AD-A274 603

AL/CF-TR-1993-0069

ARMSTRONG LABORATORY

AUTOMATED INFORMATION MANAGEMENT FOR DESIGNERS:
Functional Requirements for Computer Based Associates That Support Access and Use of Technical Information in Design

William J. Cody
William B. Rouse

SEARCH TECHNOLOGY
4898 S. OLD PEACHTREE ROAD
NORCROSS, GA 30071-4707

Kenneth R. Boff

CREW SYSTEMS DIRECTORATE
HUMAN ENGINEERING DIVISION
WRIGHT-PATTERSON AFB, OHIO 45433-7022

DTIC
ELECTE
JAN 0 4 1994
S E D

93-31292

MARCH 1993

FINAL REPORT FOR THE PERIOD SEPTEMBER 1986 TO MARCH 1992

Approved for public release; distribution is unlimited

93 12 27 045

AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6573

NOTICES

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from the Armstrong Aerospace Medical Research Laboratory. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, Virginia 22161

Federal Government agencies and their contractors registered with the Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
Cameron Station
Alexandria, Virginia 22314

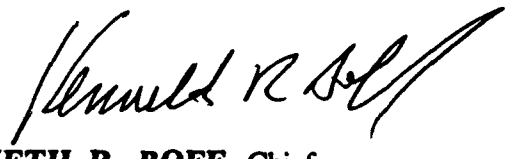
TECHNICAL REVIEW AND APPROVAL

AL/CF-TR-1993-0069

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER



KENNETH R. BOFF, Chief
Human Engineering Division
Armstrong Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1993	Final Sept 1986 to Mar 1992

4. TITLE AND SUBTITLE AUTOMATED INFORMATION MANAGEMENT FOR DESIGNERS: Functional Requirements for Computer Based Associates That Support Access and Use of Technical Information in Design

5. FUNDING NUMBERS
PE: 62202F
PR: 7184
TA: 718426
WU: 71842607
C: F33615-86-C-0542

6. AUTHOR(S)
William J. Cody
William B. Rouse
Kenneth R. Boff

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Search Technology
4898 S. Old Peachtree Road
Norcross, GA 30071-4707

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Armstrong Laboratory, Crew Systems Directorate
Human Engineering Division
HQ HSC, AFMC
Wright-Patterson AFB OH 45433-7022

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AL/CF-TR-1993-0069

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The purpose of this program was to analyze the process of complex system design to determine why designers often fail to consider potentially valuable technical information in making design decisions. Based on this analysis, the goal was to define functional requirements for a new class of supports that will enhance designers' access and use of such information. Termed Designer's Associates, these supports derive from a thorough understanding of designer's tasks and the services that intelligent computing technologies make possible.

14. SUBJECT TERMS

15. NUMBER OF PAGES
127

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

SUMMARY

The purpose of this program was to analyze the process of complex system design to determine why designers often fail to consider potentially valuable technical information in making design decisions. Based on this analysis, the goal was to define functional requirements for a new class of supports that will enhance designers' access and use of such information. Termed *Designer's Associates*, these supports derive from a thorough understanding of designers' tasks and the services that intelligent computing technologies make possible.

Relevant design information exists in a variety of places, expressed in many forms, of varying quality, etc. It includes both system-specific and system-independent elements. Information also tends to be distributed across "data islands," making availability a problem. The designer may not be aware of, misinterpret, or not value relevant information that could improve his decisions. For better or worse, each designer assesses the costs and benefits associated with information access, and acts in accord with this evaluation. A poor evaluation can effectively block information from affecting design choices. The surrounding milieu, which is composed of co-workers, organizations and technology, can interfere with access and use in a variety of direct and subtle ways.

One approach for improving information access and utilization is through use of a new concept in intelligent computer-based supports called "mixed-initiative support systems." In these systems, decision-making authority and task performance can be delegated to machine intelligence to augment human performance. Based on successful application of mixed-initiative systems in the aviation and process control domains, we examine the prospects for applying this technology to information access and use in design. We consider technological needs, system architecture, and how such a support system might be evaluated. The report ends with a future scenario of what design might be like with the support of a mixed-initiative *Designer's Associate*.

DTIC QUALITY INSPECTED 5

Accesion For	
NTIS CRA&I	A
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

PREFACE

This report documents work performed by Search Technology for the Armstrong Laboratory under contract F33615-86-C-0542. Kenneth R. Boff served as the Air Force technical director and program manager. William J. Cody served as program manager for Search Technology.

The effort involved our examining the processes by which complex systems evolve, with most examples taken from military aircraft and aircraft training systems. Our objectives were to identify support system services that will enhance designers' use of technical information and to define the technical and organizational requirements for implementation. To achieve this, we interviewed, surveyed or observed nearly 250 people in design and design-related functions in government, industry, and academia. We also developed prototypes of recommended support functions both to gauge the technical challenges of scaling up to actual systems and to obtain feedback from potential users and other stakeholders in the effort.

The recommendations offered in this report emerged from these activities and, hopefully, reflect descriptions of how design processes work and can be supported made by the many individuals with whom we interacted. We are indebted to these individuals and their sponsoring organizations for their time, interest, and insights. Listed below are individual team members who contributed to this effort and the many organizations that permitted our interacting with design personnel. These acknowledgments, of course, do not imply the endorsement of views and conclusions expressed in this report. We take full responsibility for its content.

Individual Contributors

Armstrong Laboratory
Air Force Systems Command
Wright-Patterson AFB, OH
Donald L. Monk
Tanya Ellifritt
Roy Livingston
Kristin Morton

Search Technology
Atlanta, Georgia
John T. Baldwin
Gian M. Cacioppo
V. Joseph Ceberly
Thomas A. Coonan
Phillip C. Duncan
Paul R. Frey
Norman D. Geddes
Robert J. Glushko
Charles W. Howard
William B. Johnson
David Resnick
Daryl R. Savell
Daniel R. Sewell
Mark D. Weaver
Bradley J. Wiederholt

MacAulay-Brown, Inc.
Dayton, Ohio
Martha Gordon
James R. McCracken
Kristin Rose

Hudson Research Associates
Stuyvestant, New York
Janet E. Lincoln

Logicon, Inc.
Dayton, Ohio
Sarah S. Swierenga

Systems Research Laboratories
Dayton, Ohio
Dieter J. Zirckler

Georgia Institute of Technology
Atlanta, Georgia
Alan L. Porter

ASD/EN
Wright-Patterson AFB, Ohio
Edward A. Martin

Government Agencies

Armstrong Laboratory, Wright-Patterson AFB, OH
Air Training Command, T&E Squadron 3306, Edwards AFB, CA
ATF SPO, Wright-Patterson AFB, OH
Flight Dynamics Laboratory, FIGRC, Wright-Patterson AFB, OH
F-16 SPO, Wright-Patterson AFB, OH
Training Systems SPO, Wright-Patterson AFB, OH
Naval Training Systems Center, Orlando, FL

Aerospace Industry

Boeing Aerospace Company, Seattle, WA
Boeing Helicopter, Philadelphia, PA
Douglas Aircraft Company, Long Beach, CA
Lockheed-California Company, Burbank, CA
Lockheed-Georgia Company, Marietta, GA
McDonnell Aircraft Company, St. Louis, MO
McDonnell Douglas Helicopter Company, Phoenix, AZ
Sikorsky Aircraft, Stratford, CN
Singer Link Company, Binghampton, NY

Workshop Participants

We are also indebted to the participants at four independent workshops who answered our questionnaires and, in several instances, submitted to follow-up interviews:

- Human Perception and Performance Workshop for System Designers
  Moderator: Dr. Kenneth R. Boff, Armstrong Laboratory, Wright-Patterson AFB, OH
  June 1986, Dayton, Ohio

- Human Perception and Performance Workshop for System Designers
  Moderator: Dr. Kenneth R. Boff, Armstrong Laboratory, Wright-Patterson AFB, OH
  June 1988, Dayton, Ohio

- Application of Human Performance Models to System Design
  Moderator: Dr. Grant McMillan, Armstrong Laboratory, Wright-Patterson AFB, OH
  May 1988, Orlando, Florida

- Advanced Cockpit Displays and Controls
  Moderator: Dr. Lawrence E. Tannas, Tannas Electronics, Orange, California
  February 1989, Dayton, Ohio

William J. Cody
William B. Rouse
Atlanta, Georgia

Kenneth R. Boff
Wright-Patterson AFB, Ohio

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

LIST OF FIGURES

LIST OF FIGURES (Cont.)

LIST OF TABLES

INTRODUCTION

PROGRAM GOALS

The purpose of this program was to analyze the process of complex system design to determine why designers often fail to consider potentially valuable technical information in making design decisions. Based on this analysis, the goal was to define functional requirements for a new class of supports that will enhance designers' access and use of such information. Termed *Designer's Associates*, these supports derive from a thorough understanding of designers' tasks and the services that intelligent computing technologies make possible. The capabilities and services that an associate system for designers should exhibit -- what they are, their rationale, and implementation considerations -- are the subject of this report.

BACKGROUND

This project was performed between the late 1980s to early 1990s, a period that saw profound change in product design and development processes in the U.S. Pressures for improving the process emerged from several quarters. These included disasters associated with human error in several large-scale systems (Perrow, 1984), a surge in litigation over product liability (Burger, 1991), offshore industrial competition (Packard, 1986), concerns with technology investment and ineffective transfer into practice (Buxton & Malcolm, 1991; U.S. Congress, 1989), rapid evolution of computer-based engineering tools (Begg, 1984; Floyd, 1991; Forbus, 1988; Hadley & Sommerville, 1990), and others. Numerous reports criticized U.S. global competitiveness and called for a renaissance in the scientific, technological and educational bases of design and product development (Eder, 1988; U.S. Congress, 1989).

These pressures served to focus attention on the "design process," how artifacts come into being, and also stimulated many corrective efforts. Both basic and applied research into the design process experienced substantial upswings (American Society of Mechanical Engineers, 1986; Finger & Dixon, 1989a, b; Mostow, 1985; Rouse & Boff, 1987; Steier, 1990). Managerial interventions such as Total Quality Management and concurrent engineering came into vogue (Linn & Winner, 1986; Rosenblatt & Watson, 1991; Salzberg & Watkins, 1990; Winner, et al., 1988). Many prescriptions for how to design products emerged in the public and proprietary literature (Emery & Parks, 1987; Linton, 1987; Pahl & Beitz, 1984; Rouse, 1991; Suri & Shimizu, 1989; Whitney, 1990). Makers of engineering design tools and information systems

introduced hundreds of paper- and computer-based products (e.g., Begg, 1984; Bogner, 1988; Floyd, 1991; Forbus, 1988; McMillan, 1989). Educators began exploring ways to revamp curricula to improve engineers' and technical specialists' preparation (Borovansky, 1987; Eder, 1988; Hazelrigg, 1988). These efforts shared the common objective of improving the product development process and infrastructures in hopes of improving the resulting systems.

Within this context, our initial goal was to improve the human usability of complex systems in general and crew systems for military aircraft in particular. Usability is a summary term for a host of finer-grained attributes. Put simply, a system is more usable the easier it is to learn, easier it is to operate, the fewer people it requires to operate and maintain, the more tolerant it is of misuse, and so on. We had focused on this system attribute because, despite infusions of advanced technology, newer systems were performing at lower levels and for greater costs than expected (cf. Beevis, 1987; Johnson, 1987; Promisel et al., 1985). These shortfalls were frequently traced to usability problems (Gould & Lewis, 1985; Hammond et al., 1983; Meister, 1989; Promisel et al., 1985; Wahlstrom et al., 1985).

Earlier work had demonstrated that design decisions which affect usability often were made without relevant human-related technical information (Boff, 1987a, 1987b, 1988, 1990). Such information pertains to the anthropometric, biological, behavioral, performance, learning, reasoning, and social characteristics of system users. In some instances, the information was simply not available. But in many cases, available information was overlooked, misunderstood, or misapplied. These problems especially plagued individuals trained in disciplines <u>other</u> than human engineering (Gardiner & Christie, 1987; Hammond et al, 1983; Lintz, Askren & Lott, 1971; Meister & Farr, 1967; Meister et al., 1969). Consequently, toward the goal of better integrating human engineering information into the design process, the parent program (called *Integrated Perceptual Information for Designers*) for this effort had introduced several successful products and services (Boff, 1987b; Boff, Kaufman & Thomas, 1986; Boff & Lincoln, 1988).

As we were concentrating on usability, other investigators were reporting design deficiencies along system attributes that were sacred ground to their domains. Examples include assembly, manufacturability, testability, reliability, and many others (Finger & Dixon, 1989b; Suri & Shimizu, 1989; Whitney, 1990). Deficiencies varied with product domain, but the source of the problems was similar to what we had observed -- suppliers believed that useful information about assembly, manufacturability, etc., was available, but was not being factored into design decisions. Many of these transfer failures were associated with information in archival sources that were external to the design organization (e.g., Allen, 1977; Borovansky, 1987). However, cases of

system-specific information that failed to transfer from one area to another within the same organization were equally common (Winner, et al., 1988). The latter of these anomalies suggested a "stovepipes" image of groups within design organizations, and helped create the impetus for concurrent engineering initiatives (Linn & Winner, 1986).

The similarity between others' complaints and our own demonstrated that difficulties with access and use of information in design are independent of particular disciplines and system attributes (Rouse, Cody & Boff, 1991). The problem is not confined to human-related information; it is pervasive. Moreover, it has at least three deleterious effects. First, product quality may suffer when design decisions reflect a narrower base of relevant information than is available. Second, the design process itself becomes less efficient when product deficiencies must be reworked that could have been prevented with better information transfer. Finally, investments in R&D and design efforts are lost when the concepts, principles, methods, tools, components, lessons, data, etc., that they foster are left to languish in the archives.

The pervasiveness of this problem caused us to shift from an information-centered perspective to a designer-centered perspective. Rather than concentrate on promoting a particular class of information, the goal became to specify an information system that could help any design participant obtain any type of relevant information. Aside from technical barriers associated with linking up to information, this goal required our understanding why designers use some types of information and discard others. The designer's information world is massive. It contains elements generated by the project such as requirements, conceptual designs, engineering drawings, prototypes, test data, and so on. It contains project-independent sources such as past designs, new technology, regulations, principles, tools, models and data, including human-related information. The evidence suggests that designers use a very small fraction of the total store (cf. Allen, 1977; Rouse, 1986). By distinguishing what designers use from what they ignore, we sought to understand characteristics of information that designers value. This knowledge would be instrumental to changing information, the designer, or both to facilitate use and, thereby, improve design. Thus, support systems that enhance information transfer were expected to have general appeal. Figure 1 depicts these overall support objectives in terms of the two broad classes of designer task and two approaches to task support.

Information access and utilization, illustrated by the left and right circles in Figure 1, represent the targets for support. While not synonymous with design, these tasks are integral to good practice and a part of the process that we believed would be amenable to support. For present purposes, *access* refers to activities that produce an object from the information world for

subsequent use. Access has both physical and cognitive components. *Utilization* refers to operations performed on or with an object to support problem solving and decision making.

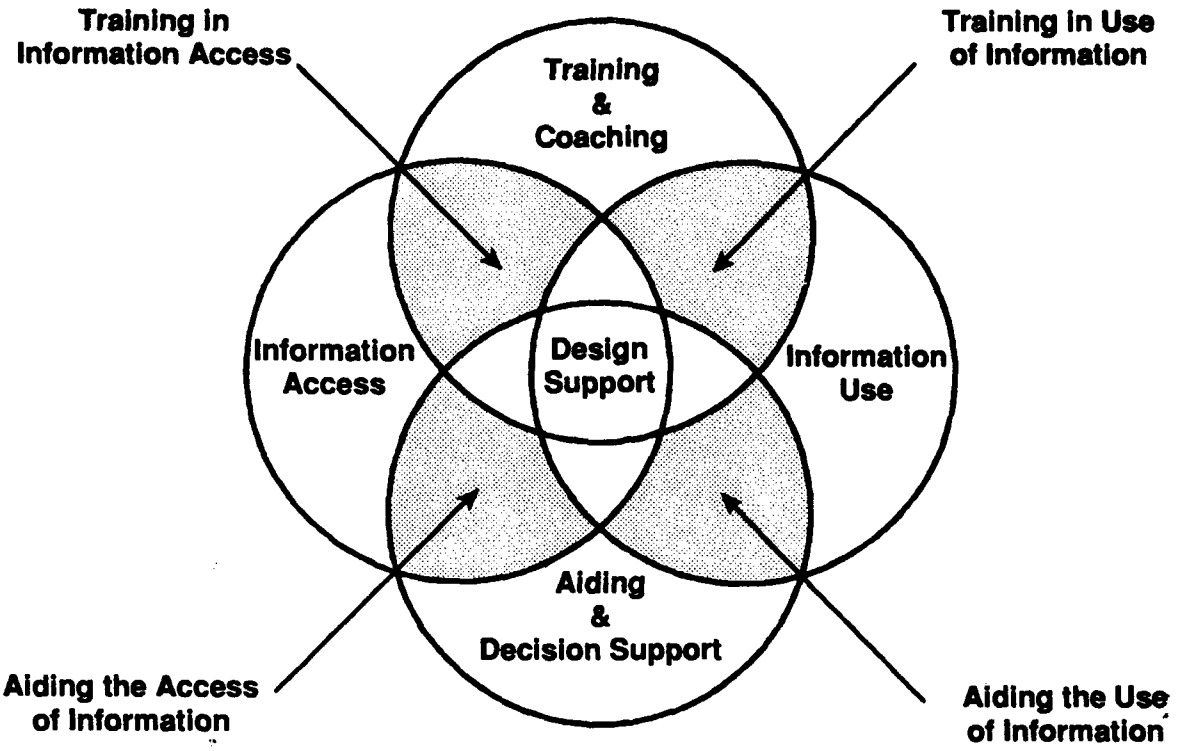


Figure 1. Four classes of design support emerge from the combination of two types of task and two types of intervention.

Figure 1 also depicts two general ways in which human performance in complex task domains can be supported (Rouse, 1991). *Training and coaching* refer to investing in people's knowledge and skills so that they possess the potential to perform without help. *Aiding and decision support* refer to augmenting people's performance directly, often through automated means. These supports represent complementary forms of adaptation (Rouse, 1986). Training is adaptation *of* the user that causes him or her to adopt more effective or efficient task strategies. Aiding involves adapting the environment *to* the user in order to complement the user's style and abilities and overcome his or her limitations.

As suggested by the darkened regions in Figure 1, mapping the two types of support to the two types of task yields four support classes. Put simply, one can train or aid the designer in accessing information, and one can train or aid the designer in applying information to solve problems and make decisions. Within each class, a wide variety of specific supports are possible. Comprehensive support systems will integrate all four forms. It is our intention through this report to foster development of such systems.[1]

OVERVIEW

This document is intended for people who develop information support systems for designers and for those who procure them. Its purpose is to present and rationalize the goals, functions and objectives for such systems. In addition, it identifies implementation requirements for delivering the proposed functions.

---

[1] We recognized the distributed and collaborative nature of system design, and were familiar with emerging supports that are emerging for group work (cf. Corcoran, 1988; Galegher et al., 1990; Kraemer & King, 1988). However, we focused on support requirements for designers when they work individually for three primary reasons. First, our investigations suggested that individuals and groups play different roles in the design process. As discussed in greater detail in this report, individual designers perform the bulk of technical tasks, whereas groups function primarily to manage individuals' work by allocating work and reviewing task results (Rouse & Boff, 1987; Rouse & Cody, 1988). There is no question that design management can be supported and improved; however, our interest was in supporting technical tasks.

Second, our investigations suggested that design processes could be enhanced by reducing cross-disciplinary communication difficulties (cf. Boff, 1987a). A variety of approaches are possible. Groupware may help achieve this. Total quality management and concurrent engineering also seek to remove disciplinary "stovepipes" through managerial intervention. We elected to take an alternative approach based on innovative computer-based education (cf. Boff et al., 1991) -- in effect, alleviating communication problems by increasing each contributor's knowledge about others' domains. A support system can introduce a person to concepts and practices in specialties other than his own so that he becomes a more effective team player. Furthermore, the support can be delivered "just in time" as the specialist needs it, rather than through traditional, and more arduous, educational practices. This approach necessarily focuses on individual's knowledge and skill, not on group activities per se.

Third, the mixed-initiative supports advocated in this report require a robust model of the process being supported. Our review of research on group dynamics in design suggested that the theoretical and empirical bases were not adequate for constructing such a model. This is not to say that emerging "groupware" is or will be ineffective. Simply, the present versions of these aids are not mixed-initiative systems.

Finally, it should be noted that nothing prevents the supports that we advocate from being used in group settings where participants discuss design issues, negotiate system characteristics, and so on. Moreover, given information about what is being discussed, a mixed-initiative system could provide the same types of support that it does for users working alone -- search and retrieval, compiling information, transforming displayed information in forms that the designer likes best, etc.

Following a brief review of study methods, we first identify designers' information needs that a support system should help to meet. In keeping with human engineering traditions, this required understanding what designers do and the types of information they value in performing these activities. Elsewhere we have called this the "human factors *of* design" to emphasize our focus on designers, not their end-users (Boff, 1990; Rouse, Cody & Boff, 1991). Our analysis concentrates on limitations that designers exhibit with respect to access and use of information in support of design tasks. Given these needs, we then define a list of support system goals. Functional requirements to achieve these goals are presented next. This leads to consideration of how system functions might be implemented.

Several implementation schemes are possible, some more automated than others. We focused on "mixed-initiative systems," a new concept in intelligent computer-based supports. In these systems, decision-making authority and task performance can be delegated to machine intelligence to augment human performance. Based on successes that Rouse and his colleagues have had with mixed-initiative systems in the aviation and process control domains (Geddes, 1989; Rouse, Geddes & Curry, 1987; Rouse, Geddes & Hammer, 1990), we examine the prospects for applying this technology to information access and use in design. We consider technological needs, system architecture, and how such a support system might be evaluated. The report ends with a future scenario of what design might be like with the support of a mixed-initiative *Designer's Associate*.

METHODS

To examine the design process and designers' tasks, we conducted a workshop on the nature of system design, collected information from practitioners in industry and government using "naturalist" methods, and reviewed the literature on design and design support.

WORKSHOP

The workshop, held in March 1986, explored design across engineering systems, software, consumer products, and architecture. Thirty individuals convened for a 3-day meeting. They represented government, industry and academia, and had backgrounds in engineering, computer science, psychology, industrial design and architecture. Participants addressed three main topics: 1) the nature of design and designers from both individual and organizational perspectives; 2) design tools in terms of their psychological impact and technological requirements; and 3) organizational environments within which designers work and how these environments influence designers' behavior. Complete results from the workshop appear in Rouse and Boff (1987).

NATURALIST METHODS

Design takes place in the industrial business environment and is not easily studied in the laboratory. Therefore, we adopted methods from social psychology and ethnology -- interviews, questionnaires, and direct observation -- to study designers on location. Our principal interests were in the tasks these individuals perform, where and how they obtain needed information, difficulties they have, and their reactions to various forms of computer-based support.

We conducted 62 interviews with aerospace crew system designers (Cody, 1989), collected 168 questionnaires from participants at four different workshops devoted to uses of human-related information in design (Cody & Rouse, 1989; Rouse & Cody, 1989a, 1989b), and took up residence for six months with a 10-member Air Force group in the early stages of their designing a new cockpit for a fighter aircraft (Sewell, 1990). Table 1 summarizes who we studied according to nature of their work, job specialty, and data collection method. Individuals included under System-Independent R&D pursued applied research and advanced technology development. People under System-Specific Design & Development worked on aircraft programs such as the Air Force's Advanced Tactical Fighter and the Army Apache helicopter, either to define system requirements or respond with design specifications. The Engineer category included people trained primarily in electrical, mechanical, structural, and software domains. Human Factors included behavioral scientists and specialists in training and education.

Table 1. Distribution of Crew System Designers Included In Data Collection

	System-Independent Research & Development		System-Specific Design & Development		
Data Collection Method	Engineering	Human Factors	Engineering	Human Factors	Total
Interviews					
Number	16	7	28	11	62
Ave years experience	15.8	11.7	21.5	17.9	
Questionnaires					
Number	32	49	56	31	168
Ave years experience	14.8	12.4	14.4	9.0	
Observation					
Number	--	--	3	7	10
Ave years experience			6.3	9.1	
Total					
Number	48	56	87	49	240
Ave years experience	15.1	12.3	16.5	11.0	14.1

Of the 240 individuals, 118 were Department of Defense employees, 105 worked in industry, and 17 were from academia. Regarding the primary purpose of their work outputs, 182 people contributed to operational systems or their technology base (e.g., a particular aircraft or helmet-coupled displays); 58 contributed to training systems or training technology (e.g., aircraft simulators). Overall, the 240 people averaged 14.1 years of job experience, ranging from 1 to 35 years.

Given our interest in common needs across design disciplines, it is worthwhile to consider whether the 240 individuals were representative of archetypal "designers." Our sample was composed mostly of people who contribute to human-machine systems and, therefore, could be questioned for being parochial to this domain. We believe, however, that our sample was representative of a broad base of designers for two reasons. First, crew system design requires expertise from over 40 technical disciplines (cf. Cody, 1989; Rouse, Cody, Boff, & Frey, 1990), many of which were represented our sample. Second, aerospace designers face problems of comparable complexity and use tools and methods that are similar to those used by designers in

the process, power, and manufacturing sectors. Consequently, we expect that they exhibit similar problem-solving and information-seeking behaviors. In contrast, our sample may differ along these dimensions from designers of consumer products and other less complex systems.

We also recognize that conclusions drawn from naturalist studies tend to be fairly qualitative and open to alternative interpretations. When we began this work, the research base on the human factors of design was sparse. Although many prescriptions for how design *ought* to be done were available (e.g., Nadler, 1985; Ostrofsky, 1977; Spillers & Newsome, 1989), there were few studies on how design actually *is* practiced and why. Under these circumstances, we felt that naturalist methods would produce greater insight into the character of this complex process than would conventional experimental approaches that focus on few factors and relationships at a time. Other "naturalists" had reached similar conclusions at about the same time (e.g., Curtis, Krasner & Iscoe, 1988). Since then, the literature on design activity has begun to grow, as exemplified by Ullman et al. (1988) in mechanical engineering, Schon (1983) in architecture and structural design, and Malhotra et al. (1980) and Guindon (1992) in software development.

LITERATURE REVIEW

Throughout the program, we reviewed papers, books and technical reports on the nature of design, design theory, prescriptions for designing, organizational and social phenomena in design, cognitive and behavioral phenomena in design, as well as design support systems. These sources represented a variety of perspectives including management sciences, systems engineering, traditional engineering fields, computer science and artificial intelligence, human engineering, several branches of psychology (cognitive, engineering, social, consumer), human factors, information and library sciences, architecture and art. In addition to the references section of this report, bibliographies of works on design are available (American Society of Mechanical Engineers, 1986; Bruns & Gerhart, 1986).

CHARACTERISTICS OF DESIGN

In order to define designers' needs, it is important to understand characteristics of design problems and the processes by which these problems are addressed.

NATURE OF DESIGN PROBLEMS

It is generally acknowledged that design is different from other sorts of problem solving such as decision making, diagnosis, and negotiation. While the overall goal of design is relatively clear, in contrast to these other varieties, design problems have been described as "wicked." The initial state, end state, and legal transformations between the two are all ill-defined (Smith, 1981). As a result, the functional demands posed by design problems tend to be unique.

Design Goals

Many definitions for the overall goal of design have appeared. One useful definition composed from several sources (American Society of Mechanical Engineers, 1986; Bruns & Gerhart, 1986; Gero, 1990; Goel & Pirolli, 1989; Hazelrigg, 1988; Mostow, 1985; Nadler, 1985; Newsome et al., 1989; Smith & Browne, in press) is as follows: *the overall goal of design is to specify an artifact whose functions and attributes satisfy requirements and criteria held by stakeholders in the effort.* Several elements of design that are captured in this definition merit comment.

Artifacts, the things that get designed, can be described as complex collections of attributes that assume particular values (Buur & Andreasen, 1989). Physical artifacts, for example, have particular values of size, reliability, and exhibit certain levels of performance. Software artifacts are more or less modular, fast, and free from logical errors. A given artifact has myriad attributes. These are related to one another in a tangled hierarchy (Eder, 1988). The designer selects values for basic or foundation attributes directly, such as materials used, layout of components, and tolerances. In turn, these choices drive the values of attributes at higher levels of abstraction and aggregation in the artifact. Higher-order attributes, such as ease of use and operating cost, are often termed "emergent" both because they result from more fundamental design choices and because their values emerge with certainty only after the artifact is implemented (Hazelrigg, 1988).

While the artifact eventually does emerge from the overall process, most design has become the production of detailed instructions or specifications, rather than of the artifact per se (Goel & Pirolli, 1989). This is due to the separation of product development into design, manufacture, distribution and use in modern society. This characteristic has two implications. First, designers spend much of their time generating and using representations of the artifact and its effects on the physical, social and economic environments as opposed to the artifact itself (Buur & Andreasen, 1989; Shah & Wilson, 1989). Second, since the designer is not the fabricator, there is a premium on clarity with which fabrication instructions must be expressed. The amount of time required to produce these specifications is directly related to the complexity of the artifact.

