RC24015 (W0608-015) August 3, 2006
Computer Science

IBM Research Report

WISE: a Wizard Interface Supporting Enhanced Usability

Joshua Hailpern
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

WISE: a Wizard Interface Supporting Enhanced usability

Joshua Hailpern
IBM T.J. Watson Research
joshua@hailpern.com

ABSTRACT

As computers become ubiquitous, the task of making them usable becomes increasingly important. Repeated failures by implementation experts to build usable systems have motivated the creation of successful design approaches by interaction designers. However, there are still many end user groups that have not successfully overcome the difficulties they perceive in computer use. Such groups include individuals with unique needs, such as physical or mental disabilities, and those with age-related challenges. As a larger percentage of Americans are considered "old" (60+), the lack of a system tailored to the needs of this age demographic has resulted in a part of the population that is disconnected from the rest of the world. The current state of software which targets older adults' ability to use computers focuses on physical issues while largely ignoring the volumes of gerontological cognitive research that could make computers not only usable but also intuitive for older adults.

This paper describes WISE, an alternative OS and application UI that specifically targets the cognitive deficits of older adults. WISE leverages both the existing body of research and the exciting Apple Cocoa API (Application Program Interface) for Objective-C and AppleScript to build a user interface layer tailored to the needs of older users. The focus was placed on a quality design by using an API and scripting language closely coupled to an operating system (OS), in this case Macintosh OS X. Further, WISE respects the guiding principles of linear interaction, effective cognitive strategy prompting, uniform limited scope, and accessibility as we explore simplified interaction methods to support older adults.

This paper is an expanded version of the extended abstract, by the same name and author, accepted to ASSETS 2006's student research competition. ASSETS is the international ACM SIGACCESS Conference.

Author Keywords

Gerontology, User-Centered Design, Linear Interaction, Effective Cognitive Strategy Prompting, Accessibility, Age-Related Challenges

ACM Classification Keywords

H5.2. User Interfaces

K4.2 Assistive technologies for persons with disabilities

1. INTRODUCTION

Accessibility is a major concern for most mass-produced computer systems in today's world. As a group, the number of people aged over 60 is growing at a record pace [3]. As commonly framed, accessibility/usability deals primarily with physical and sensory issues (e.g. changing colors or font sizes) but comparatively few systems make adequate use of gerontological cognitive research [3,10]. For older adults, smooth and productive computer usage requires a system that supports their objectives and works with knowledge of their possible cognitive challenges [10]. Thus the problem was framed as having two major aspects:

- Design an alternative to the OS and other applications' UI targeting older adults (65+).
- Use current gerontological cognitive research as guiding principles when making design decisions about the system.

To the end of making the system work, the decision was made to use the Cocoa environment to tie the interface to OS X in as seamless a fashion as possible. This decision was helpful in building a system with diverse functionality, but it also caused problems that we did not foresee at the start of the project. Some of these implementation issues will be addressed below.

The design aspect of the project was interesting. It grew in a more fluid fashion than the implementation itself. In the beginning, a system was envisioned encapsulating a simple word processor and web browser with some accessibility affordances (large text with serifs, large buttons, a single-click oriented interface, and vernacular terms for applications). However, during the process of researching cognitive functioning in older adults, a set of guiding principles emerged. These major ideas were:

- *Linear Interaction*: research by Craik and others indicates that memory retention for older adults is correlated closely to depth of processing. Split attention penalizes this depth and as such reduces retention for the aged [1,6]. The hypothesis is that a more linear interaction method might assuage some of these problems.
- *Effective Cognitive Strategy Prompting*: work by Hultsch suggests that older adults are less likely than young people to spontaneously use effective cognitive memory strategies such as imagery [5]. By providing prompting to use such strategies, he was able to raise the performance of older users. To further that line of thinking, similar strategy prompting (e.g. global metaphors) was considered while constructing the system.
- *Uniform, Limited Scope*: an overarching HCI design principle is that of uniformity in an interface. In this

new system, the goal was to build an interface that would be uniformly limited to a simplified interaction method requiring less cognitive burden to operate. This principle was where the least amount of research existed, allowing the most room for creativity in interface design.

- *Accessibility*: it would be a shame to lose accessibility features of OS X because there was to heavy a focus on redesigning and simplifying the workflow of interaction between older adults and the computer. As such, contextual affordances were built in that might help those users with reduced sensory function navigate our interface [7].

The interface of these design principles with the power of the Cocoa environment resulted in a system that called **W.I.S.E.**, *the **W**izard **I**nterface **S**upporting **E**nhanced usability*. WISE already is an easily extensible system that creates an interface tailored to the needs of older users based on gerontological research. Because the API and scripting language are closely coupled to the OS X operating system, the focus could be placed on quality of design by leveraging existing protocols and widgets

2. SOLUTION AND DESIGN RATIONALE

The body of existing gerontological cognitive research made it apparent to that an effective design must support an interaction with less cognitive stress than that presented by a traditional OS user interface. Thus, this solution is a linear, goal-oriented workflow described in more detail in the ensuing paragraphs.

