CHAPTER 4

DESIGNING VISUAL INTERACTIONS

4.1 DESIGNING FOR COMPLEXITY

Data-intensive systems are changing the scale, scope, and nature of the data to be analyzed. The tools we use to do our work are becoming increasingly sophisticated and intertwined in a mesh of data, information, and computation. Information and multidimensional data come from multiple sources and are processed using a variety of computational methods and approaches. Analyzing the data is usually a collaborative effort requiring expertise from different disciplines. Complexity is continually increasing. At the same time that users of these systems ask for simplicity, they also ask for more features and tools that extend their capabilities and allow them to explore larger datasets quickly and dynamically. Although it might seem that simplicity and complexity are opposing forces, simplicity is not the opposite of complexity (Norman, 2010). Complexity is a statement about the world; simplicity is a statement about the mind. Complexity can be tempered if a system is properly structured and organized so that it can be understood. Once understood, the effort involved in learning its structure is forgotten. A well-designed complex system, after it becomes familiar, is often described as simple or "intuitive." Simplicity is psychological. *Perceived simplicity*, which is the first impression of a system's interface based only on its look—the number of graphical and control elements that make up its visual interface—is different from *operational simplicity*. Even

Making Sense of Data III: A Practical Guide to Designing Interactive Data Visualizations, First Edition. Glenn J. Myatt and Wayne P. Johnson.

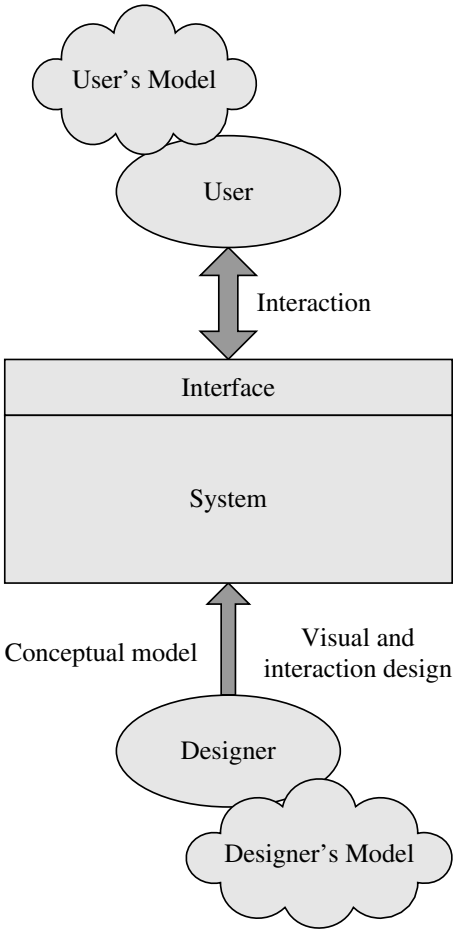

FIGURE 4.1 Two perspectives of a system (based on Norman, 2002)

a system with a complex visual interface can be operationally simple if its structure is logical and well organized, we have taken the time to learn it, and it makes interaction efficient.

As we interact with a system or its help systems, we construct in our minds a *mental model* about how a system works. We use mental models all the time. For example, if we own a gas or propane stove, even if we have not looked inside, we likely have a simple mental model of gas flowing through pipes that lead to the burner, the igniting of the gas by a spark, and the distribution of the burning gas around the burner through many small orifices. When it doesn't light, we infer from our model that something is plugged the spark is not being generated, or the pressure is low. Our mental model of how a system functions helps us operate it without memorizing arbitrary procedures and helps us make reasonable predictions about what to do when something unexpected happens. If a system is well designed, our mental model will closely correspond to the conceptual model the designers had in mind and will be consistent with the way the system behaves or responds operationally. How the two different models relate to the system is shown in Fig. 4.1.

As human beings, we have a strong psychological need to explain. The mental model we construct of a badly designed system—we always create one—may not be consistent with how the system works. When the system's behavior doesn't match our model, we get confused. We have difficulty predicting the outcome of certain actions. We cannot figure out what combinations of buttons to push. How we engage or find certain functions may appear arbitrary. We try to commit them to memory but find we have forgotten them the next time we use the system. This leads to a sense of powerlessness, frustration, and a perception that the system is complicated. We are usually willing to study and master the complexity of a system when we know it reflects and reduces the complexity of the work, but we resent and resist the effort when complexity is the result of errors of design.

A user will form a good mental model if the system's operations support the user's tasks, are logically and consistently organized, and provide feedback about how it can be used in ways that the user can perceive. The designers' conceptual model, therefore, must be rooted in a thorough understanding of the work that the system will support. This includes the goals that must be accomplished, the activities or tasks that will be performed by individuals or groups to achieve these goals, and the work environment in which the system will operate.

For visual analytics and data-intensive systems, this means understanding the nature of data analysis and exploration. The nature of this work differs between and within application domains. Between application domains, the data to be analyzed and the tools and methods used in the analysis vary widely along many dimensions. Within application domains, there are usually layers of analysis that must be done as the raw data is successively transformed until it reaches a point where decisions can be made. Each layer of analysis may be done by individuals with different subject matter expertise. Each expert may have his or her own datasets derived from the source data, questions about the data, and computational tools and methods for analysis. The context plays an important role in understanding the work, which, in turn, affects the design of the system. The application of microarray technology in the life sciences illustrates this point.

DNA microarray data is used to measure the activity and interactions of biological genes. Within the scientific research community and industry, the uses for this data include the discovery of genes, diagnosis or prognosis of a disease, and the assessment of the toxicity of drug candidates or other chemical agents. Each of these areas have distinct scientific tasks with their own sets of questions about the data. Examples of these tasks include identifying genes with similar or different co-expression patterns, looking at gene expression patterns under various stress conditions, or mapping gene expression data to metabolic pathways (i.e., a series of chemical reactions within a biological cell). Addressing the scientific questions may involve one or more data analysis tasks with quite different analytical techniques (Berrar et al., 2003).

Designing visual interactions to support the work of analysis and exploration requires sensitivity to the context in which the tasks are being performed.

The advent of tools such as Protovis should make it possible to design systems more quickly with interactions comprised of visualizations and quantitative graphics as well as user interface controls to create tools that support specific domains. The process of design is a dialogue between designers and users. Whether the dialogue is formal or informal, a good design process contains essential elements that even small projects with just one or two individuals should consider. Of these elements, one of the most important is the *conceptual model*. The conceptual model is a high-level description of the concepts—objects, actions, and attributes—that underlie the organization, appearance, and behavior of the system being designed (Johnson, 2010). The product of design also communicates with users. Its visual interface is a system of signs and conventions—a semiotic system—that reflects a conceptual model, whether the model has been carefully designed or not. If the signs and conventions have been adopted from the workplace or are familiar, and they are visually organized to fit the tasks, the system will be more easily understood, and the interaction will flow.

The remainder of this chapter is divided into two sections. The first section discusses the critical activities of design that loosely form a process: analysis of the user and work; design of the conceptual model, the architecture of the interface, and the visual interaction; prototyping as a means for reflecting about the conceptual model and exploring alternative designs for the visual interface and interaction; and usability evaluation. Although some of these activities must initially be done before others, design is a process of continual refinement. The byproducts from each stage of design, such as the various models and prototypes, may be reexamined several times over as new insights are gained. The second section discusses the physical aspects of the design of visual interaction: the form and content of the user interface, visualizations, and graphics; the alternatives for interaction; and the implications for design from what has been learned about how our minds work.

4.2 THE PROCESS OF DESIGN

As stated earlier, good design begins with a thorough understanding of the users, their work, and their environment. Within the human-computer interaction (HCI) and interaction design(ID) communities, the activities that comprise the design of interactions with technological products go by different names depending on the emphasis and which activities are included. User-centered design, interaction design, usage-centered design, contextual design, activity-centered design, participatory design, and cognitive work analysis are some of the names that are used. The emphasis varies: user or use; individuals or groups; work that is service-oriented or work that is mostly in the mind; or interaction with a wide range of technological products and devices or with primarily a computer display and input devices. But regardless of emphasis, these different approaches always begin with a focus on people and what they

are trying to do and accomplish. This is the foundation on which the rest of design relies.

Most of these approaches define a process. So that projects do not incur unnecessary overhead, some of these processes such as contextual design can be scaled from small individual projects to large projects that span corporate divisions. The simplest definition of a design process consists of three steps (Johnson, 2010):

1. Analyze the tasks.
2. Design a conceptual model.
3. Design the user interface and interaction from the task analysis and conceptual model.

This captures the basic design activities. Consider the commonly used spreadsheet application, which was initially designed to replace work done manually by accountants. In accounting, the spreadsheet was a "spread" of facing pages in a bound ledger book with columns that, among other things, were used to keep track of expenses by category. Invoices were itemized in the left margin, and the categories were entered in the headings of the columns. In the cell where the row and column intersected, the payment for an invoice was entered. One of the tasks the accountant performed was the summing of columns. In this description of the work, there is no mention of technology, interfaces, or the sequence and flow of the interaction.

The first step of design—the analysis of tasks—is done in the context of the application domain using language that would be familiar to the users—accounting, in this example. Each task, and the steps involved in doing it, must be identified. From the detailed description of tasks, a conceptual model can be designed.

The second step—designing the conceptual model—describes the operations that the system can perform and the concepts the user must know to perform them. In this example, concepts such as "spreadsheet," "column," "row," and "cell" were understood from the ledgers accountants used for bookkeeping, and operations such as "summing a column" was a task they performed. Incorporating these into the spreadsheet application's conceptual model as the objects that can be manipulated and the operations that manipulate them made it easier for those familiar with accounting to learn and understand the system.

In the final step, something concrete begins to emerge from design. The objects and operations in the conceptual model are mapped to actions and to visual signifiers—graphical elements, icons, or symbols—which are representations of familiar objects corresponding to digital representations in the system. In the example, the white space between two vertical lines of the spreadsheet application's user interface was a "column," and it appeared the way it would in a ledger. The terminology used in the system such as "spreadsheet," "worksheet," "row," and "column" would all have been familiar concepts. The mapping also has to specify the actions required to perform a task. In

today's spreadsheet applications, two actions are needed to "sum a column." By clicking the mouse on the first cell in the column to be added and dragging the mouse over the cells to be summed, and then pointing the mouse to the $\sum$ symbol in the toolbar, the user invoked the operation that summed a column.

The preceding description is a simplification of what can be a complex process, but it provides a useful outline for design: observe the work to be supported; from what you learn about users and their work, design abstract models of work and systems that keep open the options for implementation; then design concrete alternatives—prototypes—from the abstract models that can be handled and tested by users. Much has been written about the various approaches, and references will be included in the "Further Reading" section of this chapter. The design process shown in Fig. 4.2 is a composite process that includes activities common to some of the design approaches listed earlier. In the following subsections, we provide an overview of the composite process but discuss only those aspects that are relevant to the design of visual interactions for data analysis and exploration. Although the figure shows the process as though it were linear from top to bottom, it is actually highly iterative with feedback and assessments at later stages often requiring changes at one or more earlier stages. The activities grouped as "abstract" are activities done to understand the problem space and what work the system should support. Within these activities, no commitments are made as to how the system will be implemented. The activities grouped as "concrete" are the stages that design how the system will "look and feel" and evaluate how well a proposed solution achieves established goals.

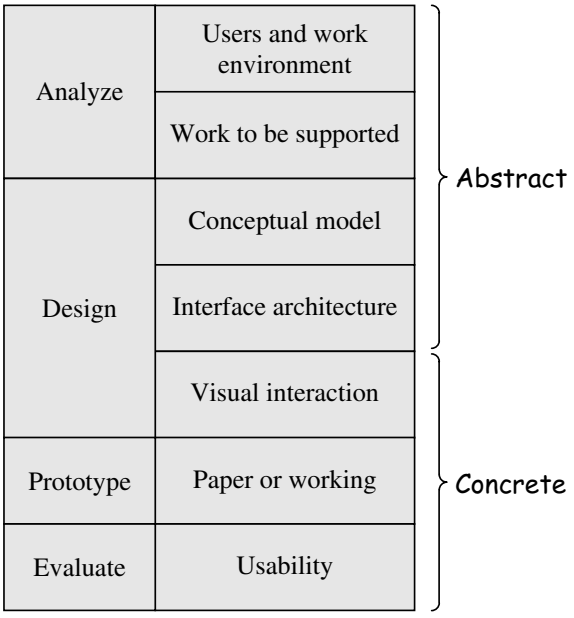

FIGURE 4.2 A composite design process

4.2.1 Analyze

Assuming a preliminary study has identified a problem area that needs to be addressed, how do we design visual tools and systems to improve the ability to analyze and explore the data or see it in new ways? To begin to answer this requires data about who will use the system, for what activities and tasks, and in what context. For example, consider what is involved in the analysis of DNA microarray data. As mentioned earlier, microarray technology is used in several types of scientific studies. Scientists measure the amount of some gene product such as protein or RNA that is expressed by genes in the DNA of a cell. Knowing the level of gene expression in a cell, tissue, or organism provides useful information. Two examples of use are identifying viral infections or exploring the sensitivity or resistance to drugs used for chemotherapy in the treatment of cancer. Figure 4.3 is an overview of the data-analysis process of a microarray study. It is not necessary to understand the details of each step. What is important is to understand that user interfaces, visualizations, and graphics are part of a larger socio-technical environment based on advanced technologies in which work is practiced. Marks on a visualization or data graphic that represent genes with unusual expression patterns may become the basis for a literature search or a search against a database for other related genes. The transition to these related tasks of searching, if not the tasks themselves, is part of the work that must be supported.

