Usability Guidelines for Interactive Search in Direct Manipulation Systems

Robert St. Amant and Christopher G. Healey
Department of Computer Science
North Carolina State University
EGRC-CSC Box 7534
Raleigh, NC 27695-7534, U.S.

Abstract

As AI systems make their way into the mainstream of interactive applications, usability becomes an increasingly important factor in their success. A wide range of user interface design guidelines have been developed for the direct manipulation and graphical user interface conventions of modern software. Unfortunately, it is not always clear how these should be applied to AI systems. This paper discusses a visualization assistant, an e-commerce simulation domain we have applied it to, and the guidelines we found relevant in the construction of its user interface. The goal of this paper is to explain how an interactive system can incorporates search-based intelligent behavior while still respecting well-established rules for effective user interaction.

Keywords: user interfaces, expert systems, simulation

1 Introduction

Designing a good user interface may appear to be straightforward, especially with the help of a user interface builder, but this ease is deceptive. The construction process involves much more than adopting the surface conventions of graphical user interfaces, mapping inputs to menus and type-in boxes, outputs to icons and related graphics. Good graphical user interfaces also follow heuristic guidelines that govern almost every aspect of design, from the manipulation properties of widgets to the decomposition and organization of tasks to the overall visual layout of an interface [Dix *et al.*, 1998]. Determining how a general guideline should be applied in any given situation is notoriously difficult.[1]

This is a newly important issue for AI, as search-based techniques find their way into conventional interactive applications. Interactive AI systems usually depend on *direct manipulation* in some form, by virtue of their integration into modern software environments. Direct manipulation systems rely on continuous representation of objects of interest, physical actions or labeled button presses for commands, and rapid, incremental, reversible operations with visible feedback [Shneiderman, 1998].

Unfortunately, intelligent behavior sometimes makes demands on the interaction process that do not naturally fit into a direct manipulation framework [Shneiderman and Maes, 1997]. AI systems have traditionally treated interaction with a user as a kind of *communication*. In contrast, direct manipulation relies on what Hutchins calls *model world* interaction [Hutchins, 1989]: the interface provides an environment and a set of tools that the user directly applies to reach a problem solution. More concisely, the user acts through a direct manipulation interface, rather than talks to it. The difference between the communication and model world paradigms is more than skin deep, in that what constitutes an effective set of rules for interaction under one paradigm may be drastically inappropriate under the other. For example, in a collaborative problem-solving process, it is common for agents to negotiate about the appropriate means to solve a problem. In contrast, tools do not negotiate with their users; they simply perform their assigned tasks. If the effectiveness of an interactive system depends on simple tools that give the user predictable, consistent behavior, then redesigning the tools so that they can communicate and negotiate with the user about their use is likely to degrade the overall usability of the system.

This is not to say that agents cannot behave as tools, or that tools cannot be extended to incorporate some agent-like behavior (e.g. [Anderson *et al.*, 2000; Horvitz, 1999; Lieberman, 1995; Maes, 1994]). On the contrary, we believe that the line between agents and tools will continue to blur. A key issue in this evolution is whether we understand the differences in user interaction that the agent-based and tool-based approaches require. With this understanding we can make better decisions about how conventional user interface guidelines should be adapted to unconventional (from a human-computer interaction perspective) application properties, such as a reliance on search.

This paper lays out a set of design guidelines for incorporating a search process into a conventional direct manipulation interface. We describe several design principles from the HCI literature and work out their general implications for systems that provide intelligent assistance through search. We further discuss how these implications affect the design of a search-based system for visualization.

[1]Even commercial applications sometimes suffer from serious usability problems; examples are found in textbooks (e.g. [Dix *et al.*, 1998]), popular works (e.g. [Raskin, 2000]), and on-line (e.g. `http://www.iarchitect.com/mshame.htm`), as well as in the technical literature (e.g. [Thimbleby, 2000]).

