Design Languages

Design languages have been used to design things as diverse as products, buildings, cities, services, and organizations. They are often used unconsciously, arising out of the natural activity of creation and interaction with created things. Yet, when consciously understood, developed, and applied, design languages can build on and improve this natural activity, and can result in dramatically better interactions, environments, and things of all kinds.
During the years when the Xerox Star was being developed, John Rheinfrank was part of a design team at Xerox, but he was not working on computer systems. Along with Arnold Wasserman, William Hartman, and a joint team from Xerox and the design firm of Fitch RichardsonSmith, he was engaged in a top-down redesign of the entire Xerox line of copiers (Rheinfrank et al., 1992). The project was guided by two fundamental principles:

1. Understand how people use the machines in real settings.
2. Think of the machine as a medium in which the designer communicates to the users. See it not just as an object with mechanical functions, but also as a statement in a design language.

The phrase design language can be interpreted many ways, but in the sense that Rheinfrank and Shelley Evenson (a prominent graphic designer) describe it in this chapter, it denotes the visual and functional language of communication with the people who use an artifact. A design language is like a natural language, both in its communicative function and in its structure as an evolving system of elements and of relationships among those elements.

Drawing on their extensive design experience with products such as copiers, industrial control rooms, and computer interfaces, as well as traditional applications of graphic design, Rheinfrank and Evenson reveal fundamental issues that are highly relevant to software design. The plasticity of software interfaces—the ease with which the look and feel can be changed at will—makes them especially amenable to the conscious development and use of design languages, as discussed in the accompanying profile on the Macintosh Human Interface Guidelines.

— Terry Winograd

At the simplest level, natural (spoken or written) language is a means for communication. It is a tool through which we create the phrases and sentences that make up our conversations with other people. At a higher level, communication—and language—is inseparably assimilated into our daily lives and activities. We talk to one anoth-

er to get things done. We comment on the world around us, trying to develop a shared understanding that will help us to make better decisions. We communicate to build and maintain our relationships with people around us. We do not just use our language: We live with it. The generation and interpretation of phrases and sentences can be understood only when language is seen within the larger context of everyday activities and experience.

Just as natural (spoken or written) languages are the basis for how we generate and interpret phrases and sentences, so design languages are the basis for how we create and interact with things in the world. And, like spoken or written language, design languages are assimilated into our everyday activities, mediating our experiences with the world (often tacitly), and contributing to the perceived quality of our lives. Design languages can be used to design things that are small or large, tangible or intangible. Design languages have been used to design things as diverse as products, buildings, cities, services, and organizations. They are often used unconsciously, arising out of the natural activity of creation and interaction with created things. Yet, when consciously understood, developed, and applied, design languages can build on and improve this natural activity, and can result in dramatically better interactions, environments, and things of all kinds.

The power of using design languages consciously has been recognized and described by other design theorists, such as Christopher Alexander (1979) for architecture and urban design, William J. Mitchell (1992) for architecture and media, Edward Tufte (1990) for visualization, and Terry Winograd (Adler and Winograd, 1992) for software design.

Design Languages in Use

Design languages are present everywhere in our constructed environment.

Most design languages have evolved through unconscious design activities. For instance, houses in Nantucket have been designed with a language of form, materials, and structure that derives its substance from the local environment, and that arose in response to physical, social, and economic influences on early settlers (Figure 4.1).
Nantucket—an island with a relatively harsh climate—was settled by people who planned to make their fortunes from the sea, such as whaling and fishing. They began by building small houses, with cedar shingles placed over a wood frame. The cedar trees that have evolved to thrive on the island are small, rugged conifers. The early settlers learned that cedar wood had water-repellent and insect-repellent properties. The shingles made from them are also small; when overlapped over the house frame, they cause water to run off the roof and sides of the house. To take advantage of the beautiful views, people built houses with large windows. To protect against the elements, they built shutters to cover those windows. Beach houses had storm fences to keep water and sand at a distance. As the original settlers made their fortunes, they added on to their homes in a piecemeal fashion, often planting gardens in the enclave created by the multiple wings. The design language of houses in Nantucket arose from necessity and was adopted by convention. The resulting style of architecture has been recognized as appropriate to the area. It has since been codified, and it is now imposed on any new buildings, even though the necessities of the overall design intent have been supplanted.

