A TAXONOMY AND RULE BASE FOR THE SELECTION OF INTERACTION STYLES

BEN SHNEIDERMAN

1. Introduction
Recent empirical research has shed much light on the design of interactive systems. As researchers, we now understand more precisely the strengths and weaknesses of system variants. As system developers we now have substantially improved evaluation techniques. Academic and industrial researchers have begun to develop predictive and descriptive theories, while thoughtful professionals have compiled useful guidelines documents for practitioners. In spite of all this progress, designers who confront a new application are often reduced to mimicking familiar examples or extrapolating intuitively from experience.

This attempt at a set of design guidelines is intended to aid designers in choosing among the primary interaction styles and in refining their design to suit specific user communities performing specific tasks. The first step is an understanding of the five primary interaction styles: menu selection, form fill-in, command language, natural language, and direct manipulation. Variants and combinations are possible, but these basic styles are a useful taxonomy (Shneiderman, 1987). The second step is an understanding of the factors that guide designers in choosing among the five primary interaction styles. A formal decision procedure would be attractive, but for the moment we must rely on informal rules which although they often stem from extrapolations of experiments need to be validated and refined.

2. Taxonomy of interaction styles
There are many ways of interacting with a computer. The variations result from differences in tasks, computer concepts, and interface devices. The five primary interaction styles described in this paper could each accomplish the full range of interactions, although some would be awkward in certain situations (see Table 1 for an overview). The remainder of this section reviews each style, shows an example, and offers guidelines for improving the design.

<table>
<thead>
<tr>
<th>Interaction Style</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Menu selection</strong></td>
<td>- Shortens training, reduces keystrokes, structures decision-making, permits use of dialog management tools, easy to support error handling</td>
<td>- Danger of many menus, may slow frequent users, requires screen space, requires rapid display rate</td>
</tr>
<tr>
<td><strong>Form fill-in</strong></td>
<td>- Simplifies data entry, requires modest training, assistance is convenient, shows context for activity, permits use of form management tools</td>
<td>- Consumes screen space, requires typing skills</td>
</tr>
<tr>
<td><strong>Command language</strong></td>
<td>- Flexibility, supports user initiative, appeals to 'power' users, potentially rapid for complex tasks, supports macro capability</td>
<td>- Requires substantial training and memorisation, difficult to retain, poor error handling</td>
</tr>
<tr>
<td><strong>Natural language</strong></td>
<td>- Relieves burden of learning syntax</td>
<td>- Requires clarification dialog, may require more keystrokes, may not show context, unpredictable</td>
</tr>
<tr>
<td><strong>Direct manipulation</strong></td>
<td>- Visually presents task, easy to learn, easy to retain, errors can be avoided, encourages exploration, high subjective satisfaction</td>
<td>- May require graphics display/pointing devices, more programming effort until tools improve, may be hard to record history or write macros</td>
</tr>
</tbody>
</table>

Table 1: Overview of interaction styles with their advantages and disadvantages.

2.1 Menu selection
In menu selection systems the computer displays a list of items from which the user selects. If the items are meaningful to the user then menu selection can be a rapid, accurate, and satisfactory approach. If the items are hard to understand or appear similar to each other, users can become confused and make errors. Menu selection is advantageous because it decomposes a complex interaction into a series of smaller steps and because it provides structure for decision-making. On the other hand this same decomposition process can be too rigid for some users and it may slow down
the knowledgeable frequent user.

For many situations menu selection can substantially reduce the number of keystrokes necessary and thereby reduce error rates and performance times. Menu selection mechanisms can be the familiar numbered menus, lettered menus, mnemonic lettered menus, or unlabelled menus. Unlabelled menus operate by movement of a highlight bar over the items, under the control of arrow keys, a mouse, a joystick, a graphics tablet, etc. Table 2 provides a brief set of guidelines while Smith and Mosier (1987) offer extensive guidelines for menu selection systems. Shneiderman (1987) reviews empirical studies and suggests practical design rules.

- Use task semantics to organise menus (single, linear sequence, tree structure, acyclic and cyclic networks)
- Prefer broad/shallow trees
- Show position by graphics, numbers, or titles
- Items become titles for trees
- Meaningful groupings of items
- Meaningful sequencing of items
- Brief items begin with keyword
- Consistent grammar, layout, terminology
- Type-ahead, jump-ahead, or other short-cuts
- Jumps to previous and main menu
- Consider: online help, novel selection mechanisms, response time, display rate, screen size

Table 2: Menu selection guidelines.

