Although there is wide agreement that design rationale is valuable knowledge, there are few means for effectively acquiring it. Technology for recording and making this knowledge easily accessible could be used to facilitate collaborative work among designers and to support communication with people who make decisions about manufacturing, operations procedures, diagnostic procedures, configuration, testing, and marketing. Yet today design rationale information is hard to elicit from designers, fragmented, easily lost, and generally not captured in machine-intelligible form.

Understanding the rationale of a design often requires a broad range of knowledge, including the design requirements (including constraints and evaluation metrics), the structure of the artifact, the reasons for choosing particular components or implementation approaches, the assumptions about the context in which the artifact is to be used, and the institutional experience of designing and manufacturing similar artifacts.

In this paper we frame design rationale capture as a knowledge acquisition problem, where the task is to acquire design knowledge in a form that can be reused with automated assistance (e.g., for information retrieval or generation of rationale explanations). Several approaches have been proposed to facilitate the capture and reuse of design rationale. The panelists will present a few exemplary approaches, and explore some of the tradeoffs. In particular, the discussion will focus on the degree of autonomy versus interactivity in these knowledge acquisition systems.

2 SOME APPROACHES TO DESIGN RATIONALE CAPTURE

Although still in the research stage, a range of approaches have been proposed for providing automated assistance to capture and reuse design rationale knowledge. Some notable points in the space include:

- Provide electronic notebooks that encourage designers to record their design processes on line [Lakin et al., 1989].
- Elicit semistructured text about the design rationale, based on argument-style discussions among designers [Conklin and Begeman, 1988; Lee, 1990; McCall, 1986].
- Elicit semistructured arguments in the context of knowledge-based design critics [Fischer, McCall, and Morch, 1989].

- Treat designs as parametric decisions (about components, etc.), and elicit reasons pro and con for alternatives in a decision-theoretic or similar framework [Boose, 1991, and Section 5].
- Elicit explicit enumerations of design alternatives, components, constraints, and classes of solutions, and present the data in graphical form to facilitate human search and evaluation of evolving designs [Shema et al., 1990].
- Record a history of design choices made by designers using a shared design memory that indexes decisions by artifact structure (e.g., component selection) and function [Mark and Schlossberg, 1990, and Section 8].
- Elicit machine-generated demonstrations of intended artifact behavior and operating assumptions from designers using simulation tools with model-formulation and explanation-generation capabilities [Gruber, 1991, and Section 6].
- Infer design rationale from engineering models of the behavior of a device and formal specifications of its requirements [Baudin, Sivard, and Zweben, 1989, and Section 4].

All of these approaches deal with the acquisition of different aspects or abstractions of design knowledge.[1] They also occupy different locations on a continuum that might be characterized as "degree of automation," ranging from relatively low automation for the notebook and hypermedia-based elicitation approaches to total automation (modulo the model formulation) of the rationale-generation approaches. Each point on this dimension makes certain cost/benefit tradeoffs and feasibility assumptions, as we shall see in the position statements of the panelists that follow this introduction. However, on careful inspection we find the spectrum to be multifaceted; there are several related (but not quite correlated) dimensions of this design space for design rationale capture (DRC) tools. To set the context for discussion, we first characterize these dimensions.

3 DIMENSIONS OF THE DESIGN SPACE FOR DRC TOOLS

Formality (structured format versus free-form data). Searle [1990] defines a *formal* representation as one that is syntactically formed and can be parsed into constituents that can be assigned a semantics. Design notebooks and similar "natural" media ( ) are quite *informal;* hypertext tools ( ) are *semiformal* in that the data are indexed by links and node types enforced by the tool; decision supports aids ( ) are semiformal in that they structure the data into formal categories such as alternatives and evaluation criteria; design memory ( ) is more formal because it provides a richer index into decisions based on formal artifact descriptions; rationale-generation schemes ( ) are very formal—they work from engineering models in forms such as differential equations.

Machine-interpretability (versus only human readable). Closely related to formality is the extent to which computer programs can interpret the captured data and provide value-added services. Systems that operate on semiformal representations can interpret the formal parts of the representations, provided the links and types can be given *operational semantics*. Consider, for example, a tool that supports a link called "decision-depends-on" between nodes of type "decision" and other data such as "availability-of-materials." The decision-depends-on link could be interpreted by a program that tracks changes to nodes and alerts the user that a particular design decision has been affected when, say, something causes a node describing availability-of-materials is changed. Note that it is necessary that the links and node types be formal, but that is not sufficient for machine-interpretability. In addition, the user must instantiate the nodes and links in a way that is consistent with their operational semantics (i.e., "decisions" are things that can be affected by changes to things linked to them by decision-depends-on). Highly interpretable representations can support more sophisticated services; an explanation of rationale based on simulation or analysis ( ) could determine whether a change in materials could actually change the required behavior of the device. Similarly, decision-support tools can assess the sensitivity of decisions (utility measures) to changes in parameters such as cost.

