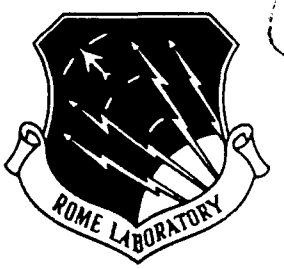
RL-TR-91-177
Final Technical Report
August 1991



INTERFACE DESIGN TOOLS PROJECT

BBN Systems and Technologies

William J. Salter, Dan Cerys, Bruce Papazian,
R. Bruce Roberts

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

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

RL-TR-91-177 has been reviewed and is approved for publication.

APPROVED: Douglas A. White

DOUGLAS A. WHITE
Project Engineer

APPROVED: Raymond P. Urtz Jr.

RAYMOND P. URTZ, JR.
Technical Director
Command, Control & Communications Directorate

FOR THE COMMANDER: Ronald Raposo

RONALD RAPOSO
Plans & Programs Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL(COES) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1991	Final Jan 89 to May 90

4. TITLE AND SUBTITLE
INTERFACE DESIGN TOOLS PROJECT

5. FUNDING NUMBERS
C - F30602-87-D-0093
PE - 63728F
PR - 2532
TA - QA
WU - 06

6. AUTHOR(S)
William J. Salter, Dan Cerys, Bruce Papazian,
R. Bruce Roberts

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
BBN Systems and Technologies
A Division of Bolt Beranek & Newman, Inc.
10 Moulton Street
Cambridge MA 02138

8. PERFORMING ORGANIZATION REPORT NUMBER
7562

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (COES)
Griffiss AFB NY 13441-5700

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-91-177

11. SUPPLEMENTARY NOTES
Rome Laboratory (RL) - Formerly Rome Air Development Center (RADC)
RL Project Engineer: Douglas A. White/COES/(315) 330-3564

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

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
This report describes an effort to provide tools that assist in the design and development of user interfaces. Human factors knowledge is used to provide an intelligent capability which can be applied to generate interface alternatives and the underlying code for those alternatives. Two gaps evident in available tools for dealing with windows were also addressed: clustering windows and linking windows. "Clustering" deals with the grouping of multiple windows, and the tools provided facilitate the specification of screen configurations. "Linking tools" provide mechanisms to specify and enforce functional dependency between windows so that when what is in one changes the other windows change in pre-specified ways.

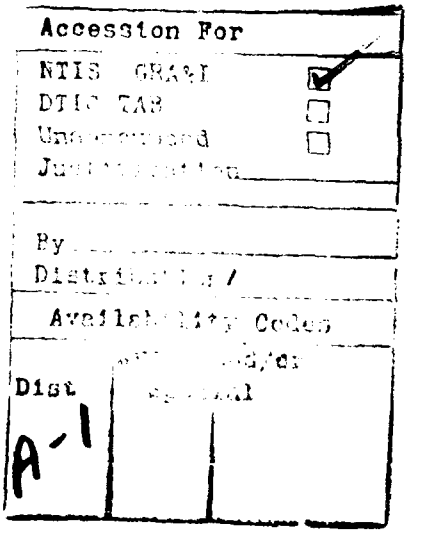
14. SUBJECT TERMS
Artificial Intelligence, Windows, Design, User Interface, Intelligent Tools

15. NUMBER OF PAGES
20

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	UL

Table of Contents



1.0 Introduction

This document is the Final Report of Task 3, sponsored by the Rome Air Development Center under contract number F30602-87-D-0093. The primary goal of this task was to develop and deliver Macintosh-based software that could be used to assist in the design of user-machine interfaces for C2 applications. The particular emphasis in this software is on incorporating Human Factors knowledge and guidelines. The software itself is discussed in some detail in BBN Report Number 7480, "Interface Design Tools Project Software User's Manual." In this document, we briefly lay out the development of our approach to the software, and present the major lessons that we have learned. We conclude with a summary of possible future activities that could support the further development of tools for user interface design and development.

The work on this task was done primarily by Dan Cerys, Bruce Papazian, Bruce Roberts, and William Salter of BBN; we built on work that also involved Richard Pew, Don Redick, Dan Tani, and Roland Zito-Wolf of BBN, and Mark Friedell and Joe Marks of Harvard University. We would like to thank our COTRs at RADC, Mike McHale and later Doug White, for the support they have provided to this activity. The statements and recommendations in this report are those of the authors, and do not represent the expressed or implied positions of the Rome Air Development Center, the Air Force, or the United States Government.