2.1 Linear Interaction

Note that there are essentially two issues facing an older adult who wishes to use a computer. First, there is the walk-up-and-use aspect facing a novice computer user who needs to be able to find information without much background knowledge about the system itself. The second is the situation in which one uses a computer repeatedly for the same task. Older adult users can experience problems associated with 'walk-up-and-use' even in situations involving repeat use. It is hypothesized that this is due to memory limitations and age-related forgetting. Thus the solution needs to support both novice interaction and provide affordances to improve remembering or at least substitute recognition for recall [3,10].

A linear style of interaction seemed to provide a solution to both issues. Knäuper suggests that working memory declines somewhat with age and that parallel processing provides the most stress on the working memory system [8]. His research centers on the ordering of questions in a survey, but it is hypothesized that the ideas behind it extend to the ordering of tasks in a user interface. By limiting the required parallel processing to a minimum, we were able to build an inherent ordering to all of the processes allowable in WISE.

Accepted cognitive psychological research indicates that a high level of parallel processing reduces the depth of processing for any individual task in the parallel array. Craik's work extends this finding, noting that such division of attention affects older adults to a greater extent than it does younger people [1,3]. WISE attempts to abate this division by never requiring a user to attend to multiple windows at any one time. Further, there is a persistent history bar that tracks all of the user's choices from start screen to destination with both icons and text. Back navigation is thus also made easy: a user can see all of his or her choices and trace the path backwards to a previous location, and return to that state with a single mouse click.

Further research indicates that episodic memory, the ability to remember autobiographical information that occurred recently, is the most grossly impaired form of memory in late adulthood [2]. This deficiency can manifest itself in an inability to remember faces or other images, making the history bar support of linear interaction even more crucial. It may be difficult for an older user to remember the recent screens associated with a path from the start of an application to some goal—the persistent history bar removes the need to remember, providing constant cues to reinforce the correct interaction. A novice user can get from point A to point B simply by making choices. Hopefully, the history bar will reinforce this user's episodic memory of decisions until the action set becomes second nature.

There are problems with linear interaction: it can limit the number of actions that a person can do at one time. However, reducing the cognitive burden on the user is of primary importance. To alleviate this, WISE will save the user's place inside of a given task (e.g. typing a document), to allow the user to resume said task. Without divided attention, older adults novice can more quickly achieve his or her goals when using a computer.

2.2 Effective Cognitive Strategy Prompting

Linear interaction methods are simply an extension of accepted cognitive psychology principles for older adults. As such, they are effective, but were not surprise as the research was. Effective cognitive strategy prompting, however, was more unexpected. Research by Hultsch looked into the differences in organization of free recall by people of different age groups. His first result was expected: young people have a tendency to perform better at such tasks than older adults. It is his second result that is more exciting: when prompted with better cognitive strategies for remembering (such as mnemonics and imagery), older people improved at a much greater rate than young people, almost catching up their performance level to that of the young [5]

What this finding indicated to Hultsch was that older adults were less likely to spontaneously use effective cognitive strategies than young people. This would account for the increased benefit of providing such prompting to older subjects. Having seen this research, the idea of providing older adults with imagery, mnemonics, and metaphors for the tasks that they are attempting to accomplish was explored.

The first insight was to provide a goal metaphor for completing tasks in WISE. This provides several advantages. First, it avoids making possibly false assumptions about cross-generational computer views by abstracting away from technical applications towards goals that do not necessarily have to be accomplished with a computer [9]. Second, it allows an older user to consider their task verbally as a set of sub-goals. Since verbal memory tends to persist better than visual memory, this is a plus for the older user. The final and perhaps most important aspect of the goal metaphor is that by dividing a task up into a set of subtasks with discrete choices. WISE therefore provides an inherent organizational and effective cognitive strategy for common computational tasks. That is, WISE abstracts away from the application-centered view of modern operating systems, and "hides" the end process (thinking about "looking up a movie time" rather than remembering to use a web browser and the URL of a website, even though the end application/task is the same).

A major concern when using prompts is that there is a body of research that indicates that visual memory is the most problematic form of memory in older individuals. Extensive research by Winograd and Simon indicates, however, that while there is a real reduction in spontaneous pictorial encoding of information in memory, images can help older individuals to remember things more effectively. They conclude that most of this result comes from increased prompting for organization rather than the pictures themselves, so one would be wise to try multiple organizational prompts when building memory training for late adulthood [**Error! Reference source not found.**].

To this end, WISE uses redundant cues for improved organization in an attempt to leverage both the possibly failing visual memory and the more stable verbal memory; each task is identified by both by an icon and by a consistent naming scheme that is always visible in multiple locations (e.g. the history bar) following the suggestions of Czaja [3]. By abstracting away from the application-centered view of OS X, we are able to build temporal organization where there was none before. WISE then repeatedly prompt the user to follow this organization.

