How do we interact with an agent: A perspective from an undergraduate human factors class

D. Scott McCrickard May Wing-Sze Cheng
Graphics, Visualization, and Usability Center
and College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
Email: {mccricks,mayc}@cc.gatech.edu URL: http://www.cc.gatech.edu/~mccricks/

Abstract

Undergraduate human factors classes explore guidelines for the design of user interfaces. However, these guidelines may not apply when dealing with agents, which run constantly without explicit direction from a user. An assignment for Georgia Tech undergrads asked them to design an interface for an agent and discuss the guidelines that were most and least applicable. This paper reviews some of their solutions.

Keywords: Information agents

1 Introduction

A software agent is a system that performs tasks in an autonomous manner in order to satisfy the needs of its users. By autonomous we mean that the agent operates for long periods of time without direct user intervention. We are particularly interested in information filtering agents that collect and process information for delivery to a user.

A number of HCI issues have emerged from the increased development and use of these agents. How should agents present the information that they collect to their users? How should users guide and direct their agents? To find the answers to these questions, we presented an agent design assignment to the students in a senior-level human factors class at Georgia Tech.

The human factors class is cross-listed in the College of Computing and the School of Psychology at Georgia Tech, so it attracts students with a wide variety of skills and backgrounds. All have had some introductory psychology and computer programming courses, but beyond that there are students from mechanical engineering, industrial engineering, information design, as well as computer science and psychology. We expected that this mix of students would result in interesting and thought-provoking designs.

As is common in human factors classes, we examined various rules for the design of interactive systems relating to consistency, cognitive load, color, animation, and so forth. Most textbooks on the subject contain a discussion of these rules [2]. However, these rules were designed with "typical" programs in mind, where a user starts the program, uses it to accomplish some task, then exits the program. Agents do not fit this paradigm: they run constantly without explicit direction from the user, and they may require input or generate output at unpredictable times.

The students were asked to design and implement an agent that would display online news articles taken from the Yahoo news page. This page is updated constantly during the day, making it difficult for a user to keep up with all of the new articles. In an accompanying writeup, the students were asked to consider how agent interfaces are similar and different from typical program interfaces and to discuss the rules that were most and least applicable to their design. This paper presents some of their designs and summarizes their writeups. A complete description of the assignment along with selected results from the students can be found on the Web at www.cc.gatech.edu/classes/cs4753_98_winter/homeworks/hw2.html

2 Rules for agent design

This section of the paper examines the rules that students thought were important in designing an agent. Ironically, the same rules that many students found important to follow, others found necessary and useful to break. We will summarize the discussions for each of these rules using text and examples from the student submissions.

Screen space One rule followed by most students was to minimize screen real estate with the display and interface. The sentiment was that the agent should be a small desktop application that can be tucked away in a corner so that the rest of the screen can be used by other applications. All too often it was the widgets that display text that suffered in size – a user only had a few lines to read the text.

A few students chose not to follow this rule. They felt it was more important to show the text in a large, readable manner than to try to cram information into a small space. The interface could be iconified when not in use, then raised when the user wants to read an article.

A compromise was to provide a small alert box that shows when new articles arrive, then to display a window when the user clicks on a button or icon. The pop-up window could then be dismissed when the user finishes reading an article. The user would still be alerted to the presence of new articles but would not have to sacrifice screen space at all times. An example of this type of interface is shown in Figure 1. One student went a step further and allowed users to jump to the article Web page using a Netscape browser, thus allowing a user to get the full multimedia effect using an application designed for displaying articles (see Figure 2).

Knowledge in the world "Put knowledge in the world" is an oft-stated rule found in Don Norman's *Design of Everyday Things* [1]. Knowledge in the world refers to the hints, labels, and signals in interfaces that inform or remind us of how they work. By using aids that users have seen before, an interface can be easy to learn and use.

Most students designed agent interfaces that are similar to those found in existing interfaces, with explanatory labels and well-established interface rules. They argued that the similarities with existing interfaces would result in an agent that is easy to use for first-timers. As a result, many of the designs consisted of a list of articles, a text display area, and a series of buttons for navigating through the list. However, several good arguments were made for violating the well-known behavior of interfaces by having common actions produce unusual results.