2.0 Approach and Rationale

We began this work with the hope that the Human Factors literature would be advanced and detailed enough to make it possible to

essentially automate the process of developing user interfaces, given a sufficient specification of user tasks. After extensive review of the literature and of the software tools based on it (presented in BBN Report Number 6932, "An Intelligent Tool for the Design of Presentations: A System Identification Study," the Final Report of Task 2 under the same RADC contract), we concluded that this goal was unrealistic. Many of the most important issues for user interface design were barely treated in the field, while much of the work focussed on lower-level aspects of interface design. For example, there was very little literature dealing with the various types of information processing tasks that users perform, and how interfaces can facilitate those tasks, while there was a considerable amount about type sizes and layouts. Although important, an understanding of these lower-level aspects of the interface is not adequate for specifying interface design.

We conceptualize the interface design problem into seven steps:

1) **construct as model of the user's tasks**. We define this in terms of information exchange primitives, which specify the different ways in which a user and a system can interact.

2) **allocate functions to the user or the system**. This is essentially the activity of scoping the application, focusing on the roles of user and system; note that these two roles must be considered and defined in tandem.

3) **develop a user-system dialogue model**. This refines the allocation of functions into interactions and sequences of interactions between user and system.

4) **assign control and display primitives to information input and output tasks**. This begins the specification of the interface *per se*; here, the abstract elements of

the dialogue are associated with specific interface components. The set of such components are constrained by the relevant hardware and software conventions; in this task, we focused on the Macintosh, and thus menus, dialogue boxes, windows, and the like were important components.

5) **group primitives into presentations**. This associates components into screens.

6) **determine interpresentation transition methods**. This addresses issues of how to move among presentations.

7) **develop a functional interface prototype.**

We drew several conclusions from our review that have helped to guide our approach. First, the **Human Factors literature included recommendations about the design process itself, as well as about facts and rules** about the actual designs of interfaces. We have used this literature, although (like most extensive scientific literatures) it is by no means perfectly consistent.

Second, and perhaps most important, since it serves to redefine the fundamental nature of the interface design task: **the design process is better conceived of as the reconciliation of conflicting constraints than as the application of clear-cut rules**.

Third, effective design requires collection and application of knowledge of several distinct types: **customer requirements, domain knowledge, computer science, Human Factors, underlying system, and user task.** In particular, we determined that a heavy emphasis must be placed on understanding user tasks and conventions from the domain when selecting dialogue and

display mechanisms. And we found that incorporating sufficient domain knowledge and conventions substantially reduces, and often eliminates, the need for applying knowledge about fundamental perceptual processes.

Our review of the state of the art of interface design tools showed that the majority of system have been built with a relatively narrow view of the interface design process. The coverage of the existing tools is often narrower than implied by the way they are discussed in the literature. In general, the majority of system focus on speeding up the development process (a worthwhile goal, to be sure) rather than on improving the resulting design. Typically, graphic design principles predominate over encoded Human Factors guidelines. This is, at least in part, because the human factors literature is not generally expressed in a way that is amenable to coding into rules, due to its situation-specific nature and the degree of background knowledge and assumptions assumed. And, somewhat surprisingly, we found that the use of constraint-based paradigms was less prevalent here than in other design domains.

3.0 The Interface Design Tools Developed on this Project

Our overall objective **was to improve the quality of C2 application interfaces by leveraging programmers' time.** This led to five specific aspects of our approach:

1) Provide embedded guidance on interface design;

2) Provide tools to facilitate interface design;

3) Use a set of mature interface primitives (the Macintosh Toolbox);

4) Extend this set of primitives to support C2 applications more fully;

5) Integrate these tools with the (Coral) Apple Allegro lisp environment.

We particularized this approach to address six **goals for the interface design process** itself:

1) Raise the level of the interface specification;

2) Enable the programmer to integrate domain, application function, and user task knowledge in designing the interface;

3) Provide sound and useful interface guidance for varying degrees of domain, application, and task knowledge;

4) Assist the programmer in identifying further information needs that can help to guide the design;

5) Allow the programmer to override system recommendations;

6) Enable the programmer to explore alternatives rapidly and efficiently.

In terms of actual interface implementation, our approach had several noteworthy features:

1) Support the assignment of interface primitives to user-computer I/O tasks;

2) Support the aggregation of interface primitives into presentations;

3) Extend the Toolbox to support the linking of windows;

4) Provide map manipulation capabilities (since these seemed to be of such importance in the particular C2 domains of relevance); and

5) Provide interface code-generation capabilities, such that the programmer could go directly from the specification to actual running interface code.

We developed an integrated suite of tools, and supported three linked, but distinct, views of interface development: the **user's view** of the interface, which provides the actual look and feel on the resulting interface; the **programmer's view** of coding the interface, which consists of a rich lisp programming environment, closely coupled with Allegro Commonlisp; and the **interface designer's view**, which is presented in terms of specifications, structures, and guidance on designing interfaces. The user's view gives the programmer access to the actual look and feel of the resulting interfaces, and also makes it possible to access the underlying lisp code for various parts of the interface; in addition, it provides access to the functional specifications contained in the interface design view, making it possible to explore the relationships of concrete aspects of the interface with the reasons for them.