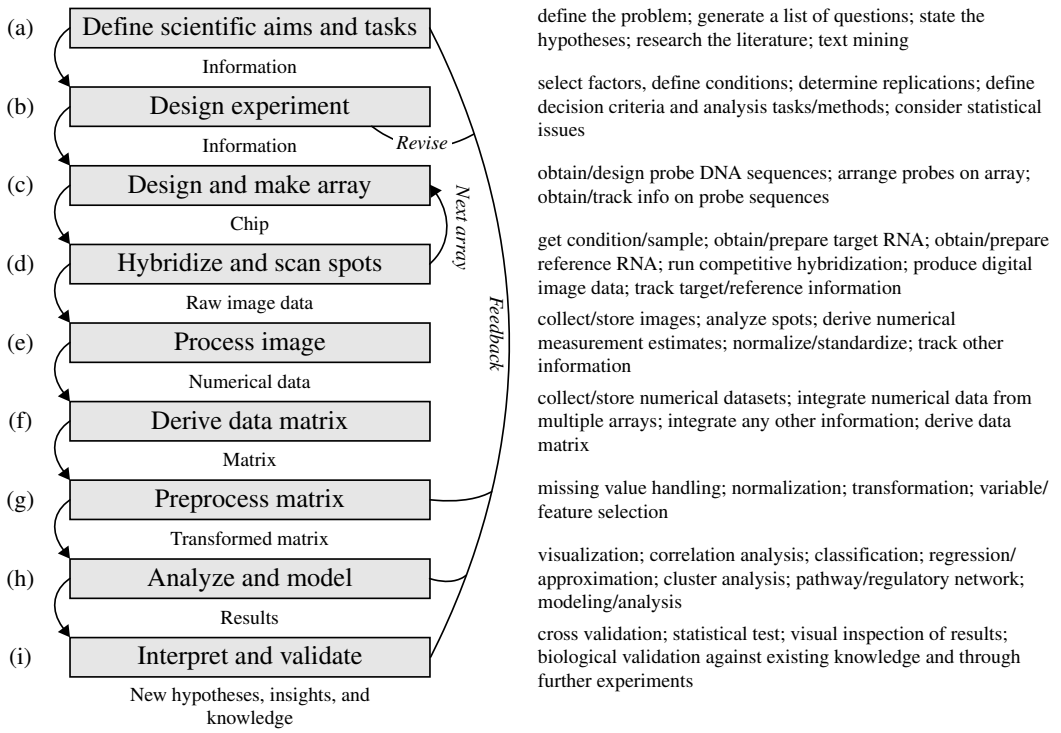

FIGURE 4.3 Overview of a DNA microarray data-analysis process (Berrar et al., 2003)

Design decisions must be based on facts and details elicited from discussions with users about their work and from observing users at work, instead of, as often happens, from speculation about users' needs. A variety of techniques exist for gathering data from work environments in a systematic way. These techniques include unstructured or structured interviews, focus groups, questionnaires, direct observation, contextual inquiry, and ethnographic interviews. Some of these techniques can provide useful information about users but limited information about how the work is actually done. The work practice has structure, but much of it is implicit, and designers must understand the work at a fine level of detail to design systems to support it. Designers need concrete data about the work as it is actually performed, not abstract summaries of it.

Of the methods previously listed, we have found variations of contextual inquiry (Holtzblatt & Beyer, 1998) to be a useful field method. The structured interviews are done on-site to directly observe the work in context as it is performed. Users, particularly experts, are so skilled at what they do that they often become unconscious of the lower-level steps they take as they perform various tasks. They may try to explain their work to the interviewer in terms they think will be understood. By engaging them in their work and asking about *how* terms, tools, or concepts are used, the details needed to develop the conceptual model will be made explicit in the interviews.

In addition to the users, the workplace (office, laboratory, or manufacturing site) also provides many clues about the work. Artifacts such as notes, documents, research papers, or even yellow sticky notes are signals that indicate the existence of information or knowledge that may be used for certain tasks. By interviewing selected individuals at work and carefully observing, the notes taken during the interviews can be interpreted and used to create models of the work environment, the work, and the resources required to do it.

The Users and the Work Environment. Work is done in an environment that imposes constraints on the design. It is important to know not only the characteristics of the types of individuals that will use the visual system being designed but also other factors that might affect its design.

Some key questions to ask of those who will use the system include the following:

- What are your goals?
- What is the system expected to help you accomplish?
- What tasks are performed? How often are these done? Which are more important and which are less important?
- What is your professional and educational background?
- Does your background include an understanding of data analysis and statistics?
- How do you stay current in your field?

The following questions can be answered through a carefully planned series of *contextual inquiry* interviews:

- What are the steps of each task?
- What data or information will be used during these tasks? Where does it come from?
- What results need to be generated by the system?
- What tools or systems do you use to accomplish your work? How often are these tools used?
- What is communicated with others involved in the work, and who are those individuals?
- What is the relationship between tasks?

Questions about the cultural and physical aspects of the work environment include the following:

- How is the work divided among teams?
- What standards, practices, and policies of the organization might constrain what you do?
- What are the guidelines and standards that IT requires of systems that are deployed?

Part of the information gained by asking these questions of different users allows the designers to create a profile for each class of individuals that will use the system. These can be generalized into *user roles*. For the microarray data analysis example, three roles might emerge: a molecular biologist, a statistician, and a computational specialist who knows data-mining methods. Each will want to examine the results from different perspectives. The results from the contextual inquiries are used to construct models of the work, which are discussed later in this chapter.

The Work to Be Supported. Much of data analysis and exploration is cognitive work. Cognition is what goes on in our minds as we carry out various activities throughout the day. One way to characterize cognition is as a set of processes. In an earlier chapter, we discussed three of these as they relate to vision: attention, perception, and memory. Others include recognition, learning, reading, speaking, listening, problem solving, planning, reasoning, and decision making. Donald Norman's *action theory* (Norman, 2002) is a theoretical cognitive framework that was developed to help designers understand the mental processes of users as they interacted with computational tools. Knowing about this theory helps to understand some of the language used in describing task analysis and conceptual models.

Norman described the interaction as the seven stages of action shown in Fig. 4.4. Our actions begin with a vague *goal* about doing something such as "going to the store." We translate the goal into a specific mental description—an

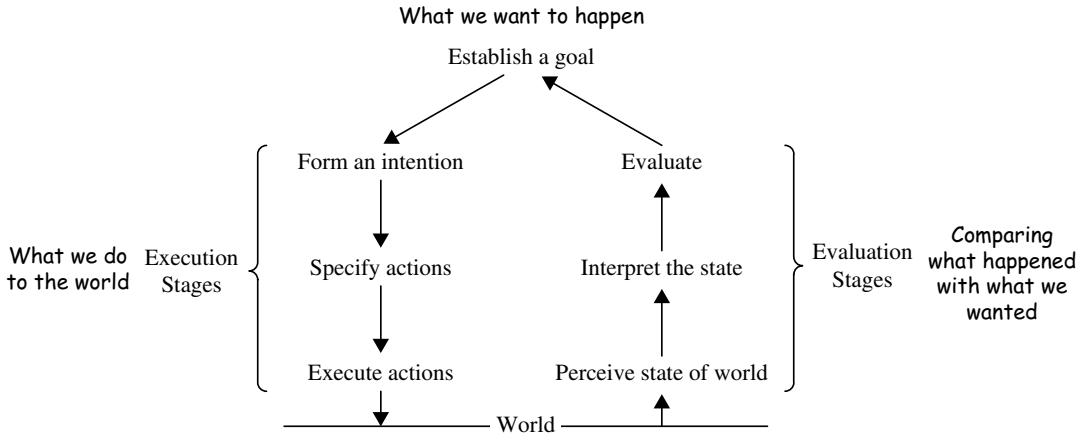

FIGURE 4.4 Action theory (Norman, 2002)

intention—of what is required to achieve the goal, but not yet specific enough to perform. The intention has to be broken into an *action sequence* of physical actions such as motion, movements, or thoughts—fine-grained steps—to carry out the intention of getting to the store. The first three stages formed the *stages of execution*. The remaining stages, the *stages of evaluation*, compare what we perceive to be happening against our expectations—the intention and goal.

Norman described two things that could make a tool difficult to use. He called these the *gulf of execution* and the *gulf of evaluation*. The gulf of execution was the difference between what the tool user wanted and the actions a tool provided to carry out the intention. If the action sequence was long—in other words, it could not be directly executed—the gulf was large and would require considerable cognitive effort to translate the intention into the steps required to carry out the intention. The gulf of evaluation was a measure of how easy it was to perceive and interpret the visible feedback provided by the system in response to the actions the user took. If the representation for the state of the tool—what was visible in the interface—closely matched the user's mental model, the gulf of evaluation was small, and the tool could be easily learned and understood.

Understanding work requires an ongoing dialogue with those who do it. To design the system's operations to support the work, the tasks the user performs needs to be understood as a series of small steps. The contextual inquiry method is designed to elicit this information, but less formal methods may be used as well. Regardless of method, the notes taken during the interviews need to be interpreted and converted into models of the work.

Various models have been devised to capture different aspects of the work. In the contextual design methodology, the models include *sequence models*, which represent the detailed steps required to achieve an intent; *flow models*, which represent the coordination and communication of work between individuals; and *artifact models*, which represent physical things—documents, reports, important research papers—produced or used in the work. In the usage-centered design methodology (Constantine & Lockwood, 1999),

essential use cases capture the detailed interaction between user and tool as a structured narrative of action and response. These are written in language the user would use and are generalized to capture only the intent of each action but not references to specific technology. This gives designers the freedom to consider implementing the actions of the task in other ways.

The starting point for the development of essential use cases is the user role. For each role, the following questions are asked of actual users:

- What is the goal?
- What must be done to achieve this goal?
- What capabilities are needed to do the task?

The fundamental question in task analysis asked of each action is "why?" Why is this being done? What is the intent? What is it accomplishing?

Note that it is often difficult to schedule interviews with domain experts because of the demands on their time. It is helpful to have someone on the design team who has an equivalent academic or professional background. This person will be able to ask focused questions relevant to the domain and explain the expert's answers later. There are also other ways to learn about the work. If the expert has some familiarity with the design methodology, he or she may write task descriptions as a starting point for further discussion. Although their task descriptions may include references to specific tools or visualizations, these experts provide insight into goals and intent. An example is shown in Table 4.1.

Another way to learn is by asking an expert to identify a set of key technical or research papers that are relevant to the work the tool will support. These can provide the background necessary to generate focused questions to ask during interviews. For scientific areas, research papers are structured to describe the problem, provide historical background, and explain the steps taken to solve the problem. Many also include a methods section that describes the work in more detail.

Analysis is iterative. The construction of models exposes gaps in understanding that lead to other questions. What was thought to be understood often requires confirmation or clarification. From the interviews and discussions, every key observation, insight, question, design idea, and breakdown—places in the workflow where problems arise—are also recorded. The results of analysis may contain the notes (or a model of these notes) and at least two key models: profiles about the types of individuals that will use the system (user role models) and the tasks they currently perform to do the work (task models).

4.2.2 Design

Design begins by developing an abstract conceptual model of the system after the analysis is well underway. From the conceptual model, an interface architecture is designed that groups related actions the user may perform into interaction spaces and shows how the user will navigate between these spaces.

TABLE 4.1 A Task Description Written by an Expert

Task-1: Group cells based on gene/miR expression.
1. Cells are grouped based on the correlation between the expression patterns of selected genes, microRNAs, or a combination of the two, and then visualized as a dendrogram.
2. (optional). Filter genes/miRs (rows of T) so that X% of the values have expression levels greater than Y; for example, 10% have expression levels greater than 7.
3. (optional). Select the *n* top genes with the highest variability in expression level, for example, the highest interquartile range or highest variance.
4. Calculate the 60×60 cell−cell correlation matrix M based on expression levels of genes/miRs from step 2. Provide options for Pearson, Spearman, and Kendall correlation statistics. There is no missing data for genes/miRs, so that is not an issue.
5. Using agglomerative clustering with $D = 1 - M$ as the distance matrix, construct a dendrogram of the cell lines. Provide options for complete and average linkage.
6. Display the dendrogram with options for coloring the cell lines based on tissue type *as shown in Fig. 2 of [MCT07 1483]*. (Note that the italicized remark does not refer to a figure in this book, but to a figure in a paper containing a visualization the expert is familiar with. Before considering other approaches, it is necessary to understand the intent behind the use of the dendrogram in that figure. The paper referred to is an artifact that provides additional context for understanding the work.)
7. Provide an interactive device for selecting cell line groups.

The visual interface and interactions are designed from the various analysis and design models.

Conceptual Model. Recalling the gulfs of execution and evaluation from action theory, the design goals should make operations available that are close to what the user intends to do and should make the state of the system visible in a visual language that can be easily understood. The graphical user interfaces of most applications communicate a world of digital objects on which actions can be performed; the interfaces embody concepts and relationships between the objects. The conceptual model is a model of the system that the designers hope the users will form as their mental model as they interact with the system. If the model reflects the users' world of work, it will be familiar. Designing the conceptual model involves asking the following questions of the task models produced in analysis (Johnson & Henderson, 2002; Johnson, 2008):

- What concepts should be presented to the users?
- What data will be manipulated, created, or viewed through the visible representations drawn on the display? What actions will be used to do this?
- What options, attributes, or parameters does the tool need to provide?

Applying this to the previous task description, Table 4.2 highlights in bold all the concepts that may need to be presented somewhere in the user interface

TABLE 4.2 Concepts Identified in a Task Description

Task-1: Group cells based on gene/miR expression.
1. **Cells** are grouped based on the **correlation** between the **expression patterns** of selected **genes**, **microRNA**s, or a combination of the two, and then visualized as a **dendrogram**.
2. (optional). **Filter** genes/miRs (rows of T) so that X% of the values have **expression levels** greater than Y; for example, 10% have **expression levels** greater than 7.
3. (optional). **Select** the *n* top genes with the highest **variability** in expression level, for example, the highest **interquartile range** or highest **variance**.
4. Calculate the 60×60 cell−cell **correlation matrix M** based on expression levels of genes/miRs from step 2. Provide options for **Pearson**, **Spearman,** and **Kendall correlation statistics**. There is no **missing data** for genes/miRs, so that is not an issue.
5. Using **agglomerative clustering** with $D = 1 - M$ as the distance matrix, construct a **dendrogram** of the **cell lines**. Provide options for **complete and average linkage**.
6. Display the **dendrogram** with options for coloring the **cell lines** based on **tissue type**.
7. Provide an interactive device for selecting **cell line groups**.