Conceptualizing artifacts in terms of attributes helps to define various terms used in design practice. For our purposes, *requirements* refer to both goals and constraints. Goals are the raison d'être for the design. Constraints are conditions that the design or design process must not violate. While not valued per se as goals are, constraints help to identify unacceptable designs. Requirements are usually stated in terms of the problem to be solved without implying a singular solution (Smith & Browne, in press). *Criteria* refer to value(s) or ranges of values that attributes can assume and still be judged acceptable. *Specifications* are precisely stated settings of attribute values, and are normally called out in a formal document. Although not inviolate, criteria tend to apply to emergent properties such as performance, reliability, and usability, whereas specifications more often pertain to basic properties like tolerance. Thus, specifications can be considered a special class of criteria.

As a final note about the definition of design goals, attributes can be thought of as "owned" by stakeholders in a design effort (Rouse, 1992, in press). For instance, people who request artifacts decide what purpose it will help them or their constituents to achieve. Product users decide, implicitly or explicitly, how easy to learn and use a system must be. Buyers, who may be neither requesters nor users, judge whether benefits outweigh costs. Stakeholders also include people who may not be interested in the artifact per se, but in its not violating important constraints. Vendors, for instance, have a stake in ease of fabrication and will balk at designs that exceed their fabrication capabilities. Similarly, project sponsors have a stake in the design process insofar as they want it not to exceed constraints on resources and schedule. Many other stakeholders and interests can be imagined.

Design States

Regarding initial state, most design efforts are marked by radically incomplete requirements and criteria for success. That is, what the problem (or opportunity) is and how solutions will be judged are both unclear. There are several reasons for this. First, design is usually carried out for a client and, therefore, the designer cannot determine all relevant goals and constraints by introspection (Smith & Browne, in press). Second, people tend to be poor at stating their needs in terms that translate easily into design objectives (Rouse, 1986). They are much better at recognizing satisfactory solutions than they are at specifying them (Guindon, 1992; Malhotra et al., 1980). This relates to a third reason. Needs that launch a project may not reflect what a client wants after he realizes opportunities that a new design makes possible (Brown & Chandrasakaran, 1989; Gero, 1990). This encourages continual discovery and refinement of requirements across the entire development effort. Put simply, in contrast with most textbook depictions, requirements definition is never quite done. As a result, problem formulation is usually a central issue in design, much more so than in other sorts of problem solving (Hunt, 1987; Rouse, 1986).

The end state of design is also marked with quite a bit of uncertainty. Proof that a particular solution meets requirements is often difficult to establish for one or more of several reasons. First, it is generally difficult, if not impossible, to forecast precisely the effects that a new artifact and its environment will have on one another (Smith & Browne, in press). Thus, conclusions that a design is a success must await implementation and, for artifacts that have wide-ranging or slowly evolving effects, this can take a long time.

Second, attributes vary in the precision with which they are defined and, therefore, in the precision with which they can be measured in support of evaluation. Some attributes can be measured directly and quantitatively (e.g., size, mass, speed). Others may be quantifiable, but are more difficult to measure (e.g., reliability, ease of assembly, durability). Still others are quantifiable only with subjective measurement practices (e.g., aesthetic appeal, acceptability, and ease of use). Compare the difficulty of measuring an artifact's size or mass with its "usability." Related to this measurement issue, criteria for acceptable attribute values are more or less precise as a function of clarity and precision of the attribute's definition. Thus, the ease of verifying that a design meets criteria varies with attribute. Establishing that the artifact meets the physical criteria is generally easier to achieve than establishing that it meets criteria related to learnability and use, for instance.

Finally, even if measuring attribute values and comparing them to criteria were straightforward, it is often difficult to identify who is the proper stakeholder to poll for a judgment. Attributes usually map to stakeholders in complex ways. Moreover, stakeholders may possess different and conflicting definitions for the same attribute or disagree with what value an attribute ought to assume in the final product (Eder, 1988). Thus, identifying who the stakeholders are, especially if the artifact introduces major changes in social or physical environments, and reconciling their differences, are often more central to design success than overcoming technical problems (Rouse, 1991).

Relationships Among Elements of Design

Figure 2 relates several characteristics about design made to this point. Focusing on the left-hand side, the figure suggests that in design, one generally proceeds top-down from observing the operational context to an artifact, although there is a great deal of iteration among steps. Developers express their observations of operational conditions in terms of the problems that people have. Problems are converted into needs which, in turn, are translated into the behavioral requirements and formal specifications for constructing the artifact. Each of these steps is typically documented in more or less formal ways. Such documentation subsequently serves in comparisons with the actual system.

Test and evaluation tends to proceeds in the opposite direction from the bottom of the hierarchy upward toward the emergent properties in the system. In principle, testing first focuses on whether the artifact "works" from an engineering perspective. Given that it runs, demonstrates no fatal flaws, etc., then the artifact can be used in comparison with documented results of the design effort. "Does the system meet specifications?"; "Does the system achieve its behavioral requirements?"; and so on up the scale from basic system properties to the broader questions of utility and cost effectiveness. Although not apparent from the figure, the artifact used for testing may range from a rough sketch on paper, to a low fidelity mockup, to a prototype or a production copy of the actual system. The rigor of test procedures are usually adjusted according to the level of artifact fidelity (Rouse, 1991).

The right hand-side of Figure 2 illustrates the mapping of attributes to stakeholders. System developers tend to be interested primarily in lower-order questions which collectively ask "Does the system work?" Given the system works, system users and buyers are much more interested with questions of validity, acceptability and viability, i.e., "Is the solution worth it?" As all successful product developers eventually discover, a development effort can derail at any level

in the hierarchy. Perhaps behavioral requirements are not achieved. Or an engineering marvel turns out to be unacceptable to users. Or a well-engineered solution that users like turns out to be financially untenable.

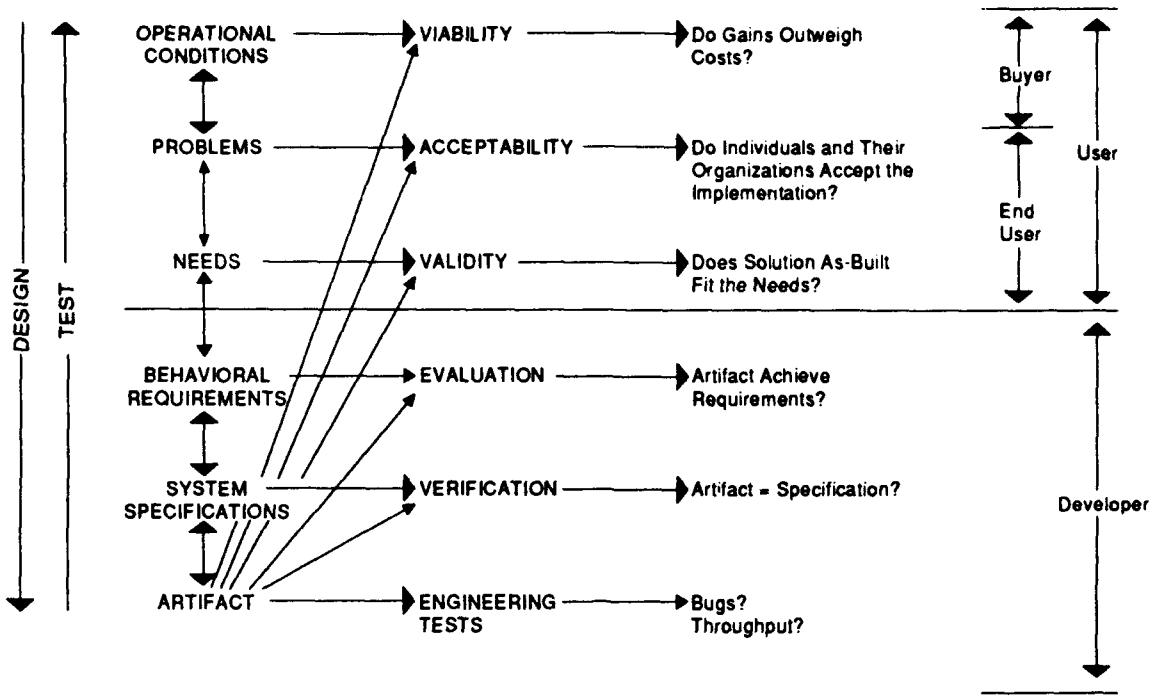


Figure 2. Design and test are complementary processes that create and evaluate, respectively, different expressions of the artifact. Artifact properties are of more or less concern to different stakeholders in the process.

THE DESIGN PROCESS

Understanding characteristics of the designer's goals and problem is necessary but not sufficient to specifying supports. This also requires understanding the nature of the design process -- who is involved, when, and what do these contributors do? Answers to these questions are much more subtle than at first might be imagined (cf. Rouse, 1987a; Rouse & Cody, 1988).

Who is the Designer?

We defined the designer to be anyone who intentionally influences the function or form of the evolving artifact (Rouse & Boff, 1987). With little reflection, one sees that this definition admits many contributors, especially in complex systems. Managers, specialists from several engineering and technical disciplines, manufacturers, marketers, trainers, customer support people, and several others intentionally influence the product -- many people beyond those whose job title is "designer." This is due to the variety of interests that must be balanced and to the varieties of knowledge and skill required to achieve this. Beyond members of the design team, there are also many stakeholders in most design efforts (Rouse, 1992). While these individuals may not intentionally affect the product, their opinions and values are, or should be, taken into consideration by those who do.

With the number and variety of people involved, design obviously is a collaborative, multi-disciplinary process that is distributed across temporal, organizational and geographic boundaries (Allen, 1977, 1986; Boff, 1987a; Curtis, et al., 1988; Rangan & Fulton, 1991; Rosenblatt & Watson, 1991). Within the same project, participants are usually organized according to system component (e.g., sensors, navigation) and type of analysis (e.g., aerodynamics, structures, reliability) (Allen, 1986). These groups interact in two general ways: directly through a variety of social exchanges and reports, and indirectly via the effects that their decisions and actions have on the product, design resources and shared information.

As a consequence of the distributed nature of design, decisions emerge both from individuals' activities and from activities when individuals collaborate. Further, decisions are not confined to a single early stage of the overall process. Contributors make decisions at all points in the life cycle. For instance, R&D decisions influence what technology alternatives will be available to product developers. Marketing decisions affect what functional requirements should be met in the new system. Engineering decisions govern both functional and structural characteristics. Procurement and sales decisions affect how the system is employed. Operational experience, including training, affects the definition of new requirements as the cycle repeats itself. In short, decision making in design is pervasive.

Stages of Design

Most commentators have similar characterizations of the process of design. Stage models are common, both at levels of the overall program (Figure 2) and the individual designer. Programs are said to move across stages of requirements definition, conceptual design,

preliminary design, detailed design, fabrication, and deployment (Blanchard & Fabrycky, 1981). Individuals' activities on particular issues are described in terms of problem formulation, design generation or synthesis, design evaluation or analysis, and optimization (Rouse, 1986). Regardless of the labeling of the stages, virtually every model or description of design emphasizes iteration or cycling among processes. At the level of the individual, this has been expressed as cycling among design states (Carroll et al., 1979) or among different levels of abstraction and aggregation (Rasmussen, 1986, 1988).

Within stages, the design process can be examined both at the level of groups and the organization as well as at the level of individual designers.

Groups and Organizations

Design groups or teams are central to designing complex systems. These teams are multidisciplinary. Not surprisingly, engineering disciplines usually lead the effort (Cody, 1989).

Designers spend a fair proportion of their time in group settings. For journeymen and seasoned designers, the time allocation is typically 30% in group activities and 70% in individual activities. Upper-level senior designers spend more time in group activity, serving as coaches and mentors.

The design group or team has several roles. The group is usually involved with decomposing the statement of work or other descriptions of objectives, requirements and specifications. Based on this decomposition, the group will set technical goals, as well as allocations of person-hours and schedule, for members of the group. Pursuit of these technical goals is predominantly an individual activity. The group subsequently reviews the results of these individual efforts.

The organization, both the company and the marketplace, strongly affects group and individual activities. Company policies and procedures directly influence activities. Success criteria and reward mechanisms, both internal and external to the company, influences motives and values. Corporate and market cultures influence, for example, relative weightings on performance, cost, and quality.

Several investigators have noted characteristics of group communications that pose substantive demands on individual designers (Curtis et al., 1988; Kraemer & King, 1988). In

face-to-face encounters, demands emerge from unsystematic coverage of issues, information overload per individual in real-time, failures to record information, misunderstood messages, incomplete communications due to dominance by one or a few individuals, unfocusing, peer pressure, and premature closure. Outside the meeting context, demands emerge from communications related to notifying others about decisions and actions, and from keeping abreast of others' design activities and changes.

Of particular importance are problems with cross-disciplinary communications. As Boff (1987a) has noted, the multidisciplinary nature of system design can cause design communication to suffer from a Tower of Babel phenomenon -- everyone talking at each other from different conceptual frameworks in different jargon and terminology.

Rouse, Cody & Boff (1991) explored the cross-disciplinary issue in some depth. They noted that the nature of design problems that different disciplines on a design team resolve inherently require different approaches. As a result, different disciplines must employ different tools, methods and computational approaches.

This, in and of itself, however, is not the difficulty -- design problems require these different approaches. Communications problems emerge when people use their specialized tools and methods, which are well suited to a particular class of problems, as metaphors with which they view *all* problems. Thus, for instance, many electrical engineers view all problems in terms of control systems analogies (i.e., dynamic state equations and feedback control laws). Similarly, many human factors practitioners, given time and resources, address all problems experimentally and with analysis of variance (i.e., linear statistical models). Few people realize that their representational framework so dominates their perspective. Consequently, they may be unaware that they interpret every problem as a type of nail that their hammer will fit, resulting in miscommunications.

Individual Designers

Much of our data collection, especially with questionnaires and observational study, focused on design problems, information sought in solving these problems, and possible design support functions. Understanding these issues required us to explore what tasks and activities designers perform as they seek information and solve problems.

Several initial hypotheses emerged from early data collection efforts and were tested, usually informally, in subsequent data collection efforts. This series of efforts eventually led to a more cogent description, which states that design occurs within the three-dimensional space depicted in Figure 3 and 4 (Rouse & Cody, 1989b). Figure 3 depicts the abstraction and aggregation dimensions of the design space. This characterization is based on Rasmussen's (1988) constructs. The definition of aggregation is obvious from the figure. Abstraction is more subtle.

The concept of abstraction relates to the types of representation relevant to design. The three levels shown in Figure 3 can be defined as follows:

- Purpose: Representation of design requirements, objectives to be met, problems to be solved, etc. via requirements documents, scenarios, simulations, etc.
- Function: Representation of relationships (i.e., physical, computational, temporal, etc.) via diagrams, equations, simulations, etc.
- Form: Representation of appearance (i.e., assembly, geometry, etc.) via drawings, pictures, mockups, etc.

Figure 4 depicts the task dimension of the design space. These tasks, and perhaps others that are similar in nature, can be viewed as the designer's proximal intentions as he works toward developing a complete design description. Although not shown in the figure, these activities are also pursued at each level of aggregation -- system, subsystem, etc.

Based upon interview and questionnaire studies (Cody, 1989; Rouse & Cody, 1988, 1989a), we hypothesized that design can be characterized as sequences of the activities in Figure 4 that form paths or trajectories across levels of abstraction and aggregation in this artifact space. "Moves" along the three dimensions can be seen to represent different types of tasks. Translations along the abstraction dimension are associated with *specification* (moving from higher to lower levels of abstraction), and design *justification* (lower to higher). *Decomposition* and *integration* are associated with moves along the aggregation dimension. Moves among tasks within the same levels of abstraction and aggregation are associated with the generation and reduction of variety via *synthesis* and *evaluation* activities.

In one study (Rouse & Cody, 1989a), we developed a fictional design scenario based on a representative task sequence. Practicing designers read the scenario, and then rated its realism and provided explanations for their ratings. The practitioners said that, from their experience, the scenario was a familiar and realistic portrayal of how designers behave. Although this method for

LEVEL OF AGGREGATION	LEVEL OF ABSTRACTION		
	Purpose	Function	Form
System			
Subsystem			
Assembly			
Component			

Figure 3. Design objects can be described at different levels of abstraction and aggregation.

PURPOSE	FUNCTION	FORM
Explore Problem/Need	Conceptualize Solution Functionality	Compose Form of Solution
Study current requirements (e.g. Statement of Work) -- read and analyze	Review functionality of past designs -- read and analyze	Review forms of past designs -- read and analyze
Study scenarios of operational need -- view and analyze	Synthesize/derive input-output relationships -- create and represent	Synthesize form of solution -- create, visualize and "sketch"
Review requirements for past designs-- read and analyze	Develop model of functionality -- integrate, analyze and test	Prototype/mockup solution -- integrate and fabricate
Explicate performance criteria and attributes -- integrate and decide	Predict performance (exercise model) -- calculate/simulate, analyze and interpret	Measure performance (collect data) -- observe, measure, analyze and interpret

Figure 4. Archetypal design tasks associated with different levels of abstraction.

testing the hypothesis is somewhat indirect, the results of the study supported the notions captured by the design space.

Since this study, considerable support for the model has emerged from observational studies of designers working on actual problems. Guindon's (1992) series of studies with software designers is representative. Starting from an overall requirements document for a software system, Guindon's subjects appeared to move chaotically from task to task and between knowledge domains, especially during early stages of design. They developed partial solutions that contained pieces of different subsystems and these were expressed at various levels of abstraction. These interim solutions appeared to serve as hypotheses which the designers used to test their understanding of requirements. Such tests often led to discovery of unstated or new goals and constraints and to drastic restructuring of either the problem, the solution, or both.

Subjects were also very eclectic in their choice of methods. Faced with different instances of very similar design problems, they might on one occasion proceed bottom-up to arrive at a tentative solution, and on another occasion, decompose the problem in a structured top-down manner. Similarly, when they needed additional information about some issue, sometimes they accessed sources external to the problem (e.g., a text), sometimes used a tool to produce the information (e.g., computer-based drawing package), but most often, relied on their own best judgment from past experiences with similar issues. Their verbal commentary about their strategies indicated that choice of method was driven by context-specific factors.

Findings showed that design activities are recursive. For example, information needed to address a design problem often stimulates the need to have information about the information (e.g., where it might be located; estimates of its validity). The need for information part way through a task forces the designer to suspend to original task, resolve the information need, and then pick back up with the original task. Thus, with recursion comes the need to keep track of tasks and information that are left in partial states of completion.

Finally, activities that applied to the primary artifact also pertained to information generators such as models and simulators. In a kind of "design within design," the designer must develop methods and tools that produce information which, in turn, helps him to make decisions about the primary artifact. In this way, the design world becomes populated with information generators which, like the primary artifact, must be retrieved, evaluated, explained, managed, etc.

From these and other observations, Guindon (1992) concluded that design is opportunistic and does not uniformly follow a top-down approach. Schon (1983) referred to similar behavior among architects and engineers as "iteratively uncovering phenomena and seeking explanations." Similar observations have now been reported in studies of mechanical engineering (Ullman et al., 1988), software design (Malhotra et al., 1980), and consumer product design (Ballay, 1987), lending credence to the generality of the description in Figures 3 and 4.

Aside from knowing about typical task sequences, it is also important to understand the demands posed by the individual tasks in Figure 4. Of particular interest are the designer's use of systematic methods for achieving these tasks and his creation and use of design representations.

Design Methods

As suggested above, design is generally described as a heterarchical, somewhat chaotic process involving transitions back and forth between synthesis and analysis at several levels of abstraction and aggregation. Various attempts have been made to organize design activities into prescribed steps, typically involving a top-down hierarchical process. These approaches have the advantage of fostering systematic evolution of solutions, but the disadvantage of potentially inhibiting insights and innovation.

Design methods can be distinguished along several dimensions. One dimension is the task supported. There are methods for problem definition, decomposition, synthesis, evaluation and optimization. Another distinction is between general and focused methods. Examples of general methods include work breakdown analysis and overall systems design (Nadler, 1985). Focused methods on narrow tasks are typically discipline-specific. In human engineering, for example, methods for allocating functions between human and machine (Rouse & Cody, 1986) or for design of displays for human users (Frey et al., 1984) exemplify focused methods. Methods can also be distinguished as formal or informal, and as discipline-specific or discipline-independent. and to another due to differences in the types of phenomena and representational forms used to understand and manipulate these phenomena (Rouse, 1982; Rouse, Cody & Boff, 1991).

For purposes of developing design supports, it is important to recognize that, as with representations, designers have a wide variety of methods with which to achieve their tasks. No doubt, selection of a particular method under any one set of conditions is driven by the designer's awareness of alternative methods and his perception of their relative costs and benefits. This does not imply that formal tradeoff analyses of methods either *are* the norm (they are not) or *should be*

(they need not). Rather, the key insight is that a support system which understands that designers can accomplish tasks in several ways can help generate and evaluate methods for the designer's consideration that appear to be "missing" as he works toward a goal.

Use of Representation in Design

Whether applying a particular method or not, designers spend a great deal of time using representations of the artifact to accomplish the tasks in Figure 4. A representation or model is defined as a partial reproduction of the attributes of an object.[2] Designers use representations to gain insights into properties of the not-yet-completed artifact.

Models vary along several dimensions. Those of particular interest in design are shown in Figure 5 grouped into three categories. The first category reflects the modeling activity. It includes properties related to why a model might be built, the object and attribute(s) of concern, and the beneficiary of the activity. The model user may be the designer himself as he tries to understand the nature of the problem, evaluate his ideas, etc. Models are also used to describe ideas or actual functionality to colleagues and customers. Similarly, computers and numerical control machines "use" models to predict artifact properties or control factory processes.

The second category includes the abstraction and aggregation dimensions discussed in connection with Figure 3. The most commonly used representations in design capture artifacts at various combinations of these two dimensions -- requirements documents, sketches, functional models, equations, detailed engineering drawings, mockups, production versions of the product, etc.

The final category focuses on properties of the model itself. Models derive from concepts and techniques in different disciplines. They also vary in their physical similarity and precision in detail to the artifact -- a rough block diagram of how a system component might function is an abstract, low precision model. In contrast, a manufacturing prototype for the same component is both concrete and fully detailed.

---

[2] Note that "object" is intentionally a very broad concept here, meant to indicate any type of entity in the design world. It certainly applies to the artifact. It also applies, however, to the design process (tasks, assignments, schedule), collateral processes that the designer must consider (production, testing, customer support, etc.), and methods and tools that the designer may need to construct (e.g., a simulator). Each of these objects can be captured at various levels of abstraction and aggregation, with more or less precision, more or less expressiveness, in formal and informal ways, etc.

PROPERTIES RELATED TO MODELING ACTIVITY	
Purpose of Model	Description, idea generation, evaluation, specification
Object of Interest	Artifact, design tool, design process, production process...
Attribute(s) of Interest	Basic (tolerance, materials) to emergent (reliability, usability, safety)
Primary User	Designer himself, colleague, customer, user, draftsman, computer, numerical control machine...
PROPERTIES OF THE ARTIFACT BEING MODELED	
Level of Abstraction	Purpose, function and form
Level of Aggregation	System, subsystem, component, part...
PROPERTIES OF MODELS	
Disciplinary Origins	Electrical, mechanical, chemical, software, behavioral, social...
Form similarity	Abstract to concrete
Precision	Coarse (rough or sketchy) to detailed
Expressiveness	Number of attributes and interrelationships that are represented
Formality	Informal (e.g., sketching), semi-formal (e.g., diagramming) to formal (basis in mathematics & logic)
Code	Natural language, symbolic, spatial
Medium	Text, speech, graphic, video, 3-dimensional (virtual, paper, foam core,...)
Machine Processability	Low (e.g., text) to high (e.g., equations)
Human Understandability	Low (machine instructions) to high (text and graphics)

Figure 5. Artifact representations vary along several dimensions.

Expressiveness refers to the variety and interconnectedness of attributes that are captured in the model. Most models are relatively low in expressiveness, as they portray an artifact from a single perspective. We will return to this aspect of models in particular as it relates to functional demands on the designer.

Formality refers to whether the model emerges from a system of primitives and combining operations or from a more heuristic basis. Formal models, like equations, have a mathematical or logical basis and are amenable to machine processing. Models such as operational sequence diagrams and circuit diagrams are semi-formal insofar as they derive from systematic, but heuristic bases. Sketches are informal models.

Finally, models can be prepared in various codes (e.g., natural language, electrical symbols, numbers, etc.) and media (e.g., paper, computer display, speech at a design meeting). These properties plus their basis in formal systems of logic and mathematics make a given model more or less amenable to machine and human understanding.

Clearly, models are essential to design. They capture relationships among attributes, thereby allowing the designer to explore implications of design choices as well as discover unstated goals and constraints. They support the designer's apprehension of the problem and offload memory by externalizing these relationships (Buur & Andreasen, 1989). In this role, models enhance communication among people who contribute to design. Models also enable product evaluation, serve as the record of design decisions, and help maintain coherence within the design (Gero, 1990; Maher, 1990).

All models, however, are inherently limited in four ways which impose different types of functional demands on the designer. First, any one representation can capture only a small fraction of the pertinent attributes and interrelationships, i.e., "expressiveness" of the majority of representations is low relative to the artifact itself. Thus, the designer must access several different models to obtain an overall understanding of the design state (trying to "apprehend the elephant" from many highly specialized views). From a practical perspective, only a few representations can be viewed simultaneously with contemporary display surfaces. Consequently, low expressiveness per model also induces much swapping in and out of views.

Second, most representations are static and nonexecutable. The designer must mentally simulate the behavior of the design solution, a task which suffers from the limited capacity of human memory (Goel & Pirolli, 1989; Zachary, 1986).

Third, models possess properties that do not belong to the artifact and which are irrelevant to the purpose of modeling. For example, a mockup may match the structural properties of an actual product, but induce task procedures that will not be required in the final product. Thus, the designer must know to ignore irrelevant features of representations, a particularly difficult task when the representation is from a discipline other than one's own (Buur & Andreasen, 1989). This relates to the last demand.

The types of representation and languages for creating and manipulating models are practically boundless. Most are discipline-specific. Moreover, having emerged during the "craft stage" of their parent disciplines, most representations have no formal foundation in logic or mathematics (Webster, 1988). This property makes principled conversion of one representation into another tricky (Rinderle et al., 1989). Since translation is a problem, effective designers must be fluent in several representational languages (Eder, 1988; Meister, 1989). If the designer is unfamiliar with or does not value the concepts, data, methods, and design types from another discipline, two problems emerge. He is often cut off from a stock of ready-made and proven solutions that could be adapted to the problem at hand (Smith & Browne, in press). Also, if he creates representations at all, chances are they will be inappropriate (Lintz et al., 1971; Meister & Farr, 1967).

Working with multiple representations or "views" of the same object induces several demands for information management (Goel & Pirolli, 1989; Guindon, 1992). For instance, when the designer edits a system element in one view (e.g., structural), updates are invariably required in other views (e.g., kinematics, behavioral) of the same element. This is due to the systemic character of designed artifacts. This effect emphasizes the need for dependency tracking, version control, and change control, issues that have received a great deal of attention in engineering information systems in terms of storage requirements, tagging, security and other artifact-related matters (Linn & Winner, 1986; Roussopoulos et al., 1991; Salzberg & Watkins, 1990; Winner et al., 1988). We only add here that they suggest support requirements for the designer as well.

SUMMARY

Design is a complex goal-oriented activity that resolves problems which, by their nature, involve many conditions and stakeholders. Designers include a variety of individuals and groups who affect the form and function of the system in direct and indirect ways. This characteristic

introduces needs to collaborate and share information, processes which run into difficulties due the multi-disciplinary knowledge required.

Demands on individuals derive from uncertainty associated with what the problem to solve is, who will judge the eventual solution, and along what criteria they will base their judgments. Requirements and criteria for success tend to shift; the interdependencies among attributes of the artifact are complex. To cope with these complexities, designers must be fluent in a wide variety of methods and representational schemes with roots in different disciplines. Each method and representation tends to support only a narrow portion of the overall effort. Coupled with the artifact's emergence in stages, these characteristics force the designer to shift back and forth among them to make progress. Each shift potentially suspends a partially completed task, thereby introducing substantial task management problems.

This review of characteristics of design is relatively brief, and for more extensive coverage, readers are encouraged to explore the wealth of available literature on design (cf. Bruns & Gerhart, 1988). Traditional engineering (Dixon, 1966; Finger & Dixon, 1989a, 1989b) and architecture (Broadbent, 1988) have especially rich literature.

INFORMATION ACCESS AND UTILIZATION

The above characterization of design suggests that there is considerable variability among the types of problem faced in design. Since supports should match the need, this might be discouraging to those seeking to improve the design process -- who exactly does one support and how to assure improvement?

Fortunately, there is a common denominator. Across all designers, tasks, and methods of performance is the access and use of different types of information (Boff, 1987a; Rouse, 1986, 1987a). Access refers to activities that produce an object from the design information environment for subsequent use. Utilization refers to operations performed on or with objects to answer a variety of questions about the not yet finished artifact. "Will it be strong enough?" "What does or will the product look like?" "Why does performance deviate from expectations?" "How do users react to it?"

We focused on this general need in order to develop supports that would have broad appeal and, thus, a potentially broad impact on design. The following discussion examines information access and utilization in design, beginning with a characterization of the information world.

INFORMATION WORLD

Figure 6 depicts the world of design information. By any measure, this world contains a vast collection of time-varying data that are captured in numerous formats and media, distributed across organizations, and often specialized by discipline or problem. At the center are the designer's ideas. Around these ideas, information can be categorized into that which serves as input to the design process, that which is generated by the process, and output information that results from the process.