Interactivity (interactive elicitation versus automated acquisition / generation). In cases where a complete model of the domain is available to the machine (including the engineering principles, possible alternatives, utility functions, etc.), a tool could infer the rationale for a design. This is the ideal case for a system of category . In practice, all design capture tools need to interact with humans. On the semistructured end of the scale ( ), tools depend on the users to invent useful categories and relations, and use the representation consistently. Tools of the more structured sort ( ) have domain-specific knowledge of the representation built in (e.g., relations between component choices and cost; indexing by part-of relations). These tools can give the user a palette of choices for these built-in relations, and do simple syntactic checking. Tools that depend on generative models ( ) need to provide model-formulation assistance: help with the task of constructing models. They could provide model libraries, for instance, that offer pieces that can be assembled into coherent engineering models. Note that the amount of interactive

[1] Some of the cited work covers more than one of the approaches as categorized here (e.g., mixing formal models and semistructured annotations). This introduction is not intended as a proper survey of the literature. See the latter sections and the cited papers for more detail.

assistance that a tool can offer may increase with use. This is the strategy of design memory ( ); as designers use the tool, they contribute to a growing memory of previous designs that can be reused.

Analyzability (amenable to completeness and coherence analysis). Given a partial theory of design (e.g., as maximizing expected utility in a decision space, or as constraint satisfaction), the machine-intelligible portion of captured data can be analyzed for completeness and coherence (e.g., consistency with constraints). For example, design critics ( ) can check design choices for constrain violations. Decision-support tools such as can inform the user of highly correlated (redundant) criteria and estimate the utility of alternatives. Issue-based hypertext tools ( ) can check for positions without support. Model-based tools ( ) can check for underconstrained behavior models, such as underspecified initial conditions in a simulation.

Expressiveness (representational adequacy). The degree of structure imposed by a formal representation, or the limited set of services provided by a tool, can inhibit the ability of the designer to "say what she means" — and therefore limits the capture of information potentially useful to a human observer. Design notebooks ( ) are optimized for expressiveness. Other tools may allow a "trap door" to free text. However, the presence of a comment field does not make a medium expressive if the tool offers no incentive for the user to fill in the field.

Cost of acquisition. It is often stated that designers resent anyone "getting in their way" during design. But design knowledge ultimately must come from those in the know. A design rationale capture system must address the issue of the cost of eliciting the desired information, whether by lowering the barriers (as aimed for in) or raising the incentive to provide the data (with design support services, such as design checking rules ( ), information management ( ), or information retrieval ( ).

Utility of captured knowledge. Of course, a fundamental dimension for the design of design rationale capture tools is the service they potentially provide. Most proposals address problems of organizational communication and memory (designers repeating the mistakes of the past; designs thrown "over the wall" to manufacturing, etc.). Measures of utility include design quality, engineering productivity, maintainability of products, usability and acceptance of tools by designers. Is there an inherent tradeoff between cost of acquisition and utility of captured knowledge? The issue is complex, since the cost and benefits may be experienced by different people.

Panelist Statements

In the following sections, panelists present their approaches to providing automated design rationale support.

4 THE INFLUENCE OF USE ON DESIGN RATIONALE ACQUISITION

Catherine Baudin and Jody Gevins*

Design rationales—reasons for design choices—are perishable pieces of information that are tedious to record during the design process and difficult to recover after the design has been carried out. Understanding reasons for design decisions is important both to enable new designers to learn from previous experience and to automate the extraction of design knowledge from design cases. Accordingly, design rationale capture can serve different purposes such as: (1) **conservation of corporate memory**—when a design is taken over by another team of engineer or to guide later design changes, (2) **tutoring**—to teach design to novices, (3) **automated acquisition of design knowledge**—to refine a set of existing body of design knowledge [Mitchell, 1985], to facilitate design reuse [Mostow and Barley, 1987], or to track the decisions that should be revised when the design specifications are changed.

One way of acquiring design rationale is to capture it informally as text strings or to have the expert record on an audio device the reasons of his/her decisions. The advantage of such informal approach is that the information is relatively easy to acquire. The main problems are: (1) canned-text can only be interpreted by humans; (2) not all decisions can be documented.

The problem with having the designer entirely controlling the acquisition of design rationale is that he must decide which decisions are important to explicitly justify. This choice, however, has an impact on how the acquired knowledge can be used and by whom: experienced designers, novices, or programs that need everything spelled out. When design rationale is geared toward novices or programs, almost all design operations have a purpose that is important to explain. This consideration motivates the search for techniques to automate design rationale acquisition.

In fact, if the goal of design rationale acquisition fits into the larger picture of acquiring formal design records (to facilitate automated design reuse, computer aided design or tutoring for instance), a plausible research is to focus on how general knowledge about the objects being designed relate to design rationale capture.

