Human–Computer Interaction – INTERACT '87
H.-J. Bullinger and B. Shackel (Editors)
Elsevier Science Publishers B.V. (North-Holland)

A FRAMEWORK FOR COMPARING SYSTEMS WITH VISUAL INTERFACES

Ted Selker
Cathy Wolf
Larry Koved

Thomas J. Watson Research Center
IBM Research Division
P.O.Box 218
Yorktown Heights, N.Y. 10598

A computer program presents its capabilities and domain of application through a user interface. With the advent of inexpensive graphics hardware, systems with visual user interfaces are proliferating. New interface technologies offer opportunities for improving the usability of programs. It is important to understand how to employ these new techniques in the design of better user interfaces. A review and comparison of visual interfaces prompted the need for a vocabulary and systematic framework to describe them. This paper presents a framework developed for describing and comparing visual user interfaces.

Communication between computers and humans has often been described in linguistic terms. This paper uses the term visual language to refer to the systematic use of visual presentation methods to convey meaning to a user. The framework includes a description of the interface in terms of the elements, operators and syntax of an interface language, the rationale governing the use of visual elements, the power of the language, interface characteristics such as the interaction style and input/output device dependencies, and the domain and purpose of the application.

The framework has been useful in identifying important differences between visual interfaces and has provided a vocabulary for the discussion of visual language. Elements of the framework are illustrated with examples from existing systems.

1. INTRODUCTION

Many modern computer systems use visual interface techniques to enhance or replace text. Efforts to demonstrate the utility of visual interface techniques have more often taken the form of systems rather than scientific analysis. Grail [1] was one of the earliest systems to employ interactive graphics in the user interface. Systems such as the popular Apple Macintosh [2], modelled after the Xerox Star [3], use spatially organized screens with icons to represent objects and actions. The success of VisiCalc-style electronic spreadsheet programs has been attributed in part to their spatial presentation of information.

The appearance of icons, graphics and other visual language techniques on computer screens does not automatically improve interfaces any more than the printing press assured that written material would be worthy of reading. One behavioral experiment [4] found that users' performance and preference differences among seven interfaces were not determined by whether the interface style was iconic, menu or verbal command language. The authors concluded that careful design is more important than interface style.

A prerequisite for careful design is to understand the range of techniques available and the reasons for choosing one technique over another. Efforts have been made to apply the principles of graphic design to visual user interfaces. Tutorials, such as ones given by Aaron Marcus at the ACM CHI conferences, have put forward these design principles [5]. A recent effort demonstrates automated choice of visual presentation based on the structure and kind of information being viewed [6]. Other recent work has taken on the challenge of classifying visual languages along a number of dimensions [7, 8]. These analyses have not been detailed enough to provide guidance in the design of visual language.

As a first step towards providing such guidance, we have found it helpful to analyze existing visual languages according to a number of characteristics. These include the domain and purpose of the system, elements of the visual language, power of the language, and interface characteristics.

This paper includes an example analysis based the user interface of the Javelin business software package [9]. Examples from other computer systems employing visual language are included as needed.

The use of Javelin as this paper's primary example is not a judgment that this software demonstrates exemplary use of visual language. Rather, the choice was made because it contains a number of visual language ideas and is available on a popular microcomputer.

2. DESCRIPTION OF SYSTEM UNDER CONSIDERATION: DOMAIN AND PURPOSE

Javelin is a business software package oriented towards the financial community. It provides spreadsheet and graphics capabilities. It is directed specifically towards users who need to perform analyses in which one of the independent variables is time.

Unlike other spreadsheet packages, Javelin provides several modes of interaction (views) with the data. Some of these will be used as examples later in the paper:

- The *Diagram view* shows how variables are related to one another. This is useful for showing dependencies among variables in a spreadsheet. See Figure 2.
- The *Chart view* uses bar charts to present data values. See Figure 1.
- The *Quick graph view* uses a cartesian graph to present and change data values of a variable. See Figure 3.
- The *Worksheet view* is very much like a traditional spreadsheet. Values and formulas in cells are entered and manipulated.

Some Javelin views are manipulable, while others are depictive (output) only. The Diagram view presents the relationship between variables in the system, but data can not be changed (depictive only). The Worksheet view allows entry of data and modification of relationships between the data elements (formulas). These concepts will be explored further in this paper.