Design languages can also be created to accelerate the assimilation of a new technology into everyday activity. For example, recent Xerox photocopiers have been designed with a language of form and graphics that communicates to users how they can be used (Figure 4.2). This language includes elements from other design languages that are familiar to users. Doors on these copiers have obvious handles that indicate how they are to be opened. Color and value coding distinguish various areas of the product. For example, areas where originals are to be fed into the machine are green, areas where paper is loaded are blue, and copy-output areas are red. Although users are not taught these colors explicitly, the consistent use of color and shade throughout a line of copiers produces tacit recognition, much in the way that
we learn most of our native language through using it, rather than through having its structure pointed out.

In addition, like other machines with which users are familiar, copiers have an overall frontal orientation. A language of graphic elements offers information at the point of need, with links to more detailed information elsewhere. Users do not need to be trained, nor do they need to refer to complex instructions. They quickly recognize handles, focus on colored areas as points of important interaction, and come to rely on informative graphics to orient them and to provide resources for understanding the intended interaction.

Design Languages and Meaning

Design languages are central to how designers contribute to experiences; they play a significant role in people’s everyday experience of the world.

Natural languages consist of words and rules of grammar, and are used to create meaningful utterances. By analogy, design languages consist of design elements and principles of composition. Like natural languages, design languages are used for generation (creating things) and interpretation (reading things). Natural languages are used to generate expressions that communicate ideas; design languages are used to design objects that express what the objects are, what they do, how they are to be used, and how they contribute to experience. People’s knowledge of natural language is the basis for their interpretations of what other people say; similarly, a design language can be the basis for how people understand and interact with a product. The best design languages create experiences of use that are simple and straightforward—then they go one step further, to make interactions between people and objects pleasant and continuously meaningful in the context of everyday life.

Design languages play an important role in the expression of the unfolding of meaning of objects. Essentially, design languages are the means by which

- Designers build meaning into objects, so that objects express themselves and their meanings to people.

- People learn to understand and use objects.
- Objects become assimilated into people’s experiences and activities.

Traditional conceptions of language have seen meaning as strictly associated with form and style, relatively independent of context. Design languages can be used most effectively when meaning is seen not just as the built-in sense of an object, but also as the quality of sense making that objects have and can produce, especially with respect to their surroundings. In other words, doing design requires more than making meaningful objects; it requires crafting whatever it is about objects that lets them participate in the creation of meaningful experiences. According to this view of meaning, the sense of an object cannot be separated from the experience that the object simultaneously sits in and helps to create (see Rheinfrank and Welker, 1994, and the discussion by Brown and Duguid in Chapter 7).

Typically, design languages consist of

- Collections of elements. Building blocks that designers and users employ to communicate, such as shapes, textures, colors, actions, and metaphors
- Sets of organizing principles. Descriptions of how the elements might be composed to build things that have meaning
- Collections of qualifying situations. Examples of how elements and principles of composition might change based on the context

Designers can use all these components to generate new uses of the language (products or services) that are meaningful in particular contexts of use, and that honor a particular set of underlying assumptions about people and people’s uses of the product or service. Design languages are not fixed formal languages. They are generative: They contain an imperative (an underlying intent or need for interaction) and social mechanisms for their own revision through ongoing interaction. Users of objects respond to the resources given to them by designers. By using (or failing to use) these resources, users provide the basis for the revision of the language. As the expectations of designers and users change, the language evolves through invention, accident, and other events that create an impetus for transformation. By consciously developing the language, we can
dramatically accelerate the process through which a new design language is naturally and meaningfully assimilated into activities and experience.

Design languages usually evolve gradually. They become a deeply held tradition, are difficult to challenge, and are even more rarely questioned. People tend to assume that they are valid, and to continue to work through them, rather than to think about them and their appropriateness. This habit is particularly dangerous in a time of accelerating change. A business might be using a language that no longer allows it to produce products that make sense to customers; thus, it becomes particularly vulnerable to a competitor whose product line suddenly addresses a more appropriate set of needs.