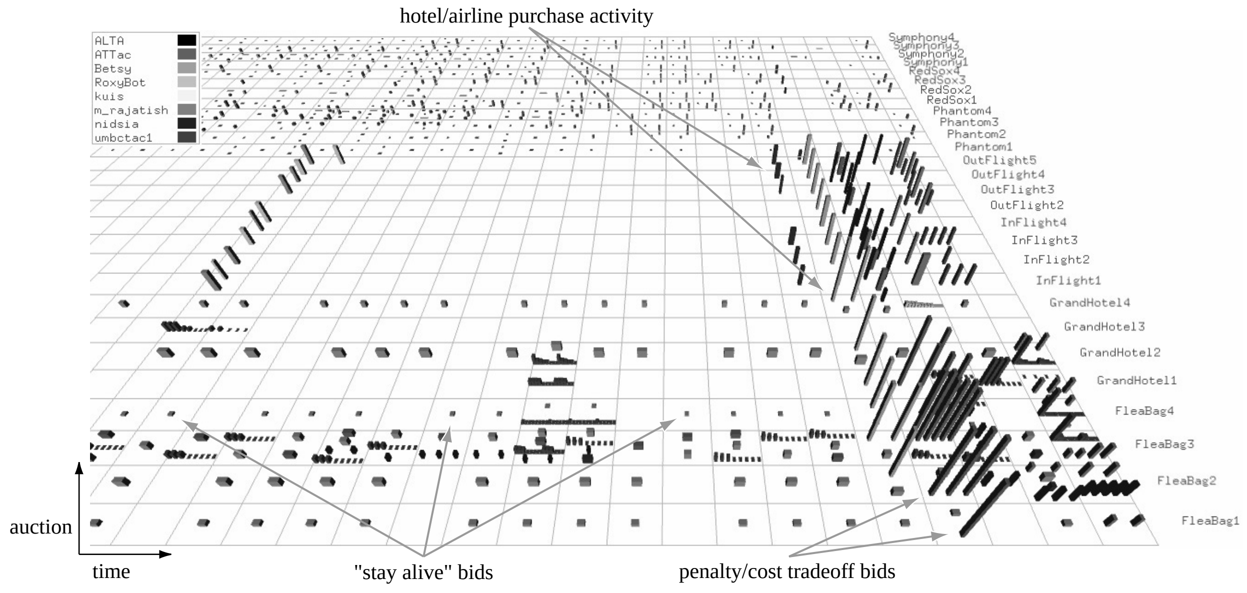


Figure 1: ICMAS ’00 TAC data visualization (in grayscale), with *agent ID* $\rightarrow$ *color*, *price* $\rightarrow$ *height*, *quantity* $\rightarrow$ *width*

2 A visualization domain

We begin with a example, using a problem domain that illustrates our motivation as well as producing results of general interest in AI. ViA is an intelligent assistant for building scientific visualizations. Scientific visualization is the conversion of collections of strings and numbers, or datasets, into images that allow viewers to perform visual exploration and analysis. Very large datasets with millions of data elements representing multiple independent data attributes are not uncommon. The challenge is to design methods that present some or all of this information simultaneously in a single display, without overwhelming a viewer’s ability to make sense of the resulting images. The choice of which visual features to use (e.g. color, size, or contrast) to represent each data attribute is called a data-feature mapping. ViA takes as input a dataset, a short description of the dataset’s properties, and a set of viewer-defined analysis tasks, and produces as output a set of appropriate data-feature mappings.

One current testbed application, the Trading Agent Competition,[2] illustrates the goal of ViA. The TAC is a simulated e-commerce auction environment run on the Michigan Internet AuctionBot platform.The AuctionBot is a TCP-based auction server that implements numerous types of auction rules. This allows the simulation of a wide variety of market games. Intelligent auction agents are designed and tested within these markets to study different buying and selling strategies.

During the TAC, each agent acts as a “travel advisor” whose goal is to assemble a travel package for eight fictitious customers. A travel package consists of a round-trip flight from TACtown to Boston, a hotel reservation, and tickets to entertainment events (a baseball game, the symphony, and the theater). Customers specify preferences for the different aspects of their trip: which days they want to be in Boston, the type of hotel they prefer (economy or luxury), and the entertainment events they want to attend. Obvious dependencies must be met; for example, customers need hotel rooms for the duration of their trip, and can only attend entertainment events during that interval. The goal of the agent is to maximize the total satisfaction of its customers (i.e., the sum of their utility functions).