TABLE 4.3 List of Concepts from a Task Description

cells (biological), correlation, expression pattern, gene, microRNA, dendrogram, filter, expression level, select, variability, interquartile range, variance, correlation matrix, Pearson correlation statistic, Spearman correlation statistic, Kendall correlation statistic, missing data, agglomerative clustering, complete linkage, average linkage, cell line, tissue type, cell line group

when it is designed in a later phase. From the contextual inquiries done during analysis, the designers know that the datasets being manipulated in the task are one of two kinds that are generated from carefully controlled microarray experiments. One dataset contains the expression levels of genes, and the other contains the expression levels of microRNA. The task would be performed in step (h) of the data analysis process shown earlier in Fig. 4.3.

The list of concepts extracted from the description is shown in Table 4.3. Much of the terminology is from biology and statistics and is used to describe the analysis of microarray data.

The next step is to carry out an object/action analysis and ask which of these concepts to expose in the interface as *objects*, *actions* that the user can perform on the objects, *attributes* of objects (that can be set somewhere in the user interface), and various *relationships* between objects. Different kinds of relationships can exist between objects: in a *supertype*/*subtype* relationship, one object is a specialized type of another (vehicle/auto); in a *whole*/*part* relationship, one object is a part of another (car/wheel); and in a *containment* relationship, one object is inside another (document/folder).

An explanation of a full object/action analysis requires a background in data mining, statistics, biology, and the problem domain of chemical genomics,

TABLE 4.4 Results of an Object/Action Analysis

objects	gene dataset, microRNA dataset, gene, microRNA, dendrogram, correlation matrix, cell line, tissue type, dendrogram, variability
actions	filter, correlate, cluster, group
attributes	interquartile range, variance, Pearson correlation, Spearman correlation, Kendall correlation, agglomerative, complete linkage, average linkage
relationships	correlation method: Pearson, Spearman, or Kendall clustering method: agglomerative clustering linkage: complete linkage, average linkage microarray dataset: gene dataset, microRNA dataset

which is beyond the scope of this book, but an initial analysis might generate the results shown in Table 4.4 for just one of many tasks. The design challenge is to keep the conceptual model as simple as possible. Actions should be generalized to apply to as many objects as possible. For example, in step 3 of the task, "Select the n top genes . . . ," the selection is really another way to filter, so a separate action is not needed as long as the filtering action provides a way to select the "top n genes" using some metric for variability.

The conceptual model has several functions:

- *Identify and structure the concepts of the domain and the work before considering how these will be presented.* A conceptual model is needed to design the interface architecture, the next step of the process.
- *Serve as a reminder to software developers that the interface is a way to communicate with users*. It provides a vocabulary for the user interface. By creating the conceptual model from the task models, which are derived from discussions with users, the user interface will embody the concepts of the workplace rather than concepts software developers generate when they implement the system.
- *Measure the complexity of the interface*. The more objects, actions, and attributes that are added, the more the user will have to learn and the more combinations of objects and actions there are to consider. Each new action added could be applied to any of the objects that already exist in the model; similarly a new object could be operated on by any action that already exists. The complexity grows exponentially.

Not all design methodologies include conceptual modeling. Contextual design uses *visioning* and *storyboarding* to conceptualize alternatives for new ways to do the work by looking across all abstract tasks and issues extracted from the interviews to see what they have in common or where they differ. The storyboards provide the detail needed to either implement paper prototypes that are used to test the ideas with users (for small projects) or begin the design of the interface architecture (larger projects).

Interface Architecture. The objects, actions, and attributes that the system will make visible as tools and materials for doing the work have been identified in the conceptual model, but we are still not ready to begin sketching something that will be visible. None of these tools or materials have yet been organized. Just as physical work is done more efficiently when the tools and materials relevant to several tasks have been organized and laid out in different places, cognitive work is also done more efficiently if the visual representations of objects and actions are organized into *interaction spaces* where related tasks are performed. Interaction spaces in user interfaces are called windows, panels, dialog boxes, pages, views, wizards, and so on. If a related task cannot be done in the same interaction space, it distracts the user from the work and forces the user to think about an action sequence that will help navigate to a different space where the related task can be done. The organization and structure of actions and objects into interaction spaces along with the specification of links that show how to navigate between them is called the user environment (Holtzblatt & Beyer, 1998) or the interface architecture (Constantine & Lockwood, 1999). Like the architectural floor plans for a home, the interface architecture specifies the objects and actions that belong in the same space because they are frequently used together.

This is best illustrated with a commonly used application. Consider Gmail™ (also known as Google Mail). An application's interface reflects its internal structure. Three of Gmail's interaction spaces have been extracted as shown in Fig. 4.5. In each interaction space, the functions and objects needed for a set of related tasks are defined, as are the links to other spaces. For the system to be coherent, it not only must have a consistent user interface but also an orderly flow of interaction. Using an interface architecture to design the visual interactions results in a system where the interactions flow naturally with the work.

The analysis and design to this point have generated various abstract models, including two abstract design models: the conceptual model and the interface architecture. In the next stage of design, the abstract becomes concrete.

Visual and Interaction Design. To make the abstract concrete, the visual interfaces and interaction must be designed and prototyped. User interfaces and visualizations are composed from graphical elements (such as points or lines) with visual properties, or higher level elements such as predesigned or customized components that include controls (radio buttons, menus, hyperlinks) and containers (panels, windows, dialog boxes, pages). Graphics are composed of marks, scales, and guides organized by coordinate systems or layout schemes as described in Chapter 3. Although the interaction style could have been one of several different types, we assume interactions will directly manipulate visual representations of digital objects. Finally, cognition and perception are a factor to consider in whatever is designed. We will cover all of these topics in more detail in the last section "Visual Interaction Design."

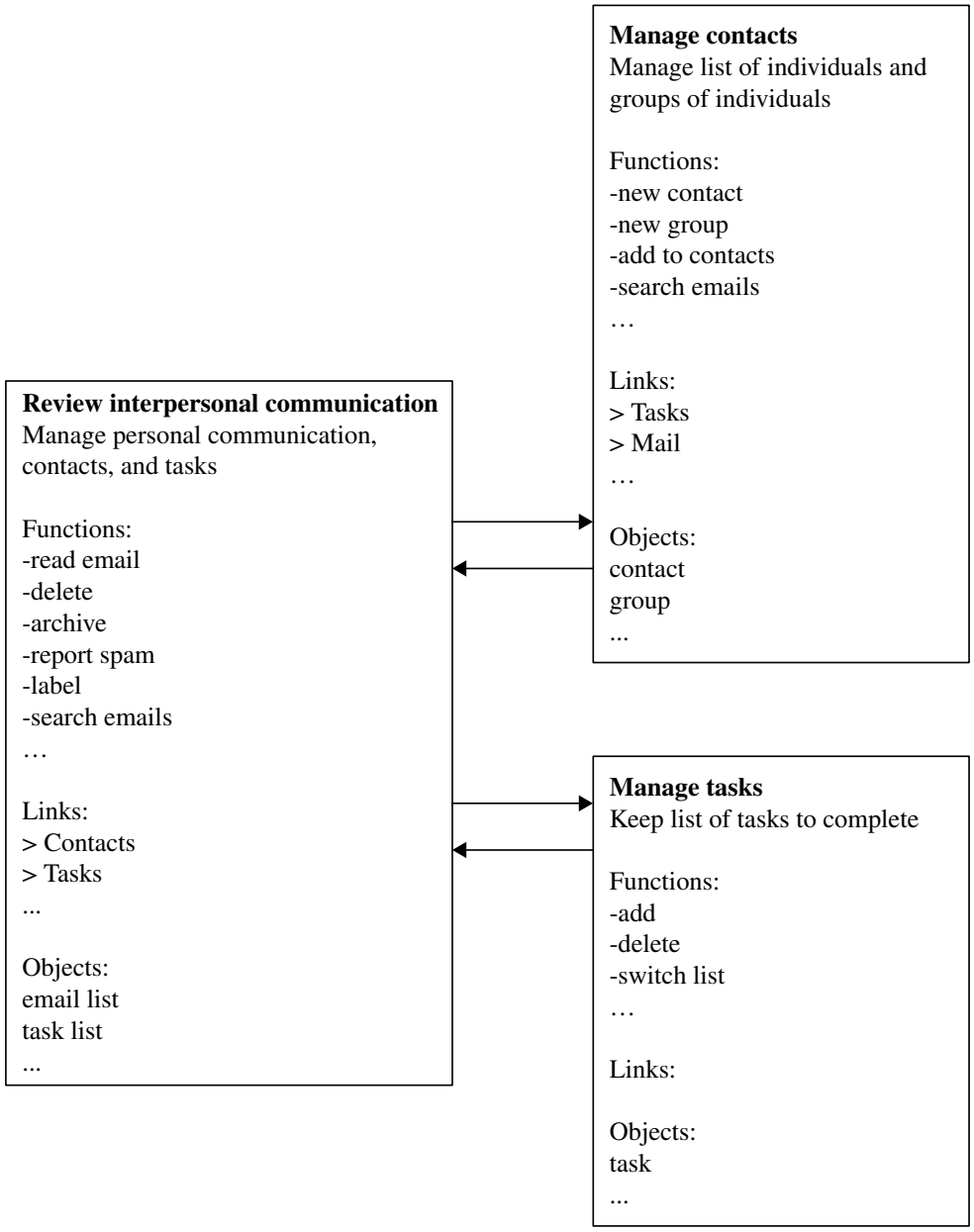

FIGURE 4.5 Example of an interface architecture for Gmail (see the User Environment Design model in Holtzblatt & Beyer, 1998)

The various work models, the conceptual model, and the interface architecture should be seen as a specification for the physical user interface design. Using these models as a guide, different ideas could be sketched as alternatives or implemented as low-fidelity prototypes (see the next section). The models keep the physical design decisions grounded in the data gathered from users and the workplace.

Note again that there is considerable back-and-forth activity between the abstract and concrete stages of design. Design is not only problem solving. Design is also "creating beyond what the problem calls for" to find a good fit to

the users' needs (Kelley, 1996). There is no hard-and-fast border between these two stages of design. Questions emerge in trying to design the details of how visually to support the work and in prototyping that will result in a reexamination of the conceptual model and the structure of the interface. Also, while we have presented the stages as if there is only one set of design models, the exploration of alternative designs might give rise to several sets of models, each of which may advance into the concrete stages of design and early-stage prototyping to determine whether they have merit.

4.2.3 Prototype

Prototypes are like the architect's drawings for a building. They are a language for communicating with the user about design alternatives and the details of interfaces and interaction. They provide a way to envision and experience how the new system will affect work, to explore technical alternatives, and to discuss flaws in the design and specific ideas for how it might be improved. Just as the architect's design moves from sketches of initial concepts to detailed drawings of floor plans and site plans to blueprints, the abstract design moves from low-fidelity to high-fidelity prototypes (or beta software).

Other ways to discuss design ideas with users include demonstrations of the visual interface, discussions about the written descriptions of requirements, and descriptions of scenarios. But ideas about visualizations and interaction are difficult to convey without something graphical to interact with. The first design alternatives of user interfaces are sometimes implemented as paper prototypes. Paper prototypes are one kind of *low-fidelity prototype* and, while there may be other choices, they have several qualities that should be present in whatever low-fidelity prototype is used:

- They are quick and inexpensive to build.
- Because they are hand-crafted and rough, discussions in early design will focus naturally on the structure of the system rather than on irrelevant interface issues such as the style of a particular user interface control.
- They are easy to change during the prototype interview in response to a suggestion or feedback.
- They can be used as a prop to discuss the details about what is needed, or what will or won't function in the work environment. Some issues do not surface until users need to reflect on the details of how something works.

Low-fidelity prototypes are sketches that include controls such as buttons, menu items, hyperlinks, or text fields. The user will simulate the actions of a real system by mentally pointing to controls or typing into fields. One or more interaction spaces may be mapped to a single "page," depending on the kind of user interface container selected. The sketch may be hand-drawn on a piece of paper with colored sticky notes simulating controls, or they may be digitally

drawn as slides in Microsoft PowerPoint® or as artwork created by Adobe Illustrator®. The digital sketches may be saved as a file of slides, as pages in a PDF file, or as HTML pages.

High-fidelity prototypes are working software with interfaces and interaction. They cost more to build and are less flexible than low-fidelity prototypes. They are used to investigate technical issues that might affect design. For example, some interaction styles require the system to provide very responsive feedback. To measure this, certain operations may need to be implemented to determine whether the time criterion can be met. Designers might implement all or some of the interface and interaction to be experienced but with limited functional capabilities. In *evolutionary* prototyping, the high-fidelity prototype eventually becomes a product. A high-fidelity prototype acts as a working prototype where core sets of functionality are developed in stages. The users continue to evaluate this prototype as it evolves. The outcome of this later stage of design is a proof-of-concept prototype that can be used to demonstrate the merit of the initial design concept. In *throwaway* prototyping, the high-fidelity prototype becomes the specification for a product and is eventually discarded.

4.2.4 Evaluate

The goals of a usability evaluation for design differ from the goals for testing usability of preproduct applications. Design evaluation is intended to provoke discussion about better ways to structure the system, discover unnecessary tasks, or reprioritize the importance of tasks. It is iterative and must be a fundamental part of the design process from the beginning. Prototyping may be thought of as a series of interviews similar to contextual inquiry but with a focus on observing design ideas in use.