Digital cameras that use a design language of professional 35-mm film-based cameras either will be uncompelling—interpreted as nothing new or special—or will be overly constrained by the behavior of light-lens optics, and will therefore fail to take advantage of the primary innovations possible with digital technology. In another way, cellular telephones now emulate the form and features of wired telephones, rather than shifting to a design language that is grounded in the diverse communication needs of people who are mobile. We are just beginning to see the breakdown of this rigidity with the introduction of products such as Sony's MagicLink and Motorola's Envoy and Marco. These personal communicators embrace a person's need to deal with the complexity inherent in mobile work styles. In addition to the primary communication functions, they include personal organizers and tools for managing communications, and they support the use of images and data during communication.

Taking advantage of design-language innovations, we can produce the core elements of successful corporate design strategies (Rheinfrank et al., 1993). Companies can consciously consider and design new design languages, with the intent of creating new product paradigms. Sony developed the My First Sony line of consumer electronics for children a few years ago. Each product has a balance of elements that suggest "cool audio devices" with elements that suggest "childlike and playful." The products in the line that express these attributes most obviously have established a clear benchmark for products in this category.

**Benefits of Design Languages**

Explicit use of a design language is beneficial from the three essential perspectives of interpretation, generation, and assimilation into everyday life.

**Interpretation: Design languages and learning**

Designers can use a design language to give people strong cues and powerful resources for learning by using. They can do so by shaping their designs to take advantage of what people already know and have the potential to do. For example, a designer of photocopiers who understands the design language of door handles and applies this understanding to the design of "opening things" can provide people with physical cues for how to open things by using handles that correspond to the contexts of the artifacts being designed and that clearly communicate "open here" in those contexts.

One thing that designers can do to help people learn is to reveal functionality through transparent-box design, rather than to conceal it through black-box design (Lave and Wenger, 1991). In a black box, all functionality is opaque, or hidden from view, and people accomplish their goals by pushing buttons that signify nothing whatsoever about the inner workings of the object. The technology is made foolproof. (Note the degrading reference to the assumed competence of the user.) Point-and-shoot cameras that take care of everything and produce acceptable photographs are black boxes, even when they are sold in bright colors. From one perspective, they de-skill users and lead to impoverished communities of practice. From another perspective, they make picture taking easy, freeing the user to attend to the scene and to capture an event without struggling with settings and technical adjustments. As users, we both gain and lose when we are highly constrained by the point-and-shoot limitations of the camera.

In a transparent box, functionality is revealed, and people are provided with the opportunity to comprehend the inner working of the artifact that they are using. A camera with more flexibility can enhance the capabilities of the user to capture events in a variety of ways, even using specialized features to enhance the sensory capacities of the picture taker and to create special effects. Rather than foolproofing, we
can create a transparent control space, designed so that the depth and breadth of options are clear to the user when they are appropriate. A well-designed transparent object selectively reveals to people just enough information about how to use the artifact and how the artifact works for people to accomplish their goals or to do tasks. A transparent camera could be optimized for certain situations of use (family gatherings, sporting events, and so on) and could reveal technical functionality and suggestions for use that would allow users to enhance the quality of their photos and to develop their skills at a pace of their own choosing. We can arrange this selective revelation to allow the meaning of the full experience of using the artifact to unfold gradually according to need, over time, as the artifact is used.

Design languages also make it easier for people to learn, by allowing patterns of use to be transferred from one artifact to another. Designers can help users to transfer patterns of use by consciously and consistently understanding and applying a design language across a family of product offerings. People can then take advantage of the resulting similarities in how products are used, often without any conscious relearning.

The members of product families resemble one another with respect to their forms and interaction styles. Thus, when people learn about one family member's form and interaction style, they can transfer this learning to other members of the same product family, adding to the overall usability of the entire product offering. Most consumer power-tool manufacturers have employed a simple language across their products. The trigger on a drill is the same color as the switch that starts the belt moving on a sander. The highlight color means tool on and is used consistently across the product range. The standard Apple Macintosh user interface (see Profile 4) is another good example of this consistent use of language: Users can easily learn new applications, because they take advantage of their understanding of similar and familiar objects, and the associated conventions of interaction.