- Resizes reveal additional information: Since different users have different amounts of space that they are willing to provide for the interface, one student varied the number of widgets and the amount of available information based on the amount of space that was provided to the user. Then, if users want to see more information, they can make the interface bigger, and if they want more space for other windows, they can shrink the interface while preserving some of the information. See Figure 3 for an example.

- Buttons double as displays: Typically the button widget has a consistent and non-changing label, but because space is at a premium, several students used buttons to show information about a change as well as to provide functionality when pushed. Figure 2 shows an example of this.

- Keyboard controls rather than space-consuming buttons: Several students provided keyboard controls in addition to or even instead of space-consuming widgets. A user could hide the widgets and use keystrokes to access the interface, even though this necessitates "knowledge in the head" (a user must remember the keystrokes since the visual clues from buttons and other widgets are not present).

Although unusual actions produced interesting results in many of the designs, students argued that simple and unobtrusive error handling would encourage users to experiment with the interfaces and discover these features. They did not provide any functionality to the user that was not easily reversible except allowing them to exit the interface. Since

users do not perceive a threat, they would be more likely to play with the interface and discover the new functionality.

Use of color and animation Color and animation are two common techniques for attracting the user's attention. Rules dictate that up to four colors can be used to highlight interesting items, with other colors used for rare and important occasions. Animation (such as blinking, continuous scrolling, or changes in color) should be used sparingly to avoid annoying the user. All too often amateur designers will overuse these techniques, so we were worried that the class agents would reflect this. While we did see some poor uses, in several places it was used in an interesting and informative manner.

Color was used to highlight important or dangerous widgets. The benefits of this is questionable – the positioning generally directs novices to the correct action, and advanced users tend to not need and even actively dislike this use of color. As seen is Figure 2, color change can be used to indicate that articles have been updated. However, it is unclear that a user would immediately realize that if a button is one color (red) it means there are new articles and if it is another (blue) it means there are not.

Two animation widgets, a ticker-tape widget and a fade widget, were provided to the students (see Figure 4). Often they used them in places where a simple change in text or inversion of colors would suffice. However, several used a tickering display of headers or fading between the headers could alert the user of the information contents without requiring button presses or other physical effort.

3 Conclusions

This paper presented several information delivery agent designs created by undergraduate students. On occasion students violated certain well-established rules for interface design to support the unique nature of agent-style programs. For example, animated tickering and fading widgets were used to show more information without user interaction, and knowledge in the world was sacrificed to save space.

However, all too often the agent designs looked much like designs for typical non-agent programs. While these well-established designs should not be dismissed lightly, it is important to remember the fundamental difference in the way an agent works and consider how it impacts the manner in which user input is collected and information is displayed. An agent works autonomously without explicit direction from the user, but at any time it can identify interesting information that needs to be communicated to the user. The importance of certain information must be balanced by the needs of the user – an agent should display information when and only when a user needs it, ideally with a minimum amount of effort from the user.

References

[1] Donald A. Norman. *The Design of Everyday Things.* Currency Doubleday, 1988.

[2] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley, 1998.