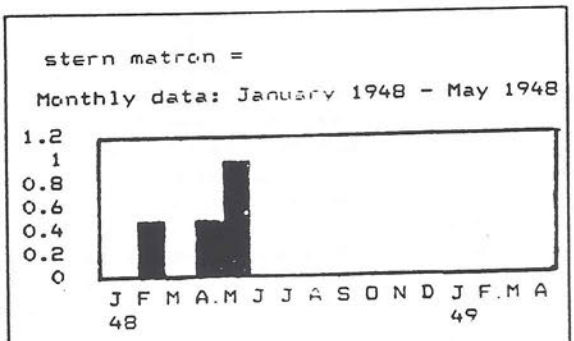

Figure 1. Chart view.

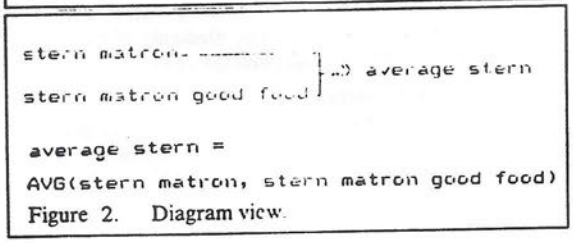

Figure 2. Diagram view.

3. CORRESPONDENCE - VISUAL INTERFACE VS. UNDERLYING FUNCTIONALITY

For the purposes of this paper, we are primarily interested in the visual component, and how it represents the functional component of a software system. A system provides a mapping between the user interface and the system's underlying functionality. Javelin provides multiple mappings through its different views. The output functions, such as the Chart, Quick graph or Worksheet views, present unique mappings from the system functionality to the display as output. Based upon the *same* underlying functionality, different mappings yield different presentations or interfaces.

A functional mapping is also performed on user input. Javelin allows users to enter data or formulas into spreadsheet cells, or change the height of bars on bar charts by moving the mouse or pressing cursor keys. The input actions, moving the mouse or pressing keys, result in changes being made to the underlying data representation. In turn, the output function provides positive indication that the user's input had an effect.

4. ELEMENTS OF THE VISUAL LANGUAGE

A formal definition of any language contains a set of terminal symbols (alphabet) and a grammar to specify legitimate sentential forms. A visual language is defined by a set of visible objects, a set of operators to manipulate them, and the syntax which prescribes relationships between objects and permissible operations on them. For example, in the Chart view, the bars, axes and labels are elements of the alphabet. The "language" of a system may be divided into several different sublanguages. Javelin includes several sublanguages for input (menu selection, data entry, etc.), output (Worksheet, Quick graph, Diagram, etc.) and navigation (getting from one part of the system to another).

The syntax of a language defines the relationship between its elements. If a visual language is to describe bar charts (Javelin's Chart view), then the syntax prescribes the shape of the bars (rectangular), the use of borders and fill pattern, orientation (vertical or horizontal), height, spatial relationship to other bars and axes, axes labels, etc. The relationships among the elements of the bar chart convey

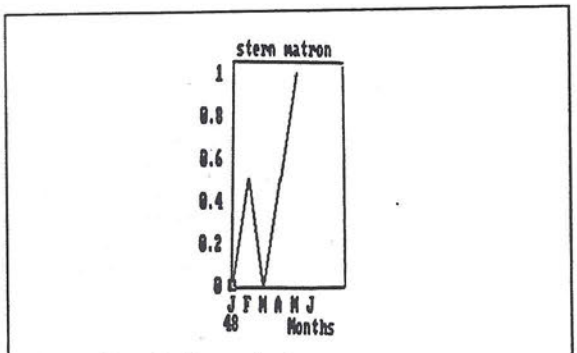

Figure 3. Quick graph view.

information to the user. Placing one bar on top of another conveys different information to the reader than if they are placed side by side. Similarly, the use of shading or the fill pattern used in the bars also conveys information.

5. RATIONALE GOVERNING USE OF BASIC LANGUAGE ELEMENTS

Visualization techniques are a central component of user interfaces. These techniques include: text, shape, position, texture, color, temporal relationship, abstract icons, symbol icons, and pictures.

Systems offering similar functionality can have completely different visual interfaces. To a degree, these differences can be explained by priorities applied to the design of a system. Communication goals and "real world" constraints are two types of design priorities which affect how a system's interface uses visual language. Communication goals are concerned with what the designer wants to communicate to the user. Designers may use multiple visualization techniques to delineate the structure of a system, represent the system's functionality, represent data, and to give multiple views of information. Real world constraints include compatibility with existing and earlier systems, or software and hardware constraints. We typically infer the designer's goals from the characteristics of the interface.

