ADAPTIVE, MULTIMODAL, APPLICATION INDEPENDENT USER INTERFACES FOR PEOPLE WITH DISABILITES USING COMPUTERS

by

PADMAPRIYA SAMBATH

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTERS OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2005

DEDICATION

New days and new moons,
eyes with dreams, one goal in mind,
their love knows no end.

To my father, Sambath Srinivasan.
To my mother, Prema Sampath.

ACKNOWLEDGEMENTS

This thesis could not have come to fruition without Prof. David Levine's technical and non-technical guidance. I thank Prof. Levine for encouraging me to do this thesis with enjoyment, which in turn enhanced my whole educational experience at the University. I take this opportunity to thank him for guiding and supporting me during several difficult times of my studies and research; he has even told motivational stories to keep my research going.

My sincere gratitude goes to the Connect project's faculty members, Dr. Farhad Kamangar, Dr. Manfred Huber, Prof. David Levine and Dr. Gergely Záruba from whom I have learnt the meaning of the research. Their earnest involvement in utilizing technologies to assist people with disabilities is my inspiration to pursue more research in that field. I thank them for giving me the opportunity to be a part of the Connect project. Their technical insight and brainstorming sessions helped me to gain the required knowledge in this thesis area.

I thank Dr. Kamangar for showing me how to analyze a problem from its big picture. Understanding the big picture not only made me look at this research from an abstract view, but also helped me to acquire better design skills in general. With his infectious knowledge and enthusiasm, he has helped me to grasp several advanced technical design concepts required to implement this work, by relating them to simple everyday examples.

I thank Dr. Huber for reviewing this work completely and giving his opinions. His suggestions made this writing more explanatory and clearly understandable.

I thank Dr. Záruba for his contribution towards the Connect project development, and for his technical assistance, especially for the user profile schema design and the user profile object implementation, which are crucial parts of this thesis.

I acknowledge the support of all my colleagues in the Assist lab, especially to Godfrey Mwangi, Tuli Nivas, Sarah Pinar and Amanda Szucs for their motivation and technical assistance.

A special thanks to Sarah Pinar for helping me to collect results for this thesis. The data collection task became a breeze due to her assistance.

A special thanks to Amanda Szucs for the server side implementation, because of which, the usability analysis became an easy task.

My gratitude goes to the Connect faculty members, the Computer Science department at The University of Texas at Arlington and SensorLogic for providing me financial support at various times of this research.

Thank you to Ravivarman Periasamy for his unconditional love, friendship, support and belief in my efforts. A special thanks for listening to all my thesis adventures on a daily basis, even though I started from the beginning, most of the times. Without his encouragement and understanding, I could have not pursued my higher studies so smoothly.

November 17, 2005

ABSTRACT

ADAPTIVE, MULTIMODAL, APPLICATION INDEPENDENT USER INTERFACES FOR PEOPLE WITH DISABILITES USING COMPUTERS

Publication No. _____

Padmapriya Sambath, M.S.

The University of Texas at Arlington, 2005

Supervising Professor: David Levine

The goal of this work is to create more adaptive and flexible user interfaces for people with disabilities and presenting information to everyone and on every device at any time. Delivering information at any time leads to situations such as a person walking with a mobile device having a small screen wanting to make a flight reservation. The intention of presenting information to everybody immediately raises the question of how much of the available information reach people with disabilities. We have suggested a content based user interface language that may be used by external services to interact with the user. The user interface component delivers information, adapting to the user's preferences, device characteristics, and also by adopting multimodality principles. As a proof of concept, we have implemented a reminder

service to deliver reminder messages to people with different disabilities and we have collected data on users' experiences.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION AND BACKGROUND

The constantly increasing research in a technology and its resultant growth makes it available in large quantities at a reduced cost, which results in the increase in the diversification of end users. The typical examples are automobiles, airlines, telephones, televisions and today's trend is computers, cell phones and Personal Digital Assistants (PDAs). The modern concept of user interfaces in the computer world is that of displaying graphical elements to the user on a big computer screen, interacting visually with the user, and optionally getting keyboard and/or mouse inputs from the user. However, this idea limits the user to handling a computer with a big monitor at home or in the office all the time, which is not always desirable considering the availability of Personal Digital Assistants and cell phones able to handle significant work. The challenge now-a-days is to have technologies accessible from everywhere and to everyone; here everyone means a larger variety of people in a group. The diversification of end users may be that some are students, teen-agers, elderly people, people with different impairments, people who are mobile, users in noisy environment, users in a sunny outdoor environment and many others.