These considerations raise three issues: (1) What design knowledge should be acquired in order to capture design rationale? (2) How should design rationale be represented? (3) When should this knowledge be captured? During the design activity? after the facts?

*of Sterling Federal Systems, also at NASA Ames.

Model-based acquisition of design rationale

We have tried a model-based approach [Baudin, Sivard, and Zweben, 1990] to design rationale acquisition with the goal of acquiring design representations for automated redesign (to assist a designer in modifying previous design to accommodate a change in specifications). We experimented with the method for the design of a class of mechanical devices.

The method infers design rationale by relating design choices to the satisfaction of the design requirements and constraints. The system uses knowledge about the structure and behavior of a device and knowledge about the designer's goals to justify design choices. The input of the program is a library of physical components, a design state (assembled by laying out elements selected in the component library), a set of alternative extensions to this design state, and a choice. The program evaluates the impact of each design extension on the specifications and attempts to justify the designer's choice with respect to the requirements that are satisfied by the solution. In fact, the system identifies the requirements and constraints that may be relevant to explain the choice and interacts with the user to validate its explanation. At this point the user can validate the criteria selected by the system, rule out some criteria, or enter new criteria which are added to the existing design requirements.

The advantages of this method are: (1) it relates decisions to the design requirements and constraints. Thus when a specification changes, the system can evaluate the impact of the change on the requirements and can retrieve from its memory a more suitable alternative; (2) The knowledge needed to infer design rationale can also be used to help the designer evaluate the impact of his/her decisions *at design time*. Design rationale is derived from *analytic* knowledge of how to evaluate a design; (3) new design criteria are acquired opportunistically by interacting with the user when the decisions cannot be automatically explained ; □ the method has the potential to explain *any* decision.

This method relies on the acquisition of such knowledge as: (1) a model of the device structure and behavior, (2) representations of design requirements and constraints that are amenable to evaluation, (3) alternative design solutions.

Acquiring design models

One way of acquiring formal design records is to integrate their acquisition with an automated design aid. In principle, an approach that acquires design rationale by interacting with the user *during* the design activity appears to be ideal because design rationale is captured as a side effect of having the user interact with the computer aided design tool. We implemented a prototype whose goal is to help a user evaluate the impact of a design choice on the design requirements. However, this type of design aid is easier to implement for routine or detailed design where a set of predefined components can be identified, than for conceptual or innovative design (where design rationale capture might be most needed).

In the early design stages, most decisions rely on simplifying assumptions and approximate computations. To adapt our performance tool to the early stages of mechanical design, we investigate methods to assist designers in *their* modeling activity by helping them select, manipulate equations, and keep track of the assumptions that support the use of approximate models.

Conclusion

At first sight our model-based approach appears to have shifted the problem of acquiring design rationale to the (possibly) more difficult task of acquiring formal models of the objects being designed. On the other hand the benefit of acquiring device models goes beyond the sole purpose of capturing design rationale. In our case we wanted to acquire design records for computer assisted redesign and design rationale was only an (important) part of these records.

In general, the selection of a rationale acquisition method should take into account: (1) how this task fits in the larger picture of acquiring knowledge about how artifacts are designed; (2) who will be the consumer of this information—experienced designer, novices or programs. The context in which design rationale is to be used also has an impact on *when* this knowledge should be acquired. A fact such as: "the designer inserted a bearing to reduce the friction between these two components" is common sense knowledge to any domain expert and can be documented any time after the design has been achieved. On the other hand a fact such as: "a type x component has been preferred to the type y component because there were no type y parts in stock at this time" would be difficult to recover two years after the decision was made.

5 BLENDING MACHINE LEARNING AND INTERACTIVE ELICITATION FOR DESIGN RATIONALE CAPTURE

John Boose, Jeff Bradshaw,* David Shema*

Knowledge Acquisition and Design Knowledge Capture

Currently, much of the information regarding decision alternatives and trade-offs made in the course of a major program development effort is not represented or retained in a way that permits computer-based reasoning over the life cycle of the program. The loss of this information results in problems in tracing design alternatives to requirements, in assessing the impact of change in requirements, and in configuration management.

* also at the Boeing Advanced Technology Center.

To address these problems, we are studying the problem of building an intelligent, active corporate memory facility which would provide for the capture of the requirements and standards of a program, analyze the design alternatives and trade-offs made over the program's lifetime, and examine relationships between requirements and design trade-offs [Boeing Computer Services, 1989a,b; Boose, 1991]. Early phases of the work have concentrated on design knowledge capture for the Space Station Freedom. We have demonstrated and are extending tools that help automate and document engineering trade studies, and we are developing another tool (DART) to help designers interactively explore design alternatives and constraints. Many of these tools are based on techniques that were previously developed for knowledge acquisition systems [Boose, Shema, and Bradshaw, 1989].