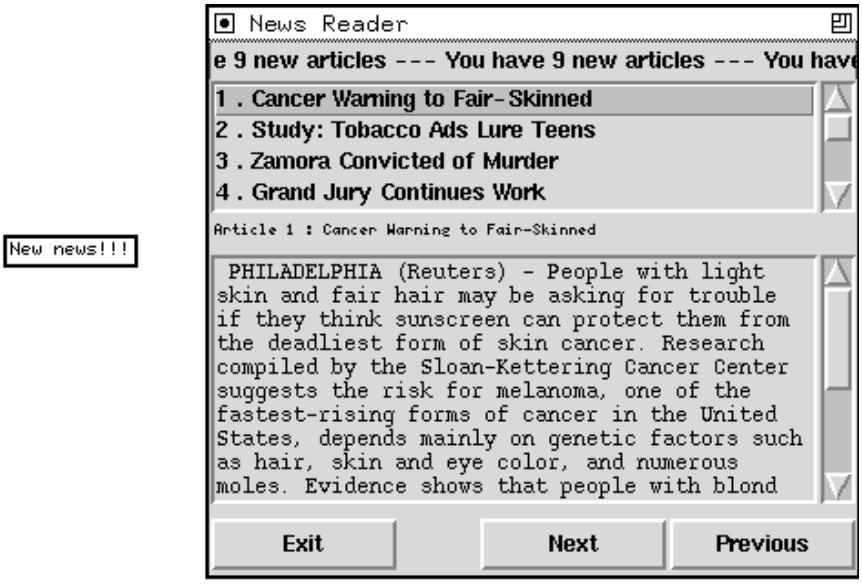

Figure 1: A news agent in iconified form (left) and expanded form. The designer suggests that users will typically keep the news reader in iconified form, and the icon (and an audio cue) will tell them when new news arrives. Users can click on the icon to expand the interface to its full form (at right) and read the articles.

Figure 2: A news agent (left) and its accompanying news viewer. The agent alerts users of new articles by changing the color of the image and displaying the "New News" message. The lower message is a ticker widget that scrolls through the news headers. If a user sees an article of interest, the news viewer can be raised by pressing the image. If a user wants more information or wants to view the article on the Web, the "Jump To Web" button will pull up the appropriate page in the Netscape browser.

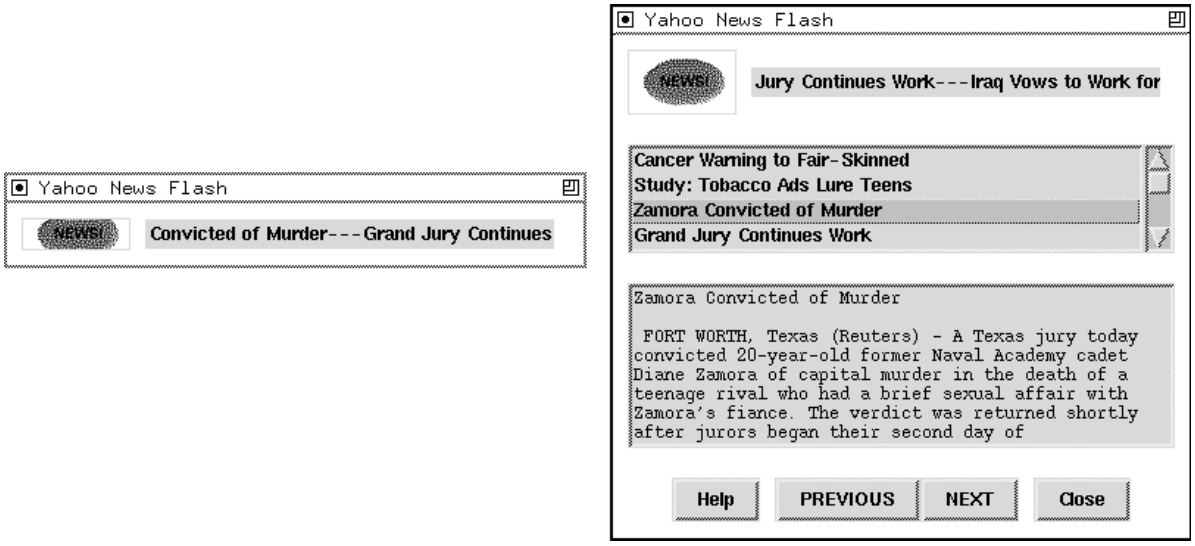

Figure 3: A news agent at a large size (right) and resized to a smaller size. The designer suggests that in general a user would keep the agent at the small size so it would fit in a corner of the desktop. When new articles arrive, the "News" image changes color and the tickering header display updates. If the user wants to read an article, the agent can be resized so the full interface is visible.

Figure 4: Five snapshots in the operation of a fade widget that displays the scores of baseball games. The first frame shows an initial block of text. The next two frames show how the text fades away into the background, and the final two frames show how the new text will appear in the same place.