
Rua das Cerejeiras, 22, casa 5
88040 Florianopolis-SC
Brazil

*Visible Language*, XXIV:3/4
Gui Bonsiepe, pp. 262-285

Rhode Island School of Design
Providence, Rhode Island 02903

# Interpretations of human user interface

## • Language

Gui Bonsiepe

Several interpretations of human user interface are reviewed and a proposal is developed for understanding the new category of tools in the form of computer programs. The advantages of a graphical user interface is compared to that of a character based interface. The contribution of the designer to the articulation of the retinal space in which these tools appear is outlined. The theoretical part is accompanied by a detailed case history of the design of an electronic mail application. The relationship between an interface science and an interface design is commented on but the proper domain of interface design is distinct from both science and art.

The interface design of computer programs and applications constitutes a new field for graphic design. The prevailing computer-science approach to the role of interface design is often dismissed as cosmetic. This is comparable to other engineering conceptions of design in which the domain of users and aesthetics with its many intangibles is fuzzy at best. It is not difficult to measure how many seconds it takes to transmit an electronic message in a local area network; but it is not so easy to find out why an application is assessed as elegant or practical or easy to learn, i.e., to handle those domains that are not accessible to the methodology of counting. Interface design is particularly challenging because of the interaction of dynamic graphics and language, and because it is the domain where programs are assessed by users and buyers of applications. It is, indeed, a fuzzy area—and in that aspect it does not differ fundamentally from other areas of design—both in practical and theoretical aspects.

Though in recent years the issue of interface design under the headings of "look and feel" has turned into an area of dispute between companies in the courts, the question of what constitutes an interface has not been answered convincingly. There is no consensus about what interfaces are, what constitutes an interface, and what makes an interface a good one.

According to the dominant paradigm, the human user interface is a:

**"means by which people and computers communicate with each other."**[1]

This widely shared communication paradigm appears also in the following definition:

**"A human interface is the sum of all communication between the computer and the user. It's what presents information to the user and accepts information from the user."**[2]

Though one can understand the penchant for looking at the relation between user and computer in terms of a communication process, where, as it is maintained, information is exchanged, it shifts the focus from perceiving computer programs in terms of tools and *action*, and sets them apart as a distinct class of immaterial tools.

In addition to the communication paradigm, there are those who maintain that the key issue of interface design consists in helping the user to build a mental model in his brain; a model that replicates the knowledge of the programmer, who has an intimate insight into the working details of the program, which is hidden from the user's view. The user, thus, would have learned a program once he has constructed a private counterpart to the programmer's model. Difficulty with learning and using an application is explained as either lack of a model or an erroneous model to which the user adheres. The quality of an interface design, consequently, would show up in the speed and correctness of a replica constructed by the user. This model-building paradigm is based on assumptions of how learning occurs and makes use of underlying philosophical assumptions that should not be taken for granted. One can say that the user has learned to use an application when it becomes transparent so that he no longer has "to think of it," i.e., when it disappears into the background so that he can focus on accomplishing the task at hand; when it does not get in his way.

Possibly the immaterial character of the new tool has fostered both the communication and mental-model-building paradigm, because the user has no direct access to the program; it is completely unlike entering a car and handling the steering wheel. But one can speculate that the user might be less concerned with communicating with an object called computer or building an isomorphic model in his brain, than in getting certain tasks achieved effectively. Only for this reason are tools invented, produced, learned and used. In this respect applications are not different from chairs, lathes or coffee grinders.

Yet another technical guide defines the graphical user interface as:

> **". . . a specification of the 'look and feel' of a computer system. This includes what types of objects the user sees and the basic conventions for how the user interacts with those objects."**[3]

This proposal captures more adequately the character of the interface and interface design by mentioning specifications, i.e. design of graphical components (what the user sees) and rules for dealing with these components on a screen via keyboard,

mouse, tablet or simply by touching the screen. These rules or conventions are established through language revealing the *linguistic base* of applications and their interfaces. The Macintosh operating system with its graphical components—the widely accepted desktop metaphor—has achieved the status of a proper visual culture. The graphical components in the form of *windows, icons, menus* and *push-buttons* are understood as metaphorical representations of a reality with which users in today's offices are familiar. However, these graphical objects rather than picturing a reality, are *constituting* a reality. Instead of using metaphors the designers of human user interfaces make use of *recurrence*. For the user the distinction between a metaphorical world and a "real" world is of remote interest. The user lives and works in *one* world, acting with tools, not with metaphors. Therefore, it might be more appropriate to say that the pictorial elements on a computer monitor don't picture anything, rather they *bring forth an action space*. This action space is articulated by graphical distinctions with which the retinal space of the users is structured.

The central concern that the use of communication paradigms and desktop metaphors address has to do with *ready-to-handness* and with the already mentioned idea of recurrence. Using this Heideggerian notion, the interface of applications can be characterized as the domain where the ready-to-handness of the tool (program) shows up. This notion of a domain offers richer distinctions than the assessment "user-centered" insofar as it incorporates the intrinsic possible breakdown of any tool and the different modalities in which ready-to-handness appears.