Through the series of demonstrations, we are showing a novel integration and extension of design knowledge capture ideas by:

a. Tailoring knowledge acquisition and process control tools for engineering trade studies, a significant and feasible part of design knowledge capture.

b. Digitally recording speech as an unobtrusive method of capturing design rationale at the trade study workstation.

c. Designing an interactive design alternative generation aid.

DART and Machine Learning

NASA is sponsoring the development of DART, the Design Alternatives Rationale Tool, as part of its overall design capture effort for Space Station Freedom [NASA, 1988a,b]. Methods in DART are based in part on earlier tools built in our laboratory and include interactive and automatic learning methods. DART uses an extended repertory grid model and is being applied by NASA to engineering trade studies and other areas. Knowledge acquisition tasks performed by DART include eliciting distinctions, decomposing problems, combining uncertain information, incremental testing, integration of data types, automatic expansion and refinement of the knowledge base, use of multiple sources of knowledge, use of constraints during inference, and providing process guidance. DART interviews experts and helps them analyze, test, and refine the knowledge base. Expertise from multiple experts or other knowledge sources can be represented and used separately or combined. Results from user consultations are derived from information propagated through hierarchies.

The repertory grid bears a close resemblance to a traditional trade study matrix. Design alternatives are listed and DART uses several interviewing strategies to elicit criteria. Analysis methods in DART help engineers refine and dynamically test the information in the matrix.

Machine learning in DART takes place in interactive and automatic forms. Interactive forms include implication generation, analysis, and review. Automatic forms include strategies embedded in the inference engine and methods to automatically improve knowledge bases.

```
Knowledge Base: SS_Technology
Problem       : co2_removal
Engineer      : JK
---------------- Weight Type Criterion
1   1   5   1   | 0.80  ORD  safety
                |      (1/does_not_require_hydrogen_-_safe-5/requires_hydrogen_
                |       -_not_as_safe)
1   2   2   4   | 1.00  ORD  maturity (1/mature-5/not_mature)
250 200 160 180 | 0.60  INT  weight (160/low_weight-250/high_weight)
5   1   5   5   | 0.40  ORD  chx_impact
                |      (1/system_impact_on_condensing_heat_exchanger-5/no_
                |       system_impact_on_condensing_heat_exchanger)
13  17  10  13  | 0.60  INT  volume (10/low_volume-17/high_volume)
650 600 150 450 | 0.60  INT  power (150/lower_power-650/higher_power)
-------------------------------------------------------------------------------
|   |   |   |         Alternative
|   |   |   two_bed_molsiev
|   |   edc
|   solid_amine_water_desorb
four_bed_molsiev
```

A repertory grid in-progress used as a matrix for a carbon dioxide removal trade study.

Implication Analysis

Inductive implications between criterion values are computed with an algorithm developed by Gaines [Gaines and Shaw, 1987]. A repertory grid is used as a set of examples. Criterion values are viewed as logical predicates, alternatives are the operands of the predicates, and ratings are fuzzy truth values. Implications are shown graphically or listed textually. The strength of each implication is shown, either by listing the score or by varying the thickness of the arrow on the graph. Implications show relationships at higher levels of abstraction implied by a repertory grid. If the engineer disagrees with an implication, DART helps the engineer refine the grid. Frequently, the engineer can think of an exception to the implication (a new design alternative) that disproves it. This alternative is entered, rated, and the implication strength is reduced appropriately. Sometimes implications point out inconsistencies in the way the engineer is using a criterion. In such cases a specialization and generalization dialog (laddering) is used to help the engineer decompose inconsistent criteria into consistent sub-criteria. For example, Figure 2 shows the implication analysis that "automated PPA testing" is implied by criterion values "hard to fix," "high PPA costs," and "low PPA reliability." (from an on-board circuit breaker trade study).

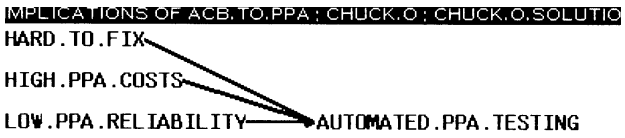

Figure 2: Results of implication analysis.

Reasoning

DART can represent connected sets of matrices for complex trade studies. In a complex trade study the engineer may not want to rate every possible cell in every possible matrix implied by DART's hierarchical matrix organization. Engineers tend to rate the leaf cells of hierarchies leaving cells at higher levels unrated. If the hierarchies are deep the engineer may rate no more than ten or twenty per-cent of the existing matrix cells. However, DART's inference engine expects all cells to be rated so that alternatives may be scored properly for each observation, preference, or constraint. DART has several mechanisms for "filling in" missing ratings as needed. Lower level ratings can be abstracted to higher levels (induction); ratings of parent values can be inherited down to child ratings (deduction); best guesses can be made by looking at siblings' ratings if they exist or by examining the functional similarity of criteria (analogy); users can supply their own application-dependent derivation functions. These mechanisms are used directly by the inference engine and to produce derived matrices that show DART's best inferences for missing ratings.