ViA was used to identify effective real-time visualizations for agent activity in a TAC run during the ICMAS ’00 conference. Five separate attributes were selected by hand for visualization: the *time*, *auction ID*, *agent ID*, *price*, and *quantity* for every bid made during the simulation. *time* and *auction ID* were used to define a bid’s x and y-position on a two-dimensional grid. Perceptual texture elements (or pexels [Healey and Enns, 1999]) that vary in their color (combined hue and luminance), height, density, and regularity of placement were used to represent the remaining attributes: *agent ID*, *price*, and *quantity*.

TAC competitors, acting as domain experts, assigned normalized importance weights of 1 (very important) to *agent ID*, and 0.5 (somewhat important) to *quantity* and *price*. They also defined the analysis tasks (searching for specific agents, identifying price boundaries, and estimating the relative frequency of particular quantities) they wanted to perform. ViA automatically identified additional properties like the spatial frequency and value range of each attribute in the dataset. This application-independent information is used together with a collection of perceptual guidelines to select the mappings that are most appropriate for TAC visualizations.

Based on dataset properties and viewer-imposed restrictions, ViA produced several perceptually salient visualizations. Figure 1 shows a modified version of a ViA-generated mapping, with *agent ID* $\rightarrow$ *color*, *price* $\rightarrow$ *height*, and *quan-*

[2] `http://tac.eecs.umich.edu`

tity $\rightarrow$ *width*. Rectangular towers are used to represent each bid made during the game. *time* and *auction ID* define a tower's location on the underlying grid. Time increases from left to right along the horizontal axis. Each row corresponds to a separate auction. A tower's color, height, and width show the bid's *agent ID*, *price*, and *quantity*, respectively. Different colors identify bids by different agents. Buy bids lie above the grid, while sell bids lie below the grid; a higher buy (or sell) price increases the height of the tower. Bids for larger quantities produce wider towers. Simultaneous bids made by an agent in a particular auction are displayed as a horizontal row of towers in the appropriate grid cell (each with its height and width defined by the particular bid's *price* and *quantity*). Bids made by different agents at the same time in the same auction are shown as rows of towers drawn on above another in a common cell. This arrangement uses spatial density to represent the level of bid activity at different locations in the game.

Many aspects of the agents' strategies and game play can quickly be identified using such visualizations. For example:

1. Some agents periodically made very low buy bids for hotel rooms to ensure the hotel auctions "stay alive" (hotel auctions automatically close after a period of inactivity).
2. Most agents deferred purchasing hotel rooms and airline tickets until just before the simulation ended, judging there was no advantage to early purchase (particularly for hotel rooms, where attempts at early purchase can drive up the final price).
3. If hotel rooms for a given customer cannot be found, the customer's entire trip is canceled, and the agent is penalized the cost of any airline and entertainment tickets they may have purchased on the customer's behalf. Some agents estimated the marginal cost c of this penalty, then made late bids for hotel rooms at a buy price of c. These agents decided that paying c for a hotel room was no worse than paying a penalty of c for unused airline and entertainment tickets. More importantly, there is a good chance that the hotel rooms will sell for less than c. If this happens, the agent will make a profit relative to the scenario of not securing the hotel room.

ViA casts scientific visualization as a straightforward search problem: finding a mapping between dataset attributes and visualization features that respects dataset and user constraints. ViA's exploration, at its core, is an incremental best-first search. The evaluation function is composed from the results of a set of critics. A critic exists for each visual feature available for use in a mapping (e.g. a hue critic, a height critic, and so on). The critic examines the data attribute associated with its visual feature, and reports on its perceptual correctness [Healey and Enns, 1999]. Critics also generate search operators for improving particular associations. ViA uses the critics' results to compute an overall evaluation of its current mapping, and to extend the search in the direction of modifications that should produce better mappings.

Our discussion up to this point has skirted the issue of user interaction. Even for the TAC domain, which poses a very small problem in computational terms, the design of an interface for ViA must address a number of nontrivial questions. How can user preferences be taken into account to guide the search process? What kind of incremental feedback should the system provide, for lengthy processing times? Should the user see all of the solutions ViA generates, or only the best? Answers are provided in the HCI literature, but require interpretation to be applied correctly.