5.1. Communication Goals

A visualization technique may be used in an interface to segment a system's structure. Icons and menus, for example, help segment the structure and function of a system. They are often used to help a user navigate through a system. Javelin uses a menu front-end to allow users to choose a view through which they will display and manipulate information. The use of different visualization techniques also helps to distinguish the views. For example, given their different visualization techniques, it would hard to confuse the Worksheet and Chart views.

The Macintosh uses visualization techniques to segment the structure and functionality of the system. Macintosh's icons represent objects and applications; text menus are used to represent actions. As a user "opens" an icon, it might start a new program. This new program might employ visualization techniques which differ from the icon menu technique used to select it.

Different visualization techniques may be used to provide somewhat different information about the same data. Several Javelin views are designed to view the same data from different perspectives. Javelin's Chart view lets the user quickly assess the approximate magnitude of data values. This is particularly useful when comparing the relative magnitude of a variable over time. The Worksheet view provides the user with the precise numeric values.

Differences in visualization techniques can also represent differences in functionality. In Javelin's Worksheet view, text formulas are used to depict and manipulate relationships among variables. In the Diagram view, lines and arrowheads depict the direction of the relationship between two variables, but not the functional relationship between them.

5.2. Real-World Constraints

The desire for compatibility with existing or previous interfaces is often a factor in the selection of visualization technique. The goal of compatibility is to enhance the transfer of skills from a system with which the user is familiar to a new system. The menu bar across the top of the screen used to access the Javelin functions is similar to that used in the popular spreadsheet program, Lotus 1-2-3 [10] In addition, the input language for selecting a menu item by typing is similar to the 1-2-3 method. To invoke the menu, one types "/" followed by the first letters of the Javelin command. For example, to invoke the Chart view, one types "/VC". Presumably, the Javelin designers deliberately chose a menu interface similar to 1-2-3 on the assumption that many Javelin users would have prior experience with 1-2-3.

The goal of compatibility with other interfaces is sometimes at odds with optimal design of an interface. A classic example can be found in the layout of the QWERTY keyboard. This layout was originally designed to slow typists down to minimize key jamming in the days of the earliest mechanical typewriters. This requirement is now obsolete, but we adhere to the QWERTY layout in the interests of compatibility.

Another type of real-world constraint involves hardware/software characteristics of the devices on which a system is expected to run. This constraint is discussed further in the section "Input/Output Devices." The point to be made here is that the visualization techniques used in an interface sometimes reflect the lowest common denominator of the hardware/software characteristics of the target devices.

6. LANGUAGE POWER

The power of a visual language describes the level of function available to the user for accessing and modifying the underlying data or functionality. Some visual languages can only present information, while others allow modification of the functionality or data. We have found the following distinctions to be useful in describing the power of visual languages:

- **depictive** -- A visual language is depictive-only when it has the capability to describe something, but not to modify it. Depictive languages have been referred to as program visualization [7]. The Diagram view of Javelin is an example of a depictive language. It employs a diagram to display the dependency relationships between variables, but does not allow the user to change these relationships.
- **manipulable** -- A visual language is manipulable if can be used to change data values. The Chart view in Javelin allows the user to change data values by changing the height of the bars in the bar graph. These new

responding functional elements of an underlying system. It can also aid in understanding syntactic rules governing combinations of elements in a language.

Javelin uses several separate metaphors in the different views. In the Worksheet view, Javelin employs the paper-and-pencil spreadsheet metaphor. Variable names or dates are entered in the row and column headings and numbers in the worksheet cells. Calculations are performed on numbers in the worksheet by entering the appropriate formula in a cell. Thus, the worksheet looks and has semantics similar to a paper-and-pencil spreadsheet. The Diagram view uses a limited flow-chart metaphor to portray dependencies among variables. Lines and arrowheads are used to connect variables which affect one another (see Figure 2 on page 3). A typical flow-chart shows flow of control information. In contrast, the Diagram view shows data dependency information. The Chart view uses physical size in a bar graph as the metaphor for representing the magnitude of a variable. Changes in the value are made by changing the size of the bar.