New Terms

In general, learning systems cannot extend or modify their initial vocabulary to generate new descriptors (new terms) when needed. There are two types of new terms: terms resulting from the compilation or decompilation of existing terms and truly new terms that are orthogonal to the original ones. DART's clustering mechanism can help engineers identify compilations of existing terms when engineers are asked to label cluster junctions. DART's automatic grid improvement mechanism addresses part of the second problem. New unlabeled criteria (terms) are produced based on sets of performance expectations, and the engineer is asked to name them [Shema and Boose, 1988]. DART identifies the need for and characteristics of a new term that is guaranteed to improve the performance of the knowledge base and the engineer is asked to supply the name of the term. For both types of new terms, allowing the engineer to interact with clustering and automatic improvement mechanisms seems to be a key to effectiveness.

Future Work

In future work we will link DART to ARMS, NASA's requirements management system for Space Station Freedom. Capturing this design information should help solve the critical problems of tracing design alternatives to requirements and assessing the impact of changes in requirements.

6 EXPLAINING DESIGN RATIONALE WITH SIMULATION

Thomas Gruber

The problem of *design knowledge capture* is to acquire, in a reusable form, the human knowledge underlying a design, including the structure of the artifact, the rationale for choosing particular components or implementation approaches, the assumptions about the context in which the artifact is to be used, and the institutional experience with designing and manufacturing similar artifacts. Today this knowledge is recorded in written form (text and graphics), if at all, which makes it difficult to access and reuse. I believe that the key to addressing this problem is to change the form in which the knowledge is represented—by designing a formalism and an interactive medium by which humans can contribute to and access a shared corpus of design knowledge.

What characteristics do we want from a representation for design knowledge capture? I claim we need a representation that is comprehensible to both human and machine. That is because a representation of design knowledge has to serve two roles: as a communication vehicle among humans and as a formalism upon which to provide automated services such as completeness analysis and dependency maintenance. Furthermore, I propose that this knowledge can be acquired from people using

tools that generate human-comprehensible explanations from machine-comprehensible models.

Consider the case at hand: acquiring a rationale for a design. The task is to acquire the information sufficient to communicate to a consumer of the information (who may be the same designer) why an artifact was designed as it was. There are several kinds of reasons a designer might give. In general, one can characterize a rationale as an *explanation* relating the design decisions made to requirements and other criteria [Gruber and Russell, 1990].

For design criteria that can be evaluated with objective metrics such as monetary cost, weight, and availability, and when design decisions can be formulated as choices among alternatives, formalisms and reasoning mechanisms from decision theory and operations research can support rationale explanations. For instance, if a certain kind of material was chosen for constructing a housing, one could show an analysis justifying the choice of material in terms of minimizing cost and weight, while satisfying the availability constraint. A design-support tool could provide electronic means for recording this information (e.g., in alternative/criteria charts) and, for operational constraints, offer design checking services.

For other kinds of rationale, such as explaining the intended function or purpose of an artifact, it is difficult to even represent the requirements, much less to automate an analysis that can prove that a given structure achieves its intended purpose. Nonetheless, it is extremely valuable to capture this *teleological knowledge*—about the intended function or purpose of a design—in a reusable form. Natural language descriptions of intended function and assumed operating conditions are typically incomplete, implicit, or missing in design documentation today. To acquire this more elusive kind of design rationale knowledge, we need a representation and interface metaphor other than alternative/criteria charts and semistructured text. For a solution, we have turned to the technology for the interactive construction of engineering models and explainable simulations.

Explainable Simulation as a Communication Medium

In engineering practice, simulation is used to predict the behavior of engineered artifacts. A simulation requires behavior models, typically in the form of equations, that describe the behavior of individual components and general physical processes. The simulation predicts the system behavior that arises from the interactions of components and the operating environment. The results of a simulation is a description of the behavior of the modeled system, presented in the form of graphs, animation, or other forms.

Simulation can also be used to *communicate* a behavior of interest, as is the practice in simulation-based tutoring. A simulation scenario focuses on particular aspects of an artifact modeled with a specific set of assumptions and approximations to describe a behavior of interest. Although the computation of behavior predictions from models is an automated process, the choices about what is relevant to model and predict is an interactive activity involving human engineers making assumptions, constructing models, and setting up scenarios to answer some class of questions about the behavior of an artifact.

We are exploring the idea of using explainable simulation as a communication medium for design rationale [Gruber, 1991; Gruber and Russell, 1990]. To document the intended function of a device, the author constructs a *demonstration* of the device performing its intended behaviors in its assumed operating context. Instead of writing text or producing a videotape mockup, the author engages in a dialog with a knowledge-based modeling and simulation environment, telling the machine what *it* needs to know to generate a demonstration of some behavior of interest. What the machine needs to know is the configuration of a simulation scenario: the engineering models, the initial conditions, the relevant behaviors of interest. Thus the user documents a design by giving a program sufficient information to *generate* an explanation of how the intended function is achieved by the design.