Usability testing, on the other hand, is done in the late stages of development. It measures the users' performance on a set of predefined tasks. It is intended to uncover small problems or areas that are found to be difficult to understand. The changes made are to polish the interface and refine the product by improving the interaction. If the prototyping has been done well, there should be no major surprises. Usability testing is important but beyond the scope of this book.

4.3 VISUAL INTERACTION DESIGN

Physical design, whether it is of something tangible such as a fountain pen or intangible such as a visual system, requires many choices and trade-offs. How these are made depends on what matters. We react to the design of a product on three levels (Norman,2004):

- At the *visceral level*, which involves the emotional system and does not involve thought or consciousness, we respond to appearance and the

physical feel of the product. This is often discounted as just aesthetics, but the emotional state of mind is a factor to consider in design. Positive feelings improve creativity and broaden our thinking as well as helping us be more tolerant of minor problems in the tool; negative feelings from stressful environments make us concentrate and narrow our thinking. For complex or stressful environments, the design focuses on function and the removal of anything in the interface or interaction that is irrelevant.

- At the *behavioral level*, we respond to what the tool does, how well it performs, and whether it can be understood and learned. Designing for this level is primarily about function, followed by how to make it understandable.
- At the *reflective level*, we respond thoughtfully and consciously and reflect more deeply on past experiences, what we have learned, and the culture we live in. We consider the strengths and weaknesses, how it might be used in the work, and many other factors.

Visual tools are semiotic systems—communication-oriented tools designed for visual interaction. The graphical primitives and higher-level elements created from these primitives are deliberately designed signifiers that represent quantitative or abstract data, information, relationships, digital objects that can be manipulated, or actions that behave like tools. To understand their meaning, users must understand the connection between the signifier and what it represents. Well-designed signifiers are instantly perceived and can be easily "read." The signifiers used to represent the content and functionality of the system must be coherent—have consistent visual characteristics and style—and be understandable by the user community they are designed for. The signifiers in Fig. 4.6 are understood because they have been learned by using graphical user interfaces. A visual language is comprised of the visual properties (color, size, shape, etc.) of a set of signifiers that are related to each other by a set of rules.

Because "user interfaces," "visualizations," and "graphics" are terms loosely used in many contexts, we provide definitions for them here. We use the term "visual interface" (Mullet & Sano, 1995) instead of "(graphical) user interface". *Visual interfaces* organize content and tools so that users can efficiently do the activities or tasks that the system is being designed to support. They have *controls* that provide ways to perform actions and *containers* that provide space—regions of the display—for either content or groups of controls. To support the activities of users, the design captured in the various models produced by the analysis of the work must be translated into a physical structure of windows, pages, dialog boxes, and so on, along with ways to

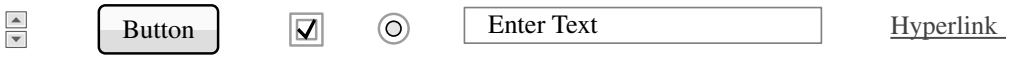

FIGURE 4.6 The controls of a user interface are signifiers for actions

navigate between them. (Note that Web applications refer to a physical interaction space—a container of content or controls—as a "page," and desktop applications refer to these spaces with terms such as "windows" or "dialog boxes." We will refer to all kinds of containers simply as pages.) Various interface schemes for doing this are discussed in the upcoming "Visual Interfaces" section. *Visualizations* communicate information using graphical representations. Examples of visualizations include charts, diagrams, tables, guides, directories, and maps. *Graphics* (Wilkinson, 2005) are the visual representations of graphs derived from statistical data. Visualizations and graphics may both have visual interfaces that allow the user to interact with the abstract or quantitative data they represent. (Note that "graphic design" is used to describe the art of communication used to create visual messages—advertising—that educate, inform, promote, and persuade people to buy products. The principles of graphic design are used in visual interfaces and in graphical design.)

The physical design of visual systems that combine visual interfaces with visualizations and graphics must consider several dimensions to know how the users will interpret the messages communicated by the system. The first three subsections discuss the guidelines, principles, and design patterns of visual interfaces, visualizations, and graphics. Each subsection includes comments about perception and cognition that constrain the design. The final section discusses real-time constraints imposed by cognitive and perceptual processes.

4.3.1 Visual Interfaces

Visual interfaces organize content and tools so that users can efficiently do the activities or tasks that the system is being designed to support. When we open an application, the first thing we perceive is its visual interface. One of the first steps that must be taken is to translate the interface architecture into a physical structure of pages.

Organizing the Application. Many different physical structures can be designed, but those commonly found in general applications are variations of one of the three schemes shown in Fig. 4.7. Multiple and tiled windows are found in desktop applications, while single-paged windows are found in browser-based applications.

In choosing a scheme, it is important to be aware of the limitations imposed by human attention. The brain has several mechanisms for attention and can make only a handful of items available to cognitive processes (such as problem solving). These items are indexed by our perceptual systems and whatever long-term memories have been activated by the focus of attention. After our attention shifts, this set is replaced by another, and the first set is forgotten. Tasks may require that something in one panel be referred to while content in another panel is modified or that a set of objects in one panel be compared with the objects in another. Interrupting the focus by requiring the user's attention

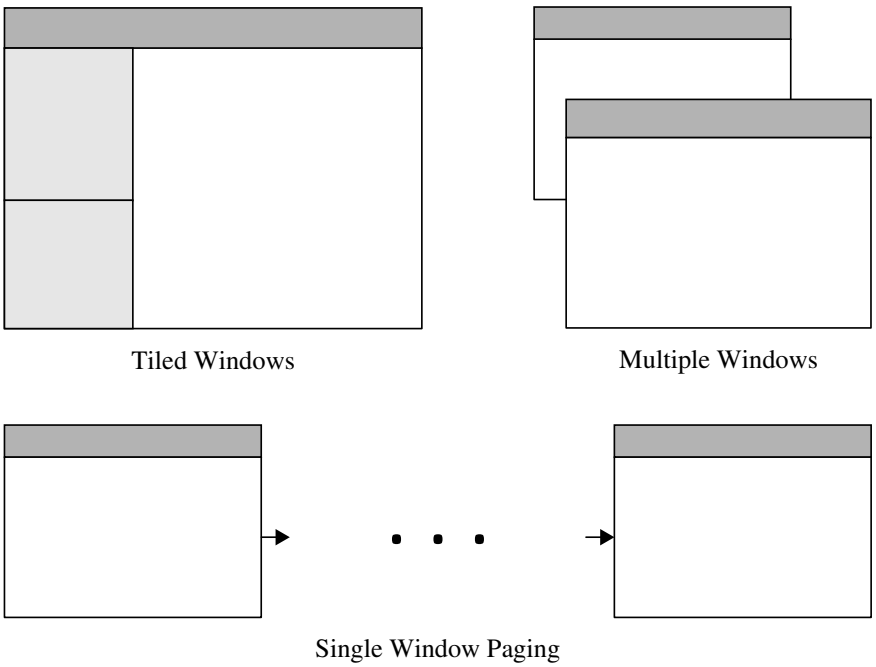

FIGURE 4.7 Different schema for physical structures of a user interface

to move elsewhere disrupts the flow of thought. The data and the kinds of tasks to be performed determine which physical structure to use.

Navigation. In complex applications, not all the functionality can be accessed from a single page. The interaction spaces of the application may be allocated to a collection of pages with various tools and views of the underlying information or data on each page as shown in Fig. 4.8. But the user must know how to get to where they need to be to perform the sequence of actions that pertain to the goal—and how to get back. This is the problem of navigation that requires organization and navigational aids to allow users to move between interaction spaces.

Navigation has a cognitive cost. Each transition to a new page is a switch in context requiring the scanning of that page to understand its structure, content, and exits. The ideal is to not require navigation—to have all the content and tools directly accessible—but the trade-off is complexity. And, in many cases, direct accessibility is not possible because there is simply too much information (or too many different actions) to design them to be all in one physical space. Other constraints result from perception and cognition.

The visual system is cued by the goals of the task. We notice things that match the goal and often don't "see"—become aware of—what is irrelevant. Further, the brain stores information in long-term memory by activating large numbers of neurons that effectively distributes what is being perceived as patterns across memory. Recognition is easier and faster than recall. When we see something familiar, it activates overlapping patterns with what has already been stored, which allows for quicker access to long-term memory. Recall

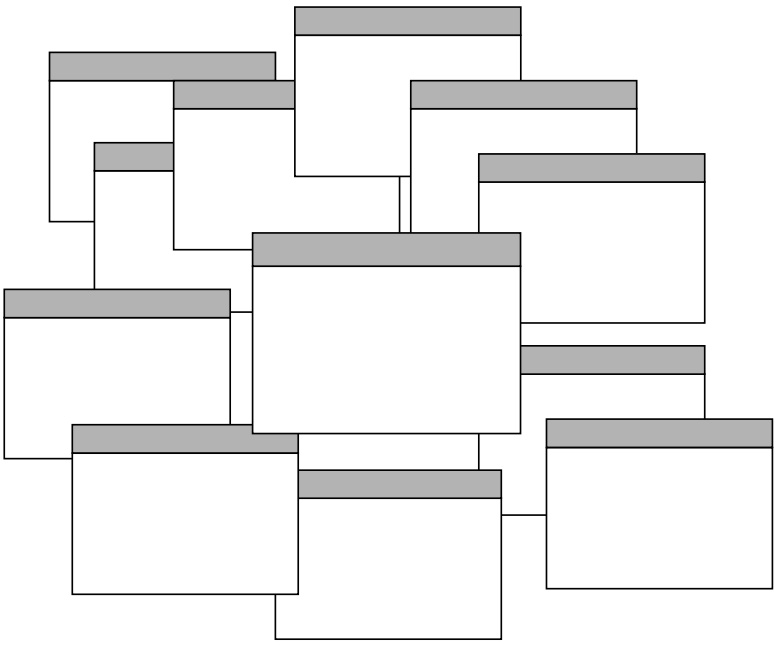

FIGURE 4.8 A complex application structure that requires organization and navigational aids

results in a slower search through memory. We memorize and prefer familiar paths that let us accomplish a task, even if there are other ways it could be done more efficiently.

Designers use signage, maps, and clues to help users know what path takes them to the place where the next required action can be performed and to keep them from getting lost. Uniformly organized, consistently placed, clear, unambiguous navigational signifiers act like highway signage (e.g., mile markers on highways give an approximate location); and when decisions must be made, overhead signs announce major intersections, and large, clearly labeled exit signs provide the name of the town or number of the highway you will be entering. The design of the interface architecture will already have functionally structured the application so that related tasks are in the same interaction space. In the mapping of the spaces to the physical structure, the goals at each decision point must be kept in mind. Fig. 4.9 shows some of the navigational signage for Gmail. The navigational structure is uniform and remains constant throughout navigation. The global map provides clear entry points to major activity areas of the application, and the local map provides entry points to interaction spaces within an activity area. The content—but not the placement—of the local map changes when moving from "mail" to "contacts."

Because of the cost of switching contexts, designers try to minimize the steps in the path. The interaction architecture is critical in ensuring that common and frequently performed tasks do not require movement through several interaction spaces. In the local maps of the mail and contact interaction spaces of

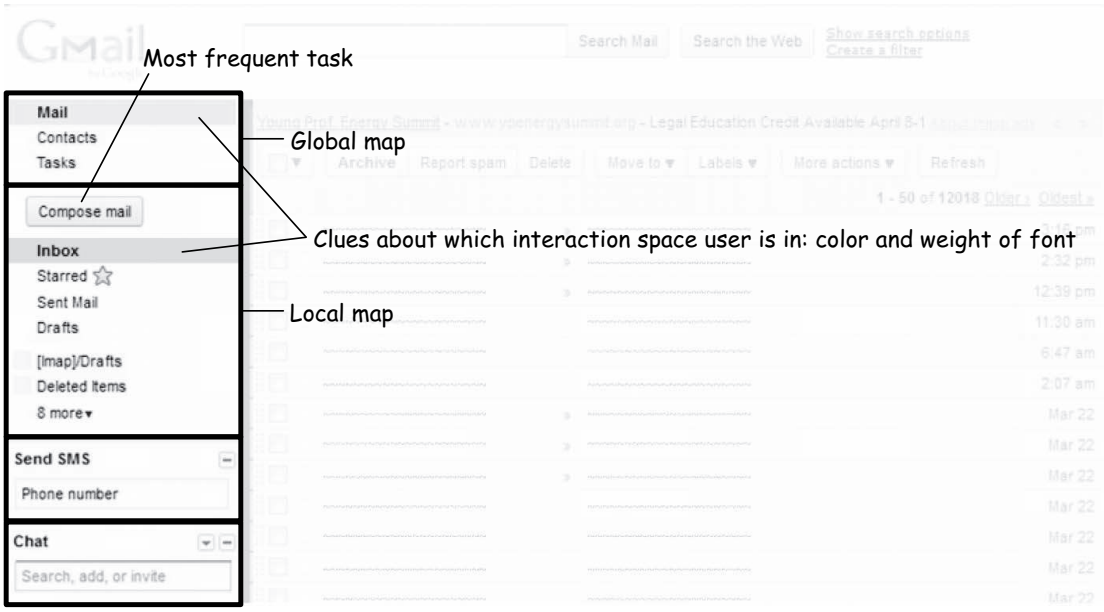

FIGURE 4.9 Navigational signage for Gmail

Gmail, for example, a button is provided for the most frequently performed tasks: "Compose Mail" (shown in Fig. 4.9) and "New Contact" (in the contact interaction space that is not shown), respectively.