There is an even greater hope afforded by WISE and its commitment to providing effective cognitive strategies. Research by Poon et al. on the effectiveness of mnemonics for older adults concurs with Hultsch's work by showing that young people are more apt to use spontaneous effective encoding strategies [5,7]. However, this work goes one step further by demonstrating that repeated exposure to effective cognitive strategies can eventually allow older subjects to generate spontaneous effective organizational strategies [7]. Such a result is the ultimate end goal of WISE: by reinforcing simple interaction, it may eventually facilitate more complex and effective interaction between computers and older adult subjects.

2.3 Uniform, Limited Scope

It is essential that any project be scoped to a proper focus in order to make it tractable. It was realized early that the same principle could be applied to WISE in order to make using it a tractable problem for an older adult novice user. Thus it was decided to use paradigms already existing on the Mac as a starting point, then to simplify the interaction as much as was feasible. The final product managed to leverage the single button MAC mouse, and create an interface based exclusively on button selections, single clicks and text entry There is no concept of right clicking or double clicking for the standard navigation actions available in WISE. By reducing the palette of interaction methods, the cognitive load resulting from multiple equivalent navigating methods, is reduced. This was in this area of limitation and scoping that the least amount of gerontological cognitive research was available, so this hypothesis was based more heavily on standard user interface practices and intuition.

In the spirit of walk-up-and-use systems, the depth of interaction with the system and the methods necessary to successfully use WISE were reduced. There are no standard navigation paths with a depth of more than five screens (e.g. depth of 4: MainWindow $\Rightarrow$ Information $\Rightarrow$ News $\Rightarrow$ CNN). Further, the information theoretic entropy of the system was reduced, because there are fewer choices to make in the course of any given interaction.

The second intent of limiting the scope was to limit and unify the look of the interface. This is a common heuristic leveraged against any justified user interface. The interface was built in such a way as to allow the 'method of loci' to be an effective way to navigate and get information from the interface. Kausler indicates that location-based memory can be an effective way to associate and encode memory for older adults, so this idea was added to the look of WISE's interface [7].

As discussed above, there is a *persistent* history bar that tracks all of the user's choices from start screen to destination allowing easy backtracking. Each page has a consistently placed title bar describing the current step in the task path from start to goal. Further, there is an extensive set of tutorial pages accessible from the same location of every screen in the WISE interface. Once one learns the location of help, it is possible to access assistance from every screen directly. There are no drop menus. Everything relevant is visible at all times, foregoing recall in favor of the cognitive simplicity of simple recognition.

This uniformity was not trivial to achieve— a variety of widgets were used to build the interface. Unfortunately, they often had differing looks and feels. As the implementation section will discuss, much time was spent trying to understand and integrate widgets written in different languages and with different functional intents. Nevertheless, the hassle was worth it: WISE's uniformity is what makes it immediately usable. Without that ability, WISE would fail at its focus of supporting a group of users that are may perform like novices (even when they are repeat users).

2.4 Accessibility

The final design concern was accessibility, an aspect already covered by many interfaces and one that WISE cannot afford to lose, as usability is the primary focus. There is a great deal of anecdotal evidence that older adults often face sensory impairments. The interesting observation is that they can often compensate for these limitations by relying on the context of information [7]. This interface conforms both to standard accessibility principles, as well as the extended context concerns.

Technical terms were removed from WISE, favoring instead a goal-oriented vernacular. Instead of media player, the term "Compact Disc Player" was used. Similarly, focusing on task, instead of "use a text editor", the user was prompted with "Type a Document". Further, menus and button titles were phrased as questions or tasks to perform rather than as one-word phrases whenever possible.

Some of the more straightforward aspects of WISE's accessibility design were the coloration and sizing decisions contained with the system. WISE was built on a desaturated set of beige and yellow hues. This was designed to reduce eyestrain and fatigue, as well as to focus on colors that are highly visible even to users with reduced visual acuity. Yellow and green are the two colors that human retinal cones are most able to distinguish [4]. Combining this with the concept of a lens that yellows with age, it was apparent that a yellow coloration would be the best choice for visibility. As a result, there is little red. What red is used, was softened. Overall, the desire was to have WISE have a soft effect on the eye, to help ease any trepidation against using a computer.

Other affordances include our use of large, high-contrast fonts with serifs and the reinforcement of text with icons and vice versa. Further, the use of large buttons not only makes them more visible, but it also reduces Fitt's Law limitations by increasing target size. This will help not only users with reduced vision but also those who may have lost some motor ability.

Past obvious accessibility affordances, context also plays a large part in the usability of the WISE system. The aforementioned history bar constantly reinforces the task metaphor, providing a goal context for every navigation action in the interface. It is this context that will hopefully blur the boundary between accessibility concerns and the actual cognitive strategies that will turn novice users into experienced users.