Input information includes both system-specific information such as marketing studies, system requirements, design rules, and management information about tasking, resources and people. It also includes a potentially massive amount of system independent information such as technology, past designs, and scientific data. Information generated includes sketches, diagrams, and system-specific models; analyses and results; and design documentation. Outputs include engineering drawings and parts lists; manufacturing plans for hardware and software; and design

artifacts, i.e., physical encodings of information. For our purposes, we decomposed this information world into four types of entity: artifacts, methods, tools, and adjunct information.

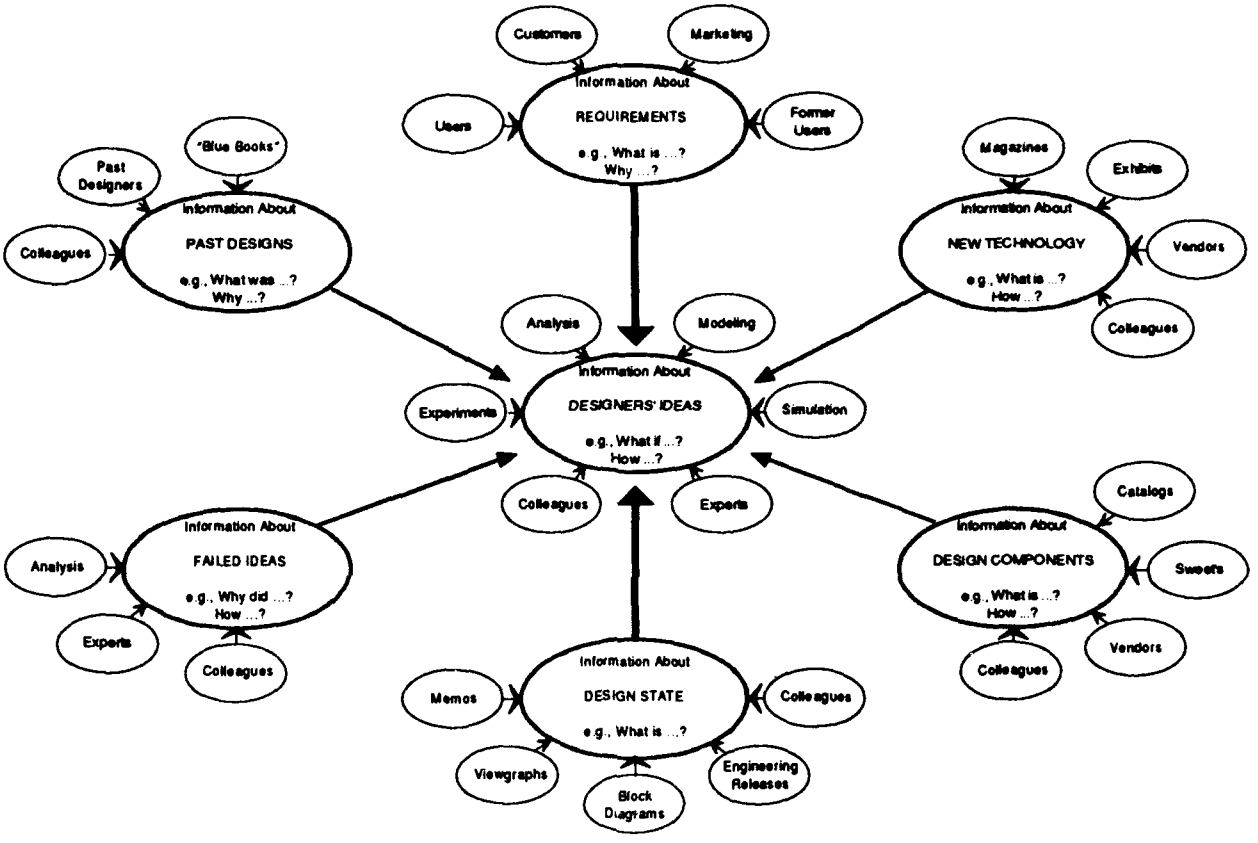


Figure 6. The designer's world contains a variety of types and sources of information.

Artifacts

Artifacts are complex abstractions composed of requirements statements, representations of function (e.g., I/O representations, computer models) and representations of form (e.g., drawings, mockups, prototypes). These data are expressed at several levels of aggregation (see Figure 3). Each representation captures a subset of attributes and values for these attributes. Values may reflect requirements and criteria, predictions based on modeling exercises, or

measurements from experimental tests. Attribute values can also be expressed as deviations from expected or desired states.

Artifacts also contain identification information such as a name, revision number, author, owner, and patent number. They contain annotations -- descriptions, explanations, critiques, test results. Constituent elements bear several types of relationships with one another, which also are part of the artifact description. Relationships include composition (A consists-of B; B is-part-of A), cross-representation (A transform from/to B), and utility (A uses B; B is-used-by A) (Linn & Winner, 1986).

Both nascent and existing artifacts can be included as members in the overall class. The principal distinction between nascent and existing artifacts is in their completion status, not their informational structure. Furthermore, our including both types of artifact under one class reflects our belief that access and utilization problems with present and past design are isomorphic.

Methods

A method refers to a combination of rules that describe preferred, default or mandatory operations that achieve a design task. Methods vary in their breadth of application from very general (e.g., systems engineering, spiral development) to specialized and narrow (e.g., particular analysis techniques). Usually the broader the method, the more likely its individual steps decompose into other methods. Also, methods may specify the use of particular tools and resources per step to produce specific outputs.

In any particular instantiation of a support system, the range of available methods may range from few to many. A broad range would include methods for requirements definition, functional synthesis, form synthesis, functional modeling and model processing, form prototyping, test and analysis. It would also include approaches to data analysis, information management (editing, storing, linking...) and design process management (planning, scheduling, monitoring). A narrow support system is also feasible in which only one or a few methods are represented.

Tools and Resources

Tools are manual or computer-based entities, plus resources, that enable execution of one or more steps in a method. Designers retrieve, modify or construct tools to facilitate development of the primary artifact. The methods that they support include representation (drawing, modeling,

prototyping), measurement, analysis, management of the design process (scheduling and resource allocation) and management of design information. Examples of design tools are drawing packages, modeling and computation packages, and prototype development packages. They also include general purpose and administrative supports such as spreadsheets and database managers. Tools often have particular disciplinary roots and are packaged with more or less usable information about their conceptual basis, instructions in use, and resource requirements.

There is a close correspondence between artifacts and tools. Some tools are reproductions of the artifact at some level of fidelity. Also, in terms of information structure, tools and artifacts may be isomorphic -- tools are composed of subsystems and modules, are driven by requirements, have an integration status, and so on. Moreover, as noted earlier, designers' activities that apply to artifacts also apply to tools. We separated tools from artifacts in recognition of the difference between expending project resources on the design of a tool versus on the design of the primary artifact.

Adjunct Information

The final class includes the vast store of additional information that designers may use. This includes requirements statements and descriptions of operational need; information about past designs with similar requirements, functionality or form; scientific and technical principles, laws, models, and data; company practices, industry standards, regulations, and guidelines; and sources for these data, including human sources. Most of these data are independent of the particular system under design.

Support information includes descriptions of the current project and past projects from a resources and management viewpoint. Elements include tasking, schedules, resources and their allocation, authority relationships, and revisions of these items across the project. Designers access these elements from past projects and the current project in order to adopt particular management strategies or assess efficiency of particular approaches.

For our purposes, it was useful to conceptualize the information world as though it were a massive centralized database. Clearly, this does not match reality. The information world is actually composed of many "data islands" of provincial databases and file systems (Rangan & Fulton, 1991). Some islands are accessible electronically, most are not. For developing support requirements, however, we were more concerned with overcoming conceptual barriers to access and use of the above objects than with present-day physical barriers.

METHODS OF ACCESS AND UTILIZATION

To obtain items from the above information world, the designer generally has at least three alternative means which can be applied to three primary sources. Figure 7 illustrates these relationships in terms of nine alternative access methods. Cell entries exemplify the types of task.

The designer can *retrieve* an existing object and use it for the new application, based perhaps on a correspondence between the new situation and situations he has experienced in the past (Chandrasakaran, 1990; Klein, 1987; Maher, 1990). The designer may retrieve and *modify* an object to suit present needs. This alternative induces the related need to access appropriate tools and information about the tools (e.g., user's manual) for such modifications. Finally, the designer can *construct* new information, perhaps by using a tool. Construction is more or less complicated depending on the type of object. For example, a sketch for a new system component may require only paper and pencil, whereas a precise geometric rendering may require sophisticated CAD tools.

Three primary sources of information are also depicted in Figure 7. *Human judgment* tends to be the preferred source (Rouse & Cody, 1989b). Designers satisfy most of their information requirements simply by recalling their own experiences, asking colleagues and talking with experts (Allen, 1977; Rouse & Cody, 1989b). Hence, the upper left-hand cell in Figure 7 represents a very popular means for producing needed information. The second source is the *archives*, which include handbooks, textbooks, journals, magazines, catalogs, company publications, and project databases. With the possible exception of trade magazines, designers access the archives only when human judgment appears to be inadequate (Gerstberger & Allen, 1968).

The third source of information is *models*. There are three types of model -- experiential, experimental, and analytical. When designers access a previous system, and the experience associated with it, as a baseline against which the new design is referenced, they are using an experiential model. This baseline is a model in the sense that at least a subset of its attributes are predicted to be relevant to the new design requirements. Of course, this prediction can be wrong if a poor baseline is chosen (Boff, 1987a).

Use of an experimental model involves producing the artifact and its conditions of use to some level of fidelity. In the case of human-machine systems, the conditions include subjects who

represent the eventual users. This is a model in the sense that the artifact, subjects and conditions provide a model of the future.

SOURCE OF INFORMATION		METHOD OF ACCESS		
		Retrieve Existing Object for New Use	Modify Existing Object for New Use	Construct New Object
Human Judgment		Adopt decision from prior case Ask colleagues for past examples	Analogize from previous case Brainstorm with colleague based on his experience	Reason from principles Guess
Archives		Conduct literature review; adopt object from reports	Modify object recalled from literature	Compose new paper regarding object
Models	Experiential	Adopt a baseline	Adapt a baseline	------
	Experimental	Re-use procedures and conditions from prior study	Modify procedures and conditions from prior study	Develop and execute new study to measure attribute
	Analytical	Re-use existing model from similar system	Retrieve and adapt existing model with appropriate tools	Develop new model with appropriate tools

Figure 7. Designers have several alternative methods and sources for accessing information.

Analytical models are typically embedded in computational tools. Use of these tools involves describing relationships at some level of abstraction (e.g., functional input-output relationships, geometric relationships in structural drawings). Parameters within the representation are then chosen, inputs are specified, and outputs computed.

The choice among these alternative sources determines which information will influence the designer's thinking and decisions, and which will not. For example, if the designer chooses to use personal judgment over another source, then that second source is blocked from influencing his design decisions. To understand why some information is accessed and other information is not, we need to examine the value of information in design.

THE VALUE OF INFORMATION

"Value of information" can be a rather elusive construct (O'Reilly, 1982; Rouse, 1986a). On the one hand, the issue seems straightforward. Value is what one is willing to pay in money or effort. This definition is reasonable for some purposes. However, it is not directly useful for developing support system requirements. To be useful, the concept of value must be defined in terms of benefits the designer experiences from using the information and the costs associated with acquiring it. In this regard, it is useful to think of design as a decision making process under conditions of uncertainty (Hazelrigg, 1988). Certainly this image corresponds with the characterization provided earlier in this report.

Benefits of Information

One benefit of having information related to an issue is that it reduces uncertainty. This may occur when the user is informed of something new or reminded of something forgotten. Uncertainty may also be reduced if the need is to have "information about information," as was discussed earlier in regard to recursion in design.

Designers may be uncertain about a variety of topics, as is illustrated by Figure 8. It shows 26 types of question, grouped into four classes, that we derived from an extensive observational study of a group that was involved in the early stages of designing a new aircraft cockpit (Sewell, 1990). Some questions focus on requirements, others on ideas for satisfying requirements, and others involve the artifact as it takes form and is evaluated and fielded. Our expectation is that information which addresses these questions and, thereby, reduces the designer's uncertainty would be perceived as beneficial.

Uncertainty reduction is necessary but may not be sufficient for the designer to perceive the benefits of some class of information. This assessment is also governed by his belief about the relative importance of the issue over which there is uncertainty (Gemunden, 1985). Issues in design are not all equally important. The payoff for being "correct" can range from trivial to enormous and the risk of being wrong can range from none to catastrophic. Whether right or wrong in his beliefs about the importance of an issue, the designer makes this assessment and, thereby, determines whether information will be useful or not. Our expectation is that failures of cross-disciplinary information transfer can be traced to designers in the "receiver" discipline discounting the importance of issues from the "sender" discipline (Lintz et al., 1971; Meister & Farr, 1967).

- **WHAT....?**

1. What were the problems with the past design?
2. What were the requirements for the past design?
3. What are the requirements for the new design?
4. What new technologies are available?
5. What off-the-shelf components are available?
6. What are the capabilities of these components?
7. What are the performance limitations of...?
8. What is the current state of the design?
   - Configuration, function allocation, physical form, schedule status...
9. What solutions to problem X are possible?

- **WHAT IF....?**

10. What will be the impact of function/form on...?
    - Performance, operability, supportability, schedule, cost....
11. What criteria will be affected by function/form?
12. What effect will this idea have on compatibility with the current design configuration?
13. What effect will this idea have on compliance with design requirements?
14. What will the design support system do if I do...?

- **HOW....?**

15. How did the past design function?
16. How can new technology be used?
17. How can available off-the-shelf components be used?
18. How can a particular "what if" question be answered?
19. How can a particular "why" question be answered?
20. How can the state of the design be updated?
21. How can the design support system be used?

- **WHY....?**

22. Why did problems occur with past designs?
23. Why are the requirements for the new design as specified?
24. Why are the results of a "what if" question other than hoped?
25. Why did the state of the design change?
26. Why did the design support system respond as it did?

Figure 8. Uncertainty in design is associated with several issues.

Figure 9 summarizes these observations about benefits in terms of a "*perceived* risk space." The axes show the designer's uncertainty associated with an issue and his beliefs about the importance (desirability) of the consequence. The corners of the space are given concrete meaning with examples. Greatest uncertainty (probability = .5) is associated with events along the central horizontal axis; the most extreme consequences occur at the left and right boundaries of the space. The two darkened regions represent situations in which the designer perceives the greatest degree of risk; this perception diminishes with distance from these regions.[3]

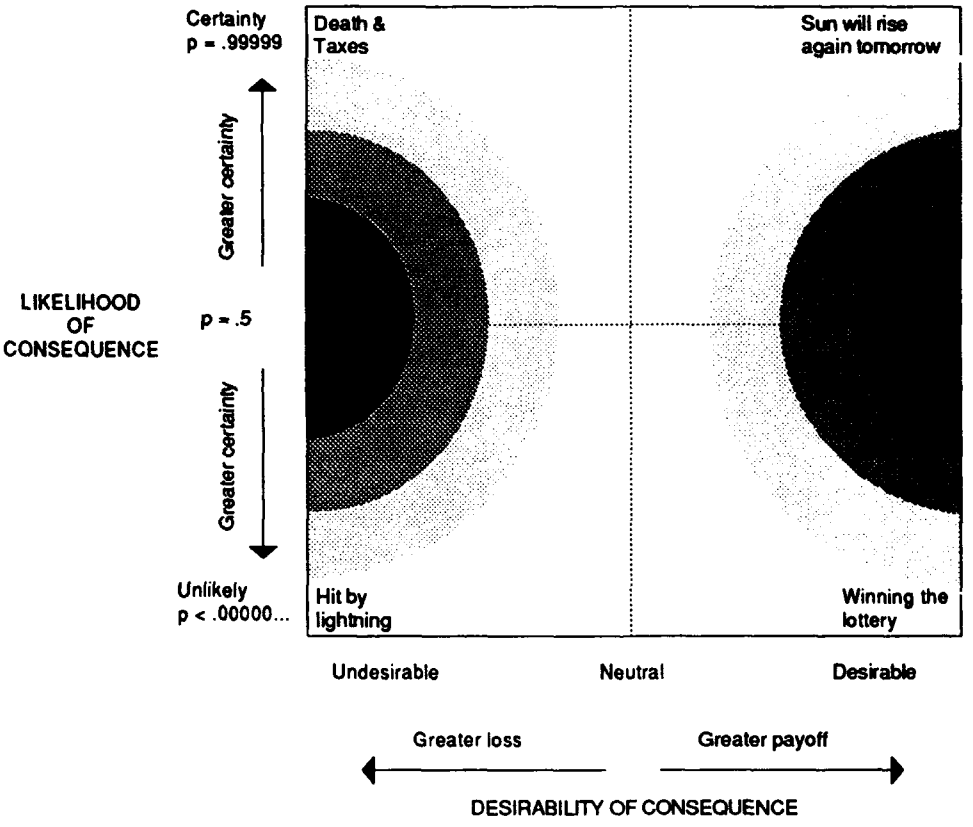


Figure 9. Perceived risk is a function of uncertainty and importance of the consequences.

[3] Although shown as symmetric, the region associated with uncertain/undesirable events is likely to draw more of the designer's attention than the uncertain/desirable region.

The designer realizes benefits with movement from one location to another in this space, and information is the vehicle for doing so. We hypothesize that, from a given starting point, the direction and magnitude of change afforded by a given class of information are directly related with the designer's perception of benefit. Information that reduces uncertainty will be valued more than that which increases uncertainty.[4] Moreover, movement toward greater certainty will be valued more when the issue is associated with extreme consequences. Thus, equivalent shifts in uncertainty at the extremes of the desirability axis will be seen as more beneficial than those in neutral regions.

Costs of Information

The second component that determines the value of information are costs associated with its access and use. Costs could be measured in terms of money, but in the present context, they are more likely measured in terms of effort required to obtain, interpret and determine the implications to design choices of a class of information. Thus, information can be such that it reduces uncertainty and pertains to an issue that the designer believes to be important, but nevertheless is not valued because it is difficult to obtain or use (Boff, 1987a; Rouse, 1986).

Information may be difficult to obtain if it is not available through electronic means, is proprietary to another group, is out of print, or has not yet become available. Information may be difficult to use for several reasons. It may be expressed in a form that is inappropriate for design. A typical example in design occurs when the designer receives qualitative information when quantitative information was sought. Information can also be difficult to use if its disciplinary origins are different from the designer's and contains unfamiliar concepts and jargon.

For some information, it may be possible for the designer to transform it so as to make it useful. However, the effort needed to make this transformation is likely to be perceived as greater than the benefit to be obtained, especially if the issue is not perceived to be important. Unfortunately, designers perceive much of the research literature suffering from these problems (Rouse, 1986).

---

[4] In the early stages of design, greater uncertainty in the form of having several feasible, rather than fewer, candidate designs may actually be valued. To be consistent with the benefit space, the condition of having "too few alternatives" might be seen as involving greater uncertainty about viable approaches that is the condition of having several alternatives. As design proceeds, and deadlines approach, more (un-eliminated) alternatives are seen as undesirable.

Figure 10 summarizes the four factors that are assumed to contribute to the perceived value of information. Reduction of uncertainty and issue importance affect utility, whereas ease of access and transformation requirements govern perceived usability. In turn, these two interim dimensions control perceived value. As Figure 10 suggests, threshold values exist for both usability and utility; below either of these levels, information will simply be discarded (Boff, 1990).

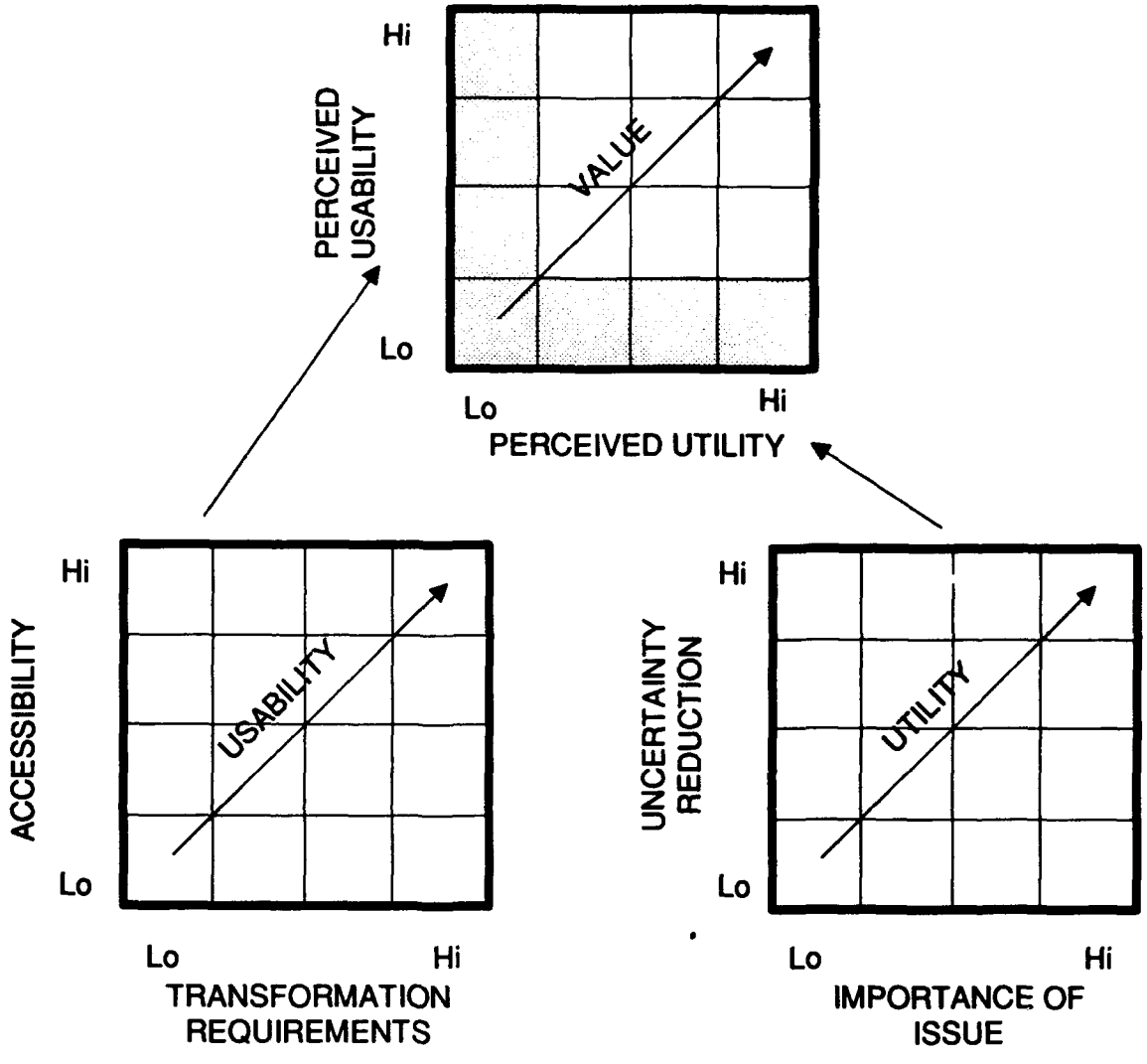


Figure 10. Perceived value of information is a function of perceived usability and utility.

The value of a given piece of information is not static. Value is inherently time-varying because uncertainty changes as design choices are made. Figure 11 illustrates this notion. Moving around the lower loop first, differences between his current knowledge state and goal create uncertainty in the designer. The magnitude and specific properties of this gap lead to consideration of what, if any, information is needed and how it might be obtained. Assuming uncertainty exceeds some threshold, the designer chooses a means (Figure 7) and acts. He then uses the information which changes his state of knowledge, presumably in the direction of the knowledge goal and, thereby, reduces the size of the uncertainty gap.

It is always possible that new information will actually increase uncertainty, if not with the issue at hand, with some related matter (Gemunden, 1985). In part, this accounts for the apparently chaotic behavior that is seen in design. For instance, the designer may work on some element of the solution until encountering implications for other elements. This realization induces uncertainty, not about the partial solution necessarily, but about other design options and commitments. This new state of uncertainty then triggers a "move" to some other level of abstraction or aggregation, which is made manifest by the designer seeking new information. Guindon (1992), Malhotra and his colleagues (1980), and Schon (1983) have reported observations of designer behavior that are consistent with this interpretation. As when working with multiple representations, abrupt shifts around the design space to chase down new uncertainties creates a trail of partial solutions and suspended tasks which then must be managed.

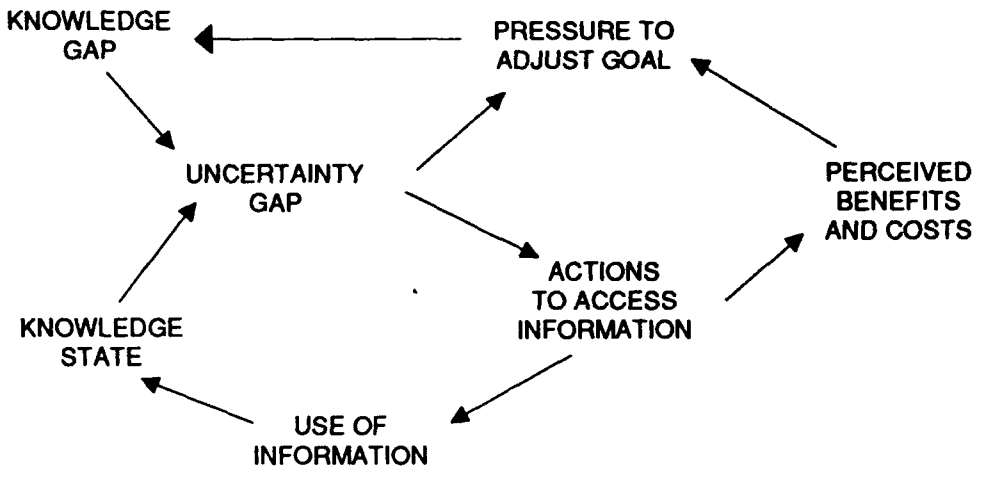


Figure 11. The value of information changes dynamically with feedback

The upper loop of Figure 11 suggests that, before acting, the designer evaluates the potential benefits and costs associated with some class of information. The designer must expect the information will reduce uncertainty, pertain to an important issue, be accessible and transformable. (Rouse, 1986). Beyond these criteria, which seldom reduce the number of alternatives to a single choice, other attributes such as access time and cost, perceived validity, and disciplinary compatibility (i.e., concepts, jargon), can also affect the choice (Cody & Rouse, 1989). Furthermore, designers' selection criteria are likely to vary from one situation to another (Rouse, 1986). For example, alternatives that a designer chooses when exploring what system requirements mean are likely to be quite different from the options he chooses when trying to confirm their interpretation with a particular stakeholder.

If costs are perceived to exceed benefits, the designer may switch to a lower-cost approach. Alternatively, he may abandon the search, and instead reduce uncertainty by discounting the knowledge goal (Gemunden, 1985; Lintz et al., 1971; Meister & Farr, 1967). As commonly occurs with information that they consider to be discretionary, designers may conclude that the trouble associated with getting information that is not readily available, in the proper form, etc., is simply not worth the effort (Rouse, 1986). As alluded to earlier, for most design issues, human judgment -- the designer's own or those of colleagues -- is perceived to yield the best value in terms of the cost of access, ease of application and consequent benefits of uncertainty reduction.

In summary, information requirements vary across design tasks. Designers evaluate sources of information in terms of their potential to reduce uncertainty, apply to an important concern, be accessible, and be in an appropriate form. As a result, the perceived value of a given piece of information can range from indispensable to irrelevant, and this assessment varies over time. For better or worse, the designer makes this judgment. The key lesson is that value is not intrinsic to the information itself. It depends totally on the designer's situation, intentions and perceptions -- value is in the eye of the beholder (Boff, 1988).

IMPLICATIONS FOR SUPPORT

The implications of this analysis for design support are threefold. First, designers access a wide variety of information, and some classes more routinely than others. Routine items include project-specific information such as requirements documents, artifact representations, and project management data. At a minimum, a system that supports information access and use in general should enable the user to access these highly valued items.

Second, to deliver relevant information beyond these items, the system needs several pieces of data: what task is being performed (Figure 4) and what type of representation(s) is (are) in use (Figure 5); what type of question is being asked and what type of answer is sought (Figure 8); and where in the abstraction/aggregation hierarchy (Figure 3) the current task is located. Without these data, information returned by a system might be trivial at best and, more likely, completely off target. This suggests that the support system must be told, or be capable of inferring, what the designer is attempting to do in a very context-specific manner.

Finally, to encourage the designer's consideration of a particular class of information, the system must increase the perceived benefits associated with its use and/or reduce the perceived costs with obtaining and applying it (Figure 10). Perceived benefits can be enhanced by making apparent to the designer how the information can reduce uncertainty about the question he currently is addressing. It is also possible to increase perceived benefits by suggesting how his failing to consider the information increases his risk of a bad decision (Figure 9). Nonspecific warnings such as "failure to consider this information could lead to problems," however, are unlikely to meet with success -- the system must help the designer determine the implications for his particular question or decision.

Perceived costs can be reduced in several ways, all of which presuppose that the system *will* have access to information. In response to the designer's specific requests, the system could decrease perceived costs by executing appropriate protocols to retrieve the desired information. The system could also help the designer formulate questions regarding the design issue at hand that he may not otherwise ask due to his background or inclinations. Along with this support, the system could help him express information needs in a manner that is compatible with the available sources. At an even more sophisticated level, the system could identify and retrieve information in anticipation of the designer's needs. This service would require the system to be capable of predicting what the designer will do next based on his present context, task, question, etc. Beyond identifying and retrieving items, perceived costs associated with transforming information might also be reduced. Transformations could be carried out at the user's command, in accord to stored preferences, or in anticipation of user needs.

