Designing the Gateway Interface: Tips and Techniques from Carnegie Mellon's Experience

THE GATEWAY INTERFACE

What, exactly, is an “interface” and why does the electronic library need more than one of them? The etymology of the term can be traced back to the Greek *prosopon*, meaning a face that is facing another face in a living, mutual relationship. In ancient Greece, an interface was a third state of being, an ontological reality achieved through communication. Similarly, some modern scholars, for example, Marshall McLuhan (1962), Walter Ong (1982), and Michael Heim (1987), define “interface” as a technological environment that slowly transforms perception and cognition and eventually induces a new state of being, a new consciousness of self and world.

Computer users learn to experience and participate in the world in a digital way through encounters at the interface. The software and hardware they use shape their digital experience and their expectations about the interface. For example, Macintosh users expect to use a mouse to select menu options, click buttons, and navigate with a scroll bar; they don’t expect to type commands or examine files without starting an application. UNIX workstation users may or may not expect to use a mouse; they do expect to type commands and examine files without starting an application. Computers are not neutral tools. They are driven by ideology. Users interiorize the ideology of their desktop computer. The successful electronic library will give users the “look and feel” and power of their desktop computer so that they can learn the application quickly, focus on the information, and share the information across applications and services (e.g., cutting, pasting, filing, and printing).

What, then, is a “gateway” interface? In the domain of information retrieval, a gateway interface essentially provides access to one or more databases in addition to the online public access catalog (OPAC). The definition may be finessed from a narrow or broad perspective. From the narrow perspective, a gateway provides access to multiple databases that are managed by one group or organization. Though the databases may be created from local or commercially licensed data and reside on the same or different retrieval servers using the same or different retrieval software, there is only one information store, that
is, one information store "owner," designer, controller, negotiator. From the broad perspective, a gateway provides access to multiple databases that are managed by multiple groups or organizations. Some databases may be locally loaded and managed; others are available over the network from other sites and managers. In this model, there are multiple information stores. The design implications and ontological ramifications of a gateway interface depend on which definition of gateway is invoked. For example, if all of the databases are locally loaded and maintained using the same database-building and retrieval software, then search syntax and retrieval protocols are easily specified and controlled. However, if databases are loaded and maintained at different sites using different software, then search syntax and retrieval protocols require rigorous standards and experimentation to achieve interoperability. In both scenarios, authentication and protection may be necessary to meet database licensing agreements. Search syntax, retrieval protocols, and authentication and protection affect user interface design and functionality.

This paper examines five lessons in interface design learned by Carnegie Mellon University Libraries in building Library Information System II (LIS):

1. Be prepared: User interface design is difficult and time-consuming.
2. Be informed: Distributed retrieval has implications for user interface design.
4. Be flexible: User interfaces need to be tested and revised.
5. Beware: Politics and egos can disrupt user interface design.

THE LIBRARY INFORMATION SYSTEM (LIS)

Some background information will provide a context for the lessons to be discussed. LIS is a distributed retrieval system of clients and servers that implements the narrow definition of a gateway interface. It currently provides access to 14 databases in one information store managed by the University Libraries. Developments are underway for LIS to provide access to multiple information stores managed by other groups on campus and by groups at remote sites.

In January 1992, LIS replaced the mainframe retrieval system operated by the Carnegie Mellon University Libraries since 1986. LIS has two client user interfaces: a Motif interface for UNIX workstations running X Windows, and a command-line ASCII interface for other machines. The command-line interface is called the VT100 interface, though it emulates many terminal types. Long-term plans include a Macintosh interface and a MS Windows interface. The LIS retrieval servers are four DECstation 5000s. Databases are built on a separate machine, then moved to the retrieval servers. Databases were initially built on a VAX 6420, but the VAX was replaced in 1995 with a DEC Alpha Flamingo to greatly increase the speed of database building. The database-building and retrieval software used in LIS is Newton, which was developed by the Online Computer Library Center (OCLC). Newton is optimized for Boolean retrieval and large databases. Experiments have begun with a second retrieval engine, Ful/Text from Fulcrum, to provide smaller databases and easy-to-use database-building tools for individuals and groups outside of the University Libraries. Plans are for Ful/Text to facilitate the provision of multiple
information stores at Carnegie Mellon and a truly campus-wide information system accessed through the LIS clients.