In context, all four of the above listed design concerns: linear interaction, effective cognitive strategy prompting, uniform and limited scope, and accessibility can interact to form one fully integrated design. It was the Cocoa API that allowed this system to be bult, but it was also Cocoa that forced a true integration of all the desired actions. Thus, implementation issues became an integral part of the design progress, warranting inclusion in any discussion of WISE's motivations.

3. DESIGN AND IMPLEMENTATION DETAILS

3.1 The History Bar

A defining aspect of WISE's design is the history bar. This tool allows the user to visually understand and recognize the choices that the user made (menu/application selections) to get to the current window/task displayed on the screen. The history bar consists of a clearly visible title, an icon and textual list of choices, and a button that executes the 'go back' function. This button is labeled as "Go back to:" followed by the name of the menu/program the user will return to if the button is pressed. In addition, if, for whatever reason, pressing the button would have no effect (i.e., only the main menu item is in the history bar) the button is hidden from the user. This prevents the user from clicking a button and having no effect occur.

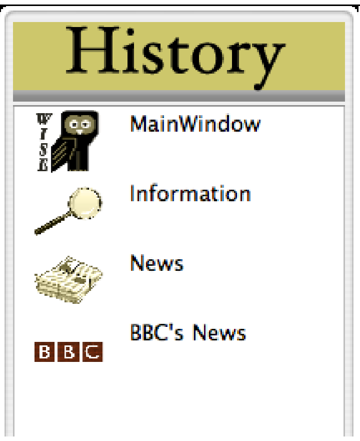

Figure 1: The History bar showing a depth of 4

Every time the user makes a choice, the menu's name and icon are placed on the bottom of the History list. The layout of the list is done vertically to help convey a 'zooming' linear perspective. As more and more options are placed on the History bar, the user will have the feeling of zooming deeper and deeper into the program without ever loosing perspective of their current menu or how to get back up the chain. Due to the simplified design of menus and choices we present the user, the depth of the bar never exceeds five. An alternative design would be to have the history bar run along the bottom of the window to present a linear concept from a horizontal perspective. However, due to limitations with the Cocoa environment, that was not possible in the current iteration of WISE. This decision will be discussed in greater detail in subsequent sections.

To conform to the limited-scope design heuristic, the history bar functions under the single click design. Without selecting a specific item from the history bar, pressing the 'go back' button moves the user back one 'step' in the history. This design choice was made to enable the user to traverse their history, even if they have no concept of selecting items in their history bar. If a menu/function is selected in the history bar, then the "go back" button displays that menu/function's name, and pressing it, will jump the user back to that step, thus allowing faster traversal of their history.

3.2 CD Player Functionality and Features

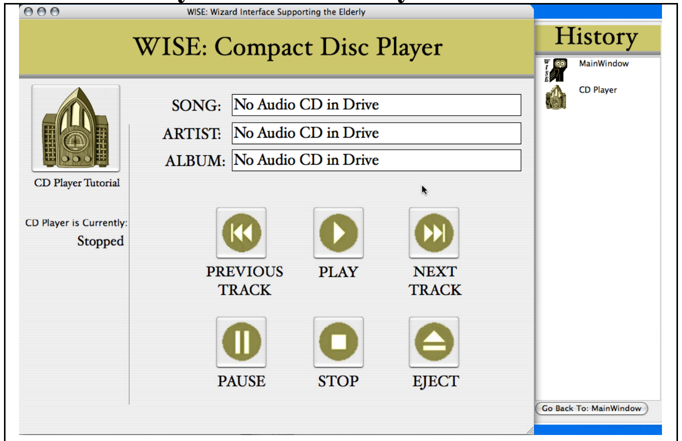

Figure 2: The CD player screen of WISE.

The goal of the Compact Disc Player's design was to allow the user to have a simple, easy-to-read view of their current CD. Modern media players tend to have large play list features, visualization options, and relatively small text and buttons. In addition, they often combine the play and pause buttons, as well as the scan and skip buttons. This button overloading can lead to confusion. As a result, our CD player's functionality is limited to six separate buttons:

- Previous Track
- Play
- Next Track
- Pause
- Stop
- Eject

Noticeably absent are the scan features. These features, though nice, are only useful when the current track's run time, and position could be displayed. Since this state information is not displayed, corresponding functionality was not included. All these functions are achieved by using AppleScript to control iTunes, the default Apple audio media player, and an application that we hide from the user.

In addition to the functionality, it was important that all information displayed to the user be clear (both in size and description). The WISE CD application displays the current song's name, the artist's name, and the album's name. In addition, the current state of the player (playing, paused, stopped) and the button functionality are clearly displayed in plain text, so that the user can easily read the state and effect of a given button, without having to know the common icons. Also adhering to the limited scoping, all buttons are single click based.