3 Interaction design for an AI system

The HCI literature contains dozens of general principles for user interface design, leading to thousands of detailed guidelines. For example, Smith and Mosier produced an early set of over 900 guidelines for text-based interfaces [Smith and Mosier, 1986]. The sophisticated user interfaces of Macintosh and Windows systems are driven by a small set of high level principles, including metaphor, direct manipulation, user control, and consistency, that expand to much more detailed guidelines. We draw on a recent concise summary of concepts due to Dix et al [1998]. The category of *learnability* in an interface includes predictability, synthesizability, familiarity, generalizability, and consistency. *Flexibility* includes dialog initiative, multi-threading, task migratability (i.e., from the user to the system, and vice versa), substitutivity of input and output forms, and customizability. *Robustness* includes observability, recoverability, responsiveness, and task conformance. Some of these concepts are applicable to all user interfaces. Those we consider in this section pose some novel requirements that we have not seen explicitly compiled and discussed in the AI or HCI literature.

Let's consider a hypothetical interactive AI system, a generalization of the ViA system, to see how these guidelines should be interpreted. The system is given a problem that it must solve through search. Its evaluation function, while accurate to an approximation, is incomplete; the search process requires input from the user to reach the best results (e.g., prefer *this* data-feature mapping to *that*, due to domain-specific interpretations of color.) This means that the system must show the user some representation of the state space, in order to elicit feedback and guidance. This interaction requirement in turn means that a significant part of the system's design must be devoted to managing interaction with the user. Unlike conventional applications, an AI system takes actions that are determined as much by the properties of the state space as by the actions of the user. The interaction is thus more likely to be opportunistic, less likely to be predictable with respect to the type of information exchanged, the duration of the entire task, how control of initiative shifts between the system and the user, and related properties.

Given this broad description, we now turn to an interpretation of the usability concepts listed above, organized into four broad areas. Following these guidelines generally improves conventional interfaces; we show how they can be applied to interactive AI systems.

Incremental processing is an important property for an interactive AI system, if for no other reason than that the user must be given some opportunity to contribute to the problem solving process. This general point has several related components.

Responsiveness. An interface is responsive when the response times of its operators match the user's expectation. Some (though not all) usability studies have found that consistency in response time is preferable to raw speed [Myers, 1985]. For example, users will generally prefer a constant response time (i.e. search duration) of, say, five seconds, to responses with a mean time of four seconds but varying between zero and eight seconds. For a search-based system, this suggests a design that combines anytime processing with continuous computing concerns [Horvitz, 1997].

Task migratability. Task migratability is supported when problem solving responsibility can be handed off from the user to the system, and vice versa. This suggests that the system and the user should have access to a common set of search operators (though user operators will often be abstractions or compositions of system operators.) This kind of facility is supported, for example, by scripting and end-user programming in conventional software, and by some systems for programming by example (PBE) [Lieberman, 2001]. Migratability is impaired if a search-based system can reach states that the user cannot reasonably evaluate, or if the user would like to override specific system behavior but cannot.

Observability implies that the user is able to infer internal properties of the system's state from its external representation. An important aspect of observability in an AI system is making clear the "maturity" of a solution. The pitfall to avoid is the system prematurely focusing the user's attention on an early solution that is likely to be superseded by a later incompatible solution. This can be viewed from the system side as a search horizon issue, from the user side as a potential anchoring problem.

Dialog initiative. An interface respects dialog initiative when problem-solving initiative can shift between the system and the user to follow the task. As with task migratability, mixed dialog initiative should be supported by presentation of information at an appropriate level of detail for the user to make a meaningful contribution.

Adaptation refers to the dynamic adjustment of the system's behavior to the user's actions. This adaptation can occur at the direction of the user or automatically.

Customizability entails that an interface be adaptable to the abilities and needs of the user. This kind of customization, for a search-based system, can be thought of as modification of preferences that influence the system's operational characteristics. Candidates for customization include the number of states incrementally searched, and the number of potential solution states retained and presented.