The protocol that enables LIS clients to "talk" to LIS servers is Z39.50 layered on TCP/IP (Transmission Control Protocol/Internet Protocol). Z39.50 specifies a Boolean query language and general search and present (display) services. The released version of LIS uses version one of Z39.50, which is predominantly site defined and therefore not robust enough to support interoperability across sites. Version two of the standard is being implemented now, with an eye towards changes coming with version three—changes that will render the standard robust enough to begin experiments in interoperability. Authentication in LIS, required by database licensing agreements, is based on Kerberos, developed at the Massachusetts Institute of Technology (MIT). Access control is provided by Transarc's PTS (Protection Server). The names, addresses, and structures of the databases are provided using the Andrew File System (AFS). However, by winter 1993, a reference server will provide database information and access control. The reference server will "talk" to both clients and (retrieval) servers and pass information dynamically upon request. It is being developed at Carnegie Mellon in preparation for retrieval from multiple information stores.

LESSON 1
Be Prepared: Interface Design is Difficult and Time-Consuming

Interface design is not for the squeamish. Read and apply the literature, but don't expect to get it right the first time. Don't expect to get it perfect—ever.

A survey of user interface programming conducted by Brad A. Myers and Mary Beth Rosson (1992) revealed that the average time spent on an interface (independent of the project application, country, or host computer system) was 45% during the design phase of the project, 50% during the implementation phase, and 37% during the maintenance phase. Use of a programming toolkit increased the percentage to around 60%. Many of the most difficult problems reported in the survey related to the design of the user interface rather than to its implementation. For example, finding appropriate test subjects, assessing user needs and expectations, accommodating both novice and expert users, understanding and conforming to style guidelines, achieving consistency (particularly across developers), selecting colors and fonts, and providing online help were raised as serious design problems (p. 201). Serious implementation issues included achieving acceptable performance and portability, finding and fixing bugs, getting enough memory, and communicating among different components of the interface and between the interface and the underlying application.

Carnegie Mellon University Libraries' experience matches the results of this survey with few exceptions. Designing, implementing, and maintaining the LIS Motif and VT100 user interfaces took more time and were more difficult than anticipated when work began in 1989. Porting the user interfaces to multiple platforms (e.g., DECstations and Sun Sparcstations) was problematic, but tracking bugs and prioritizing bug fixes across interfaces, platforms, and versions of the software are ongoing management headaches. The design
problems revolve not so much around testing but around the negotiations required to interpret research results, generate an improved design, and allocate resources to implement the design. Details about the University Libraries' design problems and process are provided throughout this paper. The remainder of this lesson focuses on general design problems and solutions that can be learned from the literature and how they match University Libraries' experience.

Computer Technology and Literate Habits

Admittedly this is not the place to review the entire body of literature on interface design, but mention must be made of key works and areas for concern. Edward R. Tufte's remarkable books *Envisioning Information* (1990) and *The Visual Display of Quantitative Information* (1983) shed considerable light on what visual excellence is, why it is important, and universal principles for achieving it. Tufte's pamphlet on user interface design (1989) and an article published in the *Bulletin of the American Society for Information Science* (1992) pinpoint four serious design problems related to the current state of computer technology and the literate habits of human beings. The following discussion is organized around Tufte's four points with related research introduced to confirm and elaborate them.

1. Resolution

According to Tufte (1992), the primary problem in user interface design is the burden placed on visual memory by the low resolution of the computer screen. In comparison with a printed book or map, a computer screen conveys very little information. The information density is so poor that user interfaces must break information—and therefore information processing—into small pieces that must be viewed or done in sequence. The cognitive processing required to keep the sequence coherent impairs the user's ability to contrast, compare, or make a choice. The constant context-switching required by menus, dialog boxes, error and status messages, etc., impairs the user's ability to concentrate.

Tufte (1989) describes two ways to lighten the burden placed on visual memory by poor screen resolution: first, reduce the noise; second, improve the signal. To reduce the noise, provide clean, sharp, precise interfaces free of unnecessary elements. Bruce Tognazzini (1992) explains that anomalies or unnecessary elements in the interface are sources of noise in the communications channel between the computer and the human. He warns: "Be wary of interface elements that detract from or overwhelm the content regions of your application" (p. 196). This is Tufte's (1989) second solution to the memory problem: improve the signal by making the organizing grid implicit or transparent. Devote more space to the data than to the data container so that users can focus on the information rather than on the design and mechanics of the interface.

2. Typography and Icons

Tufte's (1992) second problem with today's user interfaces is the design of the typography and icons. He calls for standards of quality book typography to insure the readability of electronic documents, and for typographic and artistic skills to be applied in icon design, along with a careful consideration of
vocabulary, cultural context, and computer functionality. According to Tognazzini (1992), people want multiple channels of information using both words and pictures.