3.3 Document Creator: Functionality and Features

The goal of the Document Creator was to provide a simple way for the user to do basic word processing. Many common features associated with word processors are not made available (font, font size, color etc). These features are not present so as to focus on the typing of the text, rather then confusing the user with a multitude of options. One unique notion was to provide a document title text field so as to provide the user with a term or phrase that they can constantly associate with their work. This title is constantly displayed on the left side of the Document Creator, and is used as the file name when the document is saved.

The Document Creator does not use icons to represent the functionality available. All functions available (Bold, Underline, Save, etc) are written out in large plain text, so that any literate user can understand what functions are available and what buttons perform what task.

3.4 Dictionary and Thesaurus

The initial plan was to provide these options to the user by using the built in dictionary and thesaurus provided with OS X 10.4. However, AppleScript currently cannot retrieve the content of the dictionary/thesaurus application. As a result, the decision was made to pursue other options; the two options were a dictionary/thesaurus website or an online protocol called *dict*. Dict is a simple protocol that handles requests for word information from a server. The text received is dependent on the server queried. In this case, a dictionary and thesaurus server. Other servers were available, which might provide enhanced functionality in the future, such as translation.

3.5 Web Based Applications

Following the design rationale of cognitive strategy prompting and limited scope, a WISE user never needs to understand the concept that they are browsing the World Wide Web, though the Web is used as a source of information in many aspects of WISE. By using a goal-based architecture and having 'browsing' be controlled by goal-oriented input rather than by entering URLs or invoking a browser, we achieve this application transparency. Consider an example in which a user wants to find current movie times. The old paradigm is as follows:

1. Realize you need a web browser.
2. Remember that Firefox is a web browser
3. Locate and invoke Firefox
4. Remember that there is a movie site
5. Remember that site's URL
6. Type in that URL correctly
7. Finally, search for your movie

In contrast, WISE presents the user with the choice of looking up information and then looking up movie times. The only input the user must provide once he or she decides to look up movie times is a zip code. Then, with a single button press, the browser in WISE displays the movie times. Another example of goal-directed browsing is reading the news: the user is not even prompted for input in the form of text. Rather through a series of simple menus, the user travels to the webpage of their favorite news source. WISE never prompts the user for a URL or instructs them to 'go online.' It is theorize that this decision will reduce the novice user's computer learning curve and will greatly increase the user's computer usage speed.

In addition to goal-oriented design to hide the concept of Web browsing, a Wikipedia Web page parsing algorithm designed by Sean Timothy Billig for his Wikipedia widget (http://www.whatsinthehouse.com/widgets/) was incorporated. His algorithm eliminated the general layout of the Wikipedia Web page and presented a simplified HTML layout containing text and images only (though sill properly formatted). This allowed modify the text size and color scheme to make Wikipedia articles easier to read. In time, a similar algorithm could be developed to parse other Web-based parts of WISE to make browsing the web even easier and more uniform.

A final issue pertaining to the WISE browsers is the idea of navigation scoping. To prevent confusion with going back in the history of WISE and going back in the history of the browser, a bounding box is provided around the actual Web browsing windows and the back/forward buttons associated with browser. This particular decision after some informal, non-older-adult user testing when it was noticed that users would try going backwards in the browser when they really wanted to go back in the application's history.

3.6 Tutorials

If at any point the user wishes to better understand how to use WISE or any sub feature or menu, a whole tutorial application. Is made available. Unlike most help functions in applications, WISE's tutorials use large text in tandem with images to help the user figure out how to achieve their goals. Goal-based terminology was used as well, so as to parallel the user's WISE experience to date. Tutorials can be accessed from the main page, as well as from each sub menu/application. In addition to each menu and function having a tutorial, there is also a general usage tutorial and history bar tutorial.

3.7 Uniform Window Layout

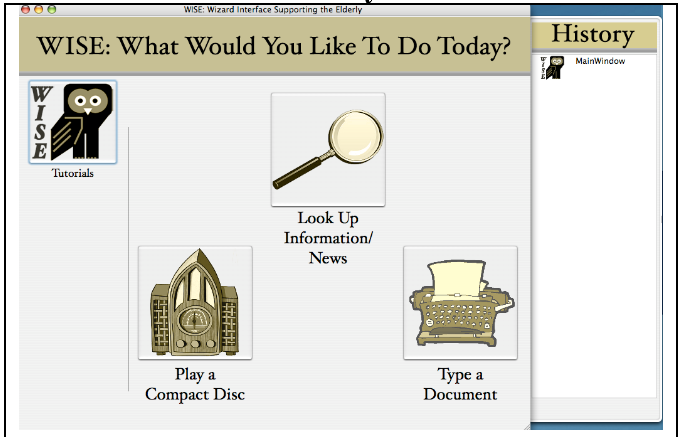
WISE: Wizard Interface Supporting the Elderly
WISE: What Would You Like To Do Today?
History
MainWindow
Tutorials
Look Up Information/ News
Play a Compact Disc
Type a Document

Figure 3: The start screen of WISE, displaying the basic layout of the program.