Figure 1 is an example of menu selection as applied to airline reservation systems when the users are air travellers making reservations from a hotel or airport lobby. This hypothetical example for Boswash Airlines will be used to illustrate all five interaction styles. Each version is merely a sketch for a system, meant to encourage discussion and to emphasize differences. A full system might have more instructional material, online help, a mixture of interaction styles for different parts of the task, special hardware, greater provision for reversibility of action, more informative feedback, etc.

In Figure 1 the user is confronted with a rigid sequence of menus that force the user to make decisions in a system-defined order. Users who would be willing to change their departure time or date to find a cheaper flight would become frustrated with this design. The numbered menus do help structure the users' work, but more knowledgeable users might be annoyed that items such as the date were broken into

---

**BOSWASH AIRLINES RESERVATIONS MENU SYSTEM**

What month do you want to take your trip?

1 January 4 April 7 July 10 October
2 February 5 May 8 August 11 November
3 March 6 June 9 September 12 December

Type the number of your choice and press RETURN: _______________ New screen _______________

What is your departure city?

1 Albany, NY 5 Philadelphia, PA
2 Boston, MA 6 Washington, DC
3 Hartford, CT 7 White Plains, NY
4 New York, La Guardia

Type the number of your choice and press RETURN: _______________ New screen _______________

What is your arrival city?

1 Albany, NY 5 Philadelphia, PA
2 Boston, MA 6 Washington, DC
3 Hartford, CT 7 White Plains, NY
4 New York, La Guardia

Type the number of your choice and press RETURN: _______________ New screen _______________

What time of day do you wish to take your trip?

1 7:00 AM to 10:30 AM
2 10:01 AM to 3:00 PM
3 3:01 PM to 6:00 PM
4 6:01 PM to 10:00 PM

Type the number of your choice and press RETURN: _______________ New screen _______________

Figure 1: Menu selection approach, with occasional form fill-in for fields such as names, phone numbers or charge numbers.
two separate items. They might prefer a form fill-in approach. Also the layout, wording of items, instructions, and error handling might all be improved. On the other hand, some form of menu selection system is likely to be the most successful in a public waiting room environment where computer and typing skills are not uniformly high.

2.2 Form fill-in
In a form fill-in interaction the users’ main task is to provide data in labelled fields clustered on one or more screens. Sometimes the data items are merely binary choices (Yes/No or Male/Female) or selections from short lists (days of the week or a set of colours), but they can also be taken from large domains (personal names or chemical formulae) or may be essentially unbounded (explanatory paragraphs or meteorological data). Of course, it is possible to replace form fill-in with a series of menu choices, but this strategy may become extremely tedious. Keyboards or other input devices are effective ways of inputting lengthy data fields. The keyboard may be used as a continuously displayed menu that permits rapid selection using a well-learned skill. Increasingly data entry is handled more automatically as is done in reading magnetic stripes on charge cards to get names and account numbers or grocery packages with bar codes.

Form fill-in requires that users learn to use the keyboard with tab, cursor control, and backspace keys. If the users are competent keyboard operators and field labels are meaningful then form fill-in is an effective approach that produces a quite rapid rate of data entry, a moderate level of errors, and good user satisfaction. There has been very little research in the design of form fill-in systems but there are several sources of design guidelines (Galitz, 1980, 1981; Smith and Mosier, 1987) (see also Table 3 for a brief list).

The Boswash Airlines example as a form fill-in (Figure 2) demonstrates the increased density of information that can be supplied on a single screen when compared with menu selection. Presumably the user can press arrow or tab keys to move around the form and perform error correction. The keyboard usage might reduce the number of potential users when compared to menu selection, but might increase the willingness of more knowledgeable and frequent users to work with the system.

Research issues include very basic questions of appropriate layout, justification, spacing, grouping, sequencing, handling of optional fields, and online help. A combination of form fill-in with menu selection might help the novice user. For example, the list of departure cities might be made available when the cursor is moved to that field; then the user could select a city instead of typing it and it would be displayed in the field. A useful conjecture, that needs validation, is that as user experience increases, the density of fields on a single screen can be increased while the field labels can be shortened.

2.3 Command language
Knowledgeable frequent users do not want to be distracted by having to locate an item in a list, nor do they want to have to view and move a cursor over a form. They can manipulate the possibilities in their mind and want concise notations for issuing commands with modest informative feedback. These 'power' users want to be able to put several commands on a single line and even create new commands that encapsulate the work of several frequently used command sequences. The learning time may be
days or weeks, but since usage is frequent the benefits of concision are great.