As part of the How Things Work project we are building a device modeling environment called DME [Iwasaki et al., 1989]. DME supports the model formulation by helping users to construct engineering models from libraries of partial models, selecting and composing relevant model fragments to produce a simulation. Explicit representation of the derivation of the model used in simulation supports rich explanations of the results. Using DME, one can build scenarios demonstrating behaviors of electromechanical devices.

The benefits in using simulation as a medium follow from the fact that the information that the engineer enters is guaranteed to be in machine-intelligible form. Because the explanation of intended behavior is generated from underlying models, the same knowledge can be used to answer questions not typically anticipated by the author of static documentation (e.g., minor perturbations on operating conditions and other "what if" queries). Furthermore, since the artifact descriptions and behavior models are constructed on-line with machine assistance, the terms in the explanation, such as the components that played a part in achieving a function, can be effectively indexed with other on-line information, such as the cost, reliability, and availability.

Furthermore, by acquiring this knowledge in an interactive environment, a design knowledge capture tool can provide structured interfaces to help with elicitation and offer computational feedback on the implications of what is entered. For example, a model-formulation system could determine when the set of initial conditions is incomplete (with respect to a simulation) and prompt the user for the missing information. In contrast, when

writing textual documentation, it is easy to forget to specify details of the assumed operating conditions or to be imprecise about the expected behavior.

New Knowledge Acquisition Problems

The major technological bottleneck in this approach is model formulation. I view this as a new frontier in knowledge acquisition research. The model formulation task is to help people construct models of structure, behavior, and function of designed artifacts. This entails providing an appropriate modeling language, a library of partial models to reuse, and intelligent assistance with making assumptions, approximations, and levels of abstraction. A related problem is machine-generated explanation of how things work. The explanation task is to provide human-comprehensible summaries of the results of a simulation, relating the answers to the initial questions for which the models were formulated. Research in qualitative reasoning about physical systems is beginning to show some progress in model formulation and explanation [Addanki, Cremonini, and Penberthy, 1989; Crawford, Farquhar, and Kuipers, 1990; Forbus and Falkenhainer, 1991; Iwasaki, 1990], but the problems remain essentially open.

Discussion

To address the hardest problems of design knowledge capture requires basic research in knowledge representation and automated reasoning, but the payoffs of acquiring design knowledge in completely operational forms would be high. In fact, they must be high or the designers will have little incentive to use the tools.

For more immediate application, approximate solutions can be pursued. If design rationale is an explanation of design decisions to requirements and criteria, weaker forms of explanations may be explored. For example, when real utility functions evaluating design decisions on criteria are not available, approximate, heuristic schemes such as rating grids may be used [Section 5]. Similarly, when the domain models are not available for generating simulations that account for intended function, it can be useful to merely record (by name) the structure and behavior of interest in achieving a function. For instance, a satellite designer may note that achieving the function of a stable, limited power supply for the instruments depends on the type of batteries. Then, when the batteries are later replaced with the newest technology, the dependency can be tracked and the user can be notified to check to see whether the output power is still to specification. (If the models were available to generate the original documentation, then the design dependency might be verified automatically.)

This intermediate solution reflects the basic tradeoffs between automation and interactivity. Greater automation requires a more formal, machine-interpretable representation (engineering models versus textual arguments), which is amenable to completeness and coherence analysis (checking for consistent sets of equations and initial conditions), and which incurs greater cost of acquisition (the model formulation problem). Less automation puts more burden on the interactive user to anticipate the relevant design information (the design constraints worth checking) and to interpret results (determine whether a flagged design dependency is actually significant).

7 NO STRINGS ATTACHED: DEEPER DESIGN KNOWLEDGE CAPTURE MEANS BETTER COST/BENEFIT

Jay Weber and Jon Schlossberg

If automated design knowledge capture is going to be worth the investment of a designer's time and energy, systems must provide a rich set of design services. Many of these services such as design experience indexing, design dependency (truth) maintenance, and requirements consistency checking require deep representations of design moves and rationale. In contrast, the state-of-the-art in design knowledge capture is free-form text structured by a representation of argumentation [Conklin and Begeman, 1988]. The state-of-practice is a word processor. Neither supports the services above. We recognize that moving to more machine-interpretable representations may involve a greater investment on the designers part, but our position is that the cost/benefit ratio is potentially more favorable with deeper design knowledge capture.

Consider the precedent of CAD systems, where formality has led to a better cost/benefit ratio. As sophisticated as they are, they place rigid constraints on what artifact constraints a designer may describe, and how s/he may describe them. Yet designers use these systems instead of the less formal pencil-and-paper medium (though not in the early stages of design) because their investment in the tool pays off in the form of services such as simulation, drafting checking, and documentation. Like with pencil-and-paper, design descriptions captured as structured text are basically only good for one thing: later examination by other designers or future selves.