Using the notion of *structural coupling* of Maturana and Varela,[4] one can say that the *interface is the domain in which the structural coupling between body and program occurs and in which the tool is brought forth*. In the same way as the handle of the hammer couples structurally the human body to the tool, the interface works as the "handle" for the program. The coupling occurs primarily through the retinal space, with minor participation of the aural space. The notion of "look and feel" refers to this process of structural coupling between body and tool through visual perception. A user has learned an application when he/she has got it into his body, when he knows which keys to press and when to click the mouse.

The retinal space is structured through graphical distinctions with which the graphic designer is familiar:

- **shape**
- **color**
- **size**
- **position**
- **orientation**
- **texture**
- **transitions or transformations in time similar to what happens in movies and TV.**

Of particular importance for the design of interfaces are *action or event triggers* such as buttons or command options listed in a pull-down menu. The selection of commands, their naming, their organization in groups, their distribution over different levels and their visual treatment is a fundamental part of interface design which requires a linguistic competence of the graphic designer not provided by established graphic design study programs. This might help to explain why graphic designers generally encounter difficulties in getting access to this new field of design. It exceeds the frontiers of what graphic designers traditionally do and are taught to do. *(Table 1 begins to define the knowledge base designers need to approach interface design while Table 2 defines the designer's role on an interface team.)*

*Table 1*

List of desirable competencies that the designer should have when working in the field of interface design.

- **know interface standards**
- **know basic distinctions of computer science and programming**
- **know prototyping programs**
- **know animation programs**
- **know scripting language (Hypertalk, Supercard)**
- **know basics of graphic design in a computational medium**
- **know techniques of usability testing**
- **have linguistic competence**
- **know learning theories**
- **know basics of expert systems**

*Table 2*

List of contributions that can be made by the designer to the development of interfaces.

- **observe, analyze and interpret tasks and work processes**
- **formulate user functionality (services that the program should offer)**
- **organize commands (action triggers)**
- **define possible flow charts of action sequences—story boards**
- **design layout of the action space (screens)**
- **design tool palettes, e.g., color**
- **design templates**
- **design technical documentation**
- **design tutorials**

Sometimes it has been claimed that from the point of view of the user, the interface *is* the application. This is a strong claim, because the application is assessed in many ways: in terms of speed, reliability (absence of "bugs"), mail programs (security against loss of mail) and "power" (list of features or what the user can do with the application). These distinctions go beyond what is the core of interface and interface design, though certainly the power and speed of an application shows up through the interface. This strong claim can, however, be backed by the fact that today the design of computer programs starts with the simulation of the interface; there is an explicit concern for the end user—how he can learn and use the application and how it opens up new possibilities of action for him. This signals a considerable shift from the first generation of program development where interface design generally tended to be dealt with as a nuisance and bottleneck *after* the questions of functionality and program architecture had been resolved. The historical merit and uniqueness of the inventors of the seminal STAR interface resides in the fact that they spent many man/years of work on the interface before focusing on other aspects or even hardware questions.

Today human user interface design is recognized as a legitimate area of specialization because programs do not simply have to function in strictly computational terms, but also have to be learned and used by a large community of users, who are not necessarily conversant or even minimally competent with programing. Moreover, the products have to be marketed—a very difficult task when dealing with applications permeated with a blindness for the issues of interface design. In the academic community, one can perceive a shift to attending to the user, for example, the last SIGCHI (Special Interest Group of Computer Human Interaction) conference of the ACM (American Computing Machinery) in April, 1990 is an indication with its title,"Empowering People."

*Table 3*

List of different program types which reveals the variety of different interface requirements.

- **text editors**
- **spreadsheets**
- **presentation (slides)**
- **layout**
- **illustration, rendering (2 and 3D)**
- **databases (relational and flat)**
- **drawing (CAD)**
- **animation**
- **communication (electronic mail)**
- **coordination of work groups**
- **games**
- **prototyping tools**
- **accessories and utilities**

## DESIGN RULES AND USABILITY

Various publications on the subject of interface reveal the emergence of a new area of design. Possibly the most widely known efforts in this field are the *Guidelines for the Apple Desktop*.[5] The arguments in favor of consistency both within and between applications are simple: consistency in and between interfaces permits faster learning and easier use. At least, this is a generally accepted claim of graphical interfaces.

The design rules deal with the components or visual building blocks with which the action space on the monitor screen is constructed. Furthermore they refer to the operational sequences (action chunks) that the user needs to perform in order to achieve certain tasks. And ultimately the acoustical signals often used as feedback for certain operations, e.g., a beep to indicate that the user hit a wrong key. Developed on the base of the pioneering work of the STAR interface designed at Xerox PARC, the exemplary value of these efforts becomes evident when looking at similar attempts of SUN Microsystems,[6] IBM[7] and OSF (Open Software Foundation).[8] They signal a trend away from character-based interfaces that are typical for MS-DOS programs to graphical user interfaces that are said to be more user-friendly. Specialists prefer to speak of user-centered design. This implies that right from the beginning the user with his incompetence and standard practices is taken into consideration. In other words: user-centered design implies the same orientation as ready-to-hand design. It developed in opposition to program-centered development. However, the notion of ready-to-handness with its different modalities has greater analytical power. It is not a positive attribute that

some programs might have and others might lack, but a name for a domain that contains different modalities like conspicuousness, obstinateness and obtrusiveness—notions that capture and throw light onto the tool-character of tools. The elaboration of these terms goes beyond the scope of the present article; they indicate the possibility to understand applications within a framework of a general theory of tools, their design and their use.