Organizing the Page. As was discussed in Chapter 2, more information is in a scene—or a page in the context of an application—than can be perceived at one time. The visual system uses various search strategies to sample and scan to find what is relevant to the current goal. We perceive the fine detail of a page only through the fovea, a very small region near the center of the retinal visual field. The peripheral vision has such low resolution that its function is to primarily provide cues that guide eye movements toward what is interesting: motion, fuzzy shapes, brighter colors, or large features. The eye is optimized to see structure, and designers exploit this to convey meaning, establish a sequence for the eye to follow, and create focal points of interest. Page layout uses techniques from graphic design to create a *visual hierarchy* of the content that gives weight to what is most important and a *visual flow* that leads the user through the page.

A visual hierarchy structures the content. The user should be able to infer the visual hierarchy of a page from its layout. Titles should be apparent. The most important content should be prominent, and less important content should appear as secondary regions. Techniques that can be used to create a visual hierarchy include the following, some of which are illustrated in Fig. 4.10:

- *Upper-left or upper-right corner*. The eye will begin to scan a page in the same place that it does for reading the text of the primary language. In Western cultures, the scan will begin in the upper-left corner.

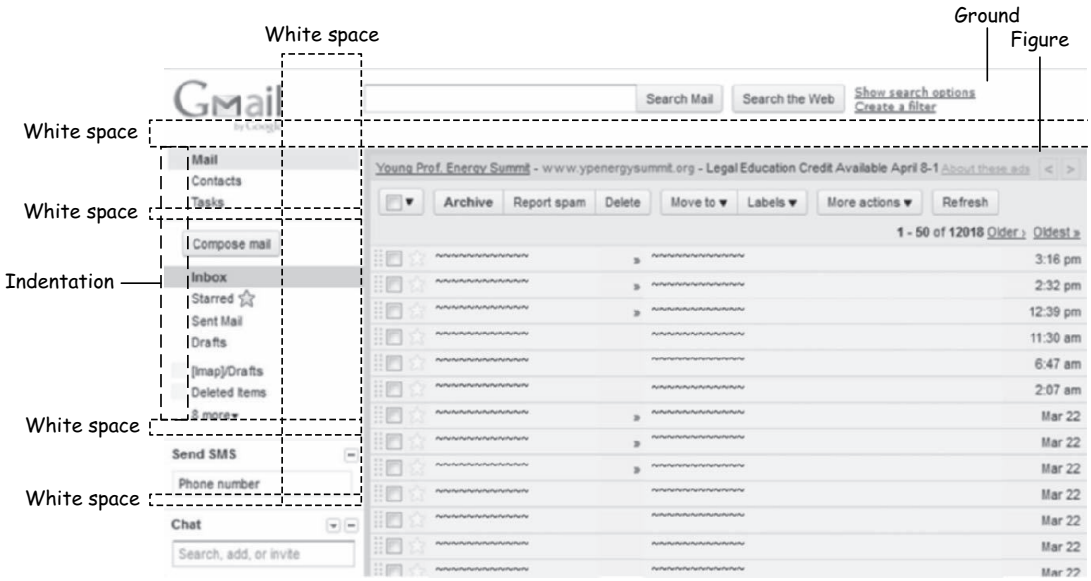

FIGURE 4.10 Techniques for creating a visual hierarchy

- *White space*. White space is an important element in constructing a hierarchy, and its uses include separating regions or establishing hierarchical order, as shown in the example.
- *Size and weight (boldness) of fonts*. The font of more important information is given a larger size or greater weight. In the example, "Mail" and "Inbox" are emboldened to identify which application and which subset of email threads are being viewed. "Send SMS," and "Chat" are titles of major sections.
- *Contrasting colors for* figure *(the foreground) and* ground *(the background)*. In the example, the white ground and the gray figure (or blue if being viewed in a Web browser) separates the navigational and search areas from the content. A darker shade of gray is the ground for the email content activity area. Lighter shades of gray separate the actions that can be performed from the email headlines that comprise the content.
- *Positioning*. In the example, the large "Gmail" lettering placed in the upper-left corner will be seen first in Western cultures. It informs the user of the application in use, and it also ensures that the branding of the application won't be missed.
- *Alignment*. Alignment is used to show a set of related items. "Mail," "Contacts," and "Tasks" are different activity areas, and "Inbox," "Sent Mail," and so on are different subsets of email threads.
- *Indentation*. Indenting text implies that it is subordinate to what is above. In Fig. 4.10, all the hyperlinks from "Mail" to "Deleted Items" are indented to show their relationship to the major activity area for email as opposed to texting ("Send SMS") or chatting ("Chat").

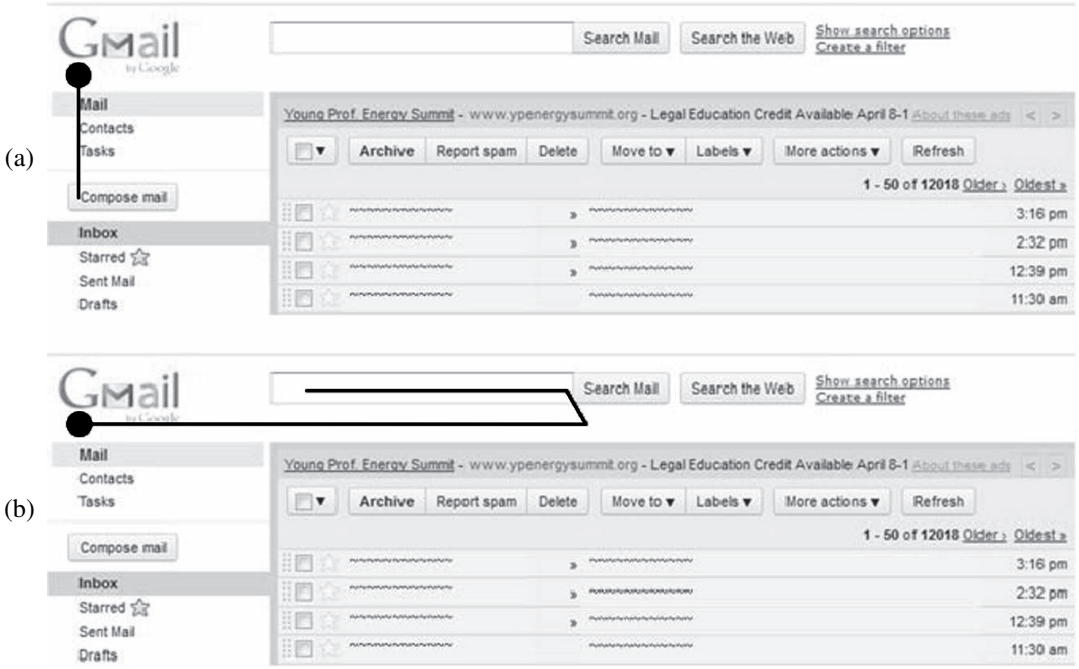

FIGURE 4.11 Possible visual flows for composing or searching mail

Visual flow is designed to manipulate the path the eye takes as it scans the page. A well-designed visual hierarchy incorporates focal points—graphical features that emphasize important elements—that lead into secondary regions of less important content. The design of focal points relies on the way the visual system processes certain graphical images. Recall from Chapter 2 that when attention is not focused on something specific, visual features with certain properties appear to "pop out" given the right conditions. Graphic designers use visual properties—lightness, color, orientation, texture, shape, size, and motion—combined with other techniques to attract the eye. (Well-designed advertising in high-quality publications provides good examples of the techniques.) The techniques include the use of white space, contrasts of color and lightness of shapes or weight and size of fonts, and converging lines or hard edges. Your gaze follows the focal points from strongest to weakest. The focal points can be overridden by the goal of the current task, the meaning in content we see, or the natural tendency to scan a page as if we were reading text. Fig. 4.11 shows possible visual flows for composing an email (a), and entering a search query for an email (b). The large, appropriately labeled buttons provide a focal point for actions that can be performed.

Visually grouping and aligning content elements indicate that they are related to each other. As shown in Fig. 4.12, four methods based on the Gestalt principles (discussed in Chapter 2) of perceptual organization are used to convey which content elements are related: similarity, proximity, continuity, and closure. In all of the methods except the one using similarity, the use of white space provides clarity by separating the clusters of related items. These

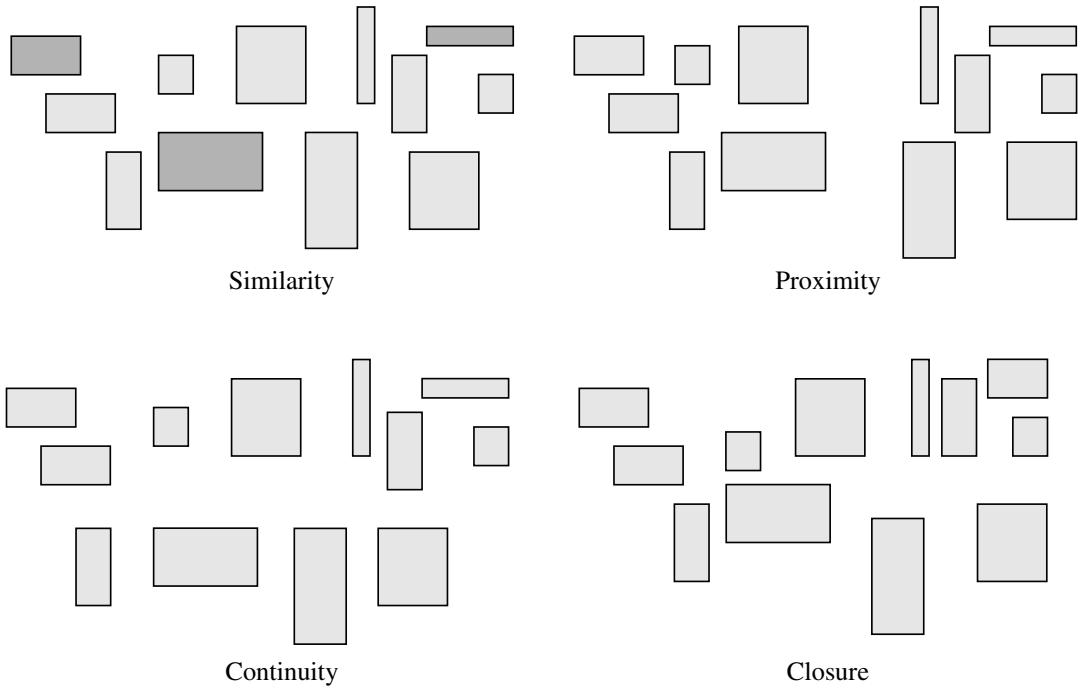

FIGURE 4.12 The four Gestalt principles used for grouping and alignment

methods are used in Gmail, for example, to group the set of buttons that provide ways to manipulate the email threads, a set of related hyperlinks that signify folders, and the search button with its query input field.

Organizing the Actions. The design of the interface so far has focused on how to create the physical structure for interaction spaces and how to present the content so that it is informative and can be quickly perceived. But to make the system fully interactive, it must be capable of taking input from the user. Although the interaction style could have been one or more of several different types (instructing, conversing, manipulating, and exploring), we will focus on what are called *direct manipulation interfaces* that are prevalent in advanced graphical user interfaces.

The first method by which we can invoke actions in the system is through visible things in the interface which we will call *visible objects*. Visible objects are controls and visual representations of digital objects that are capable of providing feedback when they are pointed at or prodded (clicked) by the mouse. The controls and visible objects are signifiers of actions that the system is capable of performing. But when the user first looks at an interface without moving the mouse, such as Gmail's interface in Fig. 4.13, how are the action signifiers discovered? And what do they do? Some of the possible action signifiers in the Gmail interface have been enclosed in black rectangles, and all but one represents an action.

Some signifiers are recognized immediately by convention and experience. Users have learned that buttons (rounded rectangles), hyperlinks (underlined

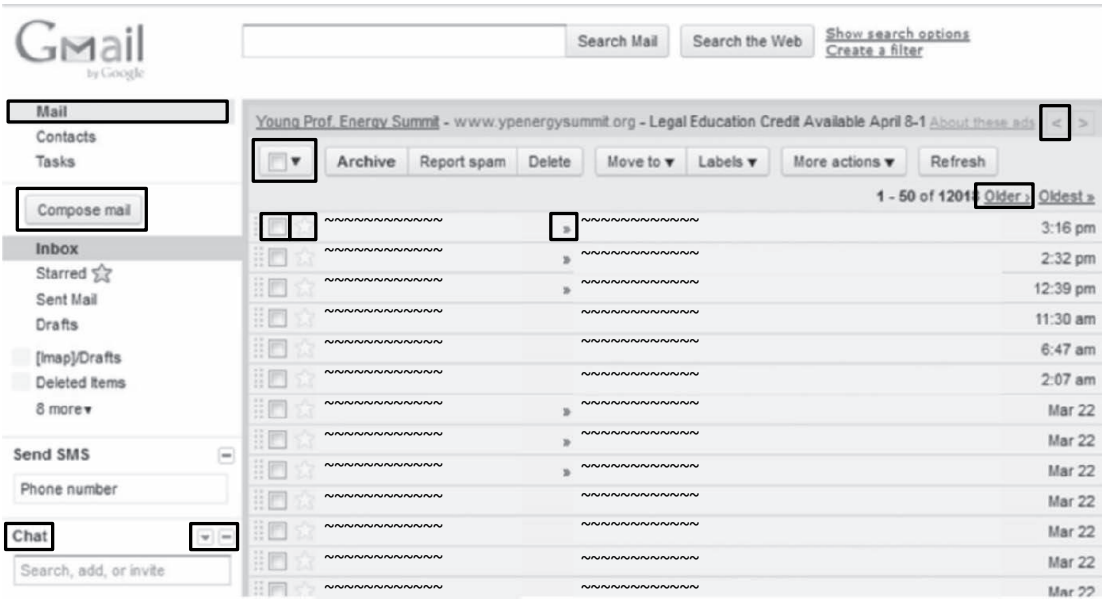

FIGURE 4.13 The possible action signifiers in a Gmail interface

text), and menus (buttons that contain upside-down triangles) are actions. These are easily identified in the Gmail interface. Style guides written for application developers for each computing platform prescribe the visual appearance and behavior of a standard set of controls. The controls in these guides should be the first source considered as signifiers of actions because the way they look and how they behave will be familiar to users.