Search adaptation is a more important point. Suppose that the system presents the user with the current best state s_i, as determined by its evaluation function f. The user reviews this information and decides that a modification is appropriate, leading to a state s_{i+1} that the system has already considered internally and given a lower value. When the system resumes its search, it must modify f or some state property to avoid again indicating that s_i is a better solution. A further useful capability is generalization of the differences between s_i and s_{i+1} to support comparisons of other states that have not yet been evaluated or presented; some PBE systems have this capability.

Coherence. Although direct manipulation systems tend to support a style of interaction in which operator sequences are short and goals interact as little as possible, an incremental search process necessarily involves maintenance of context between exchanges with the user.

Predictability and consistency are two important aspects of coherence. An interface is predictable when a user can determine, ¿from a specific operation in a given state, what the consequences will be. Consistency promotes predictability. Direct manipulation systems tend to present a comprehensive visual environment to the user, with the assumption that the user can judge relevance better than the system. In an incremental search, a system might reserve some information that is simultaneously less likely to change and less likely to be immediately relevant. Instead of giving the user a complete representation of the best n states, for example, the system might limit its display to a few selected properties, leaving itself "wiggle room," space to maneuver.

A related issue deals with predictability in response to user actions. To continue the search adaptation example, suppose that the user has selected a specific operator p to improve state s_i, and that p has been effective in the past, but this time it is not. If the reason for this is not apparent (an observability issue), the user's confidence in the effectiveness of p or the system's evaluation function may suffer. Additional explanation may be warranted to improve predictability.

Support for user orientation. One of the difficulties implied in the foregoing discussion is that of managing interaction between the system and the user, such that each party can make real contributions. Many of these difficulties can be understood in terms of user orientation during a navigation process [Kim and Hirtle, 1995].

Reachability between states in an interface generalizes the idea of recoverability; it implies that from any state, any other state can be reached, in particular non-error states from error states. For an intelligent interactive system this goes beyond the completeness of a search algorithm. In addition it means that the user can recall and re-visit past states, to review and re-evaluate earlier decisions. Navigational support (e.g. generation of landmarks) can improve orientation for this task.

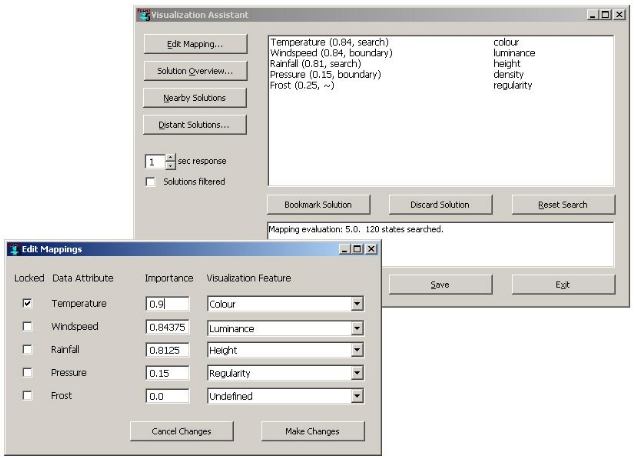
Synthesizability can be understood as the ease with which the user can form a conceptual model of the problem-solving process. In an interactive AI system, part of synthesizability means that the system partitions the state space so that the user can grasp individual portions more easily. In navigation terms, this kind of synthesizability is supported by region differentiation.

Property	*ViA mechanism*
Responsiveness	Response time tailored to platform
Dialog initiative	User controlled incremental search
Task migratability	External availability of operators, executable between search epochs
Customizability	User-editable preferences for search step size and filtering of results

Table 1: Usability properties in ViA search

Property	*ViA mechanism*
Predictability	Clustering and filtering of best states; presentation of partial state information
Synthesizability	Allowing user to explore "nearby" and "distant" solutions of comparable value, with region partitioning by windows
Observability	Progress messages and navigation support
Reachability	Navigation support

Table 2: Usability properties in ViA presentation



1 sec response

☐ Solutions filtered

Temperature (0.84, search)	colour
Windspeed (0.84, boundary)	luminance
Rainfall (0.81, search)	height
Pressure (0.15, boundary)	density
Frost (0.25, ~)	regularity

Mapping evaluation: 5.0. 120 states searched.