## A CASE HISTORY:
### *interface for an electronic mail application*

The foregoing considerations serve as a framework for a report on the development of an electronic mail application that permits "talk" from PCs to Macintosh computers and between Macintosh computers. (Visual documentation of this project follows the main text.) Certain boundary conditions were accepted right from the start, e.g., the maximum length of eight characters for the names of addressees (*see figures 13 and 16*). This inheritance from the MS-DOS world is generally considered an inadmissible breach of the MAC conventions. In orthodox MAC circles the slightest concession to the PC universe comes close to an act that deserves reprobation.

The development team[9] started with preliminary (pencil) sketches of the interface after the general specifications had been agreed upon. These specifications derive to a great extent from what users expect from an electronic mail program. Apart from the decisive technical questions of connectivity (what types of different networks can be accessed), data protection (avoiding the loss of or damage to the mail), program architecture and the questions of how electronic mail is delivered (store-and-forward, and node delivery systems), the following, sometimes obvious, user functions are considered desirable:

- **write and read notes**
- **select single and multiple addressees**
- **organize mail automatically in a private database (incoming and outgoing mail is stored without obliging the user to handle individual files**
- **retrieve mail from the data base**

according to different criteria (by date, by person,by topic)
- create groups of addressees for easier addressing
- send in different delivery modes (regular, urgent, registered/return receipt)
- maintain the data base (efficient and simple deleting of outdated notes)
- attach a document to a note
- forward a note with a comment
- get a visual and aural notification when new mail arrives (optional feature)
- guarantee privacy of the data base (password protection)
- copy address books into the personal address book (optional feature that can save the tedious work of typing large lists of names when working in a big company)
- offer different forms (in this case only a special telephone slip has been implimented).

In the first sketches vertical and horizontal palettes were used which contain the various "tools" for writing, reading, addressing, posting mail etc. (*see figure 1*). The argument for this approach was that the user should have all the tools in the form of commands immediately available around his main working area (reading/writing space), instead of accessing them through a list of commands hidden in a menu. However, this approach was discarded when more functions were added resulting at one point at what is sometimes called creeping featurism or the temptation to add features simply because they can be added from a programming point of view. The palettes became overcrowded. The dimensions of the relatively small Macintosh SE screen imposed restrictions on the number of buttons that can be arranged on a palette without impairing usability and sacrificing too much space. At some point the designer must evaluate the trade-off between direct access to the commands in the form of buttons and space that is occupied by these graphical objects. Two or three level deep dialogue boxes proved to be counter-intuitive: the procedures for getting certain results were so buried that the user might be puzzled.

The *HyperCard* principle was adopted according to which every point on a monitor screen can become an action trigger; buttons (commands) can be everywhere, not only in predetermined areas like menu bars or palettes. This principle opens new possibilities for the screen layout and offers the advantage that the tool can be brought near to the working space without interfering with it and without jeopardizing clarity. A design evolved in which standard operations like "Send" were located near the writing space, whereas other commands were grouped in one pull-down menu (*see figure 6*). Several action triggers were hidden; they appear only when the user passes the cursor over the sensitive area (*see figure 4*).

The reading-writing space was divided into two panes, because operationally reading and writing belong together as complementary actions (*see figure 6*). A horizontal pane divider is used to separate the two different areas. An arriving message is opened with an appended answer space. In that respect, the metaphorical approach of imitating real world paper procedures was deliberately left behind.

When writing a new message, the header area functions as an action space from which various actions can be triggered when the cursor is moved over the respective items. These virtual buttons are shown as push button outlines which remain visible as long as the cursor touches the sensitive area (*see figure 4*).

Clicking on the "To" button brings a dialogue box to the screen with a scroll window that shows names of individual addressees and group names, differentiated by normal and bold style (*see figure 5*). Clicking on a name puts an icon in front of it according to the type of addressee. After clicking the OK button the dialogue box disappears and the names are copied into the message header.

A major problem of electronic mail programs consists in organizing the data base and permitting fast and practical access to the records. To address this problem a list window was designed with several filters accessed through radio buttons (*see figure 7*). New incoming messages are listed in the "New Mail" listed in reverse order with the oldest unopened messages located at the top. Once opened, messages are listed in the "Old Mail

List" with the most recent message on top. Messages can be retrieved also by date range, person or topic (*see figure 8*).

A particular feature of this mail program is the possibility to group messages according to a topic. The user can browse through a set of messages which are linked to a message chain. This possibility of going forward and backward is indicated by two horizontal arrows at the bottom of the window (*see figure 6*). In the "New Mail" an arrow in front of the topic of a message shows that it deals with an ongoing matter (*see figure 7*). Such messages form part of a message chain.

In order to simplify the maintenance of the hard disk a special button "Send and Trash" has been added (*see figure 6*). In that case a message is sent, but without a copy remaining in the data base. Another feature of mail applications is the possibility to create groups of addressees, e.g. members of a work group. For this function, a standard procedure of the MAC environment was used, i.e., copying fonts or desk accessories (*see figure 9*). It is desirable that the major functions of a mail application be accessed through the keyboard; particularly when using long lists, a shortcut in the form of a selection bar jumping to the name after the first letters have been typed in adds convenience for the user. In the first version, only some of the shortcuts were implemented.