**Generation: Design languages and business**

Businesses can use design languages explicitly to create coherent ways that customers form a consistent—and increasingly more positive—impression of the corporation. This impression becomes stronger as customers are repeatedly exposed to a corporation's products. In the end, this impression plays a large role in determining future customer purchases, thereby also determining the success of the corporation. Nike uses its communication design language to target its messages to its various audiences, (youth, women, sports enthusiasts), yet its overall impression is unified and coherent: Everyone can "Just do it." This powerful message is then confirmed by Nike's product array.

The most effective design languages do more than create coherence; they also create relevance. They ensure that the product line meets customer needs, and that it plays a crucial role in customers' everyday activities. Thus, the leading design languages set an industry standard for coherence, relevance, and quality. Once adopted by customers, they force competitors either to adopt the same standards or to produce equally strong alternatives. If a design language that is appreciated by customers becomes dominant in a market, it may take rival corporations years to make the changes necessary for effective competition. Nike's products are designed with the user's context in mind. Not so many years ago, we used only sneakers. Taking up the language pioneered by Nike, we now use specialized shoes for aerobics, the water, walking, running, hiking, and even cross-training (sneakers revisited?). Reebok tried to compete by applying similar principles in another niche, to street shoes (the BOKS line), but has never been able to challenge the cultural dominance of Nike.

When a new or recast product line appears and convinces customers that it will meet their needs in a fundamentally better way, then customers move toward it steadily and irreversibly. This observation holds especially true when design languages are used to influence the whole spectrum of ways that a company communicates with its customers (advertising, customer service, literature, purchase experience, merchandising, and so on).

**Assimilation: Design languages and meaningful innovation**

One quality of design languages makes them potentially dangerous in an environment of accelerating change: They are most effective when they acquire presence over time, through their assimilation into people's lives. Design languages typically are most influential when they
have become deeply embedded, when people can unconsciously assume that they are valid and can continue to act through them, rather than think about them and their appropriateness. A reflective corporation can build on traditional assumptions and then identify, develop, and implement a new design language quickly and effectively as part of the movement toward a dramatically improved market position. For example, the first cellular radios were wireless telephones, in much the same way as we referred to the first automobiles as horseless carriages. In both cases, a primary innovation was attached to the past, through a prevailing design language. Its early development was constrained by the reference until enough momentum was achieved for the interactive evolutionary process to modify the design language. The potential of cellular communication almost certainly has as little to do with today's telephone as today's automobile has to do with buggies and buggy whips. We will one day look back at today's personal computers (TV sets with typewriter keyboards) and software as historic curiosities of the same ilk.

The Design-Language Approach

In our design projects through the years, we have identified the following five steps in the development of design languages:

1. Characterization
2. Reregistration
3. Development and demonstration
4. Evaluation
5. Evolution

We shall describe each of these steps in detail.

Characterization

Characterization is a process of describing the existing underlying assumptions and precedent-setting design languages. In many cases, when designers design, they unconsciously accept current underlying assumptions about the nature of the object that they are designing, without challenging those assumptions. For example, for many years, offices were made of traditional building materials. The difficulty of reconfiguring spaces constructed of these materials created a long tradition of closed offices and broken up spaces that required extensive (and expensive) remodeling to accommodate the natural variety of uses. In the mid-fifties, modular office systems were developed. They introduced an entirely new language for the production of offices. Along with the language came assumptions about openness and flexibility, and radically new building economies. A new industry formed around this primary innovation and the attached set of assumptions.

The office-building industry today faces the challenge of readdressing its assumption base. The original design language was deeply grounded in the productivity of single individuals during a time when the prevailing work tools were calculators and typewriters. Current work practices seem to be evolving toward conditions that call for collaboration, communication, geographically distributed work teams, and organizations with fluid structure. None of these conditions can be addressed effectively with the current office design language.

As an example of characterization, we shall describe a design-language-development program that we conducted for FRS, a division of Emerson Electric. FRS produces equipment that digitally controls and monitors complex process manufacturing facilities, such as paper, chemical, or pharmaceutical plants and refineries. When these types of plants were first built, operators used controls tied to individual aspects of the process, and monitored the production process using strip charts, sight, and sound. As analog and digital controls were introduced, operators moved to control centers and control rooms, where they continued to perform similar functions remotely. This arrangement was desirable, because the production process was usually messy, noisy, and dangerous.