The Design Memory effort [Mark and Schlossberg, 1990] at the Lockheed AI Center concentrates on the functionality made possible by deep representations of design histories. For example, design memory provides a visualization of the evolution of compatible past designs, in order to help designers chart the course of a new design. Defining a useful notion of design compatibility involves more than superficial comparisons like common keywords; it involves a system that appreciates the scope, content, and consistency of the current design context. Thus design memory has a commitment to a high-level of machine-interpretability of design knowledge, and therefore of its discourse with designers.

A key feature of design memory is that the use of deep design knowledge representations can actually lower the cost of design knowledge capture. This is possible because the system can automatically present relevant design experience, including the designer's rationale for similar decisions, in similar circumstances. The designer can then simply affirm the rationale or make minor modifications. Thus one of the goals of design memory is to make design knowledge capture a *side-effect* of artifact description: the designer says what and the system says why.

For example, Figure 3 uses a tree structure to visualize the decision paths in solid-state flight-recorder designs. Starting from the root, a designer has specified the reuse of previous decisions by traversing the "use two megabyte buffers," "use one multiplexer for all RAMs," and "make each RAM connected directly to output lines" arcs, ending at the shaded node. Because the user is adopting already-captured design knowledge, the system makes minimal demands on user-supplied information. However, as the shaded node is a "dead-end", the user most make novel modifications to continue, as pictured by the dialog box in the upper left. At this point the user is "growing memory," thereby acquiring new design knowledge, yet doing it as a step in the design process and as a natural follow-on to design decision re-use. This points to the principal commitment on the part of Design Memory: to incorporate design knowledge capture and re-use into a paradigm for doing design.

Acknowledgements

Funding for Thomas Gruber was provided by NASA Grant NCC2-537 and NASA Grant NAG 2-581 (under ARPA Order 6822).

References

Addanki, S., Cremonini, R., and Penberthy, J. S. (1989). Reasoning about assumptions in graphs of models. *Proceedings of the International Joint Conference on Artificial Intelligence,* pages 1432-1438, Detroit.

Baudin, C. Sivard, C. and Zweben, M. (1989). A model-based approach to design rationale conservation. *IJCAI-89 Workshop on Model-based Reasoning*.

Baudin, C., Sivard, C., Zweben, M., (1990). Recovering rationale for design changes: A knowledge-based approach. In: *Proceedings IEEE*, Los Angeles..

Boeing Computer Services (1989a). Information to be Retained in a Corporate Memory Facility, CMF TR1, NASA NAS2-121108, April.

Boeing Computer Services (1989b). Corporate Memory Facility Demonstration 1: Knowledge Representation and Trade Studies, CMF DEMO1, NASA NAS2-121108, April.