As shown in research, uniform layout not only looks visually pleasing, but also reduces the cognitive strain on older adults when determining WISE's functionality. The following common visually layout cues exist in almost every aspect of WISE.

- *Title Bar*: The top of every page in WISE has a large yellow bar with the word "WISE" as well as the task that can or should be performed at the user's current view. In case of disorientation, or loss of context, this large bar will allow the user to refocus on where they are and what task they are currently performing
- *History Bar*: The history bar is always present on the right side of the WISE window. Without needing to repetitively use WISE, the novice user can quickly grow accustomed to the position of the history bar and its vital nature for WISE functionality.
- *Icons*: Every menu and function has a unique and distinct icon associated with it. We use this same icon to activate the given feature, to place it in the history bar, and to identify the task in the top left of the given application's screen. This presents another form of repeated context to help keep the user aware of his or her current task and location in WISE.
- *Icon as Tutorial Button*: The icon, located in the top left of every window, can also be used at any point to activate a tutorial for the given task screen. WISE adds this tutorial screen to the end of the history, and the user can return to their menu or function at any point by using the "Go Back To:" button on the history bar. One can select the tutorial by pressing the button on the tutorial page remotely from the current menu or application, then telling the tutorial page to display itself.
- *Left/Right Sides of the Window*: Every menu and application of WISE is visually divided into a left side and right side. The left side of every window, in addition to having the icon, contains any general functionality for the given application or menu. The left side takes up about 1/6th of the window's space on every application or menu. The right side of the window is used for the application or menu itself. When presenting information (movie times, news, a document's text, etc.), the right side of the window is used, while any command functionality (main page, bold, print, etc) appears on the left.

3.8 Accessibility

All text and buttons were enlarged, using a minimum of an 18-point font when at all possible. WISE's default font has support for varying weights and serifs. Serifs make text more readable and varying weights allow for better visual hierarchy. By using larger buttons, we reduce time disadvantages predicted by of Fitt's law as stated above.

3.9 Modularity/WISE Architecture

Not only a good general programming practice, modularity provided architectural support for much of WISE's most important functionality. The following is a brief description of the architecture of WISE, first in text, then graphically.

WISE exists as three main parts:

1. The Main Window
2. History
3. All menus and applications

The main window controls the container that displays WISE. This includes the buttons, text, and images that are displayed, as well as connecting with the history part of the application. The main window has no knowledge of any of the menus and applications that WISE contains with the exception of the main option menu. To change its display, a menu or application contacts the main window controller and provides it with a bundle of information (the content to be laid out, the position of said content, the logo for the content, and a pointer to the menu or application itself). The pointer to the menu or application is only used to inform the destination if it is re-activated due to the user going backward in history.

The history part of WISE maintains what menus or applications the user has visited, the icon and name associated with each menu or application, and a reference to the menu or application. These pieces of information are provided to the history by the main window controller. The history displays the icon and name on the history bar. If the user 'steps back' in the history, the main window retrieves the pointer to the given menu or application from the history, and re-requests the layout and content to be displayed.

All applications and menus exist as separate entities from the history and all other menus or applications. Each menu or application has two parts:

1. The View
2. The Controller

The view contains all layout information; buttons, images, text, text fields, and web browsers. In addition, the views contain references to any function that a button calls. If a given button on a view activates a menu or application (e.g. the tutorial), the view knows what function to call be it in the current menu or application's controller or another menu or application's controller. The receiving function does not need to know the origin of the function call.

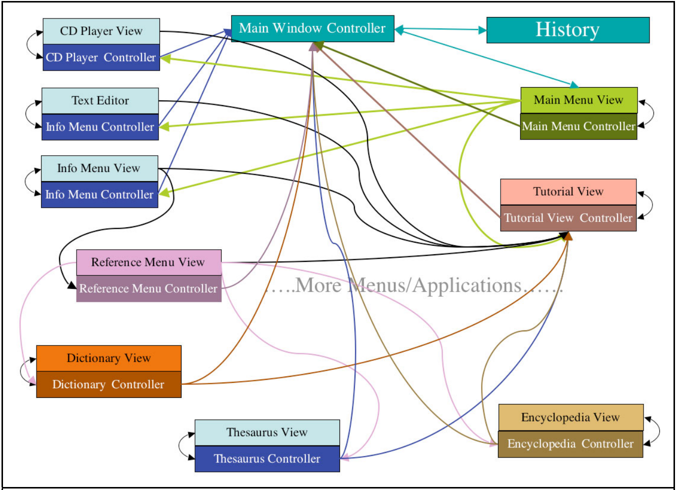

Figure 4: This is a view of the WISE architecture with one path (Reference Books). Of note, is that all of the menus/applications' controllers know about the main window, and the tutorial's controller. Each menu/application knows about the sub menus and its own controller as described by the colored arrows.

The controller contains the functionality of the given menu or application. For menus, this is just the menu's name, reference to the menu's view, the menu's name, a reference to the main window, and a function to tell the main window to display the menu itself. Applications contain everything a menu does, plus any unique functionality associated with that application. This includes any string parsing, input from the user in the form of text, or file I/O.