All components correspond to the visual framework of the MAC environment and in part are predetermined by certain routine items, e.g., type of borderlines and shape of buttons. The work of what might be labeled traditional graphic design referred then to:

- **layout of the screen**
- **location and sizes of typographical and pictorial components (distances from border lines, distances between components)**
- **design of icons** (*see figure 3*)
- **design of the "language" boxes with alerts and process messages** (*see figure 15*).

These items can be accessed and defined through a particular device in the form of a subapplication (Resource Editor).

## TEXT SUPPORT—the merging of language and graphics

Generally the user manuals are treated as a separate item—a necessary evil. According to a maximalist approach, programs should be so simple that user manuals can be dismissed. However, more complex programs require support in the form of printed user manuals. With an integrative design approach user manuals should be treated as part of the interface design, because learning the use of a tool is part of the design of the tool. This has a direct implication for the composition of application design teams in which the manual writer should be included from the beginning.

When designing a user manual the first step is to establish the different voices or text types. Thus, an action text tells the user the steps he has to perform, and should be typographically distinguished from a narrative text that talks about the program. Notes or hints, too, can be treated distinctly in order to facilitate understanding and orientation in the manual (*see figure 16*). The work on user manuals signals a merger between text writing and graphic design, and one might speculate that, in the future a new field of expertise will emerge: a specialist competent both in the domain of graphic design and writing. Personal computers make it possible for writers to become sufficiently competent to bring forth typographically the text distinctions with which they write their texts.

For the future one can anticipate that the development of applications will shift more and more from a programming-driven approach to a design-driven approach. In that respect, application design will experience a shift similar to that which occurred earlier in other branches of manufacturing like the automobile industry.

Research work needs to be done in the field of human user interfaces, with full awareness of the possibilities and simultaneous attention to the limits of scientific approaches to design. Interfaces can become objects of scientific research, but it should be kept in mind that science and design are constitutively different domains. There is no necessary link between a science of human user interfaces and the design of human user interfaces, nor is there a push or pull relation between

these two areas. Science is the domain of discourse based on the production of evidence (facts), whereas design is the domain of production of new realities subject to assessments. A future science of interfaces—as it has been claimed by representatives of scientific disciplines, particularly engineering—might eventually lead to a better understanding of interfaces, but only if it reflects on its often hidden assumptions. It is misleading to want to capture the nonscientific aspects of interfaces and particularly the design aspects of interface as "art." That notion evokes an aura of mystery and inexplicability that might be avoided by simply talking about interface design. Design is an autonomous domain not explained by categories of art. For reasons of clarity these two domains should be kept separate.