The interface to early digital control systems mimicked the readouts from the dials and face plates and the information from the strip charts. A larger span of control simply meant more discrete pages of information, reflected in separate screens. An interface for process control today could have 250 pages of electronic information and abstract representations of physical objects. The role of the operator in a plant is to watch these pages of information, to spot problems, to respond to alarms, and to intervene in the process as necessary. The control-room physical environment is dominated by furniture and
equipment consoles that have an industrial ruggedness, clearly related
to the equipment present on the plant floor. This view of the plant,
operators, and control-room regime has remained unchallenged since
the introduction of the first system.

Every design-language-development project should start with a
clear characterization of these current assumptions. It is only from this
explicit understanding that a designer can challenge—then recast—the
current set of assumptions. At the end of this first step, assumptions
such as the ones that we just described should be part of the conscious
awareness of the designer.

Reregistration

Reregistration is the creation of a new assumption set and design
framework. During this step of design-language development, designers
use a variety of methods to explore emerging trends and needs. Market
and technology surveys are conducted. Latent or masked user
needs are identified through a variety of (ethnographic) field-research
techniques, such as observing current use situations, interviewing
users, and surveying activity in related situations.

Designers then consciously create new design assumptions by con-
structing alternative design–conceptual frameworks, then acting out
these frameworks inside idealized situations of use.

In our process-control project, after we had assessed the situation,
we challenged the assumptions surrounding the role of the operator in
the plant community. Observing practice, we saw that operators were
the hub of communication in the plant. They supported a number of
other functions, such as engineering, maintenance, quality control,
operations, planning, and scheduling. We also observed that operators
were familiar with the physical layout of objects in the plant. Many
operators performed routine maintenance and troubleshooting func-
tions on alternating shifts. Through our observations, we identified
specific patterns of activities and interactions that either were un-
supported, or were supported ineffectively by the products and services
provided by the vendors in the industry. We then began to define a
new design direction. We realized that the control room was the pri-
mary center of a business, rather than a place to keep a big control
panel.

Creating a new set of assumptions fosters creativity by allowing a
design team to move away from designing according to preframed and
preanalyzed sets of assumptions, and by encouraging members to move
toward designing according to the patterns that they construct collabora-
tively from current contexts of use. Results are best when customers,
users, engineers, salespeople, and executives can all participate in this
reregistration process, since each brings unique and relevant domains
of knowledge and experience (see Profile 14 on participatory design).
The creation process is symmetric: As usage changes, the language is
modified. Seen from another perspective, the developers and users coe-
volves, with the language as the mediator of meaning.

Development and demonstration

Designing the language begins when the team gives concrete visible
form to the assumptions and to the design framework. In our exam-
ple, a designer might suggest that more representational process
objects would help operators to orient themselves to the plant.
Instead of taking the physical plant with which operators are familiar
and abstracting it, why not just represent objects more like the objects
are in the real world? Following this line of thought, elements in the
design language for a process-control interface would be representa-
tional images of boilers and other objects in the plant.

At the same time that the language elements are being developed,
they are being demonstrated, in scenarios, sketches, or prototypes.
These embodiments act to create tangible stories of the future that can
be compared and contrasted with stories gleaned from the field during
the characterization stage. This simultaneous demonstration of the
elements as they are being conceived is crucial. Although it is not diffi-
cult to picture a concept such as "more representational boilers," con-
cepts such as "communication support for the plant community" may
be difficult to imagine without tangible examples.

Demonstrations also allow designers to evaluate how elements
interact with one another, to make sure the meaning of one does not
confuse or negate the meaning of another. Another purpose for iter-
active demonstration of the language is to test the reframing hypothesis.
For example, is there enough carryover from existing process-plant
practice to support the quick adoption of this new technology?
Demonstrations also help organizations to make development assumptions explicit. Specifications come to life in the design language, and misconceptions can be discussed and resolved quickly. The interactive coconstruction of a design language is facilitated by the concreteness of the demonstrations and prototypes (see Schrage’s discussion of prototyping cultures in Chapter 10).