3. Interaction

Design elements that work well alone do not always work well together: "a complexity of marks generates an exponential complexity of shapes" (Tufte, 1992, p. 16). The result can be dancing gray spots or vibrating black lines that distract or even irritate the user. Furthermore, "Visual clutter [can result] from prison grids of window frames, empty paths, and rectangles and blocks" (Tufte, 1992, p. 16). Tufte's solution: reduce the noise and the contrast between figure and ground.

The arrangement of information and features on the screen determines what (if anything) interacts and what users will find (easy) to do. Clutter will distract the user and interfere with concentration. According to Tognazzini (1992), "Any element that does not communicate information that the user may need right now is superfluous" (p. 136). Bring to the foreground frequently used features on menus and buttons while nesting less frequently used features in dialog boxes. Put popular options at the top of lists. Display cursors prominently. Open dialog boxes over the button or menu used to request them, where the mouse is likely to be. "Close targets are faster to acquire than far ones: Keeping everything but menu bars and other edge-hugging items close to the area of interest saves the user time" (Tognazzini, 1992, p. 206).

4. Color

One way to reduce the contrast between figure and ground is with color. Color is one of the most powerful agents available for depicting complex information. Be careful, though: like all design features, colors can interact and produce jarring effects. Tufte (1992) recommends following cartographic principles for the use of color. For example, use colors found in nature like light grays, blues, and yellows. Background colors and colors in inactive windows should be muted or grayed. Foreground colors and active windows should be lighter and brighter. Strong colors can be used for emphasis.

In Preparation for Electronic Document Delivery Services

In addition to the general research on interface design and human factors, those involved in electronic library development projects should become familiar with the research on reading and writing online. Years ago OPACs spearheaded an ongoing movement into bibliographic databases, but the new trend is to talk about delivering full-text documents to the user's desktop. Readability is more of an issue when the electronic library goes beyond ASCII bibliographic records and abstracts and dares to confront users with an ASCII version of Alice in Wonderland or a bitmapped "page image" version of an academic journal. Library and computer ideologies are mixing and mingling at the interface, creating a new digital experience that is rattling print-literate habits. The technology is upping the ante on the ontology.

Research by Wilfred Hansen and Christina Haas confirms Tufte's analysis of the memory problems related to the low information density of the computer
screen. In studies of reading and writing online, Hansen and Haas (1988) discovered that the amount of information that can be seen at a glance (the "page size") determines the context for viewing and understanding the information. If the amount is small, readers must move ("page") frequently to view an entire document. Moving takes time, burdens short-term memory, and interferes with concentration, comprehension, and recall. Hansen and Haas conclude that four design features can facilitate reading and writing online:

1. Sufficient "page" size—comparable to a printed page (Tufte's despair over breaking information into a sequence of bits).
2. Legibility—font, spacing, contrast, sharpness, flicker, antialiasing, and resolution (Tufte's attention to typography).
3. Responsiveness—"page" quickly, easily, flexibly (Tufte's focus on the data rather than the mechanics of the data container).
4. Tangibility—provide spatial cues to facilitate context and recall; provide cues that users can manipulate, for example, bookmarks and annotations (Tufte's concern for the burden on visual memory).

Hansen and Haas (1988) argue that these features boost the user's confidence and facilitate learning the interface and the information.

The implication seems to be that a workstation interface is preferable to a terminal interface for full-text documents. However, research done at Bellcore on the SuperBook and MiteyBook full-text document browsers indicates that providing interactive contextual views like dynamic tables of contents and tailored text displays can compensate for the loss of context derived from the limited screen space and poor resolution of today's computer monitors (e.g., Egan, Remde, & Landauer et al., 1989; Egan, Remde, & Gomez et al., 1989).

A Note about Consistency

Efforts to support the user's visual memory and provide a sense of context are efforts to put the user in control. The user's sense of control derives from consistency in the interface. Tognazzini (1992) explains that control "arises from neither tyranny nor anarchy but from the freedom of a supportive environment constructed of reasonable and consistent rules" (p. 225). Having said that, one must know when to break the rules. "The most important consistency of all is consistency with the user's expectations" (p. 250). "Consistency with the Guidelines should be maintained unless a new solution is demonstrably and vastly superior" (p. 41). In other words, follow style guidelines as much as possible, but when the guidelines conflict with user needs and expectations, break the rules.

Experience with LIS

Carnegie Mellon University Libraries' experience with LIS matches the literature. The poor resolution of today's computer monitors, the limited screen space of different machines, and the distance between the computer screen and the computer user (almost twice the distance between book and reader) necessitate breaking information into pieces. Readability problems are not particularly
striking with bibliographic databases, where records are relatively short, but they increase with full-text databases that deliver lengthy documents to the desktop.