Locked	Data Attribute	Importance	Visualization Feature
✓	Temperature	0.9	Colour
☐	Windspeed	0.84375	Luminance
☐	Rainfall	0.8125	Height
☐	Pressure	0.15	Regularity
☐	Frost	0.0	Undefined

Figure 2: ViA user interface

4 Interaction with ViA

In the user interface to ViA we are exploring the implications of these guidelines. A functional prototype is shown in Figure 2, with an intermediate step in the search for data-feature mappings visible. (This section describes a weather dataset rather than the TAC dataset, which is too small to exercise some of the capabilities of the interactive system.) ViA's interface design accounts for most of the concerns identified in Section 3, although not with complete generality. Relevant design features can be divided into those that affect the search directly, as summarized in Table 1, and those that affect presentation and user interaction, as shown in Table 2. These design features will appear to be straightforward, even obvious, for the most part, but it is surprising how often they are neglected in interactive AI systems; hence the need for guidelines.

ViA's search is incremental, with a default but editable response time of about a second. The search step size is computed automatically when the user loads an initial dataset description, by running the critics on the dataset to gain timing estimates for the current platform. The user can incrementally modify the mappings that the system returns; these editing capabilities provide most of the functionality of ViA's search operators.

In ViA's domain, a search usually produces several candidate solutions with essentially equal evaluation results. The system maintains a list of the best ($n = 20$) solutions it has encountered in its search, and performs a simple clustering on the contents of the list at the end of each search epoch. If a cluster is found among the top-ranked solutions in which the majority of attributes are assigned to the same features, then the remaining, differing assignments are stripped out and only the common assignments of the solutions are presented. This capability is intended to focus the user's attention on the more influential assignments; whether the filtering occurs at all, however, is under user control.

ViA's search can be directed toward solutions that are near the currently displayed mapping, or those that are distant. When the user chooses to search nearby solutions, the system selects the best solution it finds that is nearest to the solution most recently presented. Distant solutions are handled by selecting farthest solution among the top candidates. This organization of search results is a compromise between showing all promising solutions at once and showing them one at a time in an arbitrary order. The nearby/distant distinction allows the user to conceptually organize the space of mappings into distinct regions during its traversal. Toward this end, a distant solution operation generates results in a new window, to create a new visual "context" for the search. As another navigational aid, the user can bookmark or discard individual solutions to impose further structure on the space.

A global search overview shows all of the presented solutions, across all nearby and distant searches. Currently the overview information is presented as text in outline form, but the intention is to build an annotated graphical representation of the search space, to show the non-linear paths that the user may have taken through it. Other areas of incomplete or missing functionality include the adequacy of explanations provided to the user (explanations currently only describe critic results and the number of states searched), the completeness of the set of search operators available to the user (discretization and task removal operators are missing), and functionality related to search adaptation.

The interface that has resulted from our work supports a more flexible, less burdensome interaction with the search component of ViA than is provided by conventional means (e.g., interaction through a command line, within a text editor, or through a simpler graphical interface dialog.) As with many systems in the intelligent user interfaces and HCI literature, however, the interactive aspects of ViA outlined above have not been evaluated in a formal sense. The application discussed here should not be treated as validation of ViA's interface design. Rather, our examples act as illustrations and an early means of formative evaluation. Evaluating a user interface of any complexity is an extensive undertaking and remains yet to be done for ViA.

5 Discussion

Our work is partly inspired by earlier work on interface softbots, which operate through the user interface of an application, rather than a programmatic interface [St. Amant and Zettlemoyer, 2000]. Interface softbots are motivated by a claim that characteristic properties and behaviors of a graphical user interface can be exploited by planning agents, due to a similarity between planning assumptions and user interface guidelines [St. Amant, 1999]. In this paper we take a more conventional route in showing how such interface guidelines can be applied to improve interactive AI systems.