Often command languages emerge from familiar notations such as mathematics, boolean logic, or music, but many command languages have been created to deal with novel task domains such as text and string manipulation, hotel reservations, or information retrieval. There has been much research in command language design for text editors and other applications. Consistency in structure, meaningfulness, orderly abbreviations, small number of commands, and congruent pairs of commands have been shown to be important determinants of rapid learning, rapid use, low error rates, high satisfaction, and easy retention over time (see Table 4 for a brief set of guidelines).

- Create explicit model of objects and actions
- Choose meaningful, specific, distinctive names
- Try for hierarchical structure
- Provide consistent structure
  (hierarchy, argument order, action-object)
- Support consistent abbreviation rules
  (prefer truncation to one letter)
- Offer frequent users the capability to create macros
- Consider command menus on high-speed displays
- Limit number of commands and ways of accomplishing a task

Table 4: Command language guidelines.

Airlines reservations systems used by travel agents and airlines employees use a very concise and cryptic command language strategy. The Boswash Airlines command language (Figure 3) is modelled after the American Airlines SABRE system that has been in use for approximately 25 years. The information is packed in a coded form without delimiters and the feedback is also quite dense, but these are advantages to the frequent user.

2.4 Natural language

Many computer scientists and computer users propose that the “ultimately desirable” way of using a computer is through natural language interaction. They argue that people already know their own natural language and therefore learning to use the computer would be simplified if the computer accepted natural language input. This vision has propelled many researchers over the years as they tried to make natural language interaction systems for programming, database retrieval, or expert systems usage. There are commercial products such as INTELLECT (Artificial Intelligence Corporation) or Q & A (Symantec) but the success could only be considered to be modest. Relieving the user of the burdens of syntax is only a small part of the problem in using a computer and is of benefit only to novice and intermittent users. The much more complex part is understanding the computer concepts and the task domain. Relief from syntactic details does not ensure that the users will know that files must be saved before quitting or that stock market purchases must be in blocks of 100 shares.
Furthermore, natural language interaction systems constantly confront the user with the problem of uncertainty about whether a particular input will be acceptable to the machine. This inhibits planning ahead since users must always be ready to engage in clarification dialogue. Improved designs will mitigate this problem and speech input might reduce the penalty of typing, but the users will still be struggling to discover the syntactic and semantic scope of the system. Therefore, this author is sceptical that natural language interaction will become widely used. The reader should be cautioned that there are still many people who hold a more optimistic view. Natural language generation and natural language parsing of textual databases are likely to be more successful applications of the technology of computational linguistics. Finally, the special case of interactive fiction and adventure games should be cited as a highly successful application of natural language interaction. The success stems from the fact that in this situation part of the challenge and fun is to discover the syntactic and semantic bounds of the program. Table 5 summarizes some of the issues concerning natural language interaction.

- Reducing syntactic load is not enough
- Computer and task semantics should be represented
- predicate calculus, boolean algebra, set theory, normalisation theory, database entities and values, permissible operations and constraints
- Make syntactic and semantic scopes visible
- Therefore, NLI might work best for:
  - Users who are knowledgeable about task domain
  - Intermittent users who cannot retain syntax
  - Users with moderate computer skills
  - Good typists (until speech I/O improves)

**Table 5: Natural language interaction guidelines.**

The scenario for Boswash Airlines (Figure 4) is meant to be positive and attractive. It is not clear that such a system could be built or that it would be sufficiently successful to be commercially viable. It might be improved by more instructional information and some display of the previously made decisions.

2.5 Direct manipulation
The first four interaction styles emerged from a 'teletype mentality' or at best a text-oriented view of the task. However, many designers are breaking away from the constraints of teletype and keyboard designs and employing screen-oriented, form-oriented, visual and graphic approaches with pointing devices such as arrow keys, a mouse, touchscreens, or a graphics tablet. These new designs often depend on bit-mapped high resolution displays, but the concepts of direct manipulation can be applied with simple text screens and a simple pointing device.

The key to direct manipulation design is to create a visual representation of the 'world of action' that includes selectable displays of the objects and actions of interest. Then with pointing, zooming, and panning the user can rapidly perform operations, see the results immediately and reverse operations if necessary. These ideas have been applied in the Xerox STAR, Apple Macintosh, videogames, many word processors, and other applications (Shneiderman, 1983, 1987). The benefits include relatively rapid learning with high retention over time and high user…

**Figure 4: Natural language style could work well in this example because the users are familiar with airline terminology and concepts. However, it requires substantial user input, clarification of possibly ambiguous phrases, and may cause difficulty in situations when unfamiliar terms are used. Also, the users may be uncertain as to what the scope of the semantics and syntax are.**
satisfaction. Errors can often be prevented because the representation shows the users the impossibility of performing a task and because typographic errors are eliminated when the user selects from a set of displayed objects. Exploration is often encouraged in direct manipulation environments, especially when reversibility of actions is ensured. There are disadvantages in current direct manipulation designs. It is often difficult to create programs and recording the history of a session can be troublesome.