In general, the popularity of a software depends mainly on having easy-to-use and easy-to-learn consistent user interfaces. Even those user friendly interfaces are not meeting the demands of "everyone's interfaces". "*Everyone interfaces* allow people with disabilities, those with low or no technological skills or inclinations, and those

with literacy and other language barriers, to effectively access and use these systems" [S01PG7]. The "General ACM [ACM1] code of ethics" states that "In a fair society, all individuals would have equal opportunity to participate in, or benefit from, the use of computer resources regardless of race, sex, religion, age, disability, national origin or other such similar factors" [ACM2]. ACM's software engineering code of ethics emphasizes to "Consider issues of physical disabilities, allocation of resources, economic disadvantage and other factors that can diminish access to the benefits of software" [ACM3]. Efforts have been made to create the universal web [H00] [W3WAI] and develop user interfaces to accommodate people with disabilities [GN01] [NG97].

From the United States census taken in the year 2003, the proliferation of computers and Internet in the United States may be understood from the fact that in the year 1983, only 8.2% of the population of the United States had personal computers at home, whereas in 2003, 61.8% had personal computers and 54.7% had Internet as well [CEN1]. The results [CEN2] also showed that about 80% to 90% of people in the age group below 55 were using computers and Internet and the rate decreases to 77% in the age group 55 to 64 and further decreases to 64% in the age group 65 and above. Regarding the statistics about people with disabilities, according to the year 2000 census results, there were about 20 million families in the United States with at least one family member having a disability which may be sensory, physical and mental disabilities, which may prevent them from independently going outside home [CEN3].

Another survey taken by a different organization in the year 2000 [CEN4] compared the usage of computers and the Internet between people with disabilities and people with no disabilities. In the age group 15 to 64 (non-elderly), 15% of people with disabilities use the Internet, whereas the usage almost tripled to 42% for people with no disabilities. In the age group 65 and above, 2.2% of people with disabilities use the Internet, and the usage quadrupled to 8.9% for people with no disabilities. This survey shows that people with no disabilities either prefer using computers and Internet more than people with disabilities or that people with disabilities could not easily use computers. This raises the question of what can be done to make computers easily accessible for people with disabilities. Surveys performed with people of different age groups [CEN2] [CEN4] statistically shows that elderly people have problems using computers and avoid them. This scenario may be improved by the development of user interfaces for all. For example, large fonts display may help elderly with low vision, and speech input may be preferred by elderly with motor impairments who are trying to use a mouse with a personal computer or a stylus pen with a PDA.

Yet another survey conducted on consumers with disabilities [CEN5] shows that computers are used for various purposes such as conducting research on disability information, finding employment opportunities, researching health care options and seeking assistive technology solutions. The survey reveals that computers are also used by people with disabilities for miscellaneous tasks such as retrieving emails, news, weather, radio, stocks, sports, locating events and disability organizations, shopping, taking online courses for schoolwork, retrieving online magazines and for merely

socializing with others. So, the Internet actually offers unprecedented opportunities to people with disabilities in several ways. A hearing impaired person may read the whole content of a speech online. People with vision impairments may read newspapers by using computers and screen readers (a form of software that converts the text to speech on computers) rather than depending on a friend. However, the information presentation may not be friendly enough for a user with vision impairments, when the user is forced to listen to a news story which might not be of importance to her. User interfaces can be created to be more adaptable and flexible compared to the existing ones. A person with no vision problems can visually scan news headlines and decide to read further or not. A similar experience must be given to a person with vision impairments by flexible and simplified user interfaces. This thesis proposes application and device independent user interfaces for everyone, focusing on people with disabilities and have them get the most out of modern computer technology.

Coupling user interfaces with the application was once a research topic; the advantages were applying software engineering principles to user interface technologies, such as reusability [BFM92]. But the recent trend is to separate user interfaces from the application logic, thus making user interfaces more consistent and easily navigated. Consistency and simplification of user interfaces benefits any end user, especially users who are limited by their abilities, and has become an important factor to enhance the user experience with computer interfaces.

In an email application, an elderly user may want only subjects of her emails to be presented one at a time and letting her decide if she wants to read that email further

or not. The same user may want to see only news headlines first and then dig into news stories if desired. A person with a cognitive disability may want to be presented with one search result at a time. Someone who is used to complex interfaces and having good vision with free movement of his hands may want to scroll through all of his emails on the PDA screen and click on the email with a stylus pen, if he is interested. So, it is the user's ability level and not the nature of the application, which determines if the title needs to be presented one at a time first for screening instead of presenting all titles or presenting titles with details. This discussion raises the need for application independent interfaces that provide a consistent navigation experience to the end user irrespective of the service used. With the increase in the usage of mobile devices, service providers face the challenge of presenting their applications in a variety of devices. The service providers may design separate interfaces for different groups of devices – Personal Computers (PCs), PDAs, phones, tablet PCs; however it is not an efficient solution. A better solution is to separate the application from the user interface, so that service providers may have to worry only about the application logic. The growing trend is to use various handy devices to provide accessible solutions rather than just depending on personal computers alone. This requires user interfaces to be not only service independent, but also device independent.