Otherwise, usability for interactive AI systems has been a small but active area of research [Hendler and Lewis, 1988; Höök, 2000; Kaasinen, 1998]. Horvitz provides a representative study, presenting a set of principles for mixed-initiative user interfaces [Horvitz, 1999]. As is commonly the case, these extend beyond conventional guidelines for direct manipulation systems, encompassing such issues as social behavior and the explicit use of dialog. Our intention is not to describe new guidelines for new technology, but rather to clarify how new techniques can be fit into an existing interaction paradigm, and to explain how existing guidelines apply.

Acknowledgments

Thanks to Peter Wurman for his contribution of TAC material and to three anonymous reviewers for helpful comments on the draft of this paper. This research was partially supported by the National Science Foundation, award numbers IIS-9988507 and IIS-0083281. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copyright notation hereon.

References

[Anderson *et al.*, 2000] David Anderson, Emily Anderson, Neal Lesh, Joe Marks, Brian Mirtich, David Ratajczak, and Kathy Ryall. Human-guided simple search. In *Proceedings of AAAI*, pages 209–216. AAAI Press, 2000.

[Dix *et al.*, 1998] Alan J. Dix, Janet E. Finlay, Gregory D. Abowd, and Russell Beale. *Human-Computer Interaction*. Prentice Hall, 2nd edition, 1998.

[Healey and Enns, 1999] Christopher G. Healey and James T. Enns. Large datasets at a glance: Combining textures and colors in scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(2), 1999.

[Hendler and Lewis, 1988] James Hendler and Clayton Lewis. Introduction: Designing interfaces for expert systems. In James Hendler, editor, *Expert Systems: The User Interface*, pages 1–14. Ablex, 1988.

[Höök, 2000] Kristina Höök. Steps to take before intelligent user interfaces become real. *Interacting with Computers*, 12(4):409–426, 2000.

[Horvitz, 1997] Eric Horvitz. Models of continual computation. In *Proceedings of AAAI*, 1997.

[Horvitz, 1999] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of CHI'99*, pages 159–166, 1999.

[Hutchins, 1989] Edwin Hutchins. Metaphors for interface design. In M. M. Taylor, F. Neel, and D. G. Bouwhuis, editors, *The Structure of Multimodal Dialogue*, pages 11–28. North-Holland, Elsevier Science Publishers, Amsterdam, 1989.

[Kaasinen, 1998] Eija Kaasinen. Usability issues in agent applications: What should the designer be aware of. Technical report, USINACTS, 1998.

[Kim and Hirtle, 1995] Hanhwe Kim and Stephen C. Hirtle. Spatial metaphors and disorientation in hypertext browsing. *Behaviour and Information Technology*, 14(4):239–250, 1995.

[Lieberman, 1995] Henry Lieberman. Letizia: An agent that assists web browsing. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 924–929, 1995.

[Lieberman, 2001] Henry Lieberman, editor. *Your Wish Is My Command: Programming by Example*. Morgan Kaufmann, San Francisco, CA, 2001.

[Maes, 1994] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):31–40, July 1994.

[Myers, 1985] Brad A. Myers. The importance of percent-done progress indicators for computer-human interfaces. In *Proceedings of CHI'85*, pages 11–17, 1985.

[Raskin, 2000] Jef Raskin. *The Humane Interface: New Directions for Designing Interactive Systems*. Addison Wesley, Reading, MA, 2000.

[Shneiderman and Maes, 1997] Ben Shneiderman and Pattie Maes. Debate: Direct manipulation vs. interface agents. *Interactions*, 4(6):42–61, November/December 1997.

[Shneiderman, 1998] Ben Shneiderman. *Designing the user interface: strategies for effective human-computer interaction*. Addison-Wesley, 1998.

[Smith and Mosier, 1986] Sidney L. Smith and Jane N. Mosier. Guidelines for designing user interface software. Technical Report ESD-TR-86-278, The MITRE Corporation, Bedford, MA, 1986.

[St. Amant and Zettlemoyer, 2000] Robert St. Amant and Luke S. Zettlemoyer. The user interface as an agent environment. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 483–490, 2000.

[St. Amant, 1999] Robert St. Amant. User interface affordances in a planning representation. *Human Computer Interaction*, 14(3):317–354, 1999.

[Thimbleby, 2000] Harold Thimbleby. Calculators are needlessly bad. *International Journal of Human-Computer Studies*, 52(6):1031–1069, 2000.