SYSTEM REQUIREMENTS ANALYSIS

The above implications for support are too general to serve as system requirements. A more detailed analysis is needed if operationally meaningful specifications are to emerge. Toward this end, we utilized an analysis method that we have found useful for designing information systems for applications in command and control (Rouse & Rouse, 1983), nuclear power (Rouse et al., 1984), and manufacturing (Rouse, 1988). The methodology begins by making explicit an overall philosophy for supporting humans in complex systems. Along with the above analysis of information access and utilization, this philosophy helps to define support system goals. Given goals, the method then identifies the functional requirements to achieve them.

SUPPORT PHILOSOPHY

Traditional approaches to developing support systems for designers can be described as artifact-driven. This involves analyzing the artifact -- what forms it can take and how it evolves from one representation to the next (e.g., requirements to functional spec, spec to drawings, etc.). Once the representations are understood, one concentrates on means for increasing the speed and efficiency of the evolution. Moreover, with advancements in automation, the design goal is often to replace the human skill, judgment and even creativity where possible (cf. Finger & Dixon, 1989a, 1989b). Resulting tools are then presented to users with the hope that they will be useful and that design outcomes will improve.

Clearly, developers who adopt an artifact-driven approach want to build products that designers value and use. The approach, however, can result in supports that devalue the human's contribution, do not address the need, are hard to use, or are not viable for economic or organizational reasons.

To avoid these difficulties, we adopted a human-centered approach to design support (cf. Rouse, 1991). The foundation for this approach is a belief that people, not their tools nor the tool designer, are responsible for achieving system objectives. Put simply, people have to "be in charge." This simple philosophy leads to the conclusion that the goal of a support system is *not* to achieve the human user's objectives for him. Rather, the purpose is to enhance the user's abilities and overcome his limitations so that he can achieve the objectives himself. The philosophy applies

equally well to people who design cockpits for pilots as it does to people who design supports for other designers.

To some, this may seem to be a minor semantic distinction. However, there are major differences between supports that derive from artifact-driven and human-centered approaches. Focusing directly on the artifact, some would conclude from the above analysis of design problem solving that the designer is the source of the problem. The conclusion might be to remove the designer from the process where possible and create autonomous systems that will "do design" (Chandrasakaran, 1990; Mostow, 1985; Newsome, Spillers & Finger, 1989; Steier, 1990). Such systems would interact with human users only to the extent that they require inputs and occasional adjustment to keep them on track. In this scheme, the human designer is relegated to monitoring progress and making adjustments. "Support" equates to reducing human discretion toward the goal of better products. Certainly, this vision of designing is far into the future, but imagines the human serving the machine rather than vice versa.

The alternative, human-centered vision is of a machine subservient to the designer who solves the problem and creates the artifact. The designer always maintains authority to reject, override, or modify suggestions or actions taken by the support system, even at the risk of a poorer product. The reason is that, while it is imaginable, and in some cases inevitable, that automation will match or even surpass human capabilities in specific problem-solving tasks, it is not imaginable that automation will ever be held legally, ethically, or socially responsible for its actions. Thus, regardless of the level of automation, people will remain responsible for the design decisions that they make (Rouse, 1991).

In sum, the human-centered philosophy suggests how responsibilities should be allocated between designer and support system. The goal of design is to specify artifacts whose functions and attributes satisfy requirements and criteria held by stakeholders in the effort. The role of the designer is to focus on stakeholders' concerns and the artifact. The role of the support system is not to optimize the artifact. Its role is to improve the designer. The system should make the designer aware of pertinent information, aware of the effects of his design decisions, and confident about options he selects. It should help him to avoid errors in approaches and tools of his choosing, and to detect artifact violations that he can choose to ignore or address. This image of support helps to establish system goals, to which we now turn.

SYSTEM GOALS

There are generally several ways to achieve a set of functional requirements. Decisions made between functionally equivalent alternatives often are based on philosophical design goals. Such goals indicate a direction to be followed as opposed to strict criteria to be met. Based on our analysis of information access and utilization in design, the following classes of support goals were identified.

Goals Related to Access

- Match the types of information sought to the user's task, context, personal preferences and intentions.
- Help the user to formulate an appropriate and complete set of questions regarding the design issue at hand.
- Help the user to express information needs in a manner that is compatible with sources.
- Adapt the amount of information that the user must supply to the system to the degree of system output desired. (context responsive; personalized; intent responsive); therefore, keep up-to-date with designer's mental model of his circumstances.
- Enhance the likelihood that the user is aware of the full range of relevant information to his current circumstances.
- Answer the user's specific questions regardless of the level at which they are asked.
- Base responses to user questions and requests on as broad a knowledge base as possible.
- Anticipate the user's information needs.
- Provide tutoring in how to access information sources, as requested by the user.

Goals Related to Utilization

- Enhance the user's ability to understand information from disciplines other than his own.
- Support use of methodologies for applying information to design decisions.
- Help the user to determine the implications of information to his particular design problem or question.
- Enhance the likelihood that system design choices will reflect consideration of relevant information that is available to the system.
- To promote understanding, transform information to match the user's preferences, style, or discipline (adaptation *to* the user).

- To promote understanding and use of relevant information, teach the user about concepts, principles, methods, and data from the domain (adaptation *of* the user).
- Support integrating information into the design database.

General System Goals

- Allow the user to set the support system's level of autonomy on a task by task basis.
- Ensure non-redundant interaction between the user and system, i.e., provide support based on stored information about the design state and the user's intentions.
- Enhance cooperative work among distributed and autonomous design team members.
- Accommodate as many different types of user as possible.
- Impose minimal administrative burden on users and organizations with regard to installing, maintaining and using the system.

METHOD TO IDENTIFY SYSTEM FUNCTIONS

What should the support system do to achieve these goals? This section summarizes the methodology we used to answer this question.[5] It is important to note that the methodology does not identify *how* supports might be realized. Rather, it defines functional requirements based solely on perceived need. Implementation considerations for one particular approach are covered in the next section of this report.

Step 1: Define Tasks. This step is concerned with identifying the tasks to be supported. Based on our analysis of design, we adopted the 12 tasks shown in Figure 4 as the targets for support, each of which could occur at various levels of aggregation. We also identified 36 types of object to which these tasks apply. Figure 12 shows these objects organized according to level of abstraction. The meaning of these objects are quite straightforward, with one exception. Both the singular and plural of drawings of form, model of functionality, and prototype appear in these lists. The plural refers to retrieving relevant candidates, while the singular refers to creating a candidate

Step 2: Map to General Tasks. This step involved mapping the design-specific tasks to a set of 13 domain-independent tasks that characterize human interaction in complex systems. These general tasks, and 17 associated forms of support, emerged from an extensive analysis of over

[5] The methodology is elaborated more fully in Rouse (1991).

100 support systems (Rouse & Rouse, 1983). Thus, the general tasks provide an intervening mechanism for linking the design-specific tasks in Step 1 to potential forms of support.

PURPOSE	FUNCTION	FORM
Requirements for past designs Information on operational need Requirements for current design Past designs (requirements) Performance attributes and criteria	Functionality of past designs Information on functions Explanations of functions Past designs (functions) Off-the-shelf functions Input/output representations Relevant input/output representations Modeling tools/packages Model of functionality Models of functionality Model's variables Model's predictions Deviations of predicted performance	Forms of past designs Information on forms Explanations of form Past design (forms) Off-the-shelf forms Drawing tools/packages Drawings of form Drawings of forms Prototyping tools/packages Prototype Prototypes/mockups Off-the-shelf prototypes Experimental variables Data collection plan Performance Performance data Measured performance Deviations of measured performance

Figure 12. Primary objects in design can be classified into three types

Two analysts independently performed the mapping from each of the 12 design tasks to the 13 general tasks. Results were then compared and consolidated, yielding 64 links, or an average of somewhat over 5 links per design task. Each link was annotated with the reason for the link and a design-specific interpretation of the connection. This set of 64 design-specific instances of the general tasks served as the input to the next step.

Step 3: Map to Limitations. The third step considered limitations that designers are likely to exhibit in performing the 64 tasks. Limitations can be characterized in more than one way. Sage (1981), Silverman (1990) and Zachary (1986) have considered *psychological limitations* associated with decision making and problem solving in complex systems. The difficulty with tabulations of psychological limitations for developing support systems is that they are very

general. With a little stretching, it was easy to imagine mapping each of the 64 tasks to all of the psychological limitations. Such a result was not considered useful. We found it more useful to think in terms of *task limitations* that were tailored to the context of interest -- information access and utilization that support each of the 64 design tasks. Figure 12 presents ten limitations that we considered, five related to access and five related to utilization.

We reviewed each of the 64 tasks to determine whether or not each of the 10 limitations in Figure 13 was likely to affect the task. We assumed a journeyman designer and concluded that the "why" and "when" limitations in Figure 13 would not be relevant. Applying each of the remaining 7 limitations to the 64 tasks resulted in 213 task-limitation pairs that represented potential support requirements.

Step 4: Requirements Analysis. We then analyzed each of the 213 requirements to determine in a very design-specific manner what needed to be done to overcome the limitation. This was accomplished by reviewing each of the 213 in terms of the nature of the limitation (Figure 13), general task involved, and design task involved (Figure 4). Results from this analysis were used to create a computer-readable database describing the requirements to overcome each relevant combination of limitation, general task, and design task.

Step 5: Clustering Requirements. The 213 requirements, including the associated context-specific interpretations, were sorted into clusters with common limitations and general task attributes. This type of sorting was needed in order to map to the aforementioned list of support concepts which is indexed by general tasks. Limitations were used as a second sorting attribute because the interpretation of support concepts is influenced by the nature of the limitation that the support is to help overcome. This process resulted in 43 clusters of requirements which served as input to the next step.

Step 6: Map to Support Concepts. Each of the 43 clusters was then mapped to one or more of 17 general support concepts (see Appendix A; also see Rouse, (1991) for the most recent exposition of these concepts) . This mapping was based on the general task associated with the cluster. Each potential support concept was then interpreted within the context of the limitation associated with the cluster, as well as the design tasks (Figure 4) with links to this cluster.

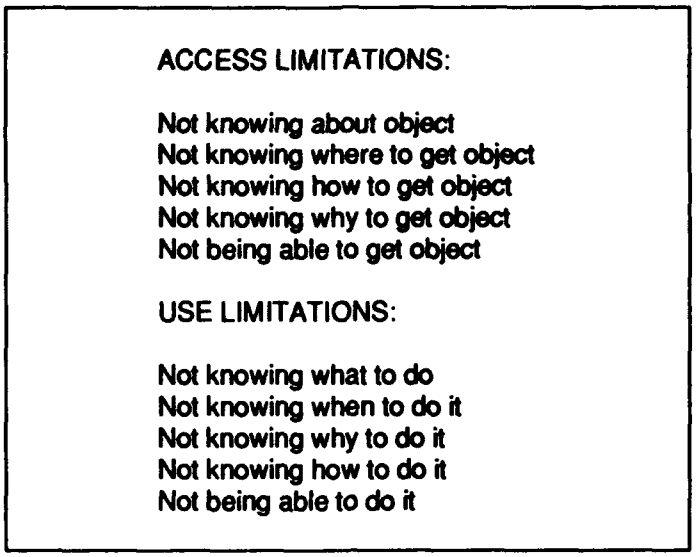


Figure 13. Designer's potential limitations apply to both access and use of information.

We found it necessary to develop a structured vocabulary for expressing supports to perform this analysis. Each instance of support was composed of four types of terms:

- Support verb — Actions of the information system to support the designer.
- Designer verb — Actions by the designer that are enhanced by the support system's actions.
- Primary object — Object of the designer's actions (Figure 12)
- Modifying object — Noun plus preposition that modifies a primary object.

Figures 14 and 15 show the specific terms and definitions for support verbs and designer verbs, respectively, that we used. After several iterations, 613 instances of support resulted. The complete list appears in Appendix B, organized according to support system action. An example entry is: (Execute procedure to) (access) (drawing tools/packages).

It is interesting to note that with 6 support verbs, 10 designer verbs, 8 modifying objects, and 36 primary objects, there were over 17,000 possible combinations. The 613 actual instances of support account for roughly 3.5% of these alternatives. Thus, the results of the analysis represent a much more structured and focused conclusion than a purely combinatorial tour de force would indicate.

Search	Use attributes or labels to identify and locate objects.
Execute	Perform procedures to access, construct, evaluate, measure, obtain, run, and select objects.
Indicate	Display variables, relevant procedures,, necessary activities, etc.
Transform	Modify, filter, and highlight observed or computed variables.
Explain	Interpret procedures, measures, variables, explanations, transform, etc.
Tutor	Coach in the use of procedures to access, construct, evaluate etc.

Figure 14. The support system performs six major operations.

Access	Open and manipulate an artifact representation or tool.
Construct	Create or modify an entity; meant to include *construction* of artifacts and tools as well as *generation* of methods and information (e.g., explanations).
Evaluate	Assess attributes of an entity through reasoning, prediction, or measurement; includes notions of comparing objects with one another or requirements.
Identify	Determine the existence of and label for an entity.
Locate	Determine whereabouts of an entity.
Measure	Determine performance through observation.
Monitor	Observe processing or execution of methods and tools.
Obtain	Acquire custody of an entity including artifacts, tools, and information about these objects.
Run	Execute a functional model to generate information.
Select	Choose among alternatives.

Figure 15. Designers perform ten operations in accessing and using information.

Step 7: Cluster Support Concepts and Compose Requirements Statements. Finally, each of the 613 supports was annotated with whether it was associated with inputs or outputs from a support system. The data base was then sorted along this dimension, and the resulting clusters were used to composed a consolidated list of 36 requirements.

To perform this step, we adopted a general framework described by Riley (1989) for analyzing human-support system interactions.[6] Figure 16 depicts this framework. It shows the information world, designer, and support system interact via three communication loops. In two autonomous inner loops, the design world provides information independently to the designer and support system, and both parties provide information back in the form of actions and procedures. In an outer cooperative loop, the designer and support system interact with one another through various interface mechanisms.

Focusing first on the designer, he receives inputs from the design world via artifact representations, reports, tools, people, etc. (see Figure 6). He also receives input directly from support system displays and indirectly by perceiving actions that the support system takes on the design world. Based on this information, he infers the states of design world and support system, and plans his next move according to his internal goals. The designer's outputs include changes to design entities (in either temporary or permanent store) and selections of functions of active tools, including the support system.

The support system exhibits analogous activity. The lower path through the support system input quadrant is devoted to sensing and interpreting changes that take place in the design world; the upper path acts on information from or about the designer. This information is used to plan actions that will support the designer according to a set of goals. In the output quadrant, the support system acts on the design world, such as to retrieve information or run a model, and constructs displays for the designer.

We grouped the 613 requirements for supporting information access and utilization according to four main processing elements depicted in Figure 16: monitoring the designer, monitoring the design world, acting on the design world and constructing displays for the designer. Results from this final step of analysis are presented next.

---

[6] The consolidated list of requirements which emerged from this step of the analysis can be achieved by a human-based system, machine-based system, and various mixes of these two. The scheme shown in Figure 16 does not presuppose a specific implementation.

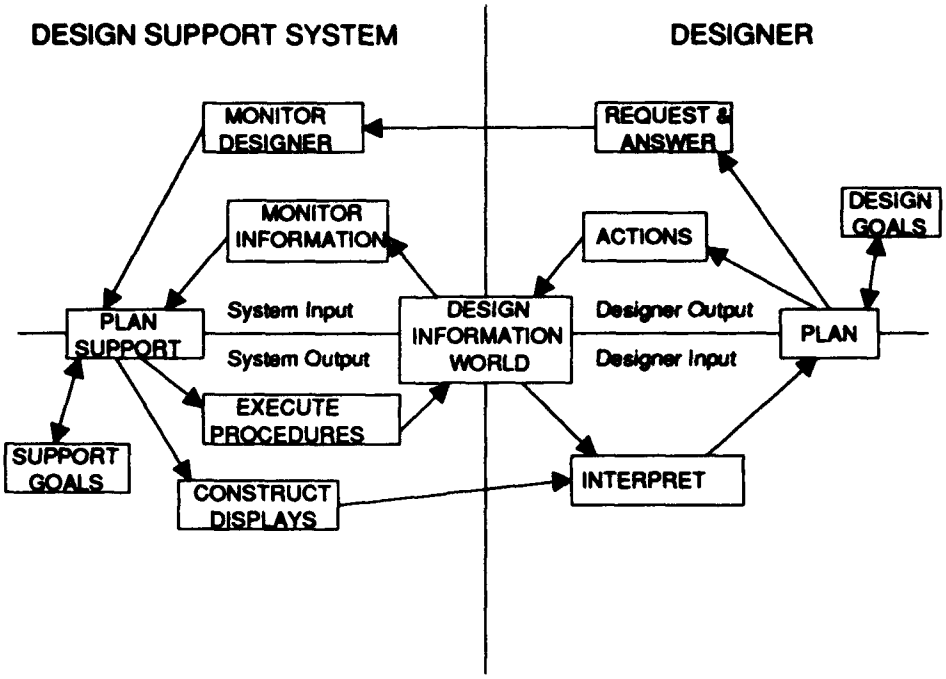


Figure 16. The designer and support system interact with each other and independently on the design information world.

CONSOLIDATED REQUIREMENTS

1.0 Monitoring the Designer

The first category of requirements for the support system is concerned with sensing and interpreting the designer's actions. His actions include those directed at the design information world as well as those issued directly to the support system (e.g., requests for processing or explanation). A computer-based system might monitor the designer's behavior in surrogate form by identifying which representations, tools, tool modes, information sources, etc., the designer currently has active as well as the changes these entities exhibit at the hands of the designer. Assuming that a means to interpret the designer's actions is available, the system could deduce from changes in these entities what the designer is attempting to do. Interpretation of the designer's activities serves to identify his goals, current task queue, possible errors and misunderstandings. Outputs from the monitoring function guide the associate's support planning process.

1.1 A system for supporting access and utilization of design information must be able to monitor the designer's actions, or be able to construct a facsimile of his actions (e.g., from information provided directly by designer), with respect to tools, artifacts and information sources. Monitoring the designer's activities is necessary for interpreting his intentions *and, therefore, his information needs.* In addition, monitoring the designer's activity relative to a known method or tool enables detecting inconsistencies and anomalies that may be indicative of inadequate, inappropriate, or incorrect application of the method or tool.

1.2 The system must be able to identify knowledge and skill required to apply a method or use a tool in order to support explanation, tutoring or augmenting the designer's performance.

1.3 The system must be able to identify tools the designer is using. Identification includes the name, version, type, resource requirements and tool purpose. Identification enables context-sensitive support in the forms of accessing needed resources for tool use, checking for inappropriate use, explaining tool behavior to the designer, and taking the initiative in applying tool functionality on behalf of the designer.

1.4 From information about the situation and the designer's actions, the support system must be able to identify the active set of methods that characterize the designer's behavior.

1.5 For a given method, the system must be able to track steps that the designer has completed, has suspended, and has not yet started. Also, it must be able to manage information about suspended tasks and tasks not yet started in order to notify the designer of incomplete tasks or, if capable, to take the initiative in completing the task(s) itself.

1.6 For a given method and information about designer's actions, the support system must be able to monitor execution for inconsistencies to enable construction of appropriate messages (e.g., queries about his intentions) to the designer.

2.0 Monitoring the Design Information World

This category of requirements specifies the support system's ability to track and explain changes in the design information world, to evaluate entities (candidate designs, tools, test results, information sources, etc.), and to select from among alternative entities according to rules. We assume that the information world could be affected by many contributors who are distributed organizationally and across time. Requirements in this category are independent of any one designer's actions. To track and interpret design state changes, a support system must be able to reason about abstractions such as "requirements," "criteria," and "specifications," and relate changes in design representations to these concepts. It must also be capable of locating information that pertains to these abstractions.

2.1 The support system must be able to identify the origin of requirements as explicitly given, inferred, or discovered following design decisions.

2.2 For a given situation and set of candidate information sources about requirements, function and form of a design, the system must be able to evaluate alternative information

sources and select the source that best matches the designer's needs. This capability includes assessing the appropriateness of tools to tasks.

2.3 Given a representation of requirements to be achieved, the system must be able to construct, evaluate and select explanations of the requirements for the current design and past designs.

2.4 For a given artifact, the support system must be able to construct, evaluate and select explanations of the artifact's functional and structural composition. This capability enables the support system to explain or tutor the designer in why an artifact was designed in a particular way and how it works.

2.5 For given requirements, resource commitments, and a candidate set of design alternatives, the system must be able to evaluate functional and structural design representations for inconsistencies, requirements violations, as well as stage of completeness (e.g., marking requirements as "met," "in negotiation," "waived,"...). Given the evaluation, the system must also be able to select the representations that best reflect the requirements. These capabilities support notifying the designer of conflicts between requirements and system state should he express an interest in such information.

2.6 Given a set of requirements and criteria, and results from modeling exercises or experimental tests with the artifact, the support system must be able to determine deviations of attribute values from expected values, and accept or reject the validity and reliability of these values. The system must also be able to construct, evaluate and select explanations of deviations between tested and expected attribute values.

2.7 The support system must be able to monitor the processing of tools in order to detect computational anomalies and error conditions.

3.0 Acting on the Design Information World

The third set of requirements pertains to the system's capacity to manipulate the design information world. Depending on its level of autonomy (see below under "mixed-initiative systems) , these manipulations range from simple searching and information retrieval from on-line bases, to assimilating information created by the designer into the artifact base, to actually applying design methods and tools to change or elaborate the artifact (e.g., filling in design details). The requirements also cover the planning process whereby the support system constructs action plans for the designer's review and approval.

3.1 The support system must be able to take initiative on tasks it has the authority to do so as a result of four types of triggering condition:

- direct designer command
- by recognizing the occurrence of specific situations that the designer has identified a priori to require DA action
- by inferring the designer's intentions

- by recognizing situations that, in past exchanges with the designer, resulted in the support system action. Thus, the support system must be able to store previous sessions with the designer and adapt its behavior based on prior experience.)

3.2 The system must be able to search the design information world for relevant entities based on attribute values or by name, and establish their existence and location for potential retrieval. Entities include artifacts, methods or procedures, tools and supporting information. Having established existence and location, the system must be able to retrieve or access the entity for use. Note, the remaining requirements in this section all are specializations of this general statement.

3.3 The support system must be able to retrieve requirements information for the current and past designs. In addition, it must support editing and reorganizing requirements as incompleteness and ambiguity of the problem specification are reduced across the design process.

3.4 The system must be able to retrieve representations of past designs based on attribute values (e.g., "all designs that met requirement X") or label (e.g., the XMD 7401 helmet-mounted display tube). For attribute searches, similarity may be based on requirements, functionality, or structural properties. As with many of the operations in a mixed-initiative system, this action merely delivers one or more candidate entities for subsequent evaluation. Evaluation may then be conducted by the designer, by the support system or cooperatively.

Retrieval will depend upon several operations. These include capabilities to formulate searches, identify attributes and labels, locate objects, retrieve objects and not just pointers to object, retrieve relevant related objects, and adjust the retrieval strategy based on user evaluation of results.

3.5 The support system must be able to identify and retrieve methods for constructing artifact representations, methods for developing explanations, and methods for evaluating entities.

3.6 Related to this, the system must be able to locate and access tools, plus the resources necessary to use them (e.g., remote computing hardware), that are called out by methods. Tool execution and management will depend on several capabilities. These include the ability to estimate processing requirements, download tools, bind downloaded objects to tools, check versions in and out, access the necessary resources for tool execution, manage tool I/O and data transfer, and handle host-specific commands and quirks. We assume that an actual system will operate within an engineering data environment that enables these functions.

3.7 The support system must be able to apply tools autonomously to construct drawings, models and prototypes of the artifact at all levels of aggregation. In keeping with the overall support philosophy, the designer has the authority to override autonomous operations by the support system, including artifact constructions.

3.8. The system must be able to construct data collection plans.

3.9 The system must be able to retrieve or construct, and then run models of artifact functionality to produce data and information. This requirement applies to calculation in general. The sense is that the support system must be capable of exercising, perhaps in the background, the same tools that the designer uses in order to predict attribute values

3.10 The system must be able to assimilate objects into the design state. This capability involves updating the design database with or for the designer. It includes actions to access relevant storage bases; locate the correct copy and version of the object; save the object; create, link, and store annotations that identify or augment the object (author, time-stamp, explanation, etc.). Specific data management operations are likely to be system and organization-specific.

4.0 Display Construction for the Designer

The final category focuses on display construction that the support system must do to enhance the designer's understanding of the design situation. Requirements pertain to enhancing outputs from tools and presenting explanations that were constructed as a result of other processing. Tutoring is also considered a form of display construction. In general, the support system must be able to manipulate displays and present information appropriate to the context, personalized to user, and responsive to user intent.

4.1 The system must be able to keep the designer informed about the state of his plans (e.g., tasks completed, suspended tasks, tasks not started).

4.2 Based on the designer's intentions and context, the support system must be able to propose approaches, methods, or tools to achieve goals.

4.3 Given a multi-task situation in which it has capability and authority to act, the support system must be able to partition and allocate tasks between itself and the designer.

4.4 The support system must be able to prioritize and schedule input messages to designer.

4.5 The system must be able to compile quantitative and qualitative data.

4.6 The system must enable the designer to view several representations of the artifact simultaneously to overcome the low expressiveness of each representation alone. Related to this, the support system must enable easy navigation between objects and views of the same object, not imposing a predetermined order of access.

4.7 The support system must be able to tailor displays to the context and designer's preferences, style, discipline, etc.

4.8 Given input and output variables from modeling or testing exercises and information about the designer's intentions, the support system must be able to transform, modify, filter, or otherwise highlight these data to match the information needs.

4.9 The support system must be able to present explanations of the following types of information:

- terminology
- principles of how past designs work based on design documentation
- nature of phenomena underlying data
- how to construct syntactically correct representations
- syntax problems encountered with models, tools,...

- computational sources of spurious results
- methods and tools
- its own search and retrieval behavior and actions its takes on the design world.
- answers it offers to direct designer's questions

4.10 The support system must be able to transition to more in-depth coverage and approaches to explaining the above topics (tutoring) if the designer demonstrates failure to understand or requests elaboration directly.

4.11 The support system must be able to propagate changes made in one view of an object (e.g., functional) to other views (e.g., structural) and display the results to the designer.

4.12 The support system must be able to notify designer about design state changes due to his actions, those of other designers and DAs.

4.13 The support system must be able to notify the designer of requirements violations.

For all four categories of requirements, the support system also must be able to adapt the type and level of support as a function of the designer's reactions and feedback, as well as to his understanding and inferred intentions. In implementation, this may require extensive dialogue between the system and designer.

IMPLEMENTATION CONSIDERATIONS

Many implementations that can achieve these requirements are feasible, some more automated than others. To give some sense of the range of possible solutions, consider a "manual" support system in which the services are performed by human specialists. A partial model of such a system is seen in the CSERIAC (Crew System Ergonomics Information Analysis Center), a government-sponsored information analysis center whose mission is to provide designers with problem-specific technical information about human capabilities. While CSERIAC personnel cannot monitor the designer's activities in real-time nor affect the artifact autonomously, they do perform functions that achieve several of the above requirements -- interpret the designer's intentions and information needs (via dialogue), search and retrieve relevant technical information, methods, tools, etc., compile results, and tailor the presentation to the particular user. Explanation and tutoring in a subject are also possible.

While human-based systems are possible, we focused on implementation issues for meeting the requirements through computer-based means and, more specifically, with mixed-initiative computer-based solutions. This section examines these issues, first, with a brief description of general mixed-initiative systems, and then with a version that is tailored to supporting information access and use in design.

MIXED-INITIATIVE SYSTEMS

Mixed-initiative systems are a new concept in automation that stem from the human-centered support philosophy discussed earlier and are made possible by advances in intelligent computing (Rouse, Geddes & Curry, 1987). They "...transcend earlier notions of 'expert systems' as self-contained, authoritative knowledge sources (Geddes & Edwards, 1991)." Instead, these systems integrate human and machine intelligence to form a joint cognitive system that is capable of sharing task demands to achieve overall system goals.

Mixed-initiative systems differ from conventional automation in how task responsibilities are allocated to the human and machine (Rouse, 1985; Rouse & Cody, 1986). Given automation, task demands in complex systems can be parsed into three groups: those in which humans are superior to machines (e.g., pattern recognition); those in which machines are superior to humans

(e.g., computationally intensive operations); and a third subset in which the two resources perform equally well although, perhaps, in very different ways (e.g., information retrieval).

Conventional schemes for meeting task demands seek an optimal allocation that remains invariant across conditions (Bailey, 1982; Kantowitz & Sorkin, 1987). In mixed-initiative systems, tasks can be allocated dynamically as conditions unfold and according to the resource levels that each system element exhibits (Rouse & Cody, 1986). Thus, initiative is "mixed" in the sense that responsibility for tasks can be taken by either the human user or the support system.

Automation Space

Rules that govern each contributor's activity in a mixed-initiative system depend upon two characteristics of the computer-based component: level of intelligence and level of autonomy. Figure 17 from Riley (1989) depicts the range of possible mixed-initiative systems in an "automation space" formed by these two dimensions. Different levels of intelligence are distinguished by availability of data and processing sophistication. Levels of autonomy are distinguished by what the support system is permitted to affect and by the override protocols between computer-based support system and human user. For the most part, level of intelligence applies to inputs to the support system input and level of autonomy applies to system outputs (see Figure 16).