**Evaluation**

The *evaluation* stage places the design language in context. After the demonstrations have been developed, they are placed within real or hypothetical situations of use. Ideally, this evaluation starts as soon as the first demonstrations are produced, to see whether the language resonates with its users. Refinements are made to the language as necessary. In an evaluation of our process-control interface, we found that an element such as “people buttons” for direct dialing to a coworker, which made perfect sense to the design team, were confusing to a user when they were placed next to buttons for powering an object. Both were perceived as buttons by the user, yet they had completely different functions, which were not revealed by their appearance.

The design team develops a stable set of demonstrations, constructing several composite objects, based on the design language, that cover the range of expected future work situations. These new expressions can also be evaluated and iterated on, so that they further contribute to the richness of the language. Applying the process-control language to portable (small-scale) and wall-sized (large-scale) products can lead to new language elements, such as process abstractions (used to create an overview of the entire process) in a tablet or on a wall, or modifications to existing elements (less representational boilers that scale well on a portable.)

**Evolution**

No matter how good the design, there will be further additions and changes to the language once it is in use. The best design language is still appropriate for only the needs of its time. Needs and practices change constantly. Designers must be sensitive to such changes, and must continue to extend the design language. Where possible, design languages should be constructed to support users in developing new forms of interaction, based on their own activities in their own situations. One of our goals for the process-control design language was to support members of the plant community in developing custom-tailored interfaces specific to their needs, giving them the power to create new elements as they need those elements.

**Conclusion**

Design languages are powerful expressions of what we know and share about the world around us. Henry Ford invented a design language for automobiles in a way that allowed him to mass produce cars and to reach customers. Innovations in the car language were fast and furious in those early days. Sadly, the language of the car has evolved slowly since then, and most of the changes have been merely of form, rather than of substance. Software design seems to be poised at the edge of a similar era of innovation. Just as running boards on cars were part of the assumption set of the car language for years, windows, icons, menus, and pointing (the WIMP interface) seem to be a part of the unchallenged assumption set for computer interfaces. Yet, games and more play-oriented activities implemented on CDs use different languages for interaction. We propose that explicitly creating software design languages based on their contexts of use, and according to the steps we have outlined, will provide new suggestions for evolution or revolution of the WIMP language, and will move us into an era of more meaning-full interaction with computers.

**Suggested Readings**


About the Authors

John Rheinfrank is a senior strategist at Doblin Group, in Chicago, where he directs programs that focus on decision strategy and on breakthrough product and service development. Prior to joining Doblin Group, he was a Senior Vice-President at Fitch, an international design consulting firm, where he was cofounder of the Exploratory Design Laboratory (EDL), an interdisciplinary team conducting project-based design research.

Shelley Evenson is a design strategist at Doblin Group. Originally trained as a graphic designer, she has worked in interaction design for the past decade, with a focus on creating compelling user experiences. Prior to joining Doblin Group, she was a Vice-President at Fitch and was cofounder of EDL. Her work has been published by the ACM, the American Institute of Graphic Arts, and the American Center for Design.

4. MACINTOSH HUMAN INTERFACE GUIDELINES

The greatest reason for the early success of the Macintosh was the perception by potential buyers that it was easy to use. This perception was not an accident; it was the result of Apple's conscious strategy in creating "the computer for the rest of us."

What was different about Macintosh applications? Most visibly, they were among the first consumer software to make use of the graphic user interface (GUI) that was pioneered by the Xerox Star (Profile 2), with its windows, icons, menus, and pointing device (leading to another commonly used acronym: WIMP). There has been much debate over the benefits and limitations of such interfaces, yet it is clear that, in the large, GUIs enhance usability for many applications, especially for novice users.

Ease of use comes from more than what you see on a single screen. A more subtle, but critical, aspect of the original Macintosh interface was consistency across applications. Applications for different purposes, built by different developers, all followed a common style of providing GUI elements and using those elements to communicate with users. This commonality was the result of a concerted campaign by Apple evangelists, such as Bruce Tognazzini (see Tognazzini, 1992). The job of an evangelist was to convince applications developers to structure their interface in the Macintosh way, rather than in their own way, even if they thought their own way was prettier or better. The evangelists carried their bible, later published as the Human Interface Guidelines (Apple, 1987), illustrated in Figure 4.3.