Given the heterogeneous computing environment at Carnegie Mellon and limited resources, all primary information displays in LIS are in ASCII format (e.g., the list of result sets, list of titles retrieved, and bibliographic records). LIS also has several full-text ASCII databases. Figures 1 and 2 show the Motif and VT100 LIS Records windows displaying the same (full-text) article from the Academic American Encyclopedia. VT100 LIS is limited to displaying 24 lines by 80 characters of ASCII text. In the current interface, 9 of those 24 lines are used for on-screen instructions, prompts, and system messages. Consequently users see very little of an encyclopedia article at one time. If users have a larger monitor, they can resize VT100 LIS windows to take advantage of the screen space and see more of the article at a glance. To reduce the noise, improve the signal, and enable users who cannot resize the window to see more information at a glance, a future release of VT100 LIS will enable users to hide and display the six lines of instructions at the bottom of the screen with a keystroke.

Motif LIS, in contrast, runs on UNIX workstations, which have more screen space and capabilities than terminals or terminal emulators. Therefore, Motif LIS can display not only more information at once than VT100 LIS, but multiple views and formats of information at once. For example, VT100 LIS must display either the list of titles retrieved in a search or a bibliographic or full-text record; it cannot display both at once. Motif LIS, however, combines these displays in one window and enables users to reallocate space between the list and the record (see Figure 1). The entire Records window can be allocated to the record, in which case the amount of information displayed is comparable to a printed page. The combination Motif Records window was the result of prototype testing in September 1990.

Though ASCII text can be displayed on any machine, it lacks the visual-spatial cues that facilitate reading longer documents. Users may have few problems reading a bibliographic record in ASCII format, but many problems reading a technical report or a novel in ASCII format because they easily lose track of where they are and where they've been. The trend for document delivery services to provide documents in bitmapped "page image" format (where the image sent to the user's screen or printer is identical to the original print publication) escalates problems with screen resolution and typography. Some typefaces scan better than others. Some scanners do a better job than others. Regardless of the dots per inch (dpi) at which a page is scanned, the dots per inch of the display monitor affects readability.

In cooperative experiments with selected publishers, Carnegie Mellon University Libraries scanned journal pages at 400 dpi for display on a 72-dpi workstation monitor. Usability tests in spring 1992 revealed that the quality of the images and the prototype image user interface were inadequate for general release to campus. Research subjects wanted to see a whole page on the screen and have the text be legible at a distance of 16 to 24 inches. Though the size of a UNIX workstation monitor makes it possible to display an entire page on the screen, it is not possible to insure readability at the requisite distance.
Renaissance humanists gave the name Middle Ages to the period between the end of the Roman Empire and their own time, which they believed was a rebirth of the civilization of Greece and Rome. In the 15th century the Italian historian Flavio Biondo regarded the sack of Rome by the Visigoths in AD 410 as the end of ancient civilization and the beginning of the modern period as 1410. A popular German textbook of the 17th century dated the end of ancient times the deposition of Romulus Augustulus as Western emperor in AD 476 and the beginning of the modern age in 1453 with the fall of Constantinople to the Turks.

**BASICPERIODS**

Within the thousand years of the Middle Ages, historians have recognized subperiods: the Early Middle Ages, to 900 or 1000; the High Middle Ages, from then to about 1300; and the Later Middle Ages, the 14th and 15th centuries. The Middle Ages, especially the early period, were also known as the DARK AGES, but historians today generally reject that term.

**The Early Middle Ages**

The Early Middle Ages was the period of the decline and fall of the Roman Empire, the invasions of the barbarians, and the triumph of Christianity. The Western Empire was broken up into barbarian kingdoms, until on Christmas Day, 800, the Frankish king, CHARLEMAGNE, was crowned emperor of the West by the pope. His empire had a fundamental weakness, however, in that it depended on the personal rule of the emperor, who could not successfully delegate authority or levy direct taxes. By 900
The fonts and typefaces of printed journals were designed for print technology, which has a higher resolution and can assume a shorter distance between document and reader. Though the prototype image user interface provided several levels of zoom for users to enlarge and shrink the page, they were not comfortable when the text was large enough to read at a distance of 16 to 24 inches—they couldn’t see enough of the page to keep track of where they were on the page. They preferred to keep the entire page visible (the text small) and move their faces closer to the screen. Users also lost track of where they were in the document because the prototype interface provided no visual-spatial contextual cues for the entire document. Detailed user interface design specifications were prepared following the prototype testing of the image user interface (see lesson 3).