Focusing first on levels of intelligence, systems that can manipulate and manage *raw data* represent the lower boundary of processing sophistication. The majority of today's information management systems (IMSs) are capable of only this level of processing. IMS users can create, edit, store, retrieve, link, and otherwise interact with information. Depending on the volume and variety of data, these functions represent significant implementation challenges from a technical perspective. Such systems, however, possess no task, context or user knowledge with which to adjust the types of support they provide.

Task knowledge refers to all aspects of the activities or methods that users in the domain must be able to perform in order to achieve their goals. This knowledge includes what the tasks are, conditions under which they are usually demanded, what outputs constitute successful performance, alternative ways in which they can be performed, and their costs. It also includes knowledge of what tools and resources can be applied and how to apply them. Task knowledge enables the system to provide two types of support beyond those capable only of processing raw data. First, task knowledge permits the system to monitor the human user's activities in order to

provide relevant information and, if needed, suggestions on how to proceed. It also enables checking for mistakes. Second, if called upon by the user to help, task knowledge permits the system to do a credible job on tasks that the user prefers to relinquish, i.e., systems with task knowledge are capable of retrieving and applying pre-defined scripts.

		LEVEL OF INTELLIGENCE						
	LEVEL OF AUTONOMY	RAW DATA	TASK KNOWLEDGE	CONTEXT SENSITIVE	PERSONALIZED TO USER	INFERRED-INTENT RESPONSIVE	USER-STATE RESPONSIVE	USER PREDICTIVE
Display Construction Only	NONE							
	INFORMATION ENHANCER							
	SIMPLE DECISION AID							
	ADVISOR							
	INTERACTIVE ADVISOR							
	INFORMATION MANAGER							
Acts On Information World	SERVANT-CLERK							
	ASSISTANT							
	ASSOCIATE							
	PARTNER							
	SUPERVISOR							
	AUTONOMOUS SYSTEM							

Figure 17. The combination of autonomy and intelligence defines a space of possible in mixed-initiative systems. Levels of autonomy are distinguished by entities that the system is permitted to affect (see Riley, 1989).

Context-responsive systems can adjust their behavior to contextual change. In design, context refers to the state of design world at both high and low levels of aggregation. At a high level, this includes the state of the overall information environment -- the artifact relative to a schedule for completion, information sources and their availability, design tools and methods that are active, the organizational distribution of work, etc. At a low level, context is defined by the designer's specific location in the abstraction/aggregation space of the artifact (Figure 3). As the

designer shifts from one location to another in this space, a context responsive support system can update its model of what the designer is doing and, thereby, adjust its support appropriately.

Personalized systems are responsive to a model of the particular user's characteristics, preferences, inclinations, historical choices, etc. In today's systems, personalization is exemplified by "user preference files" that contain settings of levels on various options. More sophisticated versions of personalized systems permit users to select preferred modes of interacting with the system and to create standing "contracts" with the system for how tasks should be shared as a function of conditions. Coupled with contextual knowledge, a personalized automation could reason that "under these circumstances, this user typically does X...." Such knowledge could be used to identify atypical behaviors as well as forecast the user's information needs in a rudimentary way.

An *intent-responsive system* uses information from user activity and stored representations of archetypal plans and goals to infer what he intends to do and how. At this level of intelligence, the system can offer a wide variety of supports. These include anticipating the user's information needs based on looking ahead to the next most likely context(s). Coupled with knowledge of task methods, an intent-responsive system can identify from the user's actions plans or goals that appear to be "missing." This function supports error monitoring and notification, tutoring, or intervention by the system in the user's task.

User-state responsive systems can adjust support based on moment to moment variations in the user's physical and/or cognitive states. This level of intelligence requires special capabilities for monitoring user state. *User-predictive* systems can forecast what the user will do next, including commission of errors.

The types of intelligence summarized above enable much of what a mixed-initiative system can do for the user. They enable tailoring displayed information to match task needs, the user's preferences and the user's state of understanding. They enable the system to perform parts of or whole tasks for the user. They also enable the system to recognize and suggest alternative ways to approach tasks that the user may not be aware of. This function may include teaching the user about some topic he might benefit from knowing about.

Mixed-initiative systems also vary in the level of autonomy granted to automation. Based on Sheridan & Verplank's (1978) concepts, levels of autonomy are distinguished by the entities that support system is permitted to affect and by the override protocol between the support

system and user (Riley, 1989). In Figure 17, systems at the lower six levels of autonomy have no authority to change the information world; they are restricted to more or less sophisticated forms of display construction (see Figure 16). Systems at the upper six levels of autonomy have increasing authority to manipulate the information world.

An *information enhancer* can compile, transform, highlight, etc., displayed data. In design, these data can include artifacts, methods, tools, etc. Note, "intelligent" information enhancement, in which the support system constructs displays based upon the context or the user's cognitive style or his intentions, is a function of the system's position along the second dimension of the automation space.

Simple aids provide basic decision support. Given criteria, this type of system can evaluate multiple alternatives and select the best one, perhaps via optimization. Aids can assess an alternative's value along a priori characteristics such as relevance and similarity. These systems can assess the match between the demands of a situation and a particular alternative, where "alternative" may apply to artifacts, methods, tools, etc.

Advisors can recommend actions and *interactive advisors* can initiate the communication of such advice. Depending on level of intelligence, these systems would also be able to explain their recommendations, and adjust recommendations based on user feedback.

An *information manager* possesses the highest level of authority with respect to display management. At this level, the system can basically decide what information should be displayed and when to the user through prioritization and scheduling.

A *servant-clerk* can perform tasks specifically assigned by the user that affect the information world. Depending on level of processing sophistication, a clerk could compensate for the user's inconsistencies or errors by reference to procedural knowledge. Note, an advisor with task knowledge could notify the user about procedural errors he makes; a servant represents the first level at which the system could intervene, at the user's discretion, to correct these mistakes.

An *assistant* can assume responsibility for tasks with standing permission or by the user's on-line consent. Assistants basically monitor for conditions that signal the need for a task, check permission contracts that the user has selected, and act accordingly.

Associates may take the initiative to perform tasks without having to obtain the user's permission. However, the user has ultimate override authority. In contrast, a *partner* has equal override authority, and a *supervisor* can override the user's decisions whereas the designer may not override the supervisor. Finally, an *autonomous system* acts outside the user's knowledge and override authority.

Autonomy/Intelligence Combinations

It is useful to consider the nature of support at a few locations in the automation space to illustrate the range of mixed-initiative systems that are possible. At the lowest ends of both continua, a support system may possess sophisticated data handling capabilities but little else. Engineering information systems (EIS), for instance, basically fall into the cell defined by "raw data" processing capability and "no" autonomy. EISs augment tool-to-tool data exchange by specifying data formats, communication protocols, and data management rules (e.g., configuration management, dependency tracking, version control, security). A given EIS may contain a discipline-specific information model (Hadely & Sommerville, 1990; Libardi et al., 1988; Rangan & Fulton, 1991; Roussopoulous et al., 1991), but, in general, these systems possess no task or user knowledge and no means to exercise autonomous task control.[7]

A simple decision aid that can process raw data would behave in a largely algorithmic fashion. Augmented with task knowledge, the same becomes capable of displaying its selection in a task-specific manner. When contextual knowledge is added, the decision aid can adjust its decision-making process to match the particulars of the user's situation. With personalized knowledge, it can portray its selection in forms that both match the context and the user's preferred ways of dealing with information. An intent-responsive decision aid is capable of not only portraying its results in terms of the present context but in a manner that reflects what the user is likely to do next.

A context-responsive advisor can support the user in dealing with unforeseen or weakly structured problems (Forbus, 1988; Leifer, 1987; Pugh, 1989). Most present-day "expert systems" fall into this cell (Geddes & Edwards, 1991). Placed where they are, however, these systems do not incorporate the user's problem solving strategies or unique expertise. With inputs

---

[7] Implementing a mixed-initiative system for design will require EIS functionality for information retrieval, tool access, tool resources access, and so on. But alone, these systems cannot supply the types of support advocated in this report.

about the user's intentions, advisors clearly could provide advice both about what next steps are reasonable and portray advice in intent-specific ways.

An assistant with task knowledge could execute procedures when called upon to do so. Many such systems are in operation or are being developed for design in particular (cf. Newsome, Spillers & Finger, 1989). An assistant's behavior is modeled on procedures and heuristics used by human designers (Steier, 1990). In design, such systems are especially useful for detailed tasks that are relatively mundane and time-consuming, such as elaborating design details once conceptual and functional solutions have been developed. In software design, a current approximation to assistants with task knowledge are automatic code generators. With knowledge about the context and user, an assistant system can provide more tailored levels of support.

At higher levels of autonomy, a partner has equal authority with the human user. Actions, therefore, must be negotiated. A partner which possesses only contextual knowledge may seem myopic in these negotiations to the human user. With intelligence about the user's style, intentions and expected next moves, however, a partner may seem more amenable to the user. Similarly, a supervisor system, which can override the human user's choices, is likely to seem more cooperative if its actions take the user's expertise and concerns into account.

Processing Structure of Mixed-Initiative Systems

Figure 16 presented the basic structure of a mixed-initiative system. Figure 18 elaborates the information processing elements required to deliver all levels of support represented in the automation space for design (Riley, 1989). The support system would need to be capable of monitoring changes in the design world. It would also need access to the designer's actions including those directed at the information world, at tools, and at the support system itself. "Designer sensors" would also be required for the system to be responsive to designer state.

Given these inputs, the system would require extensive modeling, reasoning and internal communication capabilities to infer the state of the design world as well as the user's context, state, overall goals and proximal intentions. These deductions would enable the support system to deduce the user's context-specific information needs and then to plan its own behavior. In keeping with the human-centered philosophy, goals which support this planning would focus on augmenting the user's performance, not on "perfecting" the artifact.

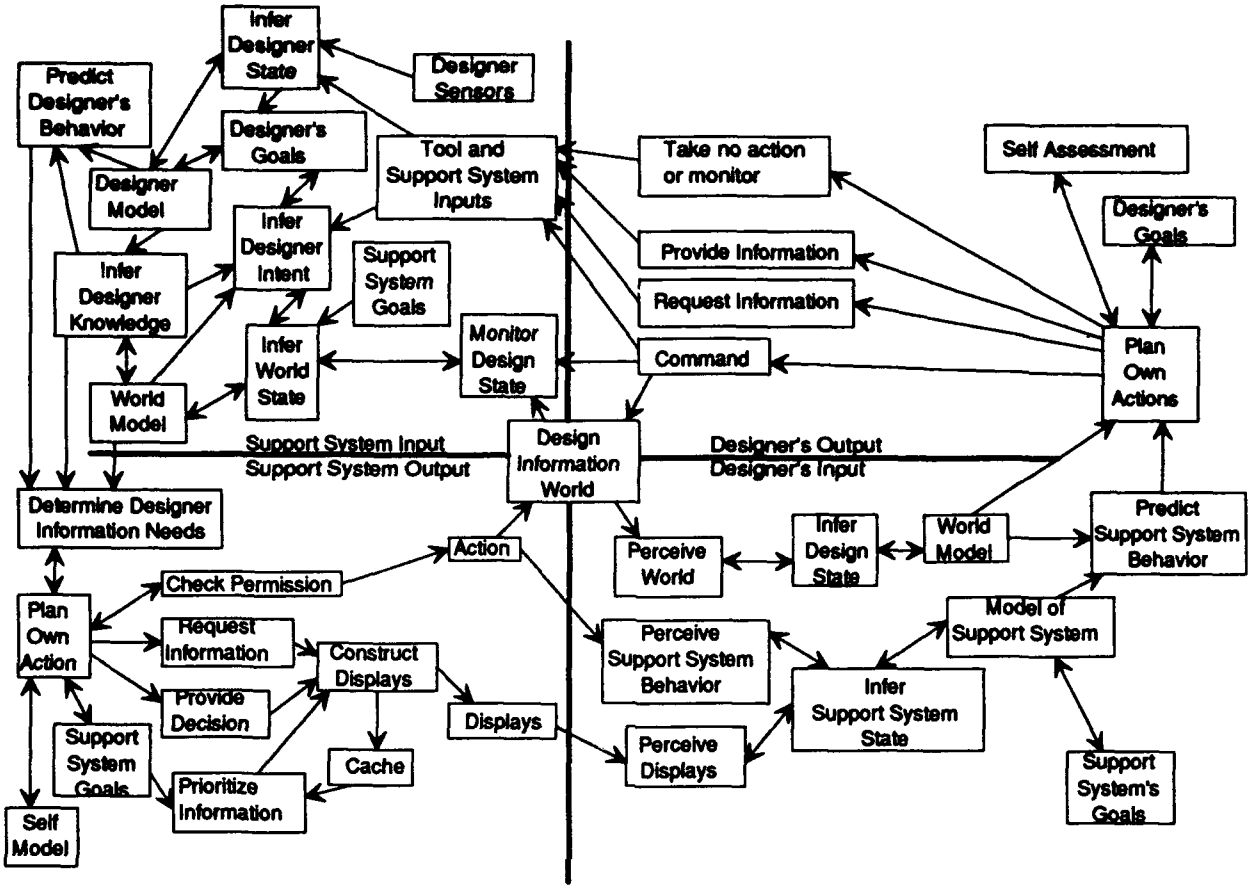


Figure 18. An elaborated form of the structure of mixed-initiative systems showing processing details within the support system and human user.

Plan details could pertain to constructing information displays for the user, to actions to be taken on the information world, or both. This would depend on system's goals as well as its level of autonomy. Override protocols are represented in Figure 18 by processes which require the system to check for permission before acting. Information management would require abilities to prioritize inputs to the user and schedule their appearance, perhaps via caching.

AUTOMATED DESIGNER'S ASSOCIATES

Perceived Value of Mixed-Initiative Supports

Each combination of intelligence and autonomy represents a qualitatively different type of support. For design, the perceived value of each combination is likely to depend on the user and particular circumstances of use. For instance, a user who is a novice with respect to a particular design discipline may value extensive support -- a system that can monitor his activities associated with that discipline, offer advice proactively, and even step in and performs tasks as needed to prevent errors. In contrast, the same individual may appreciate only limited input and tolerate nothing greater than assistant-levels of activity with regard to areas in which he is expert.

The variable nature of perceived value suggests two conclusions about implementing system requirements. First, the system must be flexible. Users must have the option to define level of autonomy on task-by-task and/or situation-by-situation basis. Second, independent of the level of autonomy the user grants, the system should act on as broad a base of intelligence about the designer and his situation as possible. Otherwise actions may be inappropriate.

Given the nature of design, we expect that the intent-responsive level of intelligence will represent an upper boundary for some time to come. Unlike more structured task domains (e.g., piloting), design consists largely of cognitive activity. Our capabilities to determine what the designer may be thinking, in non-intrusive ways, is extremely limited. Perhaps state-responsive or user-predictive levels of intelligence are possible in limited form. For instance, the system may be able to deduce designer "confusion" from his persevering on some task, issue, or tool without apparent progress. Note, this boundary does not imply that the system would be unable to clear up a designer's confusion about some question or help him to address a complex issue. It simply implies that the system would not be able to *deduce* cognitive state or predict the user's next moves without direct input from the designer. "DA. I'm confused about..."

Based on these observations, Figure 19 portrays a region of the automation space which delivers all system requirements and, we believe, will be valued by designers. A region of supports, as opposed to a single cell, is necessary to accommodate users' desire to tailor level of autonomy. Based on the region's upper boundary of autonomy, the support is labeled a *Designer's Associate*, or DA. Figure 20 shows the internal processing requirements for a DA, noting that a few components needed for user-state and user-predictive levels of intelligence become unnecessary.

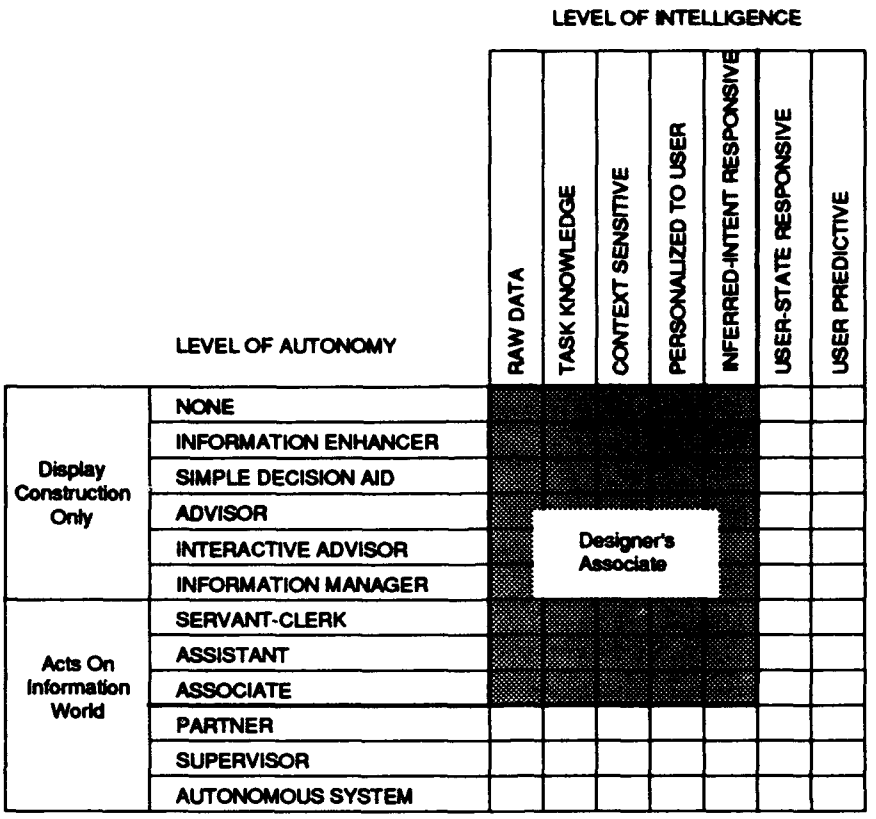


Figure 19. The support domain of a *Designer's Associate*; users can define the support system's level of authority as desired.

Interactions with a Designer's Associate

Positioned where it is in the automation space, a DA would exhibit fairly well-defined characteristics and capabilities. A DA would always be subordinate to the human designer, even on those tasks in which it matches or even surpasses the human user. This will seem foreign to artifact-driven developers who see operational achievement as the goal as opposed to supporting the human to reach the goal. The operational consequences of this approach are clear. While both the user and the associate component may initiate problem-solving activities, the designer has authority over what activities the associate is permitted to undertake as well as the choices that the associate actually makes.

The user may undertake any task or interact with the system at any level he wishes. In terms of Sheridan's levels of automation in human-machine interaction (Sheridan & Verplank,

1978), the user may choose to perform at any level of automatic support -- from none to partially assisted to fully automatic -- with the further option of changing his mind at any time.

In contrast, a DA would have discretion to take initiative only on tasks that the user has granted it standing authority to do so. As tailored by the user, an associate could perform adjunct tasks autonomously which support tasks that the user has undertaken. It could enhance the user's displays, suggest approaches to tasks, monitor activities and the world state, and indicate to the user where his attention might best be focused. It could provide information to enhance the user's performance on a task, and other supporting roles (Geddes & Edwards, 1991).

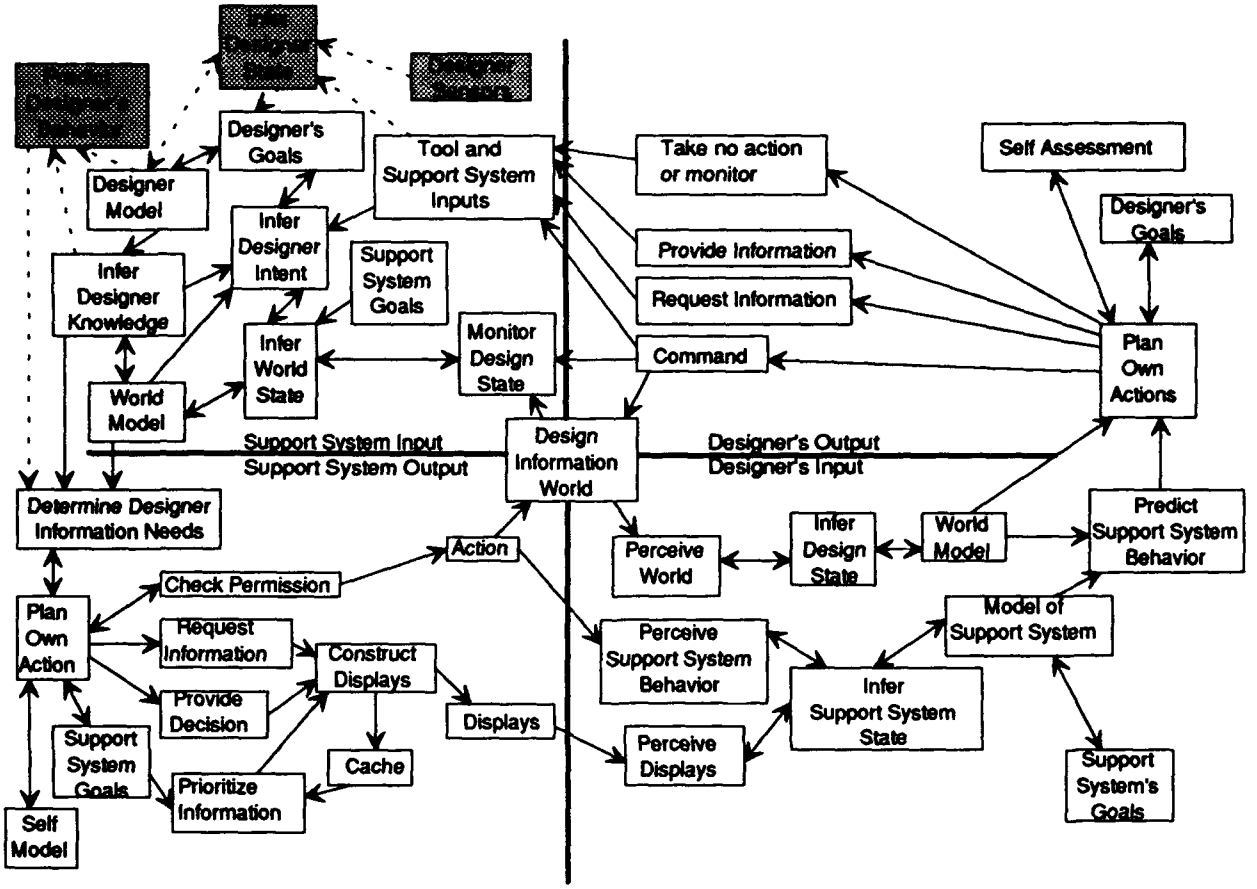


Figure 20. Limiting the DA to a region of mixed-initiative support eliminates processing elements associated with inferring the designer's internal state and predicting his behavior with much precision.

The focus on designer support also implies that a DA would continue to help the user regardless of the plan he adopts. A DA could propose design methods or solutions based on its domain knowledge and knowledge of the user. The designer could accept, reject or ignore these proposals. If the designer rejected the DA's proposals, even if the plan is superior to the user's, the associate would not go dormant. Rather, it would take this reaction into account to improve its model of the designer and situation, and marshal its capabilities to support whatever course of action the user has adopted. In this respect, a DA would exhibit no self-interest.

Humans in systems construct internal models of their situation (Rasmussen 1986, 1988). These mental models include self-assessment of their own ability to address the situation, of the resources available and how to operate them effectively, and so on. A DA would need to develop structurally similar views of the world in order to communicate sensibly with the user.

A DA's goal, however, would not simply be to identify and adopt the designer's view of things. The designer's view may be impoverished along some dimension in which the associate's knowledge is particularly good (e.g., regarding the value of a class of information to the user's decision-making needs). For similar reasons, a DA would not attempt to impose its model of the world on the user. Rather, it would identify the user's model in order to refine its own model of that part of the overall design world. It would also share this model with the designer to help him understand the DA's point of view.

Given general knowledge of design, a DA could know that "designers use methods and tools," but would not enforce a particular method or tool. Based on domain and task knowledge, a DA would also be familiar with several methods and tools, and could propose methods and tools that it deduced were appropriate to the designer's circumstances. It could also access them and the resources required for their execution, and explain their underlying principles to the designer. If needed, a DA could provide tutoring about underlying concepts and principles. It could also help the designer perform a method by executing some or all of the steps or by explaining how to proceed so that the designer could perform the steps himself. Regarding tools, a DA could provide means to estimate parameters, modify or extend the tools, and help diagnose computationally spurious results. It could perform the clerical work of constructing and executing tools, and manage tool output. If the DA determined or was told that the designer was using a particular methodology, then a DA would help him to perform its steps well.

A DA would not enforce a particular design solution. It could know about past artifacts and artifact fragments, and apprise the designer of these. A DA could retrieve representations and

information about relevant artifacts, and also explain their functional and structural properties to the designer. A DA could evaluate the match between existing artifacts and current requirements. If the DA determined that the designer was adopting a particular solution, it could help the designer to instantiate it in his context and apprise him of risks, consequences, or past problems with regard to particular attributes, e.g., operability, reliability, training requirements.

Based on the knowledge and authority relationships, a DA could provide a variety of services that the user would experience directly. It could provide *error monitoring* by observing the stream of user activity, looking for inconsistencies and anomalies indicative of inadequate, inappropriate, and incorrect use of system functionality. A DA could provide *adaptive aiding and workload management* at least by supporting the user to schedule tasks. More substantively, the associate could aid by prompting and, as permitted, initiating use of tools and information sources to actually perform tasks. It could also help with *information comprehension* by adapting displays appropriate to the situation and to user's level of expertise.

Designer's Associates Contrasted with Other Forms of Support

The types of support offered by a DA go well beyond those of present-day systems. For example, as noted earlier, EISs possess no task or user knowledge and no means to exercise autonomous task control. Design aids for specific problems (e.g., layout, structural analysis, kinematics analysis), usually embody a discipline-specific representation and can improve data visualization with advanced display technology (Baldwin et al., 1991). Like EISs, however, design aids possess no knowledge of the larger design context in which they are used or of the user's tasks, expertise, and his intentions. Today's design aids have no means to take initiative on tasks either.

SOFTWARE ARCHITECTURE

Figure 15 presents an architecture for automated versions of a DA. It shows the DA and design world as independent structures that communicate at several levels of abstraction. Referring back to Figure 20, this architecture basically elaborates the interconnections between the DA and the design world, and gathers information processing elements within the DA into a set of modules.

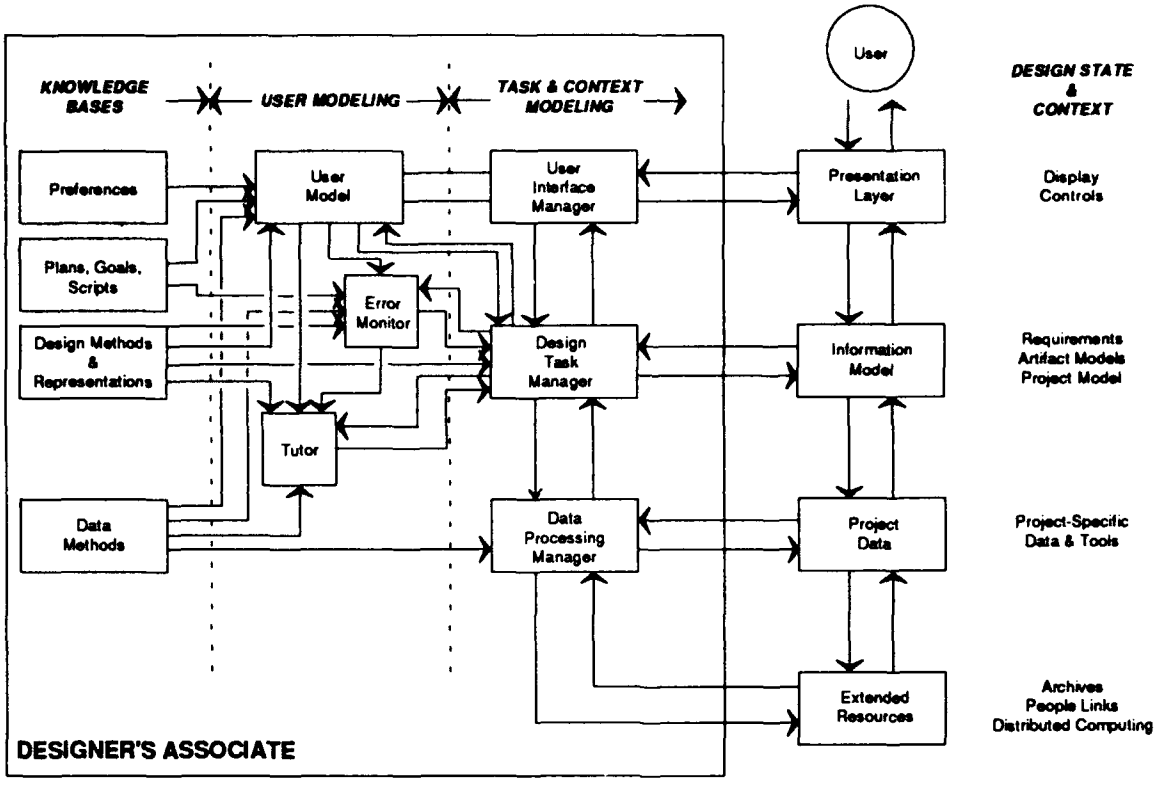


Figure 21. A software architecture for a Designer's Associate showing the independence of this system from the design world.