As a result of the architecture show in Figure 4, the implementation of the history and interconnection of the different components is directly related and made possible by our early design decisions. Without this modularity, it would become exponentially harder to add new features and applications to WISE. The modularity allows the content of the history bar to be dynamically created. A prime example of this is with the tutorial. Every menu and application references the tutorial, so at any point, the one can call the tutorial. It then is dynamically added to the history wherever in the hierarchy it should be. Yet the tutorial does not know about any of the calling pages, only when its functions are called, including its own display.

3.10 The Easy Part of using Cocoa and AppleScript

One aspect of building WISE in Cocoa was the ease of web page retrieval. A wonderful aspect of Cocoa and OS X to a large extent is its built-in support for Web pages and Web browsing. As a result, web browsing is an integral part of Cocoa, and thus doing many aspects of web browsing are extremely simple.

In addition, Xcode (the primary development tool for Cocoa) makes it simple to create the connection between our views and controllers by literally drawing many of the arrows displayed in the figure above.

Lastly, referencing many of the programs and system commands of OS X was made relatively simple though the flexible architecture of the Cocoa development language and the AppleScript scripting language.

3.11 The Hard Part of using Cocoa and AppleScript

Unfortunately, though Cocoa was a great environment to code and design in, it was difficult to make the final product have a look and feel diverging from Aqua. Cocoa and the Xcode design environment limits the placement of many objects such as the History Bar drawer, and it provides no simple support to easily place text on top of images.

Also, though web browsing was made easy by Cocoa, there appears to be a problem using AJAX, or viewing AJAX applications on web pages. As a result, we were not able to use Google's yellow pages search, which uses AJAX to display maps.

A few roadblocks were reached while attempting to retrieving some aspects of state from Cocoa objects. As of the writing of this paper, the current state of fonts surrounding the cursor in the document creator, as well as the length of the history (for backward and forward) in a web browser are unable to be retrieved. Therefore WISE is unable to display bold/italics/underline feedback for the user, and to disable forward and backward buttons for the web browser when no content exists in the given direction.

Unrelated to Cocoa, though relevant to programming, was the use of the Dict protocol. Unfortunately, the protocol does not support concatenated words such as 'picture frame.' As a result, the implementation does not provide support for concatenated definitions.

4. Conclusions

Although it was a 4-week project, WISE was designed as a launching pad for future work in the area of gerontological user interface design. The base program went beyond the original design goals, and due to the commitment to modular design it is easily extensible for future additions. The question of balancing novice performance against full expert functionality is left partially unanswered. In the scope of the stated problem, novice performance was paramount, but this does not have to be so. There are no compelling reasons to believe that some expert short cuts could be added without adversely affecting the novice senior citizen.

WISE provided quite a few unique interaction techniques, which though directly pertinent to older adults, may prove useful in other situations. Of mention is the history bar, and the goal-based architecture. Beyond the scope of WISE, it was concluded that software designers are not limited to the standard drop menu layout schemes. Rather, in WISE a simplified layout and control scheme was developed, which still provides the same functionality as the drop down menu paradigm, but which avoids the complexities inherent in that system. Such simplification may be possible in systems existing in other domains; designers would be well-served to remember this point.

4.1 Future work

As a result of the limited time frame of the WISE project, testing was unable to be conduced. Through the design process, some informal, non-older-adult, user testing was conduced. Though this provided valuable design feedback, the full implications of the design on actual older adult users, the target audience, is still unknown..

In order to test the design, the effectiveness of the task-based architecture, and the usability of the history, a three-phase battery of end-user testing is proposed. . Phase one would consist of a set of focus groups with older adults who have limited or no computer usage. This would provide initial feedback from true novice users. After this testing, we could adjust WISE based upon the results. Phase two, would be a experimental study looking at the ease of use. Again, modification to WISE would be made to reflect the findings. In addition to modifications to the system, further features would be added to the architecture. Phase three would consist of a larger 6 month to 1 year beta-test sampling of older adults having a variety of prior experience with computers. Ideally WISE would be installed in the house or living center of older adults, so as to determine the benefits of long-term use. Phase one is currently (Summer 2006) being conducted at IBM Research, NY.

WISE is currently a fairly robust system. As with most initial system designs, there are a number of points that would be beneficial to pursue given more time. The following a short list of short-term revisions that we would recommend making:

- Dictionary and thesaurus support of concatenated words
- AJAX support for such features as Google maps and Google yellow pages
- Web browser state support, so as to hide forward/back buttons from view when those options are not available
- State support for the Bold/Italics/Underline buttons, and the current state of the cursor
- Bold/Italics/Underline buttons whose text reflects the function they perform.
- Improved CSS support for web browser to blur the line between the visual design of WISE and the web pages.