While the new image user interface is being implemented, the University Libraries are finalizing agreements with vendors and publishers to acquire a substantial body of journal information in page image and ASCII formats. Plans are for LIS to provide image document delivery services in selected subject areas by spring of 1994. Work has begun on tailored text displays and dynamic tables of contents. The prototype image user interface included interactive tables of contents generated using optical character recognition (OCR) software and manual touchup. Development is underway on recognition software that will distinguish text from graphics on a page image and enable users to move from one graphic to the next or one section heading to the next in an image document.

LIS interfaces currently have no icons. VT100 LIS provides instructions in ASCII text at the bottom of each screen or display. Motif LIS provides features
on buttons and menus. Icons require a graphic artist, and the development team does not include a graphic artist, but more importantly, icons require considerable attention to cultural context. The international student population at Carnegie Mellon complicates the design of appropriate icons.

Space constraints make composing on-screen instructions or selecting the proper textual label for a menu or button difficult. Vocabulary problems also arise from the synonymy and polysemy of the language. A vocabulary must be found for features and displays that quickly conveys to users an accurate conceptual model of online information retrieval. Unfortunately, it is not easy to determine when information retrieval requires a technical vocabulary and when a technical vocabulary is needless jargon. For example, is “record” an unnecessary technical term? Is “browse” best used to denote the feature for examining database indexes to select search terms or for stepping through a hierarchical directory to find a document? Do users need to know the differences between “fields” and “indexes” or can information retrieval blur the distinction without inducing a cognitive model of information retrieval that interferes with retrieval and usability? What is “full text”? Is it searchable text? A picture of text? Text and graphics? Hypertext? Hypermedia? Experience at Carnegie Mellon indicates that information retrieval is a sophisticated business and that users need to learn certain fundamental concepts and strategies to be proficient with the technology. Interface design should bring these concepts and strategies to the fore, and vocabulary should be tested (see lesson 4).

Motif buttons present several additional problems. First, in the current implementation, they don’t resize. The University Libraries want to provide some Motif LIS workstations with large fonts suitable for the visually impaired. Motif menus resize appropriately when the font is enlarged, but Motif buttons do not. Enlarging the font makes button labels no longer fit on the buttons. Second, buttons occupy considerable screen space. User protocols indicate that they may clutter and complicate the interface and distract the user rather than simply highlight heavily used features. Using Tufte’s terms, too much space may be devoted to the data container rather than to the data. Future research and design work will determine whether Motif LIS should continue to use buttons or eliminate some or all of them. With the aid of a graphic artist, the buttons could be transformed into icons.

Though the additional screen space provided by UNIX workstations enables Motif LIS to display more information at a time and thus helps readers concentrate, comprehend, and recall the information, the additional space also introduces problems. More space is available to be cluttered with more views and formats of information. Figure 3 shows the three primary Motif LIS windows open at once. The Search window is on the upper left, the Browse window is on the lower left, and the Records window is on the right. When document delivery services are provided in image format, there will also be a Document window overlaid on the screen. The heavy contrast of black and white space all over the screen creates vibrating lines and dancing gray spots. Again, part of the problem is the quality of the monitor. Though the Motif toolkit supports the “look and feel” of three-dimensional space using subtle variations in color, the typical workstation monitor on campus is 72-dpi monochrome. The subtle variations in color and the three-dimensionality of Motif are lost in the dithering
of pixels. Indeed, depending on the monitor, lines surrounding default buttons in Motif LIS may or may not be visible. (See Figure 3.) Better monitors, preferably color monitors, will enable the University Libraries to provide better user interfaces, with active windows and information foregrounded by lighter, brighter colors and higher contrast, and background or related information subdued to be less distracting.

LESSON 2
Be Informed: Distributed Retrieval Has Implications for User Interface Design

The implications don’t always mesh nicely with user needs and expectations or library resources. Try to negotiate the best compromise between the long-term vision of interoperability with other retrieval systems and short-term needs, resources, and schedules.

One of the advantages of distributed computing is that it affords multiple user interfaces. Satisfying user needs and expectations in a multivendor world means providing multiple user interfaces. Needless to say, designing,
implementing, testing, supporting, and maintaining multiple user interfaces is more work than dealing with one user interface. The requisite labor and design problems increase exponentially.