The architecture conceptualizes the DA as an add-on or attachment to the design world. This approach accentuates two noteworthy characteristics of our image for a DA. First, the DA's purpose is to *augment* the designer's interaction with the world; it does not somehow replace that world or intervene between the designer and the design world. Rather, a DA functions to monitor both, looking for opportunities to aid the designer. Second, representing the DA as an add-on permits the developer to scale its sophistication to correspond with that of the information world. In this manner, DAs that are specialized to topics or tools become possible.

Structure of the Design Information World

Figure 21 shows the design world parsed into layers, an approach that has become fairly standard in software engineering (Denning, 1985; Kahn, 1987). The designer interacts with the Presentation Layer which is composed of controls and displays specialized for human use. Issues of "look and feel" apply at this layer. Contemporary user interfaces are based on various

metaphors (Carroll & Thomas, 1982), e.g., desktop or rooms, and permit the designer to interact with the underlying information model from several perspectives, usually one perspective per "window." Future design systems may augment or replace today's flat-panel workstation solutions with much more exotic interface technology, such as virtual reality in which information is presented and manipulated in three dimensions (Dekker, 1992). Such environments certainly will add new types of interaction for the designer's use (e.g., drawing free hand in the air, or molding virtual objects by hand). However, from an architectural perspective, these types of interaction are simply different ways to issue commands that retrieve, browse, edit, and view the underlying information.

The designer's inputs to the Presentation Layer are encoded and passed as commands and requests to the Information Model. This layer contains abstractions such as "requirements," "artifact" and "tool" and their interrelationships. Inputs to the Information Model are, in turn, converted to commands to retrieve, edit, save, etc., files of actual data records in the Project Data and Extended Resources layers. These files may contain text, graphics, audio, video, etc. data, and may be widely distributed across physical locations, databases and formats. In the return path, data retrieved from the lower layers are composed into higher-order objects at the Information Model layer. These objects are then cast into particular display formats and projected for human use.

Project Data and Extended Resources are separated to emphasize differences in ownership, not structural dissimilarity. Project Data include system-specific data and tools which may be of a proprietary nature. Extended Resources include system-independent data and tools. In this view, the Information Model and Project Data layers are sparse or empty at the start of a new project. As design progresses, this layer becomes populated with representations and data records for requirements, models, drawings, test results, etc. These files are either created with various tools by the designer or are copied from Extended Resources into the "local" storage of Project Data.

Although not shown in Figure 21, there are even more rudimentary layers in the design world. They include the local operating system, distributed operating system, communications, and hardware. Several descriptions of these types of layered system are available (cf. Denning, 1985; Kahn, 1987).

Structure and Processing Within a DA

Within a DA, as depicted in Figure 21, processes are devoted to communicating with the design world, assessing the states of the design world and user, planning, and executing plans.

Communications occur at each layer of the design world. The User Interface Manager monitors the Presentation Layer and communicates information about designer actions to the User Model and Design Task Manager for additional interpretation and processing. In addition, the User Interface Manager is responsible for communicating DA messages back to the designer. Message *content* is developed by the User Model (e.g., "Is it your intention to...?") and Design Task Manager (e.g., "The following tasks are still open..."). The User Interface Manager constructs, schedules and presents displays that result from DA processing and decisions. Hence, knowledge and principles of good presentation reside within this module. It would tailor messages according to the designer's intentions and current design context. Also, this module is responsible for presenting highlighted, transformed, and filtered versions of information that arrives at the Presentation Layer from the Information Model -- thus, its functionality extends beyond DA outputs.

The Design Task Manager monitors the Information Model, tracking and evaluating the state of the design. Since both requirements and artifact representations are assumed to reside within the Information Model, the Design Task Manager can detect requirements violations, assess level of completion, etc. The Design Task Manager is also responsible for executing design tasks that affect the artifact. Its knowledge for performing tasks resides within the Design Methods and Representations module. In addition, the Design Task Manager creates potential task allocations between the designer and DA for the designer's consideration.

The Data Processing Manager plans, schedules and executes searches of Project Data and Extended Resources based on knowledge of where information is likely to reside and protocols for accessing it. This module uses knowledge and methods that are maintained in a Data Methods module for how to search and retrieve data.

Communications also occur among the three modules at the task and context modeling section of the DA. The User Interface Manager sends designer actions and direct requests to the Design Task Manager. In return, the Design Task Manager sends messages of various types which the User Interface Manager then packages and presents in an understandable and timely manner to the designer. Message types from the Design Task Manager include the following:

- *Proposals* regarding alternative methods, task allocations, and suggestions to use DA functionality to enhance his performance (e.g., "you could use some tutoring given your level of understanding.")
- *Requests* for designer interpretations (e.g., intentions, data, plans, etc.) to improve the user or design world models.
- *Notifications* of important conditions and events (e.g., design state, requirements violations or other tensions, significant tradeoff to be considered, open tasks, potential errors, etc.)
- *Compilations* of results returned from lower-level searches and data analyses.
- *Explanations* of procedures, measures, variables, etc.
- *Tutorials*, which are viewed as a special class of interactive messages.

The Design Task Manager and Data Processing Manager have similar reciprocal communications. The Design Task Manager requests data and data processing services, such as running a model or computing statistics on a data set. The Data Processing Manager fills these requests by conducting the appropriate searches and procedures, and then returns results. It also keeps the Design Task Manager informed about its status.

Beyond communicating with the design world, the DA must also be able to assess the state of the design world to provide context-specific, personalized and intent-responsive support. Past experience in developing intelligent support systems (Rouse, Geddes & Curry, 1987; Rouse, Geddes & Hammer, 1990) has shown that the key to providing the types of support recommended in this report lies in being able to know what the human is doing and how these activities relate to his goals and plans. Determining *what* the designer is doing accomplished by the User Interface Manager and Design Task Manager. Determining how they relate to his plans and goals is a bigger problem. A potential solution to this requirement for design support emerges from our overall analysis which suggested that design behavior can be described as trajectories, or sequences of tasks, in the design space depicted in Figures 3 and 4.

If the DA knows where in the design space the designer is, in terms of task and levels of abstraction and aggregation, as well as likely trajectories he may take, then it should be quite feasible to provide the recommended types of support. The first need can be fulfilled with an appropriate Information Model from which the Design Task Manager can determine location in the design space. Determining the likely trajectory of the designer in the design space requires a structure such as a plan-goal representation of behavior. In aviation and process control applications (Geddes & Edwards, 1991; Rouse, Geddes & Curry, 1987; Rouse, Geddes &

Hammer, 1990), we have found that much of humans' activities could be described, at least qualitatively, in terms of behavioral scripts that emerge repeatedly and can provide the basis for inferring the user's goals and plans. It may be possible that "standard" trajectories in the design space can be identified and serve the same purpose.

Note, it is unlikely that inferring the designer's intentions from such pre-defined templates will be foolproof. Thus, we expect that the designer will need to provide explicit statements of intentions, particularly when intentions or representational contexts change.

Much of the tailoring emerges from processing by the User Model. This module receives inputs about designer activities from the User Interface Manager and about the present context and design state from the Design Task Manager. Its function is to determine what the designer is trying to do and what he might do next. These deductions are made by comparing the designer's activities with a model of design plans and goals and typical methods for achieving them along with the user's a priori preferences and "style" which are retained in an active knowledge base. If the User Model cannot explain the designer's behavior, the Error Monitor and Tutor may be notified to diagnose the designer's activities and to prepare a specific proposal for tutoring.

TECHNOLOGICAL REQUIREMENTS

Clearly, automated versions of the DA in the image of Figure 21 will require contributions from both conventional and advanced computing technology. New capabilities will be needed in both the design world external to the DA and component processes within DAs.

Technological Demands of the Design Information World

Alone, the task of compiling and integrating the Extended Resources for a DA is a daunting one to say the least. Volume, growth rate, syntactic differences (e.g., query languages) and semantic differences (e.g., terminology) across databases all discourage simple solutions to constructing useful bases (Rouse, Cody & Boff, 1991; Rouse, Cody, Boff, & Frey, 1990). However, two observations give the support system developer hope: storage and communications media are maturing rapidly; and it is not necessary to access all the world's technical data to provide valuable support.

First, the practical difficulties of compiling the relevant information notwithstanding, solutions to physical storage and communication at least seem plausible. Data format standards have begun to emerge from industrial and government R&D programs devoted specifically to the digital exchange of engineering information, e.g., the Computer-aided Acquisition and Logistics Support (CALS) effort. Advanced optical media already permit construction of massive electronic archives. Currently available CD ROM, with a capacity of over 600 megabytes, can store the text of about six hundred 400-page books (Kryder, 1987; Lambert & Ropiequet, 1986). The burgeoning local- and wide-area networks are laying the necessary voice and data communication paths. The integrated services digital network (ISDN) will be capable of high-bandwidth transmission of digitally encoded information of any type (voice, data, facsimile, video), and is expected to be in operation in some form in the next two decades (Kahn, 1987).

Second, design information systems that support relatively narrow, but still critical problem areas, can be constructed from small fractions of available technical data. Boff and Lincoln (1988), for instance, have compiled a wide variety of human perception and performance phenomena that pertain to display and control system design. This resource is being converted to electronic media and augmented with interactive simulations of several phenomena (Boff, et al., 1991).

The more challenging side of large database construction centers on how to organize and encode information to support timely and intelligent retrieval. Existing indexing and retrieval schemes rely on users' abilities to formulate questions and recognize useful returns. These schemes are inherently limited to the expressive power of natural language. Efforts to model the conceptual structure and pragmatic import of a data set from a particular point of view, like design, are in their infancy.

Beyond data availability, currently, there exists no standard Information Model for design. Figure 22 illustrates the rudiments of such a model for DA needs. This figure presents an entity relationship diagram (Teorey, Yang & Fry, 1986) for a structural view of an artifact. It shows the structural breakdown along a central spine with links at each level of aggregation to various types of supporting information. Vertical and horizontal links possess meanings (e.g., is-composed-of/is-part-of; uses/is used by; supports/is-supported-by) that permit the designer to access related information or navigate from one view to another.

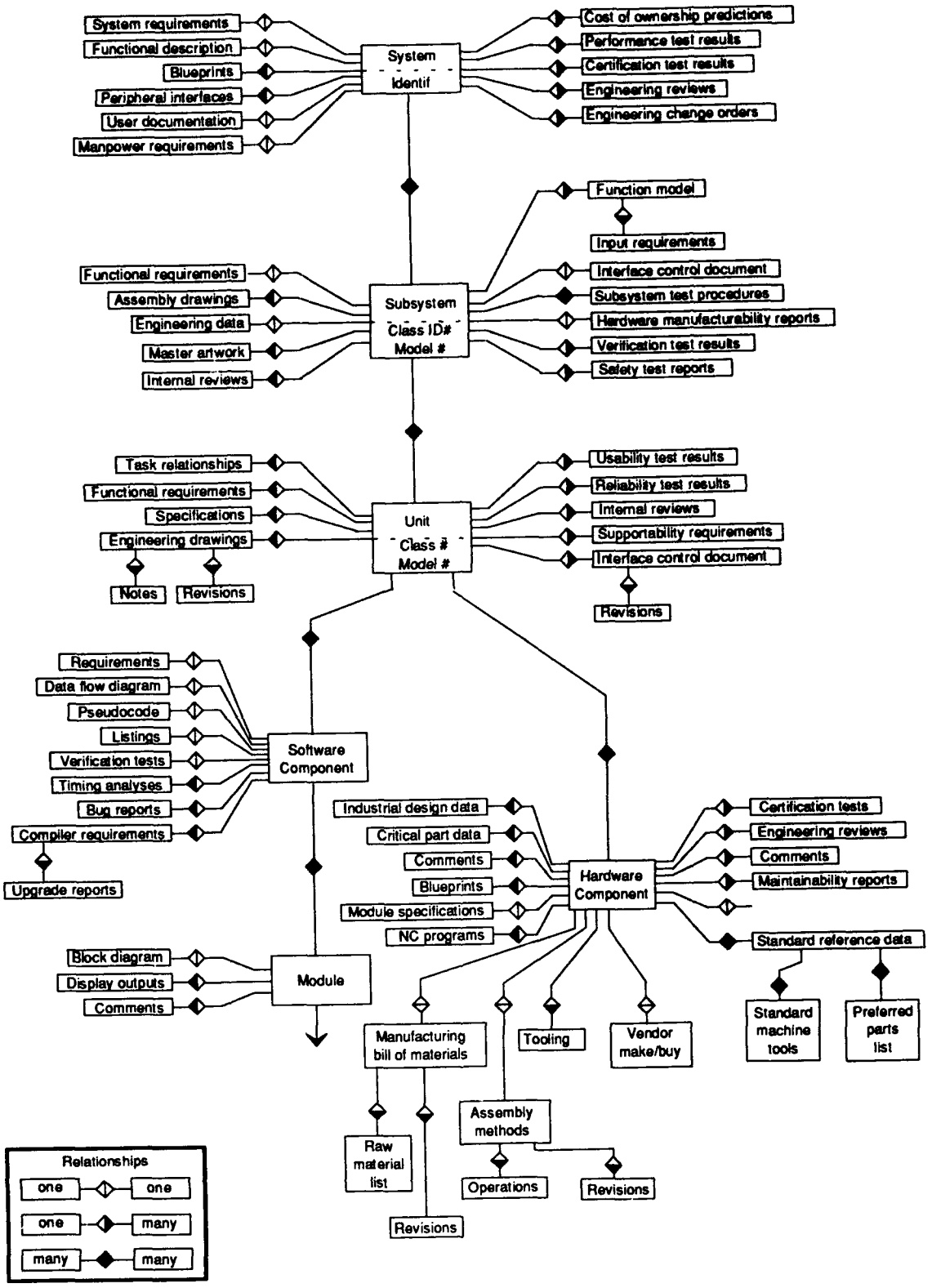


Figure 22. Associate forms of support require a rich information model of the design world.

As Figure 22 should suggest, information objects of any sort can be attached to any other object. Also, while Figure 22 shows only first- and a few second-order links, more elaborate networks are imaginable. Standards for information models are the subjects of great interest (Linn & Winner, 1986; Salzberg & Watkins, 1990; Rangan & Fulton, 1991; Winner et al., 1988).

Technological Demands of Designer's Associates

Each behavioral component of a DA illustrated in Figures 20 and 21 will require advanced computing technologies. This section considers several needs organized according to inputs, internal processing, and outputs of a DA. Of particular interest are needs that emerge from the unique properties of mixed-initiative systems.

Inputs. The DA must be able to acquire information from several levels of the design world in order to construct reasonable assertions about activity and change. At fairly rudimentary levels, this will require high-volume and high-speed data exchange akin to the needs of EISs in general. More abstract levels of acquisition will require pattern recognition, speech recognition, and natural language understanding to process design objects and to communicate with the designer.

These technological demands are fairly obvious. Much more subtle needs are associated with cooperation and shared models between the human designer and intelligent machine. The hallmark of a mixed-initiative system is that at least two actors can take responsibility for tasks. To avoid contention over goals and tasks, they must be able to communicate their observations about the world in mutually understandable ways. Hence, difficulties that attend cross-disciplinary communications among human designers emerge between the user and DA. Social psychological research on the resolution of multiple views, negotiation, and consensus formation as well as natural language dialog analysis provide some foundation for progress in this area. Of particular concern are issues related to agreement between the designer's model of the world and the system's model of the world. Disparate models would lead to chaos in subsequent planning and execution of tasks -- another Tower of Babel (Boff, 1987a).

Internal Processing. As alluded to several times in this report, a DA must be able to determine the designer's context, task, and intentions as a basis for subsequent planning. Structural approaches such as plan-goal graph representations of human activity permit machine reasoning about these matters (Geddes, 1989). These schemes appear to work reasonably well in structured environments such as air combat piloting. However, creating these structures is very

labor-intensive and the extent to which they can be adapted successfully to the loosely structured environment of design is not clear.

Beyond reasoning from relatively static models of design and the designer, the DA will also need to update various components of its model of the world continuously. Some components, such as the designer's style or level of knowledge about a topic, may change fairly slowly. Other components, such as the state of the artifact, may be more volatile and, therefore, may require vigilance.

Personalized approaches to support will require "dossiers" on individual users that contain representations of competence, style, biases, etc. in performing tasks. Furthermore, a DA would need to update these files regularly to capture slowly evolving trends in the user's knowledge or style.

Regarding mixed-initiative, if a shared understanding of the situation can be reached, then a shared understanding of what to do next must follow. Two issues in particular attend this need. First, planning has been a major area of research in artificial intelligence. Unfortunately, approaches have been driven exclusively by a model of planning as an autonomous activity, rather than as a joint activity between cooperating intelligent agents (Geddes & Edwards, 1991). Moreover, design is such that the designer must create and prosecute several plans at different levels of abstraction and aggregation concurrently. The computational and reasoning demands associated with creating, sharing and managing multiple plans between agents are unknown.

Second, cooperative planning between agents with different levels of authority leads to the issue of adapting to plan rejection. More specifically, if the designer rejects a recommendation by the DA, then the DA will need to understand why it was rejected in order to improve its model of the designer's plans and goals. Rejection could occur because the D is "on the wrong track" from the designer's point of view. Alternatively, rejection may signal that the designer is headed in the wrong direction but does nor recognize this. When rejections occur, the DA may be able to query the designer directly to find out what was unacceptable. However, depending on the frequency of such exchanges, the designer may lose trust in the system's ability to keep up and, consequently, reduce its authority to contribute.

Outputs. In order to manipulate the user's displays and affect change in the design environment, a DA will need to possess extensive task knowledge and knowledge about principles of information portrayal. It will also need the ability to select, configure and use a wide variety of

tools. On the one hand, these demands may seem insurmountable. On the other, it is not hard to imagine such capabilities given the state-of-the-art in intelligent retrieval systems and design aids.

More difficult issues relate to cooperative task execution and monitoring between the designer and DA. Coordinating these agents' efforts will require adaptive command and control strategies to prevent contention for tasks and resources. The objective is to ensure that each agent is clear about who is responsible for what.

SYSTEM EVALUATION

It is useful to consider how a DA might be evaluated. At first glance, evaluation may seem quite straightforward -- just build and try the DA and see if it works! At an extreme, one might develop a product using the DA and compare the new product's performance, reliability, usability, etc., to a product that was designed in the same environment without a DA. As might be imagined, this could be a very expensive undertaking for even a relatively simple product. More importantly, however, is the fact that one would learn very little from such a single-point comparison. Too many confounding influences beyond support system differences preclude product success from being a useful measure of support system success.

Much more insight can be gained, with substantially less cost, by decomposing the evaluation question into finer-grained issues at different levels of analysis (Rouse, 1987b). In particular, we have found it useful to evaluate intelligent support systems in terms of viability, acceptability, and validity (see Figure 2).

Viability refers to whether the benefits of adopting and using the system outweigh the costs. This issue will have very different interpretations for different stakeholders in a DA. For example, viability for the individual designer may hinge on the tradeoff between the marginal increase in perceived decision quality and added learning requirements of new functions. In contrast, viability for the designer's organization may be defined in terms of process improvements at the group and program levels versus costs to purchase and maintain new hardware and software, train people, etc.

Acceptability refers to whether the solution as implemented is agreeable to users. For design support, acceptability will be judged along several dimensions including the designer's

appraisal of his own performance and versatility with and without the DA, his appraisal of its performance (response time, reliability) and his appraisal of its "style" (e.g. intrusiveness).

Finally, *validity* refers to whether the DA improves design decision making. Direct indicators of decision making quality may be difficult to acquire. Under the assumption that better information (i.e., broader base) enhances decisions, validity may be indexed by the extent to which a DA improves the designer's ability to access and use information.

CONCLUSIONS

Design presents human decision makers with complex information access and utilization tasks. While the specific content of their information needs vary, all design contributors must identify sources, retrieve information, evaluate its implications, encode it, store it, and so on, in the process of solving problems and making design decisions (Boff, 1990; Rouse, 1987a; Rouse, Cody & Boff, 1991; Rouse, Cody, Boff & Frey, 1990). The quality with which these operations are performed is governed by characteristics of the information, the designer, and the designer's milieu.

Relevant information exists in a variety of places, expressed in many forms, of varying quality, etc. It includes both system-specific and system-independent elements. Information also tends to be distributed across "data islands," making availability a problem. The designer may not be aware of, misinterpret, or not value relevant information that could improve his decisions. For better or worse, each designer assesses the costs and benefits associated with information access, and acts in accord with this evaluation. A poor evaluation can effectively block information from affecting design choices. The surrounding milieu, which is composed of co-workers, organizations and technology, can interfere with access and use in a variety of direct and subtle ways.

One approach for improving information access and utilization is through the use of mixed-initiative support systems. Initiative is "mixed" insofar as responsibility for tasks can be shared between the human user and the support system according to a set of rules. Made possible by advances in intelligent computing, mixed-initiative systems "...transcend earlier notions of 'expert systems' as self-contained, authoritative knowledge sources (Geddes & Edwards, 1991)," by *combining* human and machine intelligence to empower the human user in achieving his responsibilities (Rouse, Geddes & Curry, 1987).

Good opportunities for mixed-initiative supports exist where two conditions prevail: the human user cannot perform effectively in a totally manual mode, but completely autonomous solutions are outside the state of the art. Fully manual operations may be impractical or impossible due to task demands that exceed human physical or cognitive capabilities, whether because they are repetitive and boring, time-critical, hopelessly complex, or require skills from many domains. Autonomous solutions may be beyond reach for two reasons. Situations in which unforeseeable circumstances are likely to emerge tend to be poor candidates for complete

automation because they prevent closed-form solutions or make automatic solutions cost ineffective. Also, situations in which a human user will be held socially or ethically accountable for system behavior are not good candidates for autonomous operation by machine. Given responsibility, the human must also be able to exercise appropriate authority over system operations.

These properties are clearly evidenced in dynamic and uncertain environments such as air combat piloting, air traffic control, and complex process control (Rouse, Geddes & Hammer, 1991). Task demands in these domains make fully manual operation impossible, while uncertainty and potential for socially catastrophic events also make totally automatic solutions infeasible.

Design differs in many ways from these operationally intensive domains (Rouse, 1988). However, design shares the chief properties that make mixed-initiative support attractive: a practically boundless solution space that requires human judgment; task complexity due to the many technical specialties, information sources and tools that must be summoned; and the present impracticality of producing design solutions automatically to any but the most mundane of problems.

While there are a range of potential mixed-initiative systems that vary in terms of intelligence and autonomy, our reading of the nature of design suggests that associate-level systems that are responsive to the user's intentions are feasible and will be well-received. This report has attempted to describe why such *Designer's Associates* make sense and requirements that such systems should meet to be valued by their users.

FUTURE SCENARIO

In order to provide a more concrete image of a *Designer's Associate* than can a list of requirements, we composed the following scenario of what design practices might be like with the support of a DA at some time in the future. While reading this story, it is useful to keep in mind the distinction between capabilities that will exist in the designer's world independent of DA technology (e.g., virtual reality interfaces) from the functions that a DA might perform. Note, the scenario does not attempt to capture all 36 of the requirements listed earlier.

Having taken a sip from the first of several morning cups of coffee, Henry turns to his wall-mounted display to pick up from last evening. He wanted to make some progress on test procedures before his scheduled time at the virtual lab. The team's prototype for the new feather-weight night vision system would be arriving from fabrication in a few weeks. Henry was in charge of the perceptual battery.

"DA?" he asked quietly in the direction of the blackboard-sized screen.

The screen jumps to life. In the lower left corner, a pleasant face appears and answers in a calm voice. 'DA' is Henry's name for the new information manager, a software system that has access to all of the tools and information that Henry's organization creates and uses. Everyone uses it -- designers, managers, manufacturing engineers, secretaries. Henry, like most of the designers, relies on the system for a variety needs. It handles his interactions with tools; stores and retrieves his designs, computer models, test data; helps him scour the literature for information; helps him determine the implications of design choices; keeps track of progress and open tasks; manages his communications with other members of the organization; and even tutors him in unfamiliar topics when he desires. In short, DA helps Henry to be a more effective designer.

"Hello," said the voice. "Sign in please."

Henry quickly scrawls a pattern on the screen with his fingertip, and this action identifies him to the system. Until now, DA only knew that Henry's station had been activated. Now that DA knew it was indeed Henry who was talking, he could get on with his work.

"OK, Henry, you're in. Do you want to pick up where you left off?"

"Yes, please."

"OK."

The face reduces to an icon off to the side. In seconds, several pieces of the test plan that Henry has been developing appear in windows on the screen -- engineering drawings of the prototype, instructions for subjects, digital photos of test patterns and objects, presentation geometry, data collection triggers. Henry scans the materials and attaches some electronic notes to a few items. Satisfied with the plan so far, he opens a new window to begin laying out the experimental design. Automatically, the other windows are resized and moved out of the way.

"DA. I need to show performance benefits of the feather-weight system compared to Models VR68 and VR73. I found out yesterday that we'll have only four subjects for this test. What do you know about test design considerations with small sample sizes?"

Being somewhat new to human performance testing, Henry has given DA a fair amount of authority in helping him to construct test procedures. If Henry's test design shows weaknesses, DA can modify the plan subject to Henry's approval. But Henry has restricted DA's role to interactive advisor when it comes to the feather-weight system itself. It's too early to turn DA loose on some of the detailed design tasks that will need to be done with the production version. Nevertheless, DA's cautions about some biomechanical issues and several manufacturing considerations have been valuable.

Following a brief pause, another window opens adjacent to Henry's. As a list of items begins to scroll into view, the DA says,

"I have a tutorial on that topic. (Pause) The laser-protection group ran a similar test last May with five subjects. Paul Thompson was the lead. (Pause) Kelly McGuire in Statistics is listed as available for consultation. (Pause). Would you like more sources?"

"That's enough. Show the test plan for the laser-protection study, please."

"OK. In the meantime, I will construct an experimental design for your consideration with three conditions and four subjects that assures acceptable power."

"OK. I'm sure I'll have a few questions."

Henry uses a mouse to click from page to page and highlights a few sections of the other group's study plan.

"Do you have questions about those passages or are you considering adopting the procedures for the feather-weight study?" asks the DA.

"Both," replies Henry. "Seems like collecting subjective measures from test subjects is a good idea. But I'm not sure how to proceed. I wish they would have included more detail about these rating scales."

Henry continues to read through the laser-protection study.

"I have the test design whenever you'd like to take a look, Henry."

"OK, DA. Let me see it."

Henry examines several charts that show a test matrix, a daily schedule, and various statistical estimates of test power. He asks several questions about how the design will test certain combinations of effects. He also notes a schedule conflict that will need to be resolved.

"OK, DA. Let's store this design as a straw man, but I'll want to re-consider it again next week. In the meantime, show me what you have on subjective measurement techniques."

The test design window closes and Henry selects a few references on measurement to read. After some time, the DA interrupts Henry.

"Henry?"

"Yes, DA?"

"It's 9:25. Your lab time starts in five minutes."

"Thanks, DA. I'll see you there."

Henry grabs his virtual reality equipment and heads down the hallway. He enters the large room that is empty except for a comfortable chair off in the corner. He buckles on a small belt pack, dons a visored headset and pair of gloves. Then he connects cables that lead from the headset and gloves to the belt pack. The pack contains a power supply and transceiver that broadcast Henry's position, movements and voice to the information manager. It also receives video images, sound and pressure signals that are displayed through the headset and gloves. Henry turns out the room lights and switches on the belt pack. In a few moments, the room reappears in its virtual form -- correctly sized and empty except for an image of the chair. A

calibration panel is floating in mid-air a few feet away. Henry turns and faces the panel straight on. Reaching out, he presses a key to launch the calibration sequence. After matching up crosshairs, touching several floating objects, and threading a virtual needle, Henry presses "Done" on the panel. The room and objects shift slightly and the panel disappears. Now ready, Henry calls his aid.

"DA?" says Henry.

A few feet in front of Henry, the visage of DA appears and issues the familiar request.

"Hello. Sign in, please."

Henry lifts his gloved hand to trace out a pattern that glows in mid-air, and then waits for approval.

"OK, Henry, you're in. By the way, I signed you off at your office station."

"Oops. Thanks, DA. Let's pick up from yesterday."

The body of DA disappears, and quickly the room is filled with Henry's test setup and documents. An optometrist's chair is located near one end of the room facing the opposite wall. Off to the side is a control console, table and chairs where the experimenters will sit. Track lights and various projectors appear suspended from the ceiling. A three-dimensional grid glows softly throughout the space between the optometrist's chair and opposite wall. A dozen or so bluish tiles appear at various positions in the grid along with numbers that summarize what their visual properties would be for someone seated in the chair.

Henry turns to a folder on the table. With a right-to-left motion of his hand over the folder, it opens to page 1 of his test plan. Henry continues gesturing to flip through the pages and stops when he reaches the section on test patterns.

"DA. Spread this section on that wall, please."

He points to the folder and then to a blank wall as he says this, and about twenty patterns quickly appear arranged neatly on the wall. After studying the patterns for a while, Henry is ready to see things from the test subject's point of view.

"DA, present patterns 5, 10, and 17 at display locations 1 through 3. Remove the test chair. Show display volume at design eye and simulate .001 mL ambient."

Patterns pop into tiles and a transparent box appears at the location where a person's head would be when he's seated in the chair. Henry stoops slightly to position his head in the box, and the light level in the lab quickly dims to starlight conditions. As Henry examines the scene and changes several parameters, the DA's voice sounds in his headset.

"Excuse me, Henry. Stuart has just left voice mail for you. Shall I play it back now?"

"Yes, please."