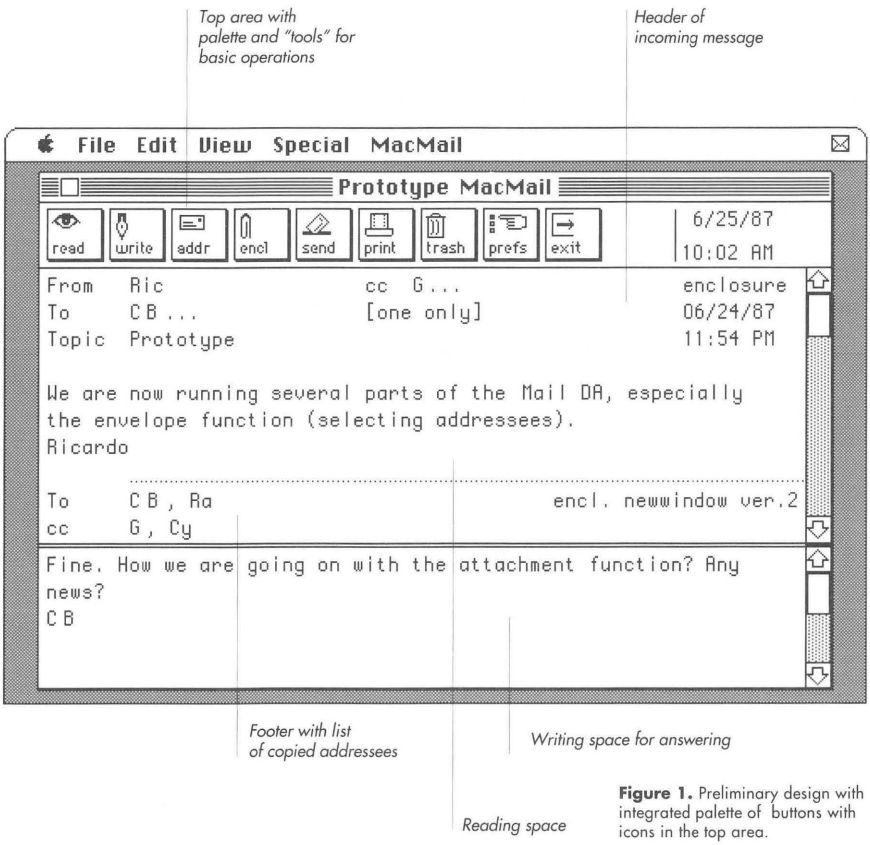



**Figure 1.** Preliminary design with integrated palette of buttons with icons in the top area.

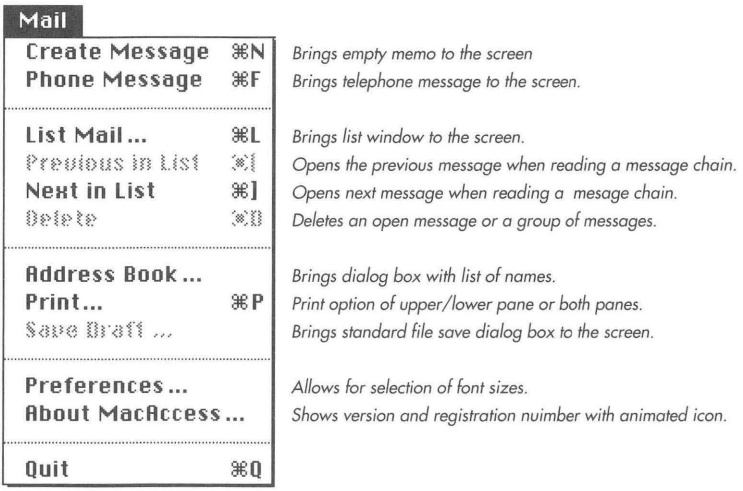



**Figure 2.** Pull down menu of the final design implemented as a desk accessory. Commands are ordered in groups according to functional affinities.

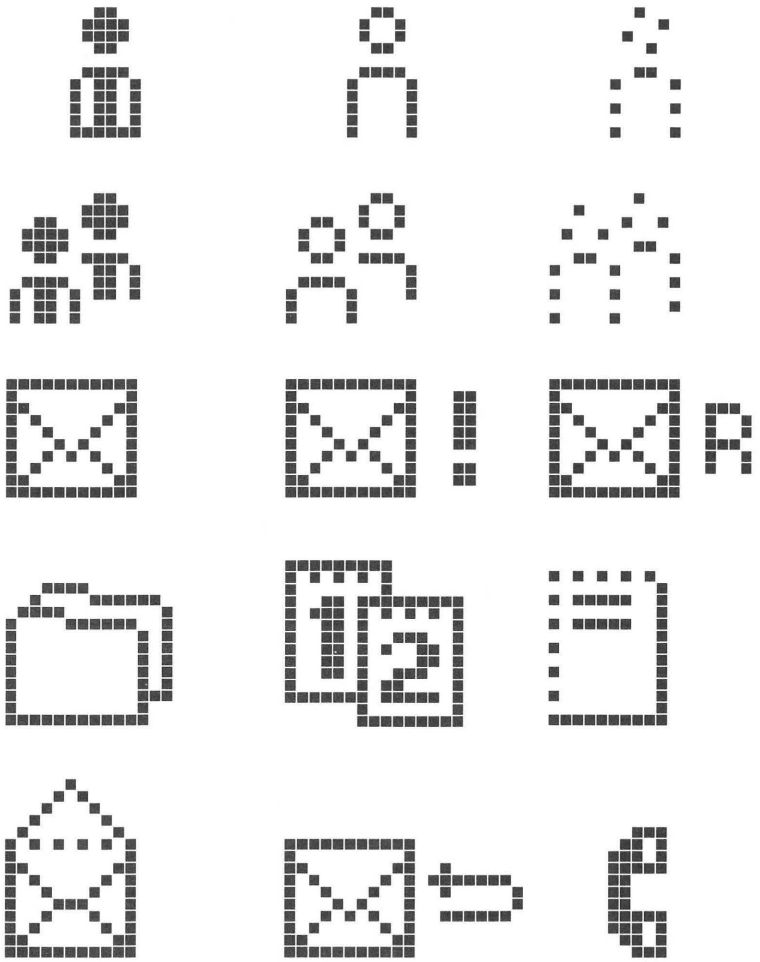


**Figure 3.**
Icon system shown in fatbits and actual size.

| Person | Copied (cc courtesy copy) | Blind copied (Bcc) |
|---|---|---|
| Group | Group copied | Group blind copy |
| New mail (unopened) | Urgent message | Registered |
| Filter by topic (folder) | Filter by date | Draft |
| Old mail (opened) | Return receipt | Phone message |

**( actual size )**

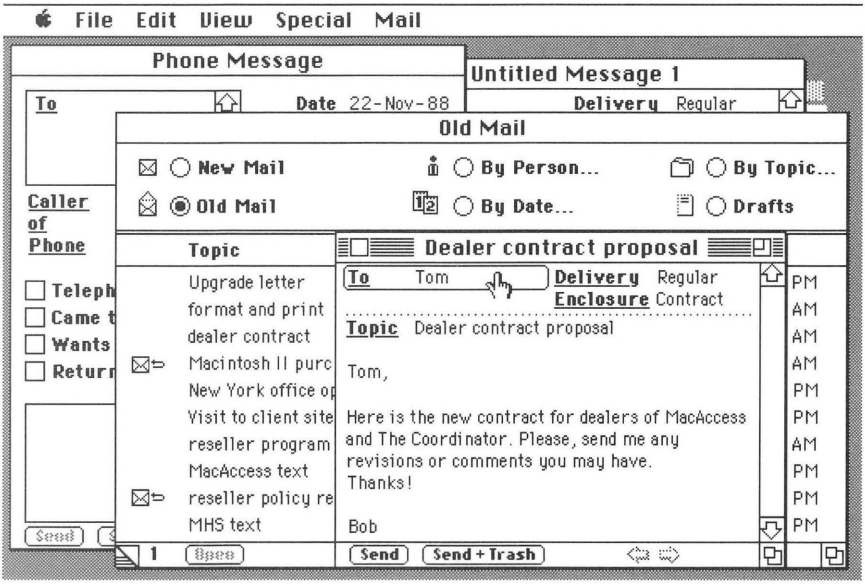



**Figure 4.** Various overlapping windows of the mail program. In the background a special phone message form like a pink slip.The writing pad can be resized. The pointer cursor changes to a hand-shaped cursor when moving over an invisible button, e.g., in the header area (To).