Controls or visible objects are also designed to change appearance when the mouse rolls over the region in which they are drawn or if the state of the digital object changes when the action is invoked. In the Gmail interface, the borders of buttons are darkened, the background of the folder labels changes color, and for all controls but the checkboxes in the email, the mouse pointer symbol changes from an arrow to a hand. Selecting an email thread by clicking on the checkbox results in a change in color of the background for that email thread. The star action is an exception to these guidelines. It provides no visible feedback unless the user clicks the star. There is no way to tell it apart from the chevron (“>>”) in the email thread headline which is not an action.

The visible action signifiers—buttons, hyperlinks, and menu items—provide a way to learn the actions available in the system. Users will often explore these to see what is available. There are also invisible actions that cannot be discovered: combinations of keystrokes, drag-and-drop operations, and double-clicking or right-clicking on visible objects. Users often expect these actions and have learned them from outside the interface.

A variety of organizational strategies are used to group actions. These include menu bars, tools bars, and ribbons. The controls in toolbars, such as the one from Gmail shown in Fig. 4.14a, are always visible and directly accessible. In complex systems, when the number of actions is large enough that

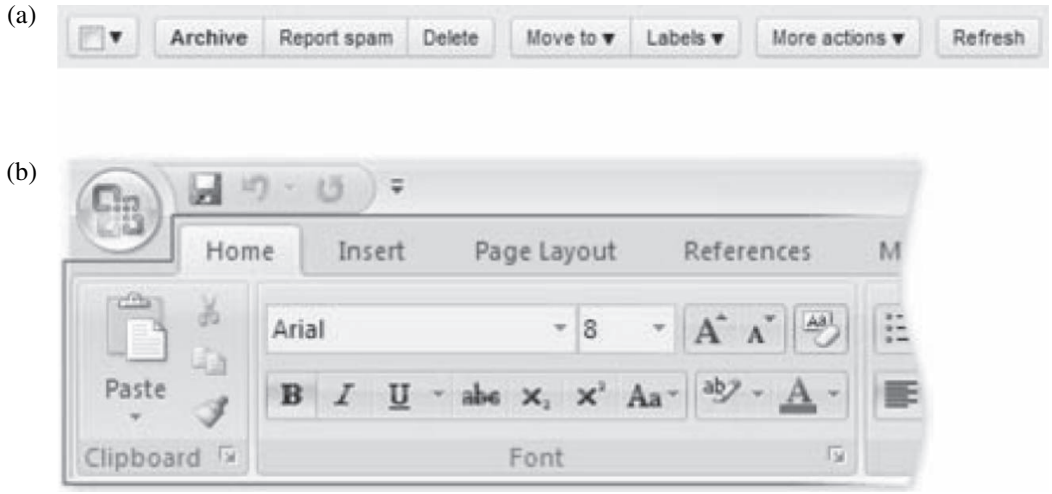

FIGURE 4.14 A simple toolbar from Gmail and a ribbon from Microsoft Word®

menu bars become cumbersome and toolbars lack sufficient physical space to display all the controls, other organizational strategies are required. A ribbon, as shown in Fig. 4.14b, is an approach that combines aspects of the toolbar with menu bars. Groups of related actions commonly used together are available as a toolbar, but the toolbar can be switched to another for a different set of controls.

4.3.2 Visualizations

> "All communications between the readers of an image and the makers of an image must now take place on a two-dimensional surface. Escaping this flatland is the essential task of envisioning information—for all the interesting worlds (physical, biological, imaginary, human) that we seek to understand are inevitably and happily multivariate. Not flatlands." (Tufte, 1990)

If the first challenge of designing interactive visualizations of complex information is how to project multidimensional abstract data into a two-dimensional space without losing its richness, the second is how to visually compress or find our way through the quantity—millions of points—of data. The third challenge is to effectively link it to the growing amount of related information.

The information being visualized is *abstract*. It is often categorical or structured data that contain attributes or properties about abstract objects that have been modeled such as biological genes, chemical compounds, documents, or financial transactions. From these data objects, secondary data can be derived that is used in data mining to cluster, classify, or find associations or other relationships. For example, descriptors can be derived from chemical compounds that summarize the number of ring systems, chains of various types of atoms, or other chemical or topological features in each

compound. The descriptors are used in clustering algorithms to group similar compounds. The emphasis in visual analytics is often on exploration: the discovery of patterns, relationships, clusters, outliers, trends, or gaps. The goal of visualization design is not to eliminate complexity but to present and interact with uncluttered images of the data where complexity can be seen alongside the detail and where comparisons can be made.

One place to learn about visualization design is from the past. Information design for presentations on paper or in print has been evolving since perspective drawing was invented as a way to draw physical objects. It has given rise to the following methods (Tufte, 1990) that are a starting point for thinking about how to present dense, complex, multidimensional information:

- *Micro/macro*. Micro/macro drawings use a design strategy where fine detail is added not only so it can be seen but also so it can be used to form the overall structure as shown in the 1739 Bretez-Turgot *Plan de Paris*. The detailed map of Paris, shown in Fig. 4.15, was drawn as 20 sheets. When a sheet is viewed from a distance, the detail blurs and becomes part of the texture of larger surfaces as shown in Fig. 4.16. But up close, as

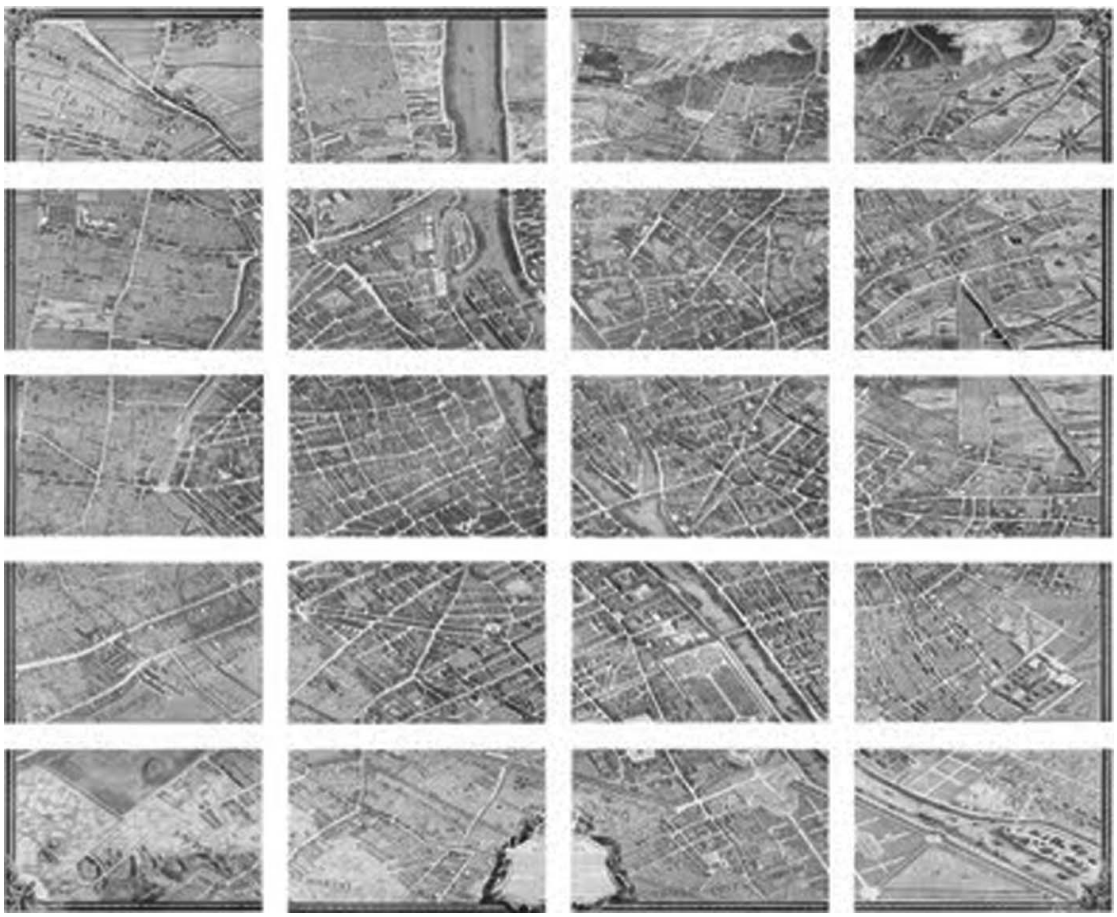

FIGURE 4.15 The 20 sheets of the Bretez-Turgot *Plan de Paris*

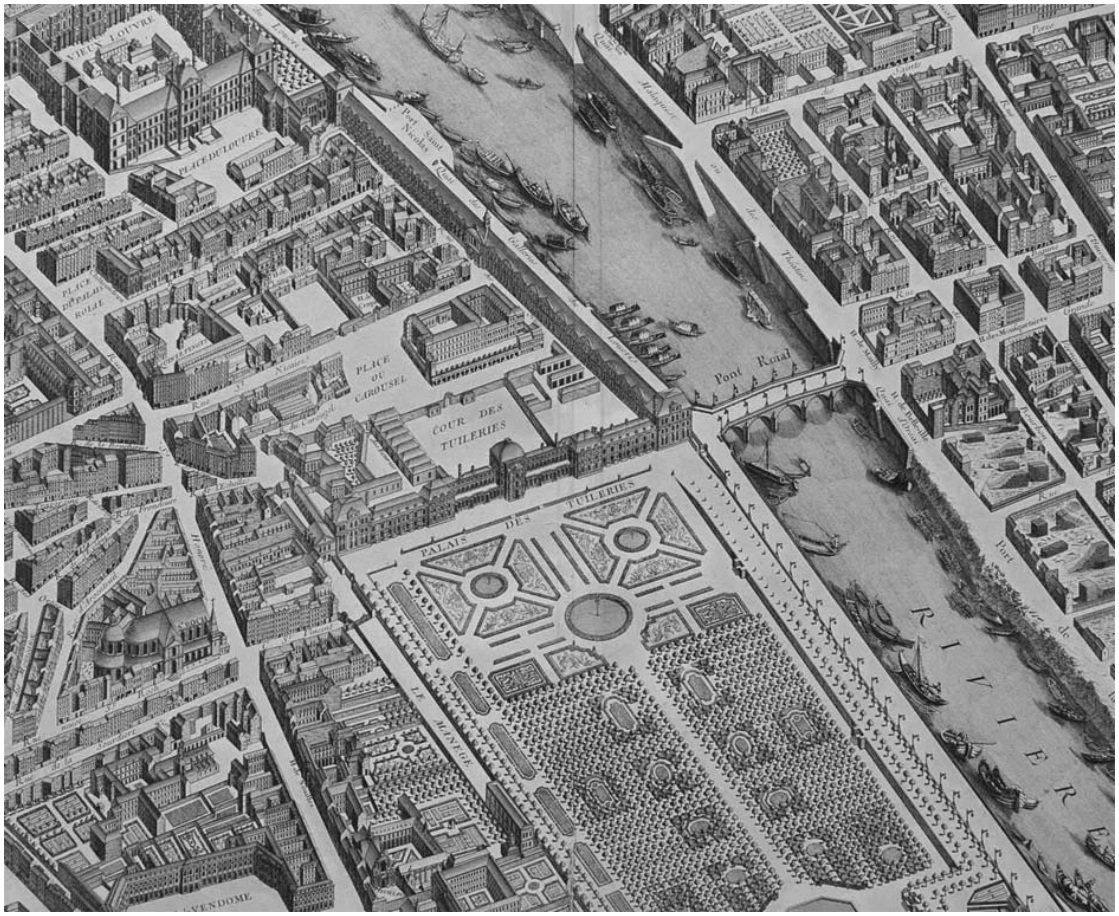

FIGURE 4.16 Sheet 15 from the Bretez-Turgot *Plan de Paris*

shown in Fig. 4.17, the details of each building and its immediate neighborhood can be seen: architecture, doors and windows, nearby buildings, and street names. The complexity is not eliminated but is controlled by organizing the information into multiple layers of context. In addition to detail becoming texture, labeling has been added to the streets, rivers, and even the roof of a hotel building to provide landmarks. Just as the design of the interface architecture brings together related tasks to avoid the cognitive costs of interrupting the flow of a task, allowing detail to be seen in its larger context avoids the cost of switching to a different image. By not stripping the complexity from the presentation, the viewer, rather than the designer, decides what information is important to see.