In addition to the sheer number of interfaces to be designed, implemented, and maintained, and the complications that arise from practical considerations like resources and schedules, each component in the distributed architecture has implications for interface design and functionality. For example, each component generates error messages. The messages are typically written by different programmers using different technical vocabularies. The language is typically foreign to users, and the information provided is typically insufficient for them to know how to resolve the problem. In Figure 4, for example, the client, the retrieval server, the reference server, the retrieval protocol (Z39.50), and the parsers that translate user syntax into Z39.50 syntax into Newton syntax and back again may generate error messages. A good interface will trap all of the messages coming from the different architectural components and convert them into something more helpful for users. To this end, the University Libraries conducted a study of error messages in May 1991. The result was a user-friendly LIS vocabulary and a model for error messages: one sentence explaining the problem, followed by a blank line, followed by one sentence explaining how to solve the problem. The vocabulary has been implemented in both Motif and VT100 LIS, but to date only Motif LIS error messages follow the model. VT100 LIS currently provides only one line for error messages. Longer messages will require redrawing the screen and may introduce performance problems.

**Figure 4. The distributed architecture of LIS**

Clifford A. Lynch (1991) explains that the separation of user interfaces from the underlying retrieval application introduces certain problems. The success of distributed retrieval hinges on a retrieval protocol standard between clients and servers. The hope is that Z39.50 will become robust enough to sustain retrieval from multiple information stores. Currently, however, clients need to know certain information about servers that Z39.50 does not address.
For example, Z39.50 assumes that clients already know the names and addresses of the servers; the "explain" service in Z39.50 version 3 will provide information about databases on known servers. If clients do not know the names and addresses of servers, Z39.50 cannot help them. Similarly, servers may have information that users want, but Z39.50 version 2 has no way to communicate it to the client. For example, a server may know how records are sorted or ranked or why a search retrieved large or zero results, but Z39.50 version 2 has no way to communicate this information to the client.

Carnegie Mellon is developing a reference server to handle information not covered by Z39.50. In communication with a client, the reference server will dynamically pass database names, addresses, record formats (e.g., MARC, ASCII), search attributes (e.g., relational predicates and defined fields), display formats (i.e., what fields or data element set names), and error and diagnostic message formats. In communication with a retrieval server, the reference server will dynamically pass access control information (i.e., the list of databases that the user is allowed to see). The reference server is being tested now for release in LIS in December 1993.

Even when different components of the distributed architecture have similar ideologies, there may be problems because the ideologies conflict with user needs and expectations. For example, early versions of Z39.50 supported only the presentation of a single (overall) result set for a query, no intermediate results. This made it impossible for Z39.50 compliant applications to indicate which term(s) in a query retrieved zero records and thus caused the entire query to fail. Similarly, the Newton retrieval software creates a single result set for a query. It does not enable users to search across databases and retrieve and manipulate (e.g., sort) one result set; Newton creates a result set for each database, and each result set must be manipulated separately.

LESSON 3
Be Smart: User Interface Design Specifications
Save Time and Aggravation

Prepare detailed user interface design specifications based on research and peer review. Make sure that the specs take into consideration human factors as well as budget, scheduling, personnel, and technological factors. Implement the specs.

Who should be involved in interface design, and how does development proceed from design to implementation? Tufte (1992) argues that user interfaces should be designed by a single guiding intelligence because of the danger of distracting interactions among elements designed by different people: "User interface design decisions cannot be made one-at-a-time. Local optimization of design will never yield satisfactory global outcomes; perfecting many little separate pieces and then putting them all together will produce cluttered and fussy screens" (p. 16). Tognazzini (1992) seems to disagree: "The most successful designs result from a team approach where people with differing backgrounds and strengths are equally empowered to affect the final design" (p. 57). Experience in the University Libraries indicates that both men are right—that
the design process requires both the participation of people with different insights and priorities and the expertise and commitment of one person whose task is to incorporate these insights and priorities into an interface that meets user needs and expectations without introducing distracting interactions.

According to Tognazzini (1992), the design model should reflect the user's needs and expectations, not the limitations of the hardware, the toolkit, or the programmer. This is the ideal. However, in Carnegie Mellon's experience, interface design is inseparable from the budget, schedule, personnel, and technology available for the project.

In the beginning, LIS programmers worked without documented user interface design specifications. Detailed design discussions were held. Some decisions were made. Some decisions were avoided. Then programmers went back to their offices and generated code. Over the years, significant problems arose from misinterpretations of needed features and functionality, from unforeseen design issues that were not discussed by the group but handled in isolation by different programmers, and from inconsistencies and incompatibilities in code written by different programmers. The current procedure in the University Libraries is to involve many people in design discussions and research, but to have one person design and document a unified vision of the interface.