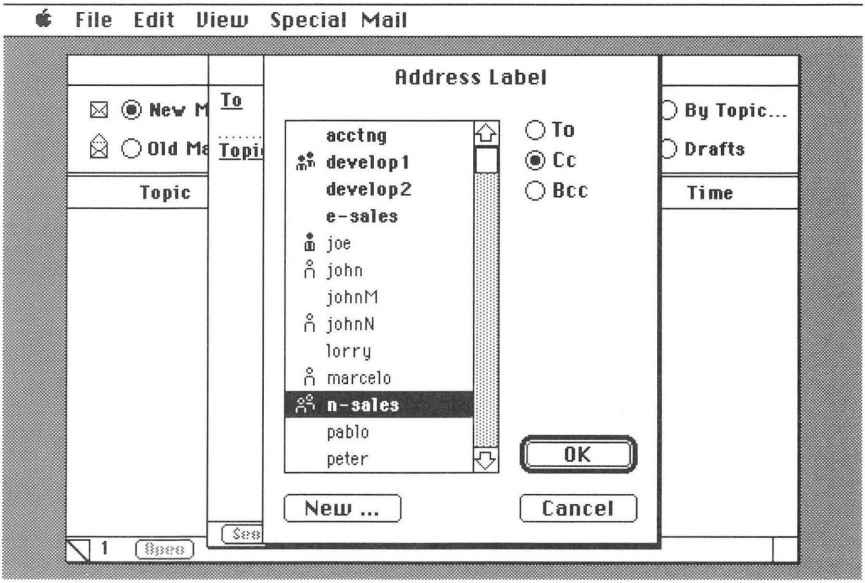
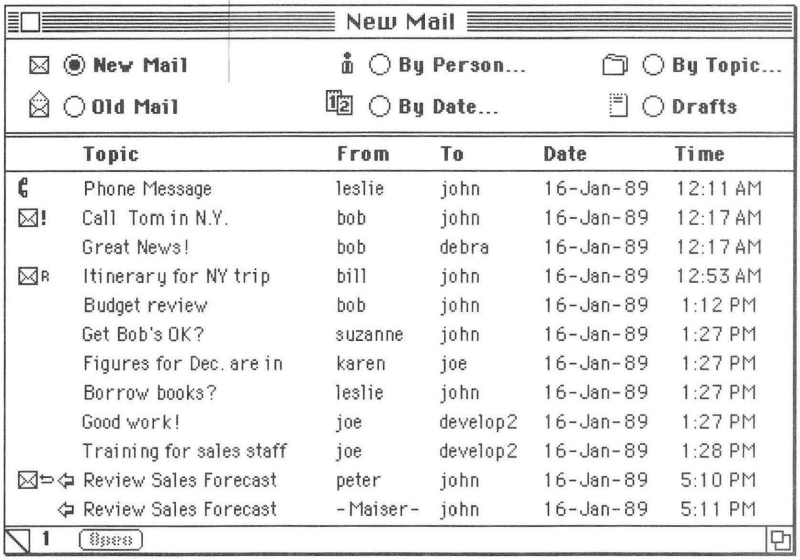



**Figure 5.** Address list with icons indicating primary addressee, secondary addressees (copies), and blind copies. The new button brings a dialog box to the screen that allows the entry of a new name.


**Figure 6.** Writing pad with appended answer plane. The name of the addressee is automatically copied into the header of the answer.

*Top area with radio buttons that act as filters to fill the list with the corresponding messages.*




*Flip-over corner for opening "pages" of the mail list*

**Figure 7.** Mail list window with icons for message type (urgent, registered, ongoing matter, phone message)

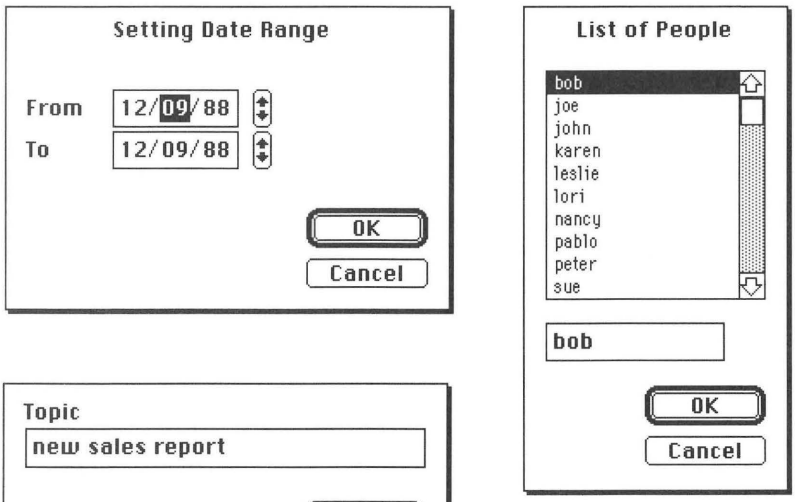



**Figure 8.**
Three different dialogue boxes that define the settings for messages that are brought to the list window:
- messages between dates
- messages related to a topic
- messages from a person..