Javelin's use of metaphors is fairly transparent in that the user may not be explicitly aware of them. In the case of the paper-and-pencil spreadsheet metaphor, the user feels that the spreadsheet *is* the underlying system. Other metaphors are more evocative than accurate. Desk-top metaphors are one such example. Filing cabinets and trash cans are not usually placed on top of desks; objects can be removed from physical trash cans, but not from desk-top metaphor trash cans; and pulling a copy of a document out of a folder in the desk-top metaphor results in a new copy being made, but this is not true in the physical world. The analogy used by metaphors may not always be a simple, complete and accurate.

8. WHY IS THE SYSTEM INTERESTING?

A visual interface may be interesting because of the visualization techniques it employs, or the ways in which it uses them:

- **New visual language techniques** -- These often manifest themselves in systems. The Xerox Smalltalk and Cedar environments were among the first system to use pictograms to activate programs.
- **New applications of visual elements** -- VisiCalc [14] was a new application of constraint language ideas to tabular layout of data.
- **Integration of visual techniques not previously demonstrated together** -- Javalin integrates several visual languages and presents multiple views of the same data.
- **Usability characteristics** -- What-You-See-Is-What-You-Get (WYSIWYG) style text editors, such as Bravo [15], have been shown in some situations to allow increased speed and accuracy.
- **New interaction paradigms** -- Laura Gould's Programming By Rehearsal [16] uses a theatrical metaphor, with a stage and actors. The user auditions animated actor images that can then be selected to become part of a program.

9. CONCLUSION: AN ASSESSMENT

In this paper we have presented a framework for comparing different visual interfaces. This framework has been useful in identifying important characteristics of visual interfaces and has provided a vocabulary for the discussion of visual language. Others [6, 5] have proposed design guidelines for visual interfaces based on the principles of graphic design.

An important next step is to relate the characteristics identified in these analytical efforts to the usability of a system through behavioral studies. Observational studies such as [17] and experimental studies such as [18] are promising first steps towards understanding usability. It is our hope that by combining behavioral and analytical approaches we can begin to provide guidance for the design of visual interfaces.

REFERENCES

1. T. O. Ellis and W. L. Sibley. The Grail Project verbal and film presentation, 1966.

2. Carol Kaehler. *MacPaint*. Apple Computer, Inc., 1983.

3. W. L. Bewley, T. L. Roberts, D. Schroit, and W. L. Verplank. Human Factors Testing in the Design of Xerox's 8010 "Star" Office Workstation. *CHI '83 Proceedings*, pages 72-77, 1983.

4. John Whiteside, Sandra Jones, Paula S. Levy, and Dennis Wixon. User Performance with Command, Menu, and Iconic Interfaces. *CHI 85 Proceedings*, 1985..

5. A. Marcus. *Tutorial 18: User Interface Screen Design and Color*. ACM/SIGCHI, 1986.

6. J. MacKinlay. *Automatic Design of Graphical Presentations*, PhD thesis, Stanford University, 1986.

7. B. A. Myers. Visual Programming, Programming by Example, and Program Visualization: A Taxonomy. *CHI '86 Proceedings*, pages 59-66, 1986.

8. Nan C. Shu. Visual Programming Lanuages: A Dimensional Analysis. *Proceedings of the International Symposium on New Directions in Computing, Trondheim, Norway*, August 1985.

9. J. Bernoff, E. Brout, and J. Waldron. *Javelin*. Javelin Software Corp., 1985.

10. J. Posner, J. Hill, S. E. Miller, E. Gottheil, and M. L. Davis. *Lotus 1-2-3 User's Manual*. Lotus Development Corp., 1983.

values are incorporated into the database available to the other Javelin views.

- **programmable** -- A visual language is programmable if it allows a user to add new data elements or function. The Worksheet view allows the user to create new functions by using existing variables, constants and built-in functions. These formulas are essentially user-defined programs. The resulting variables can be used in the Worksheet and other Javelin views. A more graphic example of a user programmable language is Pict [11] which allows the user to create programs by positioning icons for data, variables and operations and connecting them by colored lines.
- **scope** -- A visual language can be designed to work in a specific domain, such as a computer aided design system (CAD) for circuit layout, or in many domains. A "paint" program, such as MacPaint [12], can be used in many domains. It can be used to sketch a house or diagram the bonds of a chemical molecule.