In the two major design projects undertaken in 1992, the Research Manager for Library Automation (the author of this paper) conducted the research, participated in the discussions, and documented the design in great detail. I worked closely with the programmers and project leaders to understand the nuances and implications of each design element and the context in which we worked. I submitted a comprehensive draft of the user interface specifications for peer review and discussion. In keeping with Tufte's (1989) directive—"Specifications should be given by example, by detailed illustration, not by words alone"—the drafts included pictures of every window, menu, dialog box, and information display. Provisions were made for providing error messages and online help at a later date. Comments were gathered and used to revise the specifications. The final specifications were submitted to the project leaders and distributed to the programmers for implementation. The initial design and specification process took approximately six months for the basic Macintosh LIS interface and the new Motif LIS interface for document delivery in page image format. (The prototype image user interface was produced without design specifications.) Now if programmers encounter a problem or situation not addressed in the design specs, they are required to discuss it with the group rather than implement their vision in isolation.

It is not enough to document design specifications. Programmers must implement the specs. This is easier said than done (see lesson 5).

LESSON 4
Be Flexible: User Interfaces Need to be Tested and Revised

Test to save time. Time is money. Test to increase quality. Quality is defined by the user, not the developer. Test every component of the interface. Test
multiple times using multiple methods. Test multiple groups of real users. Then revise accordingly—within reason and resources.

The goal of software development is to get the best quality for the lowest cost. The fact of the matter is that user perceptions can only be discovered by testing. "Guessing does no good" (Tognazzini, 1992, p. 37). Quality software is the product of testing and revision. Contrary to popular belief, testing can be cheap and easy to do. "It can even save you money—lots of it" (Tognazzini, 1992, p. 79). Jakob Nielsen's (1989) research on cost-effective usability testing determined that the cost could be reduced by 50% if mirrors, video cameras, and research assistants were eliminated. Testing often reveals serious problems where no problems were expected, and no problems where fatal flaws were expected.

Development groups should understand the trade-offs they are making when they decide on a research method. Research by Karat, Campbell, and Fiegel (1992) indicates that empirical testing is more cost-effective than cognitive walkthroughs regardless of the application being tested. Empirical tests require the same or less time to identify problems as individual and team walkthroughs, though they require more time to prepare materials, administer sessions, and analyze the data. Empirical tests identify a larger number of problems than walkthroughs and a significant number of severe problems overlooked by walkthroughs. Team walkthroughs are more productive than individual walkthroughs. Karat, Campbell, and Fiegel (1992) recommend the use of walkthroughs early in a project or when resources are severely limited, but encourage the use of empirical tests at key points in the development cycle "where coverage of the interface and identification of all significant problems is essential" (p. 403).

Tognazzini (1992) recommends beginning a project with field analysis of the audience and scenarios of use. Who are the users? How do they differ from one another? What kind of experience do they have? What are their goals? What tasks do they need to perform? What sources and features do they need? What information will they (want to) retrieve or generate? After a design is implemented, use "think aloud" protocols and intuitive observation to discover where users get frustrated because their needs and expectations are not met.

Carnegie Mellon University Libraries conduct feasibility and usability tests before and after designing and implementing an interface. Focus groups or surveys are carefully conducted to determine what people want and expect in information retrieval before design meetings are held or specifications are documented. Development team members contribute to the design of the interfaces and the test instruments by sharing their insights, priorities, and concerns at formal meetings and informal gatherings (e.g., in the staff lounge over coffee or pizza). After a design is implemented, every component of the user interface is tested—both the data container (e.g., the buttons, menus, and prompts) and the data or information presented within that container (e.g., the list of result sets, the list of titles retrieved, the bibliographic records, and full-text documents). Multiple research methods are used, including paper designs, prototypes, surveys, protocol analysis, and intuitive observation. Approximately four to six formal studies are done each year, along with several
informal studies. A formal study typically takes a couple of months to design the test instruments, run the subjects, analyze the data, and prepare the report.

Each LIS interface is tested multiple times before it is released to campus. Typically two sets of protocols are run. The first set identifies serious problems and leads to a new design. A month or so later, when the new design is implemented, a second set of protocols identifies lesser problems and leads to minor changes (e.g., vocabulary). Then the software is released to campus. Usage monitoring of the released software will eventually lead to changes in the interface. Efforts are currently underway to build usage models from the transaction logs and design a process for using the models to guide development of the interfaces.

Tests are always conducted with multiple groups of real users, for example, librarians, faculty, graduate students, and undergraduate students. A minimum of six subjects are tested from each group. Depending on the study, subjects may self-select or be randomly chosen. Development team members sometimes participate in the testing process as research subjects.