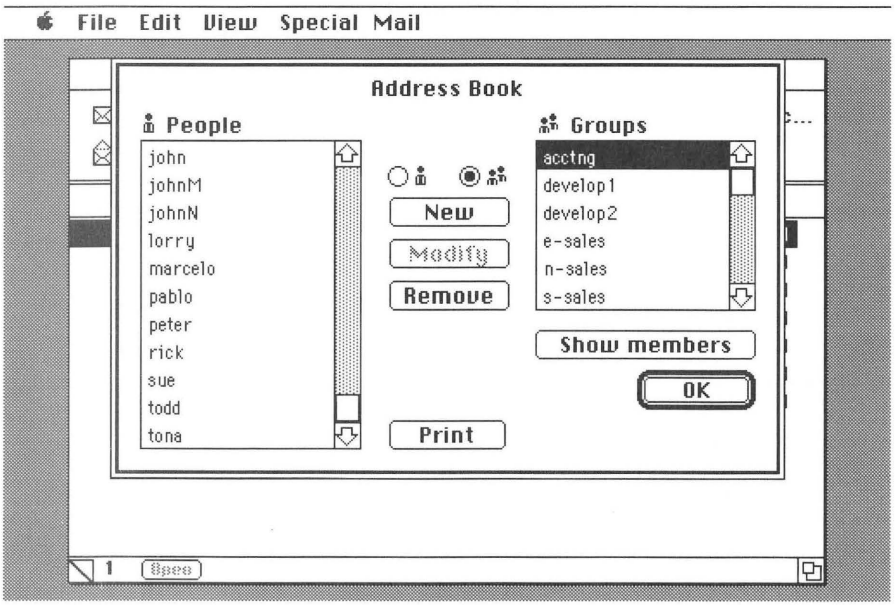



**Figure 9.** Address book that mimics the Font/DA mover. These three buttons under the single person and group icon bring corresponding dialog boxes to the screen for entering new names into the address book, modifying addresses, and removing names. Clicking on the button "Show members:" lists the members of a selected group.


**Figure 10.** Various dialog boxes for changes in the address book.

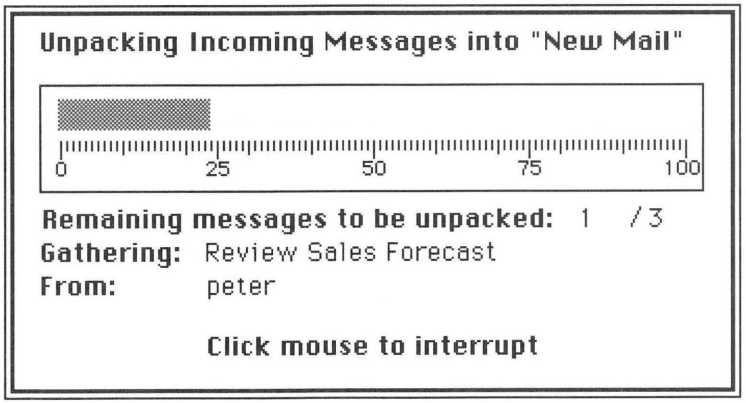



**Figure 11.**
Message box that is shown when a new mail batch is delivered and loaded into the user's private "electronic mail box" on his local hard disc.