- *Layering and separation*. This method controls complexity by visually separating the data into strata. We have already seen one example of layer and separation through the labeling of streets, building, and rivers in the *Plan de Paris*. Another example is to use color, such as red and black, to separate annotations from data. Shape, value (light or dark), size, and color can be used to separate and layer information. Layering can be difficult to achieve because of unintended interaction between the

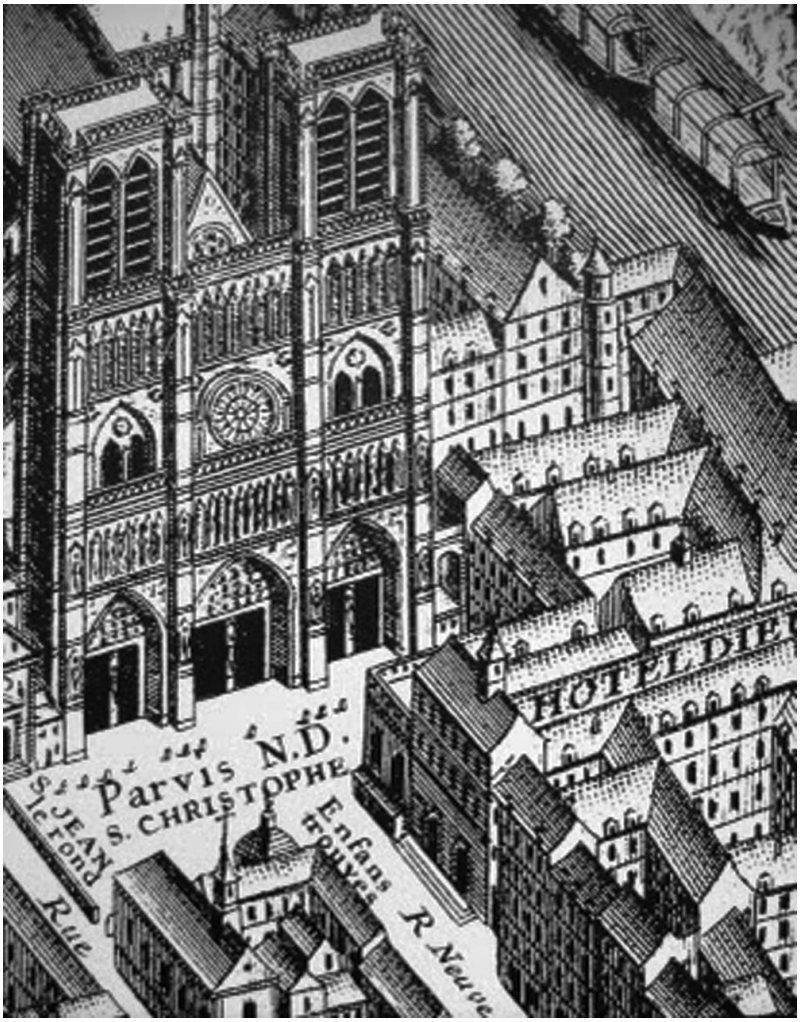

FIGURE 4.17 A section of a sheet from the Bretez-Turgot *Plan de Paris*

graphical elements. The relationships must be in the right proportions and consistent with the data being represented.

- *Small multiples*. Small multiples are repetitions of the same design structure over slices of the data. Fig. 4.18 shows an example of a small multiple for unemployment rates in specific industries over 10 years. After the first element is understood, the remaining elements can be quickly read. The small multiples show all the data and make it easy to compare changes across elements without switching contexts. How much detail to include depends on the level at which the data is being viewed.

Interactive visualizations developed since the advent of high-performance computer graphics have extended the methods in the preceding list in different or new ways. These enable users to do the following:

- *See information at multiple levels of detail*. A variety of interface schemes has been developed to provide capabilities similar to the micro/macro

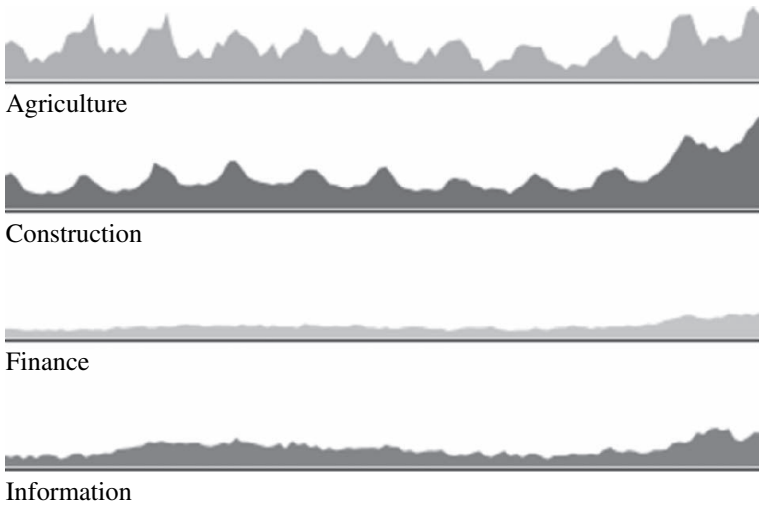

FIGURE 4.18 Example of small multiples of unemployment rate by industry over 10 years

readings of the data. These are discussed later in this section. Interaction techniques can also be used to provide additional information. For example, datatips can provide a quick summary about a visible object as the mouse pointer rolls over the object.

- *See various relationships between the items*. Relationships can be represented using the Gestalt principles of similarity, closure, proximity, and continuity, or by connected lines. Immediate highlighting of visible objects can be used to identify individual objects among many that have been selected by some criteria.
- *Filter irrelevant information*. Dynamic queries allow visible objects being displayed to be hidden or visibly changed in response to the movement of controls—sliders or checkboxes—that change the parameters of a filter action. The immediate feedback provided within a few milliseconds allows users to rapidly sift through information. Data brushing allows visible objects selected in one view to simultaneously be highlighted in other views that are linked to that view.
- *Create subsets of the information*. Visible objects can be selected, extracted, and exported into other applications or saved as files.
- *Sort or rearrange the information*. Specific attributes may be used to sort rows or columns in a table or rearrange the layout of visible objects. Sorting alphabetically, numerically, by date or time, or categorically are common. The use of statistical and data-mining algorithms, such as clustering by different distance metrics, may also be used to rearrange the data in a table.

The abstract stages of the design process for visualizations are similar to those described previously for user interfaces: understand the users, the nature of the exploration tasks they will perform, and the data. Designing the physical

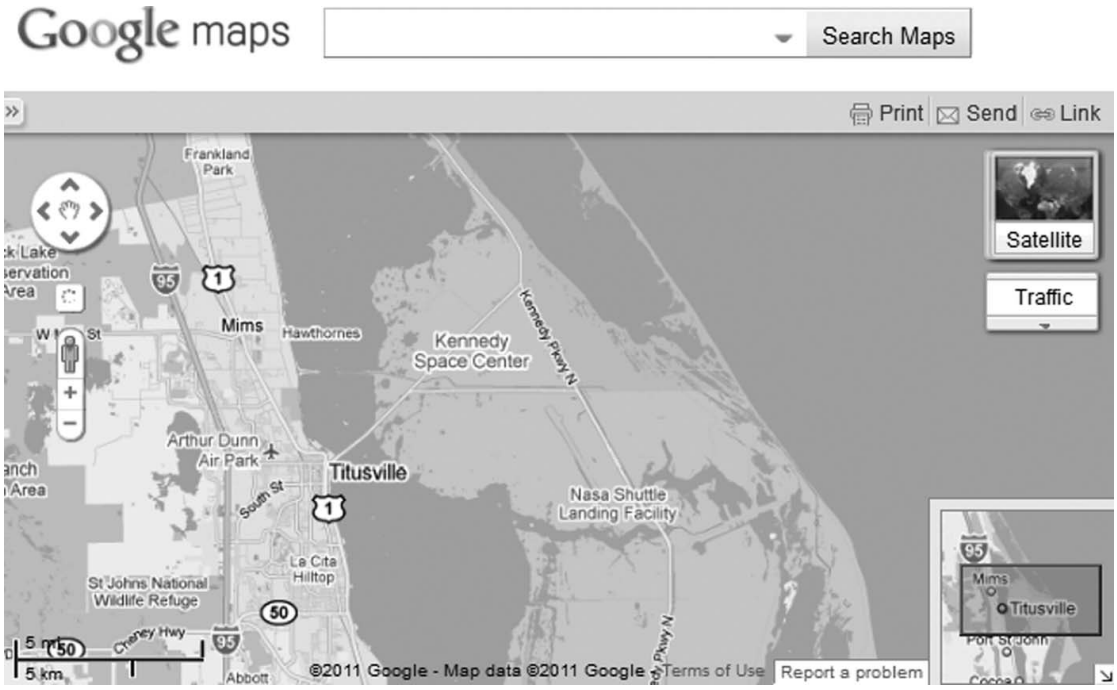

FIGURE 4.19 Google Maps provides overview + detail and zooming interface schemes

structure of an interactive visualization is often guided by a well-known mantra: "Overview first, zoom and filter, then details on demand." (Shneiderman & Plaisant, 2010) Several interface schemes incorporate the mantra and allow users to work with and move between focused and contextual views of a dataset. Some of these separate the views spatially into panels as in the tiled windows scheme described for user interfaces. Others separate them temporally through animated transitions. These schemes include the following (Cockburn et al., 2008):

- *Overview + detail (spatial separation)*. This scheme simultaneously shows an overview and a detailed view of the underlying information space. The user interacts separately with each view, but the views are coordinated so that what changes in one is reflected in the other. The overview provides context for what is seen in the detail view. Google Maps™ is an example of this structure as shown in Fig. 4.19.
- *Zooming (temporal separation)*. This scheme uses a single window that allows the user to zoom in on the dataset. Zooming controls increase or decrease the levels of scale, and the resulting changes are done in place so the views cannot be seen simultaneously. Google Maps with the overview insert hidden is also an example of this approach.
- *Focus + context*. This scheme integrates the focus and contextual views into a single seamless view. All of the content is visible, but the area within or near the focus is distorted to provide more detail as shown in Fig. 4.20.

FIGURE 4.20 Focus + context interface schemes distort to show detail in context. (Fisheye Menu courtesy of HCIL at University of Maryland: www.cs.umd.edu/hcil/fisheyemenu)

4.3.3 Graphics

In Chapter 3, the discussion of graphical design focused on how the choice of graphic schemas and the mappings of graph dimensions to variables affected the ability to create an image of the graphic that could be efficiently "read." Programming or scripting languages and libraries such as ggplot2 may be useful without knowing much more about the design of a graphic. Defaults provide most of what is needed to generate reasonably good graphics that can be used for exploration. However, grammar-based toolkits such as Protovis expect the designer to compose the elements of a visualization. Much more must be understood about design if the graphics are to communicate the data clearly, efficiently, and honestly. Quantitative graphics are visualizations that have been evolving since the 1700s. This section touches on only a few of the important principles and ideas that have been discussed elsewhere about

graphical design. The "Further Reading" section refers to other books and articles on the subject. We assume that the graphics will be designed for interactive exploration rather than for publication and that the design may evolve in the prototypying stage.

Displaying Data. Tufte's fundamental principle is "above all else, show the data." A graphic is a drawing about numbers. Even if complex, it is worth studying carefully if it allows us to see something in the data that would be harder or impossible to see without it. It should portray the data clearly and accurately, invite comparisons, and contain information that is relevant to the task. The form should be compatible with the underlying data. For example, a reference curve should not be used for integral or categorical data. As in micro/macro visualizations, the data should be accessible at several levels from overviews to fine detail.

The graphic in Fig. 4.21 is a scatterplot annotated with terminology that will be used in the following discussion. Tufte defines *data-ink* as "the nonerasable core of a graphic, the nonredundant ink arranged in response to variation in the numbers represented." The *data-ink ratio* is the proportion of data-ink to the total ink used in the graphic.

data-ink ratio = amount of data-ink / amount of total-ink

One of Tufte's principles is to maximize the data-ink ratio. (Note that more recent research on graphic reading argues that this is not always the case—see Kosslyn 2006; p. 13). Because data-ink is essential, to increase the ratio, we must erase what is not essential: *nondata-ink* or *redundant data-ink*. An example

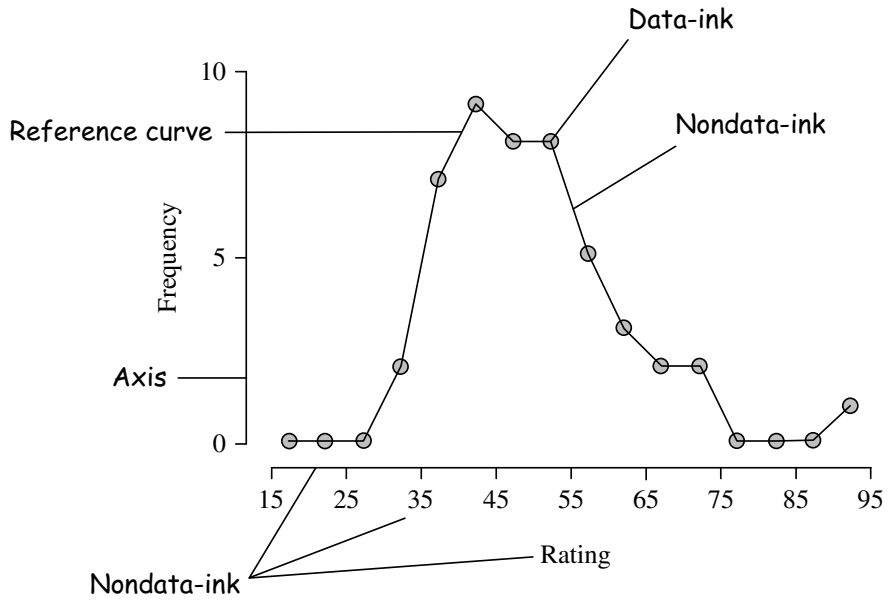

FIGURE 4.21 A scatterplot annotated with nomenclature

of redundant data-ink is a histogram bar that has its height labeled just above the bar. The number it represents is redundantly represented in at least two ways: by the height of the bar and by the label. A designer of graphics, like an editor of text, should remove what is unnecessary. The question should be asked of whatever is drawn: does this provide new information?

For clarity, the content area should be kept free of clutter. Too much information can make it difficult to see and understand the data. Labels should not interfere with the marks. Keys and legends should be kept outside the axes, and the marks should not overlap the axes.

Displaying Nondata: Scales and Grids. Grids, axes, reference curves and lines, and other accoutrements are intended to aid understanding. They are not the subject of the graphic and should be given less importance visually by being drawn in muted colors with thin lines as shown in Fig. 4.22b or, even better, erased whenever possible.