Profile Author: Terry Winograd
Document windows

Because a document may contain more information than a window can display at one time, the window provides a view of a portion of a document. Document windows also provide a graphic representation of opening, closing, and other operations performed on documents. Windows are usually, but not necessarily, rectangles. Figure 3-3 shows a standard document window and its components.

Opening and closing windows

Windows appear on the screen in different ways as appropriate to the purpose of the window. The application controls at least the initial size and placement of its windows.

A standard window has a close box. When the user clicks the close box, the window goes away. (On the Finder, this is animated—the window shrinks into the folder or icon from which it was opened.) If an application doesn’t support closing a window with a close box, it shouldn’t include a close box on the window.

Figure 4.3 Macintosh Human Interface Guidelines. These guidelines specify how each design element (in this case, a document window) should look, and what behavior it should exhibit. The correspondence of look and behavior is the key to a working design language. (From Apple Computer Inc., Human Interface Guidelines: The Apple Desktop Interface, © 1987 Apple Computer Inc. Reprinted by permission of Addison-Wesley Publishing Co., Inc.)

The evangelists’ achievement was that any user who was familiar with a few Macintosh applications could approach a new application with a reasonable sense of what it could do and how to make it perform. This sense of familiarity led to the feeling that the Macintosh was easy to use. The guidelines defined a comprehensive design language, as described by Rheinfark and Evenson in Chapter 4, and were supported by the availability of a programmers’ toolbox, which facilitated developers writing programs that followed the guidelines. In many cases, extra programming was required if the developer wanted to violate a guideline.

The Macintosh interface design language included visual and syntactic details, such as the names of standard menu items, and deeper functional elements, such as the use of a clipboard with universal cut, copy, and paste commands to be provided in a standard way in every application. In the early days of the Star, the Lisa (Macintosh’s predecessor), and the Macintosh, there was a raging debate about whether the moving and copying of documents, text, and drawings should be done via cut and paste from a clipboard, or via move, copy, and delete, commands, which were dedicated keyboard buttons on the Star. The differences were not just in the names, but also in the underlying conceptual model. The Macintosh designers may have chosen a less functional alternative (there are many people who still argue that they did so), but they did choose, and that choice led to consistency. Every application provided the same clipboard commands, making it possible for users to move text and drawings from one application to the other, regardless of the application. It was many years before this seemingly elementary form of consistency was available in either PC systems or Unix workstations.

A quick browse through Apple’s Human Interface Guidelines reveals dozens of language elements, such as the organization of commands into menus; consistent use of dialog boxes for parameters, warnings, and errors; and standard ways of organizing and managing windows. Equivalent style guides now exist for every major interface. It is interesting to see the variations that have emerged from what is fundamentally the same interface, as illustrated in Figure 4.4. The differences are reminiscent of the relationships among simple corresponding vocabulary words from French, Spanish, and Italian. The common Latin origins lead to a basic similarity, but each language has its own consistency. Also, as is true of natural languages, there is little room for argument about which communicates more effectively. There are minor variations, but, for the most part, what is important is that, within any one language, there is a consistent way of using words that is understood by everyone who speaks that language.

In all human languages, rules are made to be broken—creative innovation violates previous conventions. Design languages continue
had been effective for their design of screen fields, layout, and interaction sequences. The designers of interfaces yet to come will have the same difficulty in breaking away from the WIMP language in which they are now so fluent. The WIMP GUI is not the ultimate user-interface design, any more than Latin was the ultimate language. It has been hardy and useful, but it is tied to the hardware and systems tradeoffs that prevailed in the 1980s.

Suggested Readings


to change and evolve, just as natural languages do. The language of the Macintosh was a replacement for a long-standing and well-known language based on character-terminal interfaces, such as the IBM 3270, for mainframe applications, such as airline reservation and forms entry systems. As Tiddel describes in Chapter 2, the designers of new GUI applications had to break loose from the tacit assumptions that