In the hallway a few days before, the department manager had alerted Henry to keep some time open for an important job. Maybe this was it.

"Hi, Henry. As expected, I need your help on this X95 problem. Give me a call as soon as you can. Bye."

"DA."

"Yes, Henry?"

"Call Stuart for me, please."

In a moment, Henry hears the tone sequence and then the phone ring.

"Stuart Cashman," comes the reply.

"Hi, Stuart. Got your message. What's up?"

"Thanks for calling back, Henry. As I mentioned to you the other day, our simulator evaluations of the X95 cockpit are disappointing. In particular, the new traffic situation display is not improving flight control like we expected. The people in T&E thinks that pilot errors are somehow to blame, but haven't been much more explicit than that."

"Hmmmm. The standard diagnosis."

"Right, but who knows. I know your busy with 'feather-weight' and MedEx, but this problem must be resolved before the customer pilots arrive for OT&E."

"When do they arrive?"

"June 8. That gives us a two-week window for re-design with minimal disruption. I asked DA to recompute your schedule assuming we let MedEx slip. DA. Show the current and revised schedules to Henry, please."

Two timelines popup in a rectangle before Henry. His tasks and deliverables are highlighted. After looking over the charts, Henry replies.

"OK. I'll give it a go. But I'd like some of Sharon's time if I run into unfamiliar territory in the X95."

"Fair enough. Keep me posted," says Stuart as he signs off.

"DA."

"Yes, Henry."

"Annotate the feather-weight test plan with a reminder to check on subjective measurement, and close the file."

"Shall I clear the lab?"

"Yes, please. And then display the X95 task order."

In a moment, the visual test setup disappears. The room is empty again save a floating folder marked 'X95 Crew System.' Henry grasps the folder and heads to the chair. He gestures over the folder and begins to read. The statement of work outlines requirements for redesigning the traffic situation display, or 'TSD.' The specifications include flight control performance requirements and a real-time software budget, as well as cost and schedule constraints.

"DA. This report says that T&E put together a video. Display the video, please."

Henry leans back in the chair as the video begins, displayed in a large floating window a comfortable distance away. The tape opens with a rear view of a pilot seated in a state-of-the-art electronic cockpit. In the foreground, behind the pilot, sits another person perched on an observer's seat.

"Hi, I'm Al Long and this is Steve Edwards. We made this video in the simulator to help you diagnose the source of the performance problems. Ready, Steve?"

"Ready," replies Steve.

"Steve is going to fly landings using the new TSD. He will use the information on the TSD to adjust his controls to minimize deviations from the latest FAA standards for in-trail separation. This stacks up aircraft as closely as possible to increase airport capacity without risking vortex problems. Steve lets the onboard flight computer know that he is adjusting separation by pressing the icon on the left below the TSD."

"The TSD also indicates potential path conflicts, usually with turbulence, but also with shears, bursts and other aircraft in the vicinity. Steve responds to conflicts by slowing down or maneuvering away. He tells the computer that he is doing this by pressing the icon on the right. Let's get going. Hope this helps."

"DA. Stop the video here and send the display to that wall. Display a 3-D version of the X95 cockpit at 100%, please."

The video window moves to the wall and in a moment, a solid CAD model of the X95 cab and cockpit appears in the center of the lab. Henry gets up and walks around behind the cockpit and peers over the pilot's seatback at the all glass control panel.

"DA. Simulate a canned landing sequence and drive all cockpit displays, please."

"Which airport?"

"Someplace busy."

"I've selected Chicago O'Hare. The simulation begins at the outer marker."

For the next few minutes, Henry watches the color cockpit displays as they show what a pilot would see during a routine final approach. The cockpit is remarkable. Conventional instruments and CRTs have all been replaced with what looks like a single sheet of touch-sensitive plexiglass. The aircrew's displays are rear-projected onto this panel. Everything is driven by software. Among the displays, the new TSD, its symbology, and control icons all look great. So what's the problem?

"Excuse me, Henry. Your lab time is nearly over."

"All right, DA. Sign me out, and let's continue back in my office."

With a new cup of coffee, Henry begins watching T&E's video on his office display from where he'd left off in the lab. Every fifth landing or so, Steve executed a modest maneuver to deal with a conflict. For the rest of the landings, Steve frequently touched the left icon and adjusted the throttle or flight controls to reduce separation a bit. Everything looked ok.

"DA. Show the landing performance data."

The several graphs and tables showed T&E had problems. In some cases, average separation actually increased a bit using the TSD, and the variance was often quite large.

"DA. I think we have two alternatives. Either Steve is missing events on the display or he's responding to events that don't exist. Somehow this is affecting performance."

"There are three alternatives in this type of task," interjects DA.

"What are they?"

As DA responds verbally, a list appears on the screen.

"Misses and no execution; false alarms and unnecessary execution; correct detection and improper execution."

Henry thinks, yes, this is right. We potentially have detection and/or execution problems.

"DA. Show the X95 cockpit documentation list."

In a moment, a list begins scrolling up the screen.

1. Requirements document
2. Functional block diagrams
3. Simulation model
4. Engineering drawings
5. Parts list
6. Operating procedures
7. Maintenance procedures
8. Training system

.

.

"That's enough. Show me 2 and 3."

Two windows appear on the screen with the labels 'X95 Functional Block Diagrams' and 'X95 Simulation Model,' Henry rapidly browses each document, jumping between the two using their many embedded cross references.

It was not clear to Henry how human performance had been considered in the design. The simulation model was filled with concatenations of probabilities, but there were no definitions on the top-level diagrams.

"DA. What is PHE?"

"PHE is defined as the 'probability of human error'."

Do any other probabilities contain the words 'human' or 'performance' in their definitions?"

"PAPF is defined as the 'probability of actuator performance failure'."

"That's all?"

"No other variables have 'human' or 'performance' in their definitions."

"OK. Give me the values of PHE that were used in the simulation studies of the X95 design."

"There are no values of PHE available."

This is too bad, Henry thinks. He'll have to start from scratch. Henry feels that he needs to represent human performance in much greater detail than just a single probability. He guesses that representing event detection and maneuver execution separately might suggest a wide range of re-design options. First, however, he has to figure out how the current design worked.

"DA. What types of models do you have for visual detection?"

"I have information on six models of human visual detection and four pattern recognition models for automated inspection."

"Display one-page summaries of each of the ten models."

None of the models was exactly *what Henry was looking for.* But it looked like a reasonable approximation for his needs would involve a linear combination of features such as distances between objects, relative velocities, and symbology coding. Actually, *perceived* features had to be the variables.

"DA. Do we have any data on people's abilities to perceive distances and relative velocities?"

"There are four relevant tabulations in the Data Compendium."

"Show all four side-by-side."

While the results for large distances and velocities were quite mixed and context-dependent, small values produced surprisingly similar performance across contexts. In general, people were much better at perceiving small changes relative to some reference, rather than absolute values. In fact, without a reference, humans frequently fail to detect small changes. Henry wondered whether performance with the TSD might be improved by providing better references for judgments of separations and relative velocities? He decides to use parameter ranges in a model that represent performance with both baseline and enhanced references.

"DA. Display our flight control models."

"There are four models available. There is also a tutorial on the *GoldenArm* model. My records show that you have used none of these."

"That's true. Do you have any advice?"

"What are you trying to accomplish at the moment?"

"I want to develop a better pilot flight control model for inclusion in the X95 simulation model."

"I assume 'better' relates to the visual performance materials you have been reading?"

"Yes -- more detailed with regard to visual detection of moving display elements and with manual control performance."

"I recommend your viewing the *GoldenArm* model tutorial, but using the *Pilot* flight control model. The latter will require the fewest modifications to be compatible with the X95 simulation model."

"Great. Display the tutorial."

Following the tutorial, Henry studies the *Pilot* model. The DA constructs various inputs data files and displays the results of model runs to Henry. As Henry explores the model's behavior, the DA occasionally probes him with a brief question to sample his growing understanding of the tools and concepts.

"OK, DA. I think I see. Get the X95 simulation model again. Substitute the following expressions for PHE."

Henry types in a pair of equations, one related to detection of changes of distances and velocities, and the other related to control performance. The control equation is actually a call to the *Pilot* model along with a set of input values.

"Now I'm going to need some parameters," says Henry. "DA. Display the X95 engineering drawings and operating procedures."

Two new windows appear with the appropriate materials. Henry searches a bit and finds the TSD format definitions and the associated operating procedures.

"DA. I want to modify these temporarily. Make a copy and configure the CAD editor, please."

Henry adds reference lines to the TSD format that he believes will make it easier for the pilot to detect changes of separations and relative velocities, modifies the display symbols to better differentiate traffic from conflicts, and changes cockpit lighting slightly to reduce glare on the front control panel -- he got the glare reduction idea while reviewing the visual detection models.

When he changes the length of the yoke to lessen the chance of occluding the display, DA interrupts.

"Henry, are you aware of Specification M39256 regarding yoke mechanical requirements in the X95 cockpit?"

"DA, I'm sure you're about to tell me about a spec violation. But right now I'm just trying several things."

Henry knew that making all of the changes that he had in the model was a much simpler undertaking than convincing the entire X95 cockpit design team would be. However, he would worry about that team meeting later. Right now he needed to find the source of the performance problems.

"DA. Compute and save the following dimensions and angles."

Using a mouse, Henry moves a cursor around the X95 engineering drawings, clicking on a variety of points. As he does this, DA displays the numeric values of the dimension or angle that Henry has indicated, both on the drawing and appended to a parameter list.

"Great. Now here are my labels for the parameters in your list."

Next to each parameter, Henry types a name. These would be used in the revised X95 simulation model.

"DA. Display the values of these parameters without the changes I just made."

"Do you mean changes of the parameter names?"

"No. No, I mean display the values of these parameters in the original drawings."

"The variable Deviation from Reference Lines cannot be determined using the original drawings."

"Of course, that's right. Set it to zero."

There were now two parameter lists displayed, one labeled 'Original X95' and the other 'X95 Modifications.' The variable DRL in the original list was annotated as set to zero.

"DA. Perform two model runs, one with the original parameters and the other with the new parameters. Plot average aircraft separation and response time to conflicts as a function of traffic density. Also, plot the data from both runs on the same graph."

Henry stood up, stretched and strolled down the hall while DA scanned the network for available computing resources, scheduled time, and ran the models. When Henry returned several minutes later, a data window on his wall display is filled with a tangled mess of lines annotated with various labels.

Wow, Henry thinks. Lots of noise in these plots. I'll bet the original modelers overdid the randomness. But, no matter. Average performance should reveal the effects.

"DA. Replot these data with a ten-second moving average."

The displays disappeared and reappeared almost instantly. The tangled mess was gone. But the results were most disconcerting. The new and original configurations were nearly identical!

"DA. Plot the control components for each model."

The resulting plots showed that while the act of flight control was somewhat different for the new and original designs, the end result was the same.

"Hmmm. Looks like *Pilot* is a pretty adaptable guy," thinks Henry.

"DA. Show the statement of work again."

Henry compares the various model predictions with the criteria in the SOW. Both the new and original configurations meet the specs, with the new configuration marginally better for conflict avoidance. Something really troubled Henry, however. Regardless of the parameter variations, predicted in-trail separation performance was always *better* than the actual performance numbers than T&E reported. In other words, the models could not reproduce the problems that caused T&E to come to Henry in the first place. Something clearly was missing.

"DA. Show the original SOW for the TSD part of X95."

"There are three versions. Should I display all of them?"

"No. Just the one that went out for bids."

The original SOW provided very little additional information beyond the SOW Henry had received. The only noticeable difference was that the original indicated that two alternative operating procedures were acceptable. Why had the subcontractor who developed the TSD chosen one over the other?

"DA. Retrieve the winning subcontractor's TSD proposal and display the section that discusses proposed operating procedures."

Henry skimmed the proposal. It was too much to expect to find a rationale for their design. Maybe he could contact the designers?

"DA. Is this company on our network?"

"They were until last year when they were acquired by a conglomerate. Should I attempt contact?"

"No, disregard. We'll keep that as an option."

Henry decides that he might be able to make some headway by assuming that the designers' choice had been arbitrary. Perhaps he could figure o[illegible]y it should not have been arbitrary?

"DA. Extract the descriptions of the alternative operating procedures from the original SOW. Get the subcontractor's description from their proposal. Get the sections from the X95 documentation about TSD operating procedures. Display all of these items side-by-side."

For the next several minutes, Henry compares the three sources. He finds only one salient difference. One of the procedures, the one actually adopted, involved first pushing either the left or right icon and then adjusting in-trail separation or executing a conflict avoidance maneuver, depending on which icon was pressed. The alternative procedure, the one not adopted, involved first maneuvering and then telling the flight computer why you acted. So the button push followed the control activity. Henry couldn't see how the difference between the two could really matter.

"DA. What do you know about procedural errors?"

A list of options starts to scroll up the screen.

1. Background readings
2. Tutorial
3. Classification schemes
4. Analysis methods

.

.

Henry touches the Classification Schemes line on the screen.

"There are four schemes available," DA responds. "They include...."

"DA. Just give me the simplest one."

A list appears on the display with some terse definitions and a little more jargon than Henry was hoping for. Most of the concepts were pretty straightforward though.

"DA. What is a 'capture error'?"

"A capture error is likely to occur when a specified procedure interferes with a population stereotype...."

"What's a population stereotype?"

"A population stereotype is a standard way of doing something that usually has evolved over a long period of time."

"DA. Can you provide an example, please."

"You are driving home from work with the intention of stopping for groceries. As you pull into your own driveway, you realize that you forgot to stop for the groceries. In this situation, you were captured by the stereotypical procedure for driving home."

Henry ponders this. Interesting. But, there didn't seem to be any population stereotype for X95. Unless...

"DA. Summarize the specific differences between using the TSD to optimize separations and using it to avoid conflicts."

"To optimize in-trail separations, the pilot is instructed to press the left icon and then take control actions as necessary. For conflicts, the pilot is instructed to press the right icon and then make appropriate conflict avoidance maneuvers."

Aha! For every tenth or twentieth landing, the pilot presses the right icon and goes through the conflict avoidance procedures. For the other nine or nineteen landings, he presses the left button to squeeze in a little closer to the aircraft in front of him -- and he does this much more frequently. Henry supposes that every once in while, the pilot presses the left icon when he

intended to press the right one. As a result, the onboard computer interprets his intention is to optimize separation when in fact he is trying to avoid a conflict.

"DA. Retrieve the X95 simulation model. I have an idea. Label the original model PROC1 and make me a copy labeled PROC2. Also bring up the editor."

Henry then proceeds to change PROC2 to correspond to the procedure that the TSD designers did not choose. Beyond changing the order of operations, he also moved the icons further apart and replaced their text labels with pictures -- one portrayed a turbulence cloud and the other two aircraft in line.

"DA. Modify the parameter list to reflect the layout changes and run PROC1 and PROC2."

"The PHE equations cannot be computed for PROC2."

That's right, thought Henry. He forgot. In fact, PHE would not be right for PROC1 either.

"DA. Have you got any human error data?"

"There are 128 tabulations in the Data Compendium that include human error as a dependent variable."

"Whoa. That's not going to work. What do you have regarding procedural errors?"

"There are no entries for procedural errors. There is one compilation for actuation of buttons, switches and levers."

"Better yet! Let me see it."

The data in the table looked great for PROC2 where Henry thought he had greatly lessened the likelihood of capture errors. However, he didn't know what to do for PROC1 where he wanted to represent a high likelihood of capture errors.

"DA. I'm stuck on PROC1."

"In the past, when you've made comments like this, we have usually ended up using sensitivity analysis."

"You're right. That's it."

"DA. Set up PHE for PROC1 to range from the same values as PROC2 to ten times those values. Then run the two models."

"Do you still want a ten-second moving average?"

"Yes, that's fine."

Henry strolled down the hall, sensing he had the answer. But it was going to take more than warm feelings to convince T&E that he'd found the source of the problem. It would take even more to convince the X95 cockpit design team to modify its software.

The plots on Henry's display looked very promising. The PROC1 results matched actual performance when PHE equaled roughly twice the baseline value. It looked like capture errors might in fact be the problem. The PROC2 results actually exceeded required performance a bit. Now, Henry thought, he needed some real data to support his explanation.

"DA. I need your help again in developing an experimental test design. Figure out how many subjects and test days we'll need. Then check the schedule at the X95 manned simulator for openings to run a test."

Henry needed a quick and dirty part-task simulation in which he could vary the same parameters as he did in the models except, of course, for the human performance parameters which he wanted to measure. After some back and forth with the DA about the test design, Henry is satisfied.

"DA. Send this plan to everybody involved and convene a meeting on the network for early tomorrow morning."

.
.
.
.

Ten hectic days later, Henry was sitting again in his office looking at the experimental results Things hadn't gone quite as planned. There were some holes in data and the last few sessions were useless because of an instrumentation failure.

"DA. What do you know about statistical analysis with lots of missing data."

"There is a general tutorial as well as tutorials associated with each of three statistics packages that we have."

"OK. Run the general tutorial, please."

.
.
.

Henry finally got to results and conclusions. It required all sorts of approximations and figuring out reasonable worst-case assumption. Not much fun.

"DA. Overlay the experimental results and the PROC2 results."

The predictions of X95 performance were a little lower than the actual measurements. The differences were not large but they were systematic. Henry wondered whether he was missing something.

"DA. Are any of the displayed differences between the data and the PROC2 predictions statistically significant?"

"None are significant. The underlying variances in both sets of data are very large."

"They don't look that large."

"You asked that both data sets be smoothed via ten-second averaging for display purposes. Do you want to see the original, unsmoothed plots?"

"No, thanks. I forgot. Let's not worry about it. Display the SOW from T&E again, please."

Rereading the SOW and carefully cross-checking with experimental results, Henry believed that the new design might actually exceed the performance requirements by 5% or so. He thought about changing his new icons pictures to make them a bit easier to confuse. This is the kind of perverse thinking that's allowed when you know you've found it, Henry thought.

"DA. Let's look at the T&E video again."

Henry watched Steve make landings and imagined him using the new operating procedures, as well as the moving references lines, icons, etc. It was going to work, he thought. The human errors that had been designed in, had now been designed out of the system.

"DA. Schedule a meeting with T&E and the X95 cockpit design team, please. You have my schedule. Query theirs and pick a time."

"Done."

Henry's on-line calendar blinked and then showed a highlighted block for next Tuesday.

"DA. Now about those subjective measurement procedures...."

REFERENCES

Allen, T.J. (1977). *Managing the flow of technology.* Cambridge, MA: MIT Press.

Allen, T.J. (1986). Organizational structure, information technology and R&D productivity. *IEEE Transactions on Engineering Management*, Vol. EM-33, 4, 212-217.

American Society of Mechanical Engineers. (1986). *Goals and priorities for research in engineering design: A report to the design research community.* New York: Author.

Bailey, R.W. (1982). *Human performance engineering: A guide for system designers.* Englewood Cliffs, N.J.: Prentice Hall.

Baldwin, D.F., Abell, T.E., Lui, M.M., DeFazio, T.L., & Whitney, D.E. (1991). An integrated computer aid for generating and evaluating assembly sequences for mechanical products. *IEEE Transactions on Robotics and Automation, Vol 7*, No. 1, 78-94.

Ballay, J.M. (1987). An experimental view of the design process. In W.B. Rouse and K.R. Boff (Eds.), *System design: Behavioral perspectives on designers, tools and organizations.* New York: North Holland.

Beevis, D. (1987). Experience in the integration of human engineering effort with avionics systems development. In *AGARD Conference Proceedings, 417, The Design Development and Testing of Complex Avionics Systems*, Neuilly Sur Seine, France.

Begg, V. (1984). *Developing expert CAD systems.* London: Kogan Page.

Blanchard, B.S. & Fabrycky, W. J. (1981). *Systems engineering and analysis.* Englewood Cliffs, New Jersey: Prentice Hall.

Boff, K.R. (1987a) The Tower of Babel revisited: On cross-disciplinary chokepoints in systems design. In W.B. Rouse and K.R. Boff (Eds.), *System design: Behavioral perspectives on designers, tools and organizations.* New York: North Holland.

Boff, K.R. (1987b). Designing for design effectiveness of complex avionics systems. In *AGARD Conference Proceedings, 417, The Design Development and Testing of Complex Avionics Systems*, Neuilly Sur Seine, France.

Boff, K.R. (1988) The value of research is in the eye of the beholder, *Human Factors Society Bulletin*, 31, 1-4.

Boff, K.R. (1990). Meeting the challenge: Factors in the design and acquisition of human-engineered systems. In H.E. Booher (Ed.), *People, machines and organizations: The MANPRINT approach to system integration.* New York: Van Nostrand Reinhold.

Boff, K.R. Kaufman, L., & Thomas, J.P. (1986). *Handbook of perception and human performance.* New York: Wiley.

Boff, K.R., & Lincoln, J.E. (1988). *Engineering data compendium: Human perception and performance.* Wright-Patterson AFB, OH: Armstrong Aerospace Medical Research Laboratory.

Boff, K.R., Monk, D., Swierenga, S., Brown, C. & Cody, W.J. (1991). Computer-aided human factors for system designers. In *Proceedings of the Human Factors Society 35th Annual Meeting*, San Francisco, CA, Vol II, pp. 332-336.

Bogner, M.S. (1988). *Catalogue of MANPRINT methods*. Alexandria, VA: U.S. Army Research Institute.

Borovansky, V.T. (1987). Teaching engineering students to utilize information resources. *International Journal of Applied Engineering Education*, 3, 87-92.

Broadbent, G. (1988). *Design in architecture*. London: David Fulton Publishers.

Brown, D.C. & Chandrasakaran, B. (1989) *Design problem solving*. London: Pitman.

Bruns, G.R. & Gerhart, S.L. (1986). *Theories of design: An introduction to the literature*. MCC Technical Report Number STP-068-86. Austin, TX: Microelectronics and Computer Technology Center.

Burger, W.E. (1991) Too many lawyers, too many law suits. *New York Times*, May 12, pp. 12.

Buur, J. & Andreasen, M.M. (1989). Design models in mechatronic product development. *Design Studies*, 10(3), 155-162.

Buxton, J.N. & Malcolm, R. (1991). Software technology transfer. *Software Engineering Journal*, 5, 17-23.

Carroll, J.M. & Thomas, J.C. (1982). Metaphor and the cognitive representation of computing systems. *IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-12*, 107-116.

Carroll, J.M, Thomas, J.C., & Malhotra, A. (1979). Clinical-experimental analysis of design problem solving. *Design Studies*, 1(2), 1-9.

Chandrasakaran, B. (1990). Design problem solving: A task analysis. *AI Magazine*, 11, 59-71.

Cody, W.J. (1989). Designers as users: Design supports based on crew system design practices. In *Proceedings of the American Helicopter Society 45th Annual Forum*, Boston.

Cody, W.J., & Rouse, W.B. (1989). A test of criteria used to select human performance models. In G. R. McMillan, (Ed.), *Applications of human performance models to system design*. New York: Plenum Press.

Corcoran, E. (1988). Groupware. *Scientific American*, July, 110-111.

Curtis, B., Krasner, H., & Iscoe, N. (1988) A field study of the software design process for large systems, *Communications of the ACM*, 31, 1268-1287.

Dekker, K. (1992). A future interface for computer-aiding styling. *Design Studies*, Vol. 13, 42-53.

Denning, P.J. (1985). Computer networks. *American Scientist*, Vol. 73, 127-129.

Dixon, J.R. (1966). *Design engineering: Inventiveness, analysis, and decision making*. New York: McGraw Hill.

Eder, W.E. (1988). Education for engineering design: Application of design science. *International Journal of Applied Engineering Education*, 4, 167-184.

Emery, J. & Parks, R. (1987). Crew station design process overview. Ft. Worth, TX: Bell Helicopter Textron.

Finger, S. & Dixon, J.R. (1989a). A review of research in mechanical engineering design. Part I: Descriptive, prescriptive, and computer-based models of design processes. *Research in Engineering Design*, 1, 51-67.

Finger, S. & Dixon, J.R. (1989b). A review of research in mechanical engineering design. Part II: Representations, analysis, and design for the life cycle. *Research in Engineering Design*, 1, 121-137.

Floyd, B. (1991). The CAD to come. *Workstation News*. April.

Forbus, K.D. (1988). Intelligent computer-aided engineering. *AI Magazine*, Fall, 23-36.

Frey, P.R., Sides, W.H., Hunt, R.M., & Rouse, W.B. (1984). *Computer-generated display system guidelines: Volume I: Display design*. (Technical Report. N3701, Volume 1). Palo Alto, CA: Electric Power Research Institute.

Galegher, J., Kraut, R.E., & Egido, C. (1990). *Intellectual teamwork: Social and technological foundations of cooperative work*. Hillside, NJ: Erlbaum.

Gardiner, M.M. & Christie, B. (eds.) (1987). *Applying Cognitive Psychology to User-Interface Design*. New York: Wiley.

Geddes, N.D. (1989). *Understanding human operators' intentions in complex systems*. Unpublished Ph.D. Dissertation, Georgia Institute of Technology.

Geddes, N.D. & Edwards, G.R. (1991). Human-machine cooperation in associate systems.

Gemunden, H.G. (1985). Perceived risk and information search. A systematic meta-analysis of the empirical evidence. *International Journal of Research in Marketing*, 2, 79-100.

Gero, J.S. (1990) Design prototypes: A knowledge representation scheme for design. *AI Magazine*, 11, 26-36.

Gerstberger, P.G. & Allen, T.J. (1968). Criteria used by research and development engineers in the selection of an information source. *Journal of Applied Psychology*, 52, 272-279.

Goel, V. & Pirolli, P. (1989) Motivating the notion of generic design within information-processing theory: The design problem space. *AI Magazine*, 10, 18-36.

Gould, J.D. & Lewis, C. (1985). Designing for usability: Key principles and what designers think. *Communications of the ACM*, 28, 300-311.

Guindon, R. (1992). Requirements and design of DesignVision: An object-oriented graphical interface to an intelligent software design assistant. In *Proceedings of CHI '92: Human Factors in Computing Systems*, Monterey, CA. New York: ACM.

Hadley, N. & Sommerville, I. (1990). Integrated support for systems design. *Software Engineering Journal*, 6, 331-338.

Hammond, N.V., Jorgensen, A.H., MacLean, A., Barnard, P.J. & Long, J.B. (1983). Design practice and interface usability: Evidence from interviews with designers. In *Proceedings of CHI '83: Human Factors in Computing Systems*, Boston. New York: ACM.

Hazelrigg, G.A. (1988) The engineering method. *Engineering Education*, Nov, 118-121.

Hunt, R.M. (1987) The difficulties of design problem formulation. In W.B. Rouse and K.R. Boff (Eds.), *System design: Behavioral perspectives on designers, tools and organizations*. New York: North Holland.

Johnson, E.M. (1987). The role of man in the system design process. In W.B. Rouse and K.R. Boff (Eds.), *System design: Behavioral perspectives on designers, tools and organizations*. New York: North Holland.

Kahn, R.E. (1987). Networks for advanced computing, *Scientific American*, 257, 136-143.

Kantowitz, B.H. & Sorkin, R.D. (1987). Allocation of functions. In G. Salvendy (ed.), *Handbook of human factors*. New York: Wiley.

Klein, G.A. (1987). Analytical versus recognitional approaches to design decision-making. In W.B. Rouse and K.R. Boff (Eds.), *System design: Behavioral perspectives on designers, tools and organizations*, 175-186. New York: North Holland.

Kraemer, K.L. & King, J.L. (1988) Computer-based systems for cooperative work and group decision making. *ACM Computing Surveys*, 20, 115-146.

Kryder, M.H. (1987). Data-storage technolgies for advanced computing. *Scientific American*, 257, 116-125.

Lambert S. & Ropiequet S. (eds.) (1986). *CD ROM: The New Papyrus*. Redmund, WA: Microsoft Press.

Libardi, E.C., Dixon, J.R., & Simmons, M.K. (1988) Computer environments for design of mechanical assemblies: A Research Review. *Engineering with Computers*, 3, 121-136.

Leifer, L. (1987). On the nature of design and an environment for design. In W.B. Rouse and K.R. Boff (Eds.), *System design: Behavioral perspectives on designers, tools and organizations*. New York: North Holland.

Linn, J.L. & Winner, R.I. (Eds) (1986). *The Department of Defense Requirements for Engineering Information Systems*. Alexandria, VA: The Institute for Defense Analysis.

Linton, P. (1987). *Crew station design methodology*. Stratford, CT: Sikorsky Aircraft Division, United Technologies Corporation.

Lintz, L.M., Askren, W.B., & Lott, W.J. (1971). *System Design Trade Studies: The Engineering Process and Use of Human Resources Data*. Report No. AFHRL-TR-71-24. Wright Patterson AFB, OH: Air Force Human Resources Laboratory.

Maher, M.L. (1990). Process models for design synthesis. *AI Magazine*, 11, 49-58.

Malhotra, A., Thomas, J.C., Carroll, J.M., & Miller, L.A. (1980). Cognitive processes in design. *International Journal of Man-Machine Systems*, 12, 119-140.

McMillan, G.R. (Ed.), (1989). *Applications of human performance models to system design.* New York: Plenum Press.

Meister, D. & D.E. Farr (1967). The utilization of human factors information by designers. *Human Factors*, 9(1), 71-87.

Meister, D. (1989). *Conceptual aspects of human factors.* Baltimore, MD: Johns Hopkins Press.

Meister, D., Sullivan, D.J., Finley, D.L., & Askren, W.B. (1969). *The Design Engineer's Concept of the Relationship Between System Design Characteristics and Technical Skill Level.* Report No. AFHRL-TR-69-22. Wright Patterson AFB, OH: Air Force Human Resources Laboratory.