Developing direct manipulation systems is difficult with current software tools, but improvements are expected. There is some concern also for visually impaired users. For some tasks, frequent users may prefer command language approaches (unseen manipulation) because they prefer not to be distracted by having to locate objects and actions on the screen and then select them. The definition of direct manipulation with some of the benefits and concerns is in Table 6.

---

* Create visual representation of the ‘world of action’
  - Objects and Actions are shown
  - Tap analogical reasoning
* Rapid, incremental, and reversible actions
* Replace typing with pointing/selecting
* Make results of actions immediately visible

Benefits
- Control/display compatibility
- Less syntax reduces error rates
- Faster learning and higher retention
- Encourages exploration

Concerns
- Increased system resources, possibly
- Some actions may be cumbersome
- Macro techniques are often weak
- History/tracing may be difficult
- Visually impaired have more difficulty

---

Table 6: Direct manipulation guidelines with benefits and concerns.

The Boswash Airlines example (Figures 5a–e) shows a form that provides the context for the decision process. Users can touch a field to begin the flight selection process. Then with the form still on the screen, they might be presented with a map showing departure cities. After selecting the departure city the screen would show a map with arrival cities accessible from the selected departure city. Then to choose the date, the calendar provides familiar context, avoids the questions of data entry format, and can be very rapid. Finally, the results of each selection are displayed on the form allowing continuous visibility of status and an opportunity to make changes.

Figure 5a: Form to be filled. The user has some flexibility in order.

Figure 5b: Map to select departure city shows geographic relationships.
3. Choosing an interaction style

Like good cooking, good user interface design is a blend of measurable aspects (calorie counts, fat percentages, vitamin levels) with subjective issues (smell, texture, colour, taste) and stylistic variations (Szechuan, French, Northern Italian) plus contemporary fashion (nouvelle cuisine, par-boiled vegetables, rare meat). Designers absorb each problem situation thoroughly, then apply reason and intuition combined with experience to produce a solution. Some designers favour menu selection or command language for all situations because they are familiar with the strategies for design and implementation. In other cases the availability of form management software may sway designers to depend heavily on this approach.

As a design field, human-computer interaction is moving beyond depending only on the intuition of designers; empirical evidence is accumulating to guide designers. Still interface design is a new field and much work remains to be done before reliable designs for multiple situations are available.

3.1 Factors influencing choices

What follows is the author’s attempt to suggest rules for designers to follow in selecting appropriate interaction styles. The rules are presented in an informal rule-based notation that is similar to what is used in expert systems. Much work remains to refine these rules. Undoubtedly there are exceptions, missing criteria, or simply
poor decisions. However, this first attempt is meant to be compact, be narrowly focused, and therefore to provide a basis for discussion and improvement.

Table 7 lists some of the aspects of the task domain that influence the selection of interaction style. For example, if the task has a high degree of data entry, then form fill-in is strongly recommended, although command language can be used. If a familiar notation is already available to the users, then that notation should be converted into a compact and hopefully easy to learn command language. If exploration is important, as in circuit or automobile design, then direct manipulation can provide a flexible environment with minimum distraction from the task.

<table>
<thead>
<tr>
<th>IF high degree of data entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>=&gt; ff</td>
</tr>
</tbody>
</table>

| IF paper form exists |
| => ff |

| IF familiar notation exists |
| (e.g., arithmetic or boolean expressions, chemical formulae) |
| => cl |

| IF natural visual representation exists |
| 1 modest number of objects and actions can represent the task domain |
| => dm |

| IF multiple decisions are required |
| 1 selection from a large unfamiliar state space |
| => ms | dm | cl |

| IF poor keyboard skills |
| => ms | dm | (nl with speech input) |

| IF exploration and intuition are important goals |
| => dm |

Table 7: Task factors are an important determinant of interaction styles. This informal rule based representation expresses the author’s opinions in design situations; there will undoubtedly be exceptions and refinements. The notation uses => to mean then, & for and, and | for or. The interaction styles are menu selection (ms), form fill-in (ff), command language (cl), natural language (nl), and direct manipulation (dm).

User skill level has a profound influence on the selection of an interaction style (Table 8). With novices, menu selection or some form of direct manipulation present familiar words or visual representations of the task domain. These styles require limited keyboard skills and computer knowledge. More knowledgeable users can work more rapidly with a form fill-in approach. Knowledgeable intermittent users can work with any style. As frequency of use and proficiency increases, users will seek more rapid and productive styles with less to look at on the screen. Expert users expect short response time and macro facilities to encapsulate frequent sequences of actions into a single command.