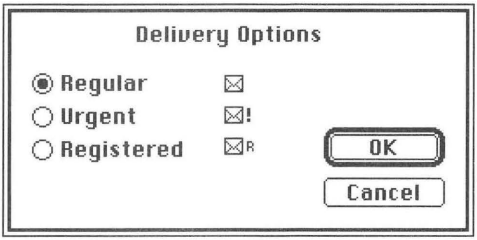



**Figure 12.** Dialog box for selecting types of delivery.

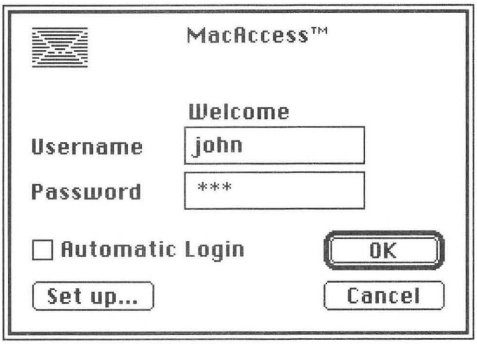



**Figure 13.** Welcome screen with text entry fields for password.

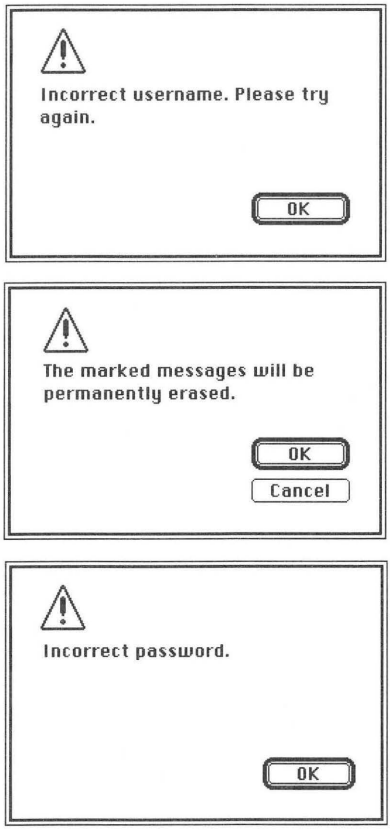



**Figure 15.** Samples of text dialog boxes.

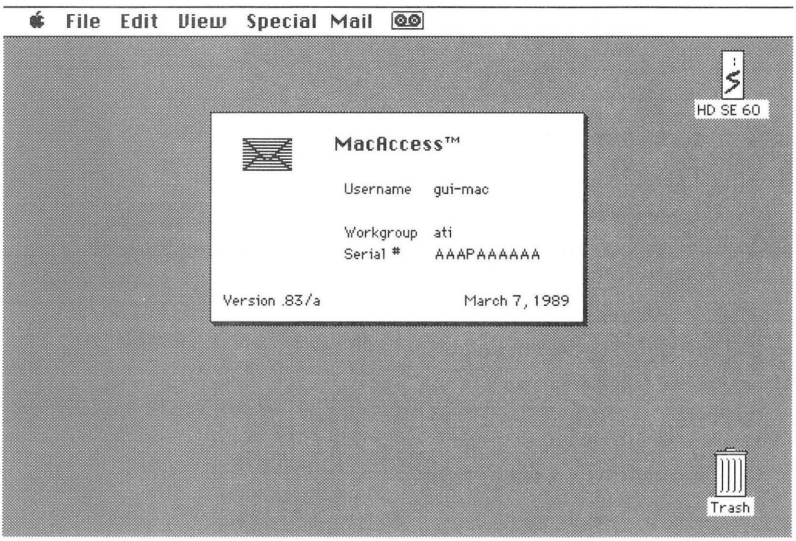


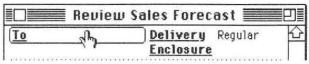

**Figure 14.** "About "screen with animated mail (envelope) icon..

**Figure 16.** Sample page of the user manual. Actions texts on gray shaded fields. Captions and secondary illustrations (for browsing) in a "fast track" column.



## Addressing a Message

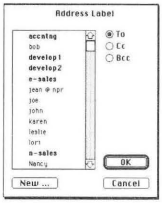
- To address the message you are working on, start by moving the pointer over the heading **To** at the upper left of your message.




***Change of cursor with hidden button***

- When the pointer becomes a hand, and you see a rounded rectangle appear around the word **To**, click.

You see the **Address Label** box, with a list of any nicknames now in your address book. (Group names are in **bold**.) You're going to create an address label to stick on the electronic envelope so that the mail carrier knows where to deliver the message.



## NOTES

**1.** *SAA - Common User Access - Panel Design and User Interaction.* 1987. Boca Raton: IBM, 7.

**2.** *Human Interface Guidelines: The Apple Desktop Interface.* 1987. New York: Addison Wesley, XII.

**3.** *The Open Look UI (user interface) Style Guide.* 1989. Mountain View: SUN Microsystems, 1.

**4.** H. Maturana and F. Varela. *The Tree of Knowledge.* 1987. Shambala, 75.

**5.** *Human Interface Guidelines: The Apple Desktop Interface. 1987.* New York: Addison Wesley.

**6.** *Open Look - Graphical User Interface Specification.* 1988. Mountain View: SUN Microsystems.

**7.** *SAA - Common User Access - Panel Design and User Interaction. 1987.* Boca Raton: IBM.

**8.** *OSF/MOTIF Style Guide.* 1989. Cambridge: Open Software Foundation.

**9.** This program has been developed at Action Technologies, Inc. during 1988 and part of 1989. The first sketches were made in mid-1987. The principal computer scientist in the development team was Pablo Flores, under the general direction of Juan Ludlow. Various specialists for Macintosh applications contributed to the program during its different stages. The author is particularly grateful to Ricardo Salas for orientation in the Macintosh environment. The theoretical framework presented here owes a debt to the groundlaying work for a new theory of design by Terry Winograd and Fernando Flores in *Understanding Computers and Cognition* (Norwood: Ablex Publishing Company, 1986). The notion of retinal space was introduced by Fernando Flores in the Ontological Design Course of Logonet, Inc. (1988) and further detailed in a workshop "Graphic Design For Effective Action" organized by Logonet, Inc. (1988). At the same workshop the notion of "action trigger" has been presented.

## BIOGRAPHY

Gui Bonsiepe, who studied and taught at the Ulm School for Design in Germany, is a senior researcher at the National Council for Scientific and Technological Development in Brazil. In the mid 80s, he founded the Brazilian Design Institute in Florianopolis. Among his publications are: *Theory and Practice of Industrial Design* (Italy, 1975), *Industrial Design, Technology and Dependency* (Mexico, 1978), *The Technology of Technology* (Brazil, 1983) as well as numerous articles on design.