Mostow, J. (1985). Toward better models of the design process. *AI Magazine*, 6, 44-57.

Nadler, G. (1985). Systems methodology and design. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-15, 685-697.

Newsome, S.L., Spillers, W.R., and Finger, S. (Eds.). (1989). *Design Theory '88.* New York: Springer Verlag.

O'Reilly, C.A. (1982). Variations in decision makers' use of information sources: The impact of quality and accessibility of information. *Academy of Management Journal*, 25, 756-771.

Ostrofsky, B. (1977). *Design, planning and developmental methodology.* Englewood Cliffs, New Jersey: Prentice Hall.

Packard, D. (Ed). (1986). *Interim report by the President's blue ribbon commission on defense management.* Washington, D.C.: U.S. Government Printing Office.

Pahl, G. & Beitz, W. (1984). *Engineering design.* New York: Springer Verlag.

Perrow, C. (1983). The organizational context of human factors engineering. *Administrative Science Quarterly*, 28, 521-541.

Perrow, C. (1984). *Normal accidents.* New York: Basic Books.

Promisel, D.M., Hartel, C.R., Kaplan, J.D., Marcus, A., & Whittenburg, J.A. (1985). *Reverse Engineering: Human Factors, Manpower, Personnel, and Training in the Weapon System Acquisition Process.* Technical Report 659. Alexandria, VA: U.S. Army Research Institute.

Pugh, S. (1989). Knowledge-based systems in the design activity. *Design Studies, 10*(4), 219-228.

Rangan, R. & Fulton, R.E. (1991). A data management strategy to control design and manufacturing information. *Engineering with Computers*, 7, 63-78.

Rasmussen, J. (1986). *Information processing and human-machine interaction: An approach to cognitive engineering.* New York: North Holland.

Rasmussen, J.R. (1988). A cognitive engineering approach to the modeling of decision making and its organization in process control, emergency management, CAD/CAM, office systems, and library systems. In W.B. Rouse (ed.), *Advances in Man-Machine Systems Research.* Greenwich, CT: JAI Press.

Riley, V. (1989). A general model of mixed-initiative human-machine systems. In *Proceedings of the Human Factors Society 33rd Annual Meeting*, Denver, Co.

Rinderle, J.R. et al. (1989). Form-function characteristics of electro-mechanical designs. In S.L. Newsome, W.R. Spillers and S. Finger (Eds.), *Design Theory '88*. New York: Springer Verlag.

Rosenblatt, A. & Watson, G.F. (Eds.) (1991). Concurrent engineering: Competitive product development. *IEEE Spectrum*, July.

Rouse, W.B. (1982). On models and modelers: N cultures. *IEEE Transactions on Systems, Man, and Cybernetics*, 12, 605-610.

Rouse, W.B. (1985). Optional allocation of system development resources to reduce and/or tolerate human error. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-15*, 620-630.

Rouse, W.B. (1986). On the value of information in system design: A framework for understanding and aiding designers. *Information Processing and Management*, 22, 217-228.

Rouse, W.B. (1987a). Designers, decision making, and decision support. In W.B. Rouse and K.R. Boff (Eds.), *System design: Behavioral perspectives on designers, tools and organizations*. New York: North Holland.

Rouse, W.B. (1987b). On meaningful menus for measurement: Disentangling evaluative issues in system design. *Information Processing and Management*, 23(6), 593-604.

Rouse, W.B. (1988). Intelligent decision support for advanced manufacturing systems, *Manufacturing Review*, 1, 236-243.

Rouse, W.B. (1991). *Design for success: A human centered approach to designing successful products and systems*. New York: Wiley.

Rouse, W.B. (1992). *Strategies for innovation: Creating successful products, systems, and organizations*. New York: Wiley.

Rouse, W.B. (in press). *Catalysts for change*. New York: Wiley.

Rouse, W.B. & Boff, K.R. (Eds) (1987). *System design: Behavioral perspectives on designers, tools, and organizations*. New York: North Holland.

Rouse, W.B. & Cody, W.J. (1986). Function allocation in manned systems design. In *Proceedings of the 1986 International Conference on Systems, Man and Cybernetics*. Atlanta, GA.

Rouse, W.B. & Cody, W.J. (1988). On the design of man-machine systems: Principles, practices, and prospects, *Automatica*, 24, 227-238.

Rouse, W.B. & Cody, W.J. (1989a). A theory-based approach to supporting design decision making and problem solving, *Information and Decision Technologies*, 15, 291-306.

Rouse, W.B. & Cody, W.J. (1989b). Designers' criteria for choosing human performance models. In G.R. McMillan, (Ed.), *Applications of human performance models to system design*. New York: Plenum Press.

Rouse, W.B., Cody, W.J., & Boff, K.R. (1991). The human factors of system design: Understanding and enhancing the role of human factors engineering. *International Journal of Human Factors in Manufacturing*, 1, 87-104.

Rouse, W.B., Cody, W.J., Boff, K.R., & Frey, P.R. (1990). Information systems for supporting design of complex human-machine systems.. In C.T.Leondes (Ed.), *Advances in control and dynamic systems*. Orlando, FL: Academic Press.

Rouse, W.B., Geddes, N.D., & Curry, R.E. (1987). An architecture for intelligent interfaces: Outline of an approach to supporting operators of complex systems, *Human Computer Interaction*, 3, 87-122.

Rouse, W.B., Geddes, N.D., & Hammer, J.M. (1990). Computer-aided fighter pilots, *IEEE Spectrum*, 27, 38-41.

Rouse, W.B., Kisner, R.A., Frey, P.R., & Rouse, S.H. (1984). *A method for analytical evaluation of computer-based decision aids*. Oak Ridge National Laboratory, Technical Report NUREG/CR-3655.

Rouse, W.B. & Rouse, S.H. (1983). *A framework for research on adaptive decision aids, Aerospace Medical Research Laboratory*, Report TR-83-082.

Roussopoulos, N., Mark, L., Sellis, T., & Faloutsos. (1991). An architecture for high performance engineering information systems. *IEEE Transactions on Software Engineering*, Vol. 17, 22-33.

Sage, A.P. (1981). Organizational and behavioral considerations in the design of information systems and processes for planning and decision support. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-11*, 640-678.

Salzberg, S. & Watkins, M. (1990). Managing information for concurrent engineering: Challenges and barriers. *Research in Engineering Design*, 2, 35-52.

Schon, D.A. (1983). *The reflective practitioner*. New York: Basic Books.

Sewell, D.R. (1990). *A study of designers' questions in a group problem solving context*. Atlanta, GA: Search Technology.

Shah, J.J. & Wilson, P.R. (1989). Analysis of design abstraction, representation, and inferencing requirements for computer-aided design. *Design Studies*, 10(3), 169-178.

Sheridan, T.B. & Verplank, W.L. (1978). *Human and computer control of undersea teleoperators*. (Tech. Report). Cambridge, MA: M.I.T. Man-Machine Laboratory.

Silverman, B.G. (1990). Critiquing human judgment using knowledge-acquisition systems. *AI Magazine*, Fall, 60-79.

Smith, G.F. & Browne, G.J. (in press) Conceptual foundations of design problem solving. *IEEE Transactions on Systems, Man & Cybernetics*.

Smith, J.M. (1981). Wicked design problems and interactive optimization. In *Proceedings of the 1981 IEEE International Conference on Cybernetics and Society*, Atlanta, Georgia, October, pp. 218-224.

Spillers, W.R. & Newsome, S.L. (1989). Design theory: A model for conceptual design. In S.L. Newsome, W.R. Spillers, and S. Finger (eds.) *Design Theory '88*. New York: Springer-Verlag.

Steier, D. (1990). Creating a scientific community at the interface between engineering design and AI. *AI Magazine*, 11, 18-22.

Suri, R. & Shimizu, M. (1989). Design for analysis: A new strategy to improve the design process. *Research in Engineering Design*, 1, 105-120.

Teorey, T.J., Yang, D., & Fry, J.P. (1986). A logical design methodology for relational database using the extended entity-relationship model. *Computing Surveys*, 18, 197-222.

U.S. Congress, Office of Technology Assessment. (1989). *Holding the edge: Maintaining the defense technology base*, OTA-ISC-420. Washington, D.C.: U.S. Government Printing Office.

Ullman, D.G., Dietterich, T.G., & Stauffer, L.A. (1988). A model of the mechanical design process based on empirical data: A summary. In J.S. Gero (ed), *Artificial intelligence in Engineering: Design*. Amsterdam: Elsevier.

Wahlstrom, B., Heinonen, R., Ranta, J., & Haarla, J. (1985). *The design process and use of computerized tools in control room design*. Nordic LIT-3.1 Project. Espoo, Finland: Technical Research Centre of Finland.

Webster, D.E. (1988). Mapping the design information representation terrain. *Computer*, 21, 8-23.

Whitney, D.E. (1990). Designing the design process. *Research in Engineering Design*, 2, 3-13.

Winner, R.I., Pennell, J.P., Bertrand, H.E., & Slusarczak, M.G. (1988). *The role of concurrent engineering in weapon system acquisition*. AD/A203 615. Alexandria, VA: Institute for Defense Analysis.

Zachary, W. (1986). A cognitively based functional taxonomy of decision support techniques, *Human-Computer Interaction*, 2, 25-63.

APPENDICES

A. GENERAL TASKS AND ALTERNATIVE SUPPORT CONCEPTS

The following description of general tasks and support concepts is taken from Rouse (1991).

In order to organize information on past experiences in developing aiding and support systems, Rouse & Rouse (1983) reviewed more than 100 past aids and support systems. Most of these efforts were in the aerospace industry. This review led to the conclusion that all of these efforts were concerned with supporting one or more of the set of 13 general tasks shown in Figure A-1. This set of tasks was sufficient to classify and describe all of the aids and support reviewed in this analysis. A subsequent analysis of aids and support systems in the process control domain confirmed the generality of the scheme in Figure A-1.

Execution and Monitoring

1. Implementation of plan
2. Observation of consequences
3. Evaluation of deviations from expectations
4. Selection between acceptance and rejection

Situation Assessment: Information Seeking

5. Generation/identification of alternative information sources
6. Evaluation of alternative information sources
7. Selection of alternative information sources

Situation Assessment: Explanation

8. Generation/identification of alternative explanations
9. Evaluation of alternative explanations
10. Selection of alternative explanations

Planning and Commitment

11. Generation/identification of alternative courses of action
12. Evaluation of alternative courses of action
13. Selection of alternative courses of action

Figure A-1. General system users perform thirteen types of task.

Two characteristics of the tasks in Figure A-1 are of particular significance. First, most of the tasks involve generation, evaluation and selection among alternatives. Use of this standard

terminology is helpful for identifying approaches to enhancing abilities and overcoming limitations in these tasks. The second noteworthy characteristic of Figure A-1 is the emphasis on alternatives in terms of interpretations of deviations, information sources, explanations and courses of action. hence, the structure of Figure A-1 is based on a three-by-four array of action words (generation, evaluation, selection) vs. objects of actions (types of alternative). This degree of structure brings an important consistency and rigor to the process of characterizing user-system tasks.

Aiding requirements can be expressed in terms of needs to support one or more of the tasks in Figure A-1. Equivalently, support requirements can be stated in terms of needs to assist users in generation, evaluation, or selection, and possibly implementation or observation. Based on the review of more than 100 systems, it was concluded that there are basically 17 alternative ways to support users in performing the 13 general tasks in Figure A-1. These 17 alternatives are summarized in Figures A-2 through A-6.

The most difficult support to provide is for generating alternatives, and there are few previous efforts to draw upon. This appears to be due, for the most part, to humans having great difficulty in specifying the attributes of desirable alternatives. Some progress has been made in using pattern recognition methods to infer attributes of desired alternatives from a set of examples. It is fairly straightforward to retrieve examples if users can define them appropriately. In general, as shown in Figure A-2, there are three basic ways to support generation of alternatives.

Evaluation of alternatives is easy to understand in that this type of activity is common within engineering analysis. The feasibility of supporting evaluation depends on the availability of appropriate models and calculation methods for the alternatives and measures of interest. The availability of models and methods can present difficulties when the phenomena of interest are complex. Finite difference methods and geometric modeling techniques exemplify evaluative supports for designers. Figure A-3 shows five approaches to supporting evaluation tasks which encompass the range of alternatives that have been pursued in different domains.

The majority of previous efforts to develop support systems have focused on selection among alternatives, in part because this type of support is most tractable. The two primary approaches to support are shown in Figure A-4.

If all of the alternatives have been specified, and the probability distributions associated with consequences of choosing each alternative are known, and the users' criteria can be assessed, then it is usually quite easy to determine the best or optimal alternative. For alternatives involving multiple stages, locations, and so on, this optimization problem can become tricky, but is, nonetheless, standard fare for control theory and operations research. This is not meant to denigrate the important function of selection support systems, but it is frequently found that identification of feasible alternatives and their likely consequences is sufficient for users to choose immediately without resorting to optimization. Thus, despite the great attention selection among alternatives has received, this task is usually not the most difficult task faced by users. Good support for generation and evaluation is typically more valuable, but also seldom available.

While support of generation, evaluation and selection is central to designing aiding and support systems, these types of support are not sufficient for a comprehensive approach to human-centered design. It can also be necessary to support observation (Figure A-5) and execution (Figure A-6).

1. For a given situation, a support system can retrieve previously relevant and useful alternatives.
2. For a given set of attributes, a support system can retrieve candidate alternatives with these attributes.
3. Given user's assessment of suggested alternatives (e.g., via ranking or rating), a support system can adapt its search strategy (e.g., via attribute weights or logical operations) to produce new alternatives.

Figure A-2. Genneration of alternatives can be supported in three general ways.

4. For a given alternative, a support system can assess the alternative's a priori characteristics such as relevance, information content and resource requirements.
5. For a given situation and alternative, a support system can assess the degree of correspondence between situation and alternative.
6. For a given alternative, a support system can assess (e.g., via simulation) the likely future consequences such as expected performance impact and resource requirements.
7. For given multiple alternatives, a support system can assess the relative merits of each alternative.
8. Given user's assessments of evaluation results (via requests for explanation), a support system can adapt its evaluations in terms of time horizon, statistical measures, etc.

Figure A-3. Evaluation of alternatives can be supported in five general ways.

9. For given criteria and set of evaluated alternatives, a support system can suggest (e.g., via optimization) the selection hat yields the "best" allocation in terms of human-system resources.

10. Given the user's assessments of selection (e.g., via ranking or ratings), reflecting perhaps individual differences and time-variations of criteria, preferences, and evaluations, a support system can adapt (e.g., by modifying criteria weights) its processing to provide suggestions that respond to these variations.

Figure A-4. Selection among alternatives can be supported in two general ways.

11. For given information, a support system can transform, format, and code the information to enhance human abilities and overcome human limitations.

12. For a given set of evaluated information, a support system can filter and/or highlight the information to emphasize the most salient aspects of the information.

13. For a given sample of information, a support system can fit models to the information in order to integrate and interpolate within the sample.

14. For given user and system constraints, as well as individual differences, a support system can adapt transformations, models, etc. (e.g., modify what information is presented and how it is presented.)

Figure A-5. Monitoring can be supported in four general ways.

15. For a given plan and information regarding the user's actions, a support system can monitor plan implementation for inconsistencies and errors of omission and commission.

16. For a given plan and information regarding the user's actions and intentions, a support system can perform some or all of the plan to compensate for the user's inconsistencies, errors or lack of resources.

17. Given information on intentions, resources available, priorities, etc., a support system can adapt its monitoring and/or implementation to reflect, for example, a change in user goals.

Figure A-6. Plan execution can be supported in three general ways.

B. DETAILED SUPPORT REQUIREMENTS FOR INFORMATION ACCESS AND USE

This Appendix contains the complete list of 613 support functions. Each function is expressed in terms of four elements:

- Support verb — Actions of the information system to support the designer.
- Designer verb — Actions by the designer that are enhanced by the support system's actions.
- Primary object — Object of the designer's actions (Figure 12)
- Modifying object — Noun plus preposition that modifies a primary object.

Supports are organized according to support verb.

```
Execute procedure to
        access
                []
                        modeling tools/packages
                        models of func.
                        model's variables
                        drawing tools/packages
                        prototyping tools/packages
                        experimental variables
        construct
                []
                        models of func.
                        drawings of form
                        prototypes
        create
                []
                        input/output representations
                        models of func.
                        drawings of forms
                        prototypes/mockups
                        data collection plan
                explanations of
                        req. info for current design
                        info on operational needs
                        req. info for past designs
                        func. of past designs
                        dev. of model's pred. from expect
                        forms of past designs
                        dev. of measured perf from expect
        evaluate
                []
                        input/output representations
                        models of func.
                        drawings of forms
                        prototypes/mockups
                dev. from expct of
                        model's pred.
                        measured performance
                explanations of
                        req. info for current design
                        info on operational needs
                        req. info for past designs
                        perf. attributes and criteria
                        func. of past designs
                        dev. of model's pred. from expect
                        forms of past designs
                        dev. of measured perf from expect
                sources of
                        req. info for current design
                        info on operational needs
                        req. info for past designs
                        info on func. of past designs
                        info on forms of past designs
        measure
                []
                        performance
        monitor
                processing of
                        modeling tools/packages
                        drawing tools/packages
                        prototyping tools/packages
        obtain
                []
                        req. info for current design
                        info on operational needs
                        req. info for past designs
                        info on func. of past designs
                        input/output representations
                        off the shelf models
                        modeling tools/packages
                        info on forms of past designs
                        off the shelf forms
                        drawings of forms
                        off the shelf prototypes
                        prototyping tools/packages
                explanations of
                        func. of past designs
                        forms of past designs
        run
                []
```

```
                                        models of func.
            select
                        []
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                        btw accpt/rejct of
                                        model's pred.
                                        measured performance
                        explanations of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        perf. attributes and criteria
                                        func. of past designs
                                        dev. of model's pred. from expect
                                        forms of past designs
                                        dev. of measured perf from expect
                        sources of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        info on func. of past designs
                                        info on forms of past designs
------------------------------------------------------------------------------
Explain attributes for search for measures to
            create
                        explanations of
                                        req. info for current design
            evaluate
                        []
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                        explanations of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        func. of past designs
                                        forms of past designs
                        sources of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        info on func. of past designs
                                        info on forms of past designs
------------------------------------------------------------------------------
Explain attributes for search for procedure to
            create
                        []
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                        explanations of
                                        info on operational needs
                                        req. info for past designs
                                        func. of past designs
                                        dev. of model's pred. from expect
                                        forms of past designs
                                        dev. of measured perf from expect
            evaluate
                        dev. from expct of
                                        model's pred.
                                        measured performance
------------------------------------------------------------------------------
Explain attributes for search to
            identify
                        []
                                        past designs with relevant req.
                                        past designs with relevant func.
                                        relevant I/O representations
                                        past designs with relevant forms
                        availability of
                                        explan. of func. of past designs
                                        input/output representations
                                        off the shelf models
                                        modeling tools/packages
                                        explan. of forms of past designs
```

```
                                        off the shelf forms
                                        drawing tools/packages
                                        off the shelf prototypes
                                        prototyping tools/packages
-------------------------------------------------------------------------------
Explain labels for search for procedure to
            evaluate
                        []
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                        explanations of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        perf. attributes and criteria
                                        func. of past designs
                                        dev. of model's pred. from expect
                                        forms of past designs
                                        dev. of measured perf from expect
                        sources of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        info on func. of past designs
                                        info on forms of past designs
            obtain
                        []
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        info on func. of past designs
                                        input/output representations
                                        off the shelf models
                                        modeling tools/packages
                                        info on forms of past designs
                                        off the shelf forms
                                        drawing tools/packages
                                        off the shelf prototypes
                                        prototyping tools/packages
                        explanations of
                                        func. of past designs
                                        forms of past designs
            select
                        []
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                        btw accpt/rejct of
                                        model's pred.
                                        measured performance
                        explanations of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        perf. attributes and criteria
                                        func. of past designs
                                        dev. of model's pred. from expect
                                        forms of past designs
                                        dev. of measured perf from expect
                        sources of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        info on func. of past designs
                                        info on forms of past designs
-------------------------------------------------------------------------------
Explain labels for search to
            identify
                        availability of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        info on func. of past designs
                                        info on forms of past designs
            locate
                        []
                                        req. info for current design
                                        info on operational needs
```

```
                                        req. info for past designs
                                        info on func. of past designs
                                        input/output representations
                                        off the shelf models
                                        modeling tools/packages
                                        info on forms of past designs
                                        off the shelf forms
                                        drawing tools/packages
                                        off the shelf prototypes
                                        prototyping tools/packages
                                explanations of
                                        func. of past designs
                                        forms of past designs
-------------------------------------------------------------------------------
Explain measures to
            evaluate
                                []
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                                explanations of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        func. of past designs
                                        forms of past designs
                                sources of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        info on func. of past designs
                                        info on forms of past designs
-------------------------------------------------------------------------------
Explain need to
            create
                                []
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                                explanations of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        func. of past designs
                                        dev. of model's pred. from expect
                                        forms of past designs
                                        dev. of measured perf from expect
-------------------------------------------------------------------------------
Explain procedure to
            access
                                []
                                        modeling tools/packages
                                        models of func.
                                        model's variables
                                        drawing tools/packages
                                        prototyping tools/packages
                                        experimental variables
            construct
                                []
                                        models of func.
                                        drawings of form
                                        prototypes
            create
                                []
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                                        data collection plan
                                explanations of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        func. of past designs
                                        dev. of model's pred. from expect
                                        forms of past designs
                                        dev. of measured perf from expect
            evaluate
                                []
```

```
                                    input/output representations
                                    models of func.
                                    drawings of forms
                                    prototypes/mockups
                          dev. from expct of
                                    model's pred.
                                    measured performance
                          explanations of
                                    req. info for current design
                                    info on operational needs
                                    req. info for past designs
                                    perf. attributes and criteria
                                    func. of past designs
                                    dev. of model's pred. from expect
                                    forms of past designs
                                    dev. of measured perf from expect
                          sources of
                                    req. info for current design
                                    info on operational needs
                                    req. info for past designs
                                    info on func. of past designs
                                    info on forms of past designs
               measure
                          []
                                    performance
               monitor
                          processing of
                                    modeling tools/packages
                                    drawing tools/packages
                                    prototyping tools/packages
               obtain
                          []
                                    req. info for current design
                                    info on operational needs
                                    req. info for past designs
                                    info on func. of past designs
                                    input/output representations
                                    off the shelf models
                                    modeling tools/packages
                                    info on forms of past designs
                                    off the shelf forms
                                    drawings of forms
                                    off the shelf prototypes
                                    prototyping tools/packages
                          explanations of
                                    func. of past designs
                                    forms of past designs
               run
                          []
                                    models of func.
               select
                          []
                                    input/output representations
                                    models of func.
                                    drawings of forms
                                    prototypes/mockups
                          btw accpt/rejct of
                                    model's pred.
                                    measured performance
                          explanations of
                                    req. info for current design
                                    info on operational needs
                                    req. info for past designs
                                    perf. attributes and criteria
                                    func. of past designs
                                    dev. of model's pred. from expect
                                    forms of past designs
                                    dev. of measured perf from expect
                          sources of
                                    req. info for current design
                                    info on operational needs
                                    req. info for past designs
                                    info on func. of past designs
                                    info on forms of past designs
-------------------------------------------------------------------------------
Explain search-by-attributes for measures to
               create
                          explanations of
                                    req. info for current design
               evaluate
                          []
```

```
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                              explanations of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        func. of past designs
                                        forms of past designs
                              sources of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        info on func. of past designs
                                        info on forms of past designs
-------------------------------------------------------------------------------
Explain search-by-attributes for procedure to
        create
                              []
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                              explanations of
                                        info on operational needs
                                        req. info for past designs
                                        func. of past designs
                                        dev. of model's pred. from expect
                                        forms of past designs
                                        dev. of measured perf from expect
        evaluate
                              dev. from expct of
                                        model's pred.
                                        measured performance
-------------------------------------------------------------------------------
Explain search-by-attributes to
        identify
                              []
                                        past designs with relevant req.
                                        past designs with relevant func.
                                        relevant I/O representations
                                        past designs with relevant forms
                              availability of
                                        explan. of func. of past designs
                                        input/output representations
                                        off the shelf models
                                        modeling tools/packages
                                        explan. of forms of past designs
                                        off the shelf forms
                                        drawing tools/packages
                                        off the shelf prototypes
                                        prototyping tools/packages
-------------------------------------------------------------------------------
Explain search-by-label for procedure to
        evaluate
                              []
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                              explanations of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        perf. attributes and criteria
                                        func. of past designs
                                        dev. of model's pred. from expect
                                        forms of past designs
                                        dev. of measured perf from expect
                              sources of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        info on func. of past designs
                                        info on forms of past designs
        obtain
                              []
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
```

```
                                availability of
                                                explan. of func. of past designs
                                                input/output representations
                                                off the shelf models
                                                modeling tools/packages
                                                explan. of forms of past designs
                                                off the shelf forms
                                                drawing tools/packages
                                                off the shelf prototypes
                                                prototyping tools/packages
-------------------------------------------------------------------------------
Search-by-label for procedure to
                evaluate
                                []
                                                input/output representations
                                                models of func.
                                                drawings of forms
                                                prototypes/mockups
                                explanations of
                                                req. info for current design
                                                info on operational needs
                                                req. info for past designs
                                                perf. attributes and criteria
                                                func. of past designs
                                                dev. of model's pred. from expect
                                                forms of past designs
                                                dev. of measured perf from expect
                                sources of
                                                req. info for current design
                                                info on operational needs
                                                req. info for past designs
                                                info on func. of past designs
                                                info on forms of past designs
                obtain
                                []
                                                req. info for current design
                                                info on operational needs
                                                req. info for past designs
                                                info on func. of past designs
                                                input/output representations
                                                off the shelf models
                                                modeling tools/packages
                                                info on forms of past designs
                                                off the shelf forms
                                                drawing tools/packages
                                                off the shelf prototypes
                                                prototyping tools/packages
                                explanations of
                                                func. of past designs
                                                forms of past designs
                select
                                []
                                                input/output representations
                                                models of func.
                                                drawings of forms
                                                prototypes/mockups
                                btw accpt/rejct of
                                                model's pred.
                                                measured performance
                                explanations of
                                                req. info for current design
                                                info on operational needs
                                                req. info for past designs
                                                perf. attributes and criteria
                                                func. of past designs
                                                dev. of model's pred. from expect
                                                forms of past designs
                                                dev. of measured perf from expect
                                sources of
                                                req. info for current design
                                                info on operational needs
                                                req. info for past designs
                                                info on func. of past designs
                                                info on forms of past designs
-------------------------------------------------------------------------------
Search-by-label to
                identify
                                availability of
                                                req. info for current design
                                                info on operational needs
                                                req. info for past designs
```

```
                                        info on func. of past designs
                                        info on forms of past designs
            locate
                        []
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        info on func. of past designs
                                        input/output representations
                                        off the shelf models
                                        modeling tools/packages
                                        info on forms of past designs
                                        off the shelf forms
                                        drawing tools/packages
                                        off the shelf prototypes
                                        prototyping tools/packages
                        explanations of
                                        func. of past designs
                                        forms of past designs
-------------------------------------------------------------------------------
Transform, filter, highlight
            []
                        []
                                        model's variables
                                        experimental variables
-------------------------------------------------------------------------------
Tutor use of procedure to
            access
                        []
                                        modeling tools/packages
                                        models of func.
                                        model's variables
                                        drawing tools/packages
                                        prototyping tools/packages
                                        experimental variables
            construct
                        []
                                        models of func.
                                        drawings of form
                                        prototypes
            create
                        []
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                                        data collection plan
                        explanations of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        func. of past designs
                                        dev. of model's pred. from expect
                                        forms of past designs
                                        dev. of measured perf from expect
            evaluate
                        []
                                        input/output representations
                                        models of func.
                                        drawings of forms
                                        prototypes/mockups
                        dev. from expct of
                                        model's pred.
                                        measured performance
                        explanations of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        perf. attributes and criteria
                                        func. of past designs
                                        dev. of model's pred. from expect
                                        forms of past designs
                                        dev. of measured perf from expect
                        sources of
                                        req. info for current design
                                        info on operational needs
                                        req. info for past designs
                                        info on func. of past designs
                                        info on forms of past designs
            measure
                        []
```

```
                                performance
monitor
                processing of
                                modeling tools/packages
                                drawing tools/packages
                                prototyping tools/packages
obtain
                []
                                req. info for current design
                                info on operational needs
                                req. info for past designs
                                info on func. of past designs
                                input/output representations
                                off the shelf models
                                modeling tools/packages
                                info on forms of past designs
                                off the shelf forms
                                drawings of forms
                                off the shelf prototypes
                                prototyping tools/packages
                explanations of
                                func. of past designs
                                forms of past designs
run
                []
                                models of func.
select
                []
                                input/output representations
                                models of func.
                                drawings of forms
                                prototypes/mockups
                btw accpt/rejct of
                                model's pred.
                                measured performance
                explanations of
                                req. info for current design
                                info on operational needs
                                req. info for past designs
                                perf. attributes and criteria
                                func. of past designs
                                dev. of model's pred. from expect
                                forms of past designs
                                dev. of measured perf from expect
                sources of
                                req. info for current design
                                info on operational needs
                                req. info for past designs
                                info on func. of past designs
                                info on forms of past designs
```

U.S.G.P.O.:1993-550-057/81031