  The languages (views) in Javelin are domain-specific in that they are well suited for business analyses. They are also task-specific in that the values of all variables must be expressed as a function of time. Thus, although Javelin is suitable for describing and manipulating monthly sales figures, it cannot be used for representing sales as a function of geographic area.

  Javelin has domain-independent aspects. The visual presentation of dependency relations used in the Diagram view could be used effectively in other domains or tasks.

7. INTERFACE CHARACTERISTICS

In this section, we consider several interface characteristics affecting user interactions with the visual language: interaction styles, input/output device dependencies, and the use of visual metaphors.

7.1. Interaction Styles

The interaction style is the means by which the user communicates with the system. Several types of interactions styles are commonly employed in the user interface. These include verbal command, menu selection of text or icons, and object manipulation interaction styles. Verbal commands require typing operators, operands, and attributes. For example, to print a copy of a file named *work1* on a laser printer, the user might type: "print work1 device=laser". Menu selection requires the user to select operators, operands and attributes from a menu of displayed text or icon alternatives. The selection is typically carried out by pointing to the desired item and performing a selection action. Typical selection actions include pressing the return key, clicking a mouse button, pressing a function key, or typing a character corresponding to the item. In the above example, the user might select the icon for a file by clicking on it, select "print" from a text menu of file commands, and select "laser" from a menu of printer types. Object manipulation style requires the user to transform or move the visual representation of the object to perform the operation. In the print example, the user might drag an icon of the file to an icon of the laser printer.

The Javelin user interface employs verbal command, menu selection and object manipulation interface styles. Verbal commands can be used to access the various views and to perform operations within a view. The typed command "/VC" selects the Chart view. The user can also select this same view by first selecting the item "View" from the main menu followed by the item "Chart" from the "View" submenu. The value of a variable in the Chart view can be changed by object manipulation. Moving a cursor up or down on a bar will change the value of the variable represented by the bar.

Combinations of interaction styles coexist in many user interfaces. In Javelin, the value of a variable can be changed by combining menu selections and verbal commands. The user selects the variable to be changed by positioning the cursor at that variable and changes its value by entering a new one. Typing in a new value implies the "change" command as well providing the new value.

The feel of a system is strongly influenced by both the interaction style and the visual language. The feel of directly manipulating a value in the Chart view comes from both the visual representation of the value as a bar and the ability to change the value by moving the cursor up and down.

7.2. Input/Output Devices

Visual languages are typically designed with a set of assumptions about the input and output devices available to the user. In many cases, particular input and output capabilities may be required to use the language.

Javelin is designed for use with personal computer hardware: a monochrome character display and keyboard. The hardware inherently limits the capabilities of the software. Simple cursor movement (up or down) is used for changing the value of variables in the Chart view. Similarly, the character display limits the resolution of the increments by which variable values can be changed.

The visual language of the Macintosh operating system is an example of a visual language which would be difficult to use with a keyboard as the only input device. Since the two-dimensional position of an icon is significant in the Macintosh, a mouse, which permits simultaneous x-y positioning, was employed. It would have been more cumbersome to require users to perform positioning operations using the four conventional cursor control keys of a keyboard.

7.3. Visual Metaphors

A metaphor is a systematic reference to a real or conceptual world. Many visual interfaces employ a metaphor to help users learn, remember or predict how to use a computer system or program [13]. A metaphor can help users understand the relationship between visual elements and the cor-

11. E. P. Glinert and S. L. Tanimoto. Pict: An Interactive Graphical Programming Environment. *Computer*, November:7-25, 1984.

12. Carol Kaehler. *MacPaint*. Apple Computer, Inc., 1983.

13. J. M. Carroll and R. L. Mack. Metaphor, Computing Systems, and Active Learning. *International Journal of Man-Machine Studies*, 22:29-57, 1985.

14. V. Wolverton. *VisiCalc*. Visi Corp., 1983.

15. S. K. Card, J. M. Robert, and L. N. Keenan. On-Line Composition of Text. *Interact '84 Conference Papers*, 1:231-236, 1984.

16. W. Finzer and L. Gould. Programming by Rehersal. *BYTE*, June:187-210, 1984.

17. J. M. Carroll and S. A. Mazur. Lisa Learning. *IBM Research Report*, RC 11427, 1985.

18. W. P. Jones and S. T. Dumais. The Spatial Metaphor for User Interfaces: Experimental Tests of Reference by Location versus Name. *ACM Transactions on Office Information Systems*, 4:42-63, 1986.