There are a number of guidelines on the use of scales, including the following:

- The minimum and maximum limits set on the scales should be chosen so that the marks fill the content area as much as possible. If multiple panels are being used to compare data, as in small multiples, both horizontal and vertical axes should be consistent. The design structure of each panel should not change. (Cleveland 1994)
- Use understandable rounded numbers for tick marks (Unwin, 2008). *Nice numbers* are familiar numbers learned in arithmetic that make mental calculations easier to do. A *nice scale* is an interval scale of numbers where the differences of the first two numbers in the scale are members of the set { . . . , .1, .2, .5, 1, 2, 5, . . . } (Wilkinson 2005). The following are all nice scales that can be used to label tick marks when appropriate:

```
{ . . . 1, 2, 3, 4, 5 . . . }
{ . . . 2, 4, 6, 8, 10 . . . }
```

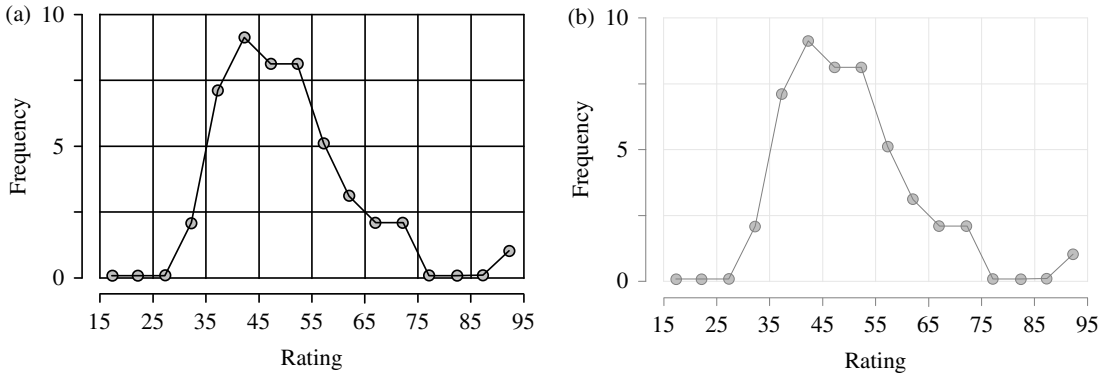

FIGURE 4.22 Creating a visual hierarchy to emphasize the data

```
{ . . . 0, 50, 100, . . . }
{ . . . 0.001, 0.003, 0.005, 0.007 . . . }
```

- There are times when it is helpful to have different scales on top and bottom, or left and right.
- Showing data on a logarithmic scale is useful when the range of values is large, but the axis label should call attention to the use of a nonlinear scale.
- Not including zero does not necessarily distort the information (Cleveland, 1994), although this is a subject of debate.
- Avoid the scale breaks that are sometimes used in charts when there are large gaps in the data values. The mind interprets uninterrupted space as continuous, and it is difficult to perceive it otherwise. If a break is needed, a different panel should be used.

Perceiving Graphics. To show the data, we must understand how we perceive it. In a graphic, the numbers and nonnumeric values have been translated into geometric objects with visual properties that have been positioned in two-dimensional space by some coordinate system or projection. Exploration involves making comparisons: How much? How much change? How much proportionally? How similar or different? The answers are inferred from comparisons made of the geometric relationships of the objects and of their visual properties: How long is this line, angle, or area? How much longer is this line, angle, or size than that one? How much lighter is this color or grayness from another? Is this point closer to this group of points or to that group? What matters is not only the actual measurements on the physical surface but also what is perceived. The following includes just a few of the findings about perception relevant to design:

- We notice large perceptible differences. We discussed earlier how focal points can be created to direct a user's attention by using brighter colors or larger features to visually emphasize certain elements. This emphasis may be used to separate the data elements from the nondata elements—such as grids, reference lines, and reference curves—and to draw attention to what is important. Visual properties, however, are relative. It's the contrasts that matter.
- We can discriminate between two values of a visual property only when the difference is proportionally large enough. For example, we can perceive a difference between a line that is 25 cm and 26 cm long no better than we can perceive a difference between a line that is 2.5 cm and 2.6 cm long.
- We group elements into units. The Gestalt principles described in a previous section also apply to the graphical elements. Spatial proximity, in particular, is a powerful way to emphasize relationships between data entities. But if grouping is not intended, the perceptual tendency to

structure the elements can make it difficult to see other patterns in the data or make it easy to see patterns that don't exist. The Gestalt principles have been effectively used in visualizations that combine a scatterplot graphic with dynamic query controls. The controls are used to filter the data. Grouping by proximity allows outliers to be seen in the data, and grouping by similarity (usually color) is used to find related items.

- When we map real numbers of a linear scale to values of a visual property (the physically measured values such as the length of a line or some value in a color scale), what we perceive (the sensation) is on a nonlinear scale. The perceived scale is the actual scale raised to some power. We estimate fairly accurately the length of a line (the power is between 0.9 and 1.1). We underestimate the area of a shape such as a square, rectangle, or circle (power is 0.6 to 0.9), and underestimate still more the volume of a solid (power is 0.5 to 0.8). (Ward et al., 2010)
- We can perceive individual marks immediately under certain conditions. This phenomenon was discussed in Chapter 2 in the section on distributed attention. For example, in a scatterplot, a few red points among a large number of gray points will be immediately noticed. The points appear to "pop out" with no cognitive effort. The degree of contrast of a specific feature in pop-out points and the other points give rise to the result. Color, brightness, orientation, size, motion, and stereoscopic depth can all be used to produce a pop-out effect. This perceptual mechanism has been effectively used in heatmaps. Heatmaps are 2-D color tables with a bipolar color scale. For each cell of the table, a lower value will be mapped to a shade of one color and a higher value to the shade of the other color. Blocks of cells of similar color and intensity immediately stand out.

Graphical Integrity. For a graphic to have integrity, its visual representation must be consistent with the data. The design choices establish visual expectations for what is represented by the physical space and the marks in it. The design of each graphic should be uniform, invariant, and clearly labeled. The scales used for the content area should have regular intervals without breaks. The mappings of data variables to visual variables should remain constant for all marks in the graphic. Although perception can affect how the graphic is interpreted, what is physically measured on the surface of the graphic should at least be in direct proportion to the values in the data. More is more—a longer line means a greater magnitude of the value of some variable—and less is less. The variation of a reference curve in the graphic should correspond with a variation in the magnitude of the variables it represents. All the data that has been selected by the user should be shown so that comparisons can be made in context.

Aesthetics. The elements and principles that make a graphic pleasing are elusive. Within any of the fields of functional design—graphic design, industrial

design, architecture, visual interface design—certain words are used to describe the goal. For elegance and simplicity: scale, contrast, and proportion are used; for organization and visual structure: distinctiveness, integrity, comprehensiveness, and appropriateness are used. For each of these attributes, there are principles, techniques, and many examples to study. But graphics for data exploration is at its finest when it has the visual representation best suited for interacting with some specific data. The elegance of Gmail is in the way it supports the fundamental tasks it was designed to support.

Graphics or Tables? Graphics are not always appropriate. Interactive data tables or spreadsheets are the best way to show exact numerical values when the datasets are not too large. Data tables or spreadsheets provide ways to filter, sort, and perform various calculations on the numbers. Tables may also be preferred when a dataset consists of highly labeled numbers.

4.3.4 Real-Time Constraints

Interactive systems must be responsive. Responsiveness, unlike performance, is measured on a human time scale. The time it takes to physically react, to notice a lag in the system's response, to keep our attention from drifting to something else, or to wait for a response to an email impose design constraints. A system is responsive if it provides what we need within the expected time constraint. Table 4.5 shows a generalized threshold of time constraints that are important for interaction design (Johnson, 2010). Early stage perception takes about 10 milliseconds. Above 100 milliseconds, we begin to lose the sense of cause and effect between an action we take and the reaction of the system. The longest time we expect there to be a lull in a conversation is 1 second. If the system has

TABLE 4.5 Human Time Constraints for Interaction (Based on Johnson, 2010, p. 161)

Threshold	Type of Interaction
0.01 second	feedback for stylus-based input with electronic ink on display
0.1 second	feedback for hand-eye coordination (pointer movement, resizing, scrolling, drawing with mouse, etc.)
	feedback for click on button or link
	show "busy" indicators
	longest interval between animation frames
1 second	show progress indicators
	finish various operations (e.g., open window, auto-save, etc.)
	time to wait before next visible response
10 seconds	complete one action of a multisequence action (e.g., an edit in a text editor)
	complete user input to an operation

not responded, we wonder why and start to become impatient. At 10 seconds, we will have completed an action in a multisequence action. To design interaction that flows, these constraints must be met as appropriate for each action the user will take.

4.4 SUMMARY

The complexity of data-intensive systems is continually increasing. But even a system with a complex visual interface can be operationally simple if its structure is logical and well organized, and it makes interaction efficient. To help designers better understand the mental processes of users as they interacted with computational tools, Donald Norman devised a theoretical cognitive framework of seven stages he called action theory. From a high-level goal, users form an intention, which they break into sequences of actions that they perform. As they perform each action, users evaluate and interpret the results. A system is difficult to use if it requires too many action steps or the interface is difficult to interpret.

Good design begins with a thorough understanding of the users, their work, and their environment and is done in four stages:

- *Analysis*. Design decisions must be based on facts and details elicited from discussions with the users and observations of the work. These are used to create models of the work environment, the work, and the resources required to do the work.
- *Design*. Design begins by developing an abstract conceptual model. From this, an interface architecture is designed that groups related actions into interaction spaces and shows how the user will navigate between these spaces. The visual interface and interactions are designed from the various analysis and design models.
- *Prototyping*. Prototypes provide a way to envision and experience how the new system will affect work, to explore technical alternatives, and to discuss flaws in the design and specific ideas for how it might be improved. Prototypes should be quick and inexpensive to build and relatively easy to change.
- *Evaluation*. Evaluation is intended to provoke discussion about better ways to structure the system, or to uncover tasks that aren't necessary or may be more or less important than initially thought.

Visual interfaces organize content and tools so that users can efficiently do the activities or tasks that the system is being designed to support. The following steps are required to design a visual interface:

- *Organize the application*. Complex applications require organization and navigational aids that allow users to move between interaction spaces.

Designers use signage, maps, and clues to help users know what path takes them to the place where the next required action can be performed and to keep them from getting lost.

- *Organize the page*. Page layout uses techniques to create a visual hierarchy of the content that gives weight to what is most important and a visual flow that leads the user through the page. A visual hierarchy structures the content that incorporates focal points that lead into secondary regions of less important content. Visual flow is designed to manipulate the path the eye takes as it scans the page.
- *Organize the actions*. Actions may be visible or invisible. Visible actions must provide visual feedback when they are manipulated by the user. Invisible actions are invoked through combinations of keystrokes, drag-and-drop operations, and double-clicking or right-clicking on visible objects.

Designing the physical structure of interactive visualizations is often guided by a well-known mantra: "Overview first, zoom and filter, then details on demand." Several interface schemes incorporate the mantra: overview + detail (spatial separation), zooming (temporal separation), and focus + context. Interactive techniques such as filtering, creating subsets of the information, or sorting and rearranging the information provide support for creating different views of the information.

The fundamental principle of displaying graphics is "show the data." Graphics should portray the data clearly and accurately, allow the data to be compared, and contain information that is relevant to the task.

Interactive systems must be responsive. The time it takes to physically react imposes design constraints that vary by task. The real-time constraints that affect perception and cognition range from 10 milliseconds to upward of 10 seconds. A system is responsive if it provides what we need within the expected time constraint.

4.5 FURTHER READING

The following are three well-known Web sites for interaction design and usability:

- The Nielsen Norman Group (www.nngroup.com/)
- Tog's First Principles of Interaction Design (www.asktog.com/)
- Jakob Nielsen on usability (www.useit.com/)

Design Methodologies. The contextual design methodology is fully described along with plenty of practical advice in *Contextual Design: Defining Customer-Centered Systems* (Holtzblatt & Beyer, 1998). Variations of this methodology tailored for smaller projects with more focused goals have been

further described in *Rapid Contextual Design: A How-To Guide to Key Techniques for User-Centered Design* (Holtzblatt et al., 2005). A methodology with an emphasis on *usage*-centered rather than *user*-centered design can be found in *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design* (Constantine & Lockwood, 1999).

We have found, particularly for domains that are not well understood, that by working through a small set of core tasks and functions to be supported and combining it with low-fidelity prototyping, the designer can more quickly learn the domain. A deeper understanding of the domain makes it is easier to make decisions about how to expand the scope of the effort. Kuniavsky (2003) discusses user research methodologies used to inform the design of the user experience. Nielsen (1993) describes the importance of incorporating a usability engineering lifecycle throughout the design of software products. And, Yaffa (2007) describes the effort involved in the design and usability assessment of something as apparently straightforward as designing fonts for highway signs.

Visual Interfaces and Interactive Design. *Designing Visual Interfaces* (Mullet & Sano, 1995) explains how graphic design principles and techniques are used to design visual interfaces. *Designing Interfaces* (Tidwell, 2005) discusses some of the same principles but provides more detail of interest to software developers. The book contains concrete strategies and design patterns for many aspects of visual interface design. Cockburn et al. (2008) survey the major interaction techniques for user interfaces such as overview + detail, zooming, and focus + context, while Fekete and Plaisant (2002) discuss the challenges of interaction design when applied to large datasets of more than a million data points.

Visualization. There are two good places to start to learn more about visualization. The first is *Readings in Information Visualization* (Card et al., 1999), which is a collection of papers that cover a variety of topics on visualization. The second is *Envisioning Information* (Tufte, 1990), which describes general principles used in the design, editing, and analysis of data representations.

Graphics. An introduction to some of the issues in graphics design is *Good Graphics?* (Unwin, 2008). Texts on quantitative graphics include works by Bertin (1983), Cleveland (1993), Kosslyn (2006), Theus & Urbanek (2008), Tufte (1983), Wainer (1997, 2005), and Wilkinson (2005). Unwin et al. (2006) discuss issues related to large datasets.