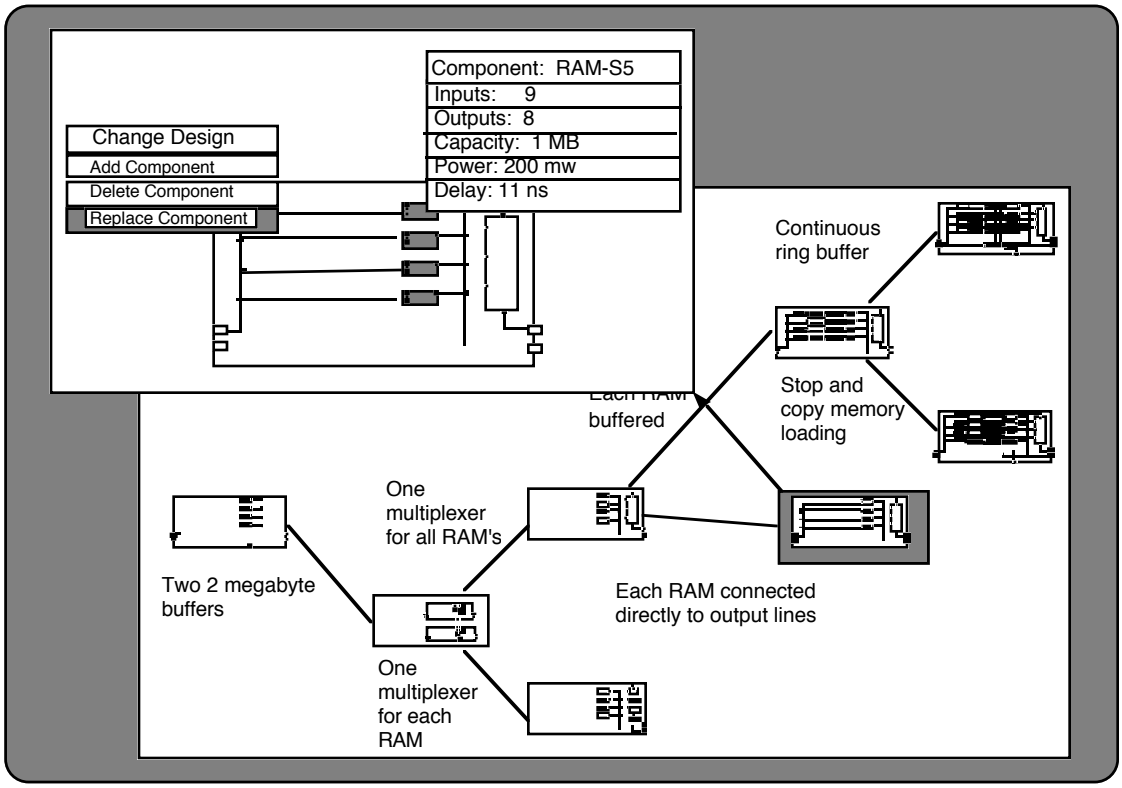

Figure 3: Growing Memory

Boose, J. H., (1991). Knowledge-based design rationale capture: Automating engineering trade studies, in M. Green (ed.), *Knowledge Aided Design*, Academic Press: London (in press).

Boose, J. H., Shema, D.B., Bradshaw, J.M. (1989). Recent progress in Aquinas: A knowledge acquisition workbench, *Knowledge Acquisition: An International Journal*, Vol. 1, No. 2, pp. 185-214.

Conklin, J. and Begeman, M. L. (1988). gIBIS: A hypertext tool for exploratory policy discussion. *Proceedings of the 1988 Conference on Computer Supported Cooperative Work (CSCW-88),* Portland, Oregon.

Crawford, J. M., Farquhar, A., and Kuipers, B. (1990). QPC: A compiler from physical models into qualitative differential equations. *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, August.

Fischer, G., McCall, R., & Morch, A. (1989). JANUS: Integrating hypertext with a knowledge-based design environment. *Hypertext '89,* pages 105-117,

Forbus, D. K. and Falkenhainer, B. (1990). Self-explanatory simulations: An integration of qualitative and quantitative knowledge. *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, July.

Gaines, B. R., and Shaw, M. L. G. (1987), Induction of Inference Rules for Expert Systems, Fuzzy Sets and Systems, Vol. 8, No. 2, p. 315-328.

Gruber, T. R. (1991). Interactive acquisition of justifications: Learning "why" by being told "what". *IEEE Expert*, in press.

Gruber, T. R. and Russell, D. M. (1990). Design knowledge and design rationale: A framework for representation, capture, and use. Stanford Knowledge Systems Laboratory, Technical report KSL 90–45.

Iwasaki, Y. (1989). Two model abstraction techniques based on temporal grain size: aggregation and mixed models. *Proceedings of the Third Workshop on Qualitative Physics*. Available from Stanford University, Knowledge System Laboratory, report KSL 90-63.

Iwasaki, Y., Doshi, K., Gruber, T.,Keller, R., and Low, C. M. (1989). Equation model generation: Where do equations come from? *Proceedings of the IJCAI-89 Workshop on Model-Based Reasoning*.

Lakin, F., Wambaugh, J., Leifer, L., Cannon, D., and Sivard, C. (1989). The electronic design notebook: Performing medium and processing medium. *Visual Computer: International Journal of Computer Graphics*.

Lee, J. (1990). SIBYL: A tool for managing group decision rationale. *Proceedings of the conference on Computer Supported Cooperative Work (CSCW-90),* Los Angelos.

Mark, W. and Schlossberg, J. (1990). Design Memory. *Proceedings of the Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, November.

McCall, R. (1986). Issue-serve systems: A descriptive theory of design. *Design Methods and Theories*, *20*(8):443-458.

Mitchell, T.M., Mahadevan, S. and Steinberg, L.(1985). LEAP: A learning apprentice for VLSI design. *Proceedings IJCAI-85*, Los Angeles, 573-580.

Mostow, J. and Barley, M. (1987). Automated reuse of design plans. In W.E. Eder (Ed.) *Proceedings 1987 International Conference on Engineering Design* (ICED87). Boston, 632-647.

NASA (1988a). Process Requirements Document for Design Knowledge Capture, Space Station Program, March 28, NASA Parkridge Space Center, Reston, VA.

NASA (1988b). Space Station Program, Design Knowledge Capture Plan, Work Package One, Draft 1, D683-10267-3, December 22.

John Searle (1990). Can machines think? *Scientific American*, April 1990.

Shema, D. B., and Boose, J. H. (1988). Refining Problem-Solving Knowledge in Repertory Grids Using a Consultation Mechanism. *International Journal of Man-Machine Studies*, p.447-460.

Shema, D., Bradshaw, J., Covington, S., and Boose, J. (1990). Design knowledge capture and alternative generation using possibility tables in CANARD. *Knowledge Acquisition*, *2*(4):345-364.