The programmer's view gives direct access to the lisp code that implements the design; it is fully integrated with the Allegro Commonlisp environment, including the Dialogue Designer. And it also provides access to the design constraints that have shaped the specific design. The interface designer's view mediates between the user's view and the programmer's view. It provides an efficient means to discriminate among design alternatives using functional

information. It also supports using partial specifications to yield a subset of design alternatives and associated requirements for specifying additional information to select among the alternatives. It generates explanations as information is supplied and alternatives are eliminated. And it permits the programmer to override the system's recommendations.

4.0 Possible Next Steps

We believe that the tools we have developed can facilitate the interface design process, particularly for Macintosh applications (as defined in the RADC SOW) in the C2 domain. However, we also have identified five areas in which we think further advances can be made, advances that are both technically feasible and cost-justified in terms of incremental efficiency of the interface design process. We briefly discuss these areas in this section.

First, there should be **additional design support for the presentation of groups**. By "groups" we mean sets of alternatives or choices where the set size is greater than one (such as topographical features to be displayed on a map). Specification of groups pervades applications, and a number of design approaches are supported within the Macintosh style guidelines and by the Toolbox. We can build on the structure-function mapping paradigm we used for the work described above. and would focus on specifying the detailed semantics of the user-machine interaction, the particular selection characteristics of the task, and the physical constraints (and resources) of the display monitor. We note that a variety of dialog box types, window types, and palettes supported by the Toolbox must be addressed.

Second, **expanded on-line design advice** could be useful. Such advice could build on models of human performance and interface usability. The advice could address two linked problems: alternative

design comparisons for underspecified alternatives, and the use of *a priori* constraints to narrow the design space. Metrics exist for **number of required mouse clicks, display complexity,** and **choice time**. Such metrics can be integrated into the set of design tools and could be used to help in the evaluation, or initial selection, of interface design alternatives.

Third, we believe that substantial advances can be made in the **interpretation and utility of prototype interface usage data**. A number of approaches to interface design have "instrumented" prototypes, recording the time and location of each mouse click. In general, such approaches result in masses of data that are quite time-consuming to analyze and generally do not justify the effort and costs of using them. We recommend an approach centered on the time history of the use of selection mechanisms. The basic idea is to find the access paths taken to get to various functions within the application, and to optimize access time across functions, taking account of frequency and sequences of use. We are convinced that this basic approach can be quite useful. In particular, it can help in identifying: functional choices that are hard to find, choices that are buried too deeply, ambiguous menu terminology, and selection or access mechanisms that can be grouped functionally.

We also believe that enhancements can be made in **tools for redesigning prototypes**, based on usage. Access to interfaces within prototypes can be enhanced using both the Apple Interface Designer and our own suite of tools. Links should be developed between the sort of time history discussed above and design views of the prototype, so that the implications of the usage data can be quickly and directly incorporated into the developing design. And tools could be extended to support an efficient iterative process of design, prediction of use, testing, and redesign.

Finally, we also believe that **further treatment can be given to layout and aesthetic considerations**. Such considerations can be applied after the functional specifications and design have occurred. They would provide more detailed advice on Macintosh layout conventions and graphic design considerations, and would give the designer more flexible control of the layout of choice selection mechanisms.

In closing, we note that the increasing power and scope of computers enhances the need for clarity and consistency in the design of interfaces. The proliferation of useful applications can, paradoxically, make it harder for a user to find required functionalities, since differences across applications in conventions of usage, commands, and the like can make it more difficult to use a new application, or to move among applications. Guidelines and tools for application and interface design are more needed -- and more possible -- now than ever before. We believe that such guidelines and tools will make it easier for application builders to deliver utility to users, rather than constraining them in their choices, and we plan to be a part of this critical developing area for some time to come.

MISSION

OF

ROME LABORATORY

Rome Laboratory plans and executes an interdisciplinary program in research, development, test, and technology transition in support of Air Force Command, Control, Communications and Intelligence (C^3I) activities for all Air Force platforms. It also executes selected acquisition programs in several areas of expertise. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C^3I systems. In addition, Rome Laboratory's technology supports other AFSC Product Divisions, the Air Force user community, and other DOD and non-DOD agencies. Rome Laboratory maintains technical competence and research programs in areas including, but not limited to, communications, command and control, battle management, intelligence information processing, computational sciences and software producibility, wide area surveillance/sensors, signal processing, solid state sciences, photonics, electromagnetic technology, superconductivity, and electronic reliability/maintainability and testability.