Beyond these point fixes, there are many longer term that re-designs that would better immerse the user in the WISE experience. These features range from simply improved layout to brand new functionally. The following a list of possible re-design directions for WISE:

- Support for a full screen mode at various screen resolutions
- Utilization of OS X's Voiceover Utility to synthesize text to speech
- Better and more consistent parsing of Web pages and other external sources, possibly based on current work by Billig on parsing Wikipedia
- Reinforcement of linear interaction via screen transitions, possibly in the form of a page turn/flip in the direction of the change in scope
- Support for more applications such as a mailer, health/medicine information/search, chat client , weather reporter, and calendar while still respecting the design decisions already present in WISE
- Better exploitation of Cocoa and OS X's graphical capabilities to make our system both more attractive and more accessible
- True integration of WISE into OS X as a non-application interface layer.

Clearly not all of these goals are feasible in the near future. However, only by setting the bar high now, can our society move closer to a future where a system is available commercially and can bring the older adult community into the 21st century.

4.2 Chat Client

One of the most exciting future extensions to WISE is the Chat Client. Maintaining the "one task at a time" philosophy of WISE, places the notion of online chatting in a new territory. Online chatting is intended to have multiple conversations at a time., as currently phrased However, this design in and of itself flies in the face of accepted gerentilogical usability research.

Having given this problem some limited attention a queue based chat system could conceivably be implemented. Consider the model where a user can see a list of individuals who whish to speak with them, in the order they have requested conversation. At which point, the user can choose whom they wish to speak to, one at a time. When a conversation is concluded, a new one can be started. Thus, chatting becomes more like a phone call, involving only two people, though having the "call waiting" beep occurring, letting one know others would like to speak with you. Other solutions could be conceived with more thought, and extensive user testing. Such a concept is hoped to be explored in further iterations of WISE.

5. ACKNOWLEDGMENTS

I want to thank William Haines, Assistant Professor Jason Hong, Vicki Hanson, Mark Laff, Shari Trewin, John Richards and Chin Chin Lee for their assistance. The author would also like to thank Sean Billig for his open-source Wikipedia Widget, which helped us enormously with parsing the Wikipedia data. It is available online at http://www.whatsinthehouse.com/widgets/.

6. REFERENCES

1. Craik, F. I. M. Age Differences in Memory: The Roles of Attention and Depth of Processing. *New Directions in Memory and Aging*. Ed. L. W. Poon, J. L. Fozard, et al. Lawrence Erlbaum Associates, Inc. (1980): 95-112.
2. Craik, F. I. M. Memory, Aging, and Survey Measurement. *Cognition, Aging, and Self-Report.* Ed. N. Schwartz, D. Park, et al. Edwards Brothers (1998): 95-115.
3. Czaja, Sara J. Computer Technology and the Older Adult. *Handbook of Human-Computer Interaction Second Edition*. M. Helander, T.K. Landauer, P. Prabhu. Science B.V. (1997): 797-812.
4. Hong, Jason. Private Communication. *Software Architecture User Interfaces – 05431*. Carnegie Mellon University (Fall 2005).
5. Hultsch, D. F. Adult Age Differences in the Organization of Free Recall. From *Developmental Psychology*, Vol. 1, No. 6 (1969): 673-678.

6. Hultsch, D. F and C. A. Pentz. Encoding, Storage, and Retrieval in Adult Memory: The Role of Model Assumptions. *New Directions in Memory and Aging*. Ed. L. W. Poon, J. L. Fozard, et al. Lawrence Erlbaum Associates, Inc. (1980): 73-94.

7. Kausler, D. H. *Learning and Memory in Normal Aging*. Academic Press, Inc. (1994).

8. Knäuper, B. Age Differences in Question and Response Order Effects. *Cognition, Aging, and Self-Report*. Ed. N. Schwartz, D. Park, et al. Edwards Brothers (1998): 341-364.

9. Kwong See, S. T. and E. B. Ryan. Intergenerational Communication: The Survey Interview as a Social Exchange. *Cognition, Aging, and Self-Report*. Ed. N. Schwartz, D. Park, et al. Edwards Brothers (1998): 245-263.

10. Newell, A. F. and Gregor, P. Human Computer Interfaces for People with Disabilities. *Handbook of Human-Computer Interaction Second Edition*. M. Helander, T.K. Landauer, P. Prabhu. Science B.V. (1997): 813-824

11. Poon, L. W., L. Walsh-Sweeney, and J. L. Fozard. Memory Skill Training for the Elderly: Salient Issues on the Use of Imagery Mnemonics. *New Directions in Memory and Aging*. Ed. L. W. Poon, J. L. Fozard, et al. Lawrence Erlbaum Associates, Inc. (1980): 461-484.

12. Winograd, E. and E. W. Simon. Visual Memory and Imagery in the Aged. *New Directions in Memory and Aging*. Ed. L. W. Poon, J. L. Fozard, et al. Lawrence Erlbaum Associates, Inc. (1980): 485-506.