After testing, then what? According to Karat, Campbell, and Fiegel (1992), "The identification of usability problems is not an end in itself. Rather, it is a means towards eliminating problems and improving the interface" (p. 403). The goal is to revise and improve the interface based on the research results. However, caution is in order: changes to the interface should be disciplined and relatively infrequent. Users need consistency and stability. They do not need interfaces that appear to change randomly. According to Tognazzini (1992), "If it ain't broke real bad, don't fix it" (p. 158).

Carnegie Mellon University Libraries' revision process—like the design process—evolved over time. In the beginning, when development proceeded without design specifications, research was conducted and changes were made in a closely coupled yet undisciplined cycle. Programmers changed the user interfaces in reaction to empirical research results that matched their experience or in reaction to indignant user comments posted on electronic bulletin boards or overheard on the stairs. The result was clutter, inconsistency, confused users, and bewildered management—not to mention a frustrated researcher. Concerted efforts were made in 1992 to integrate the revision process with the research and design process. Changes are more holistic, now, in that they are approached conceptually and contextually—in terms of the budget, schedule, personnel, and technology—rather than helter-skelter.

A relatively new approach is to devote considerable time and planning to the presentation of research results. Experience indicated that the significance of the research results was often overlooked because the results were presented quickly at the end of a meeting that had already run overtime, and to an audience that was not necessarily interested in the results. The new method carefully selects the time, place, and audience. For example, initial usage models were presented to the Library Professional Council rather than to the development team. The presentation lasted almost an hour. Empirical data and anecdotes will be included in future presentations along with carefully designed text and graphics. The goal is to engage everyone in the audience, to move them to
understand and appreciate the problems that users encounter, and to invite their participation in interpreting the data and solving the problems.

LESSON 5
Beware: Politics and Egos Can Disrupt User Interface Design

Researchers can be zealots. Programmers can be mules. Managers can be bigots. Nobody's perfect. Love one another and what you do or get out of the business.

A development team is a group of people with different strengths and weaknesses who are more or less dedicated to the same goals. Working closely together to solve difficult problems in an environment constricted by limited resources inevitably creates tensions and trade-offs. The sense of camaraderie among the group affects the quality of life in the workplace and the quality of the work. Everyone on the team has the potential to facilitate or disrupt the development process.

Experience at Carnegie Mellon indicates that development team members are likely to exhibit certain shortcomings that can be overcome by good communication. For example, researchers and user advocates can be zealots lacking in diplomacy. They need to be made aware of the "big picture" of the development project, given a "reality check" of constraints in addition to human factors that affect the design and implementation of the software. Researchers seem to be more easily swayed by empirical data than anecdotal evidence, so concrete information about budgets, personnel, schedules, etc., will help them make informed (realistic) design decisions and specifications. Programmers tend to read programming manuals, not style guides. They can be stubborn and reticent. They need to be drawn into design discussions to explain how the technology (e.g., the programming toolkit) shapes the design, and to learn how human factors (e.g., the resolution of the human eye) affect usability and customer satisfaction. Programmers seem to be more easily swayed by anecdotal evidence and direct confrontations with users than by empirical data. Managers can be temporarily motivated by budget concerns or inflexible deadlines or power struggles among the group. They need to adjudicate conflicting needs and priorities based on a thorough understanding of all facets and forces in the project, which means that team members must bring their expertise and concerns to the table.

Management must somehow orchestrate communication among members of the development group in such a way that requisite information is shared, informed decisions are rendered, and slow-and-steady progress is made toward the goal. In the University Libraries, team members communicate in person and by using electronic mail and bulletin boards. Constant and open communication is encouraged by the peripatetic management style of the project leaders. Regular meetings are held and everyone is invited to voice their issues, concerns, priorities, and constraints. There is no excuse for not being heard or taken into consideration. In addition to constantly encouraging dialogue among team members, management also works to create an atmosphere of caring and good will. For example, pizza delivery, coffee and donuts, jokes, and Library
Automation sweatshirts are standard practice for the development team. The most recent sweatshirt says: “Library Automation on patrol. Trust us. We’re professionals.”

The solution to the problems of politics and egos is to love what you do and to love the people with whom and for whom you do it. This researcher prays for charity—to be charitable to users and colleagues. I reiterate, with Andrew Carnegie: “My heart is in the work.” I recommend reading the book *Love and Profit* by James A. Autry (1991). “Research and development” really means “negotiate and compromise.” It means “grow and learn.” It leads to a Zen experience wherein you know that the team has done good work, that users have some of what they need, and that you can do better—given the time and resources. I hope to contribute to the design of digital experiences that express the full plenitude of human beings, not the agenda of a privileged few. And I hope to do this in such a way that my colleagues appreciate and enjoy my presence on the team.

**REFERENCES**