In addition to the interaction style, there are a host of variables that have to be adjusted as a function of the users’ knowledge and frequency of use (Table 9). For example, novices need meaningful labels, while knowledgeable intermittent users prefer abbreviations so that more information can be shown in a single display. Expert frequent users may prefer to eliminate some labels and use concise codes, thereby speeding up screen scanning and permitting still denser displays. Similarly, the desire for informative feedback when actions are taken generally decreases with increased expertise and frequency of use. These rules describe some of the design features that vary with the users’ profile. There are certainly other features and exceptions to these rules.

3.2 Blending interaction styles
The five primary interaction styles are meant for exposition and education. In real systems, designers blend styles where appropriate. For example, in the Boswash Airlines menu selection example, the date was requested as a fill-in item, because menu selection would have been preposterous. Similarly, personal names are usually handled as a form fill-in item. For a short list, say 2 to 12 items, if the display rate is reasonably fast, a menu selection strategy is often the best. As the list gets longer the designer must weigh the users’ familiarity with the items, the potential and
IF novice
& high density of meaningful labels
& high density of informative feedback
& slower pace
& introductory tutorial/demo
& limited subset of actions and functionality

IF knowledgeable intermittent
& modest use of labels
& modest use of informative feedback
& moderate pace
& online help to explain objects and actions
& chance to move up to more powerful actions, but protection danger

IF frequent user (knowledgeable about task/computer/syntax)
& short, sparse, or no informative feedback
& faster pace
& online reference with elaborate search mechanisms
& abbreviations, shortcuts, user-defined macros, with access to system internals

Table 9: User skill levels are a key factor in determining a variety of aspects of the interaction design. These rules are an informal representation of the author's opinions that will undoubtedly have exceptions and will need to be refined. It is meant as a first attempt to provoke discussion.

seriousness of an error, the speed of interaction required, and the length of the items. All but the final criteria are difficult to assess quantitatively, leaving the designer with the responsibility for making a tough judgment.

A few versions might be built and subjected to empirical testing. Sometimes a blend of interaction styles can help resolve the problem. For example, maybe the user is presented with a form fill-in for the departure and arrival cities. Those who wish to can type in the city name or the short three letter airport code if they know it. Alternatively, the placement of the cursor in the field may generate a pop-up window with a menu of cities that can be selected by pointing (mouse or arrow keys). Then the selected city name is automatically displayed in the form. Form fill-in can be nicely supplemented by pop-up menus for less knowledgeable users.

Similarly, command language systems can be supplemented by a form fill-in or menu selection strategy for less knowledgeable users. For example, many text editors (e.g., Wordstar or Finalword) start with a command language approach that has a hierarchical structure. After pressing the first key for the first part of the command, the user can immediately type more characters to complete the command, but if they are less familiar with the commands a menu will appear automatically after 4 – 6 seconds.

Some users prefer to make the menus visible all the time, others prevent them from ever appearing. A quite similar approach, that might be called command menus, has been implemented in LOTUS 1-2-3. The top line of the screen shows a menu with items that are selected by arrow keys or typing the first letter. Then another one-line menu appears, or arguments are typed in, going down as many as five levels in the tree structure. Novice users walk slowly through the menus while experts type ahead rapidly enough that the menus appear on the screen for only a fraction of a second.

Another blending approach is to offer a form fill-in and allow the knowledgeable user to type a long sequence of commands in the field. Direct manipulation systems often revert to menu selections approaches when a visual representation cannot be found. Some direct manipulation systems offer a menu of visual items, such as a colour or texture palette. Tennant, Ross and Thompson (1983) offer an imaginative blend of natural language and menu selection. The users select from a menu of syntactically and semantically acceptable sentence fragments. As each choice is made, the permissible sentence fragments change. With a rapid display and pointing device, the user can quickly assemble a syntactically and semantically correct query for a database. These are but a few of the possibilities. Identifying useful and appropriate blends for different users in different situations is an important goal for researchers and system designers.

4. Conclusions
Designers of interactive systems have had to work from their own experience and intuition, validating their designs by costly and time consuming iterative testing. There is some hope that we can more precisely understand when to apply a particular interaction style and how to refine it so that it produces rapid user performance, low error rates, high satisfaction, ease in learning, and ease in retention over time. Blends of interaction styles can be very effective in serving a range of users and in dealing with certain interaction tasks. A massive program of empirical research will be necessary to guide this research, but the payoff in more reliable and effective designs can be enormous. Finally, a useful psychological theory, tied to the interaction styles would be a tremendous benefit.