The categories of users due to their impairment or disability include "mobility impaired, dexterity impaired, visually impaired, hearing impaired, speech impaired, language impaired" [S01]. There are different types of contents such as text, speech and pictures and not all of them are accessible to each kind of user. A simple example is that

the visual representation of a text content cannot reach a person with vision impairments. “Adaptation solutions” [S01] may have to be developed to transform the original content to accessible content. Some examples of adaptation techniques for people with hearing disabilities are that a speech may be converted to text and sounds such as music and environmental sounds may be converted to a textual representation. Still pictures may be converted to vocal description, and text and graphics may be converted to speech for people with vision impairments. “For people with dexterity impairments, reduced function of arms and hands makes activities related to moving, turning or pressing objects difficult or impossible” [S01]. An adaptive interface solution is to provide easily accessible two big buttons, one on the right and the other on the left hand side. A person working in an outdoor environment may want to hear messages when the screen on her PDA is almost invisible due to the sunlight. A user who reads text messages on his device’s screen may prefer to listen to his messages while driving, while on the contrary, a noisy environment may require speech to text adaptation. Privacy reasons may require speech messages to be represented as text on the screen such as a work assignment from a user’s boss.

Different groups such as the multimodal interaction activity group [W3MMI], the device independence working group [W3DI], and the web accessibility initiative group [W3WAI] of the World Web Consortium have been working in parallel to make the information on the web available to everyone and everywhere. Even though interests of these groups might seem to be distinct, the research area overlaps with each other. Finding a solution to multimodal interaction may actually be one of the solutions

to web accessibility; for example, a speech output modality may benefit a user with vision impairments. The modality preferences are service independent and hence modalities need to be considered at the user interface level. While using the news service, a hearing impaired person may want a news interview to be displayed in a textual form and while using the reminder service, she may want the reminder to be displayed on the screen. The application level does not have to worry about user modalities and this is another reason to decouple applications from their user interfaces. These are the research questions that appeared while reviewing the background work and they are addressed individually in the "implementation" section.

There are various applications, such as email readers and news readers for people with disabilities. Do they provide simple and accessible interfaces for people with different disabilities? Why do service providers have to develop a separate user interface for every service and for every user instead of just focusing on service functionality development? Even if a common user interface system is used by service providers, is it usable by people with different disabilities?

There are many learning and adaptive interfaces developed for advanced users. However, is the adaptation simple enough for disabled users without interfering with the user's preference? How can user interfaces adapt to users with different disabilities, as well as users in different environments and situations? How could user interfaces be made adaptable to user preferences, irrespective of the service being used?

CHAPTER 2

RELATED WORK

2.1 Adaptive systems

During the early nineties, adaptive systems were required with the introduction of graphics (menu, icons) interface when some users preferred command interfaces [BSHKM93]. Users with different background and skills were selected for experimenting with a database system, which was developed with command interfaces, menu interfaces and adaptive interfaces. Some differences between users were accommodated simply by training and education and by improving the simplicity of interfaces. The adaptive interfaces were considered when differences between users were beyond the training level. Five factors were considered while determining user groups: “spatial ability, verbal ability, field dependence (the ability to distinguish an object from its environment), short term memory and a thinking/feeling personality test” [BSHKM93]. From the results of the database system usage, the most significant difference observed was that people with high spatial ability performed faster using command line interfaces than people with low spatial ability. Based on the error rate while using the command interface database system, the interface was dynamically changing from command to menu whenever required, thus exhibiting an adaptive interface behavior. The user model was always available to the user for inspection and amendment. So, in spite of any suggestion made by the system, the user could still stick

to the original interface mode if desired, thus the adaptive system was not forcing the user to use a specific interface type against his preference.

The focus of research shifted from adaptation based on the statistical data analysis to a user centered approach by understanding the user's preferences using computer aided analysis. Kühme developed a user centered approach for adaptive interfaces [K93] in which two solutions were discussed; one was to have the user answer a set of questions and present interfaces accordingly, and another approach was to monitor dialogs between the user and the system, deduce information and change interfaces accordingly. Problems were found with both approaches; the problem of the first approach being that the user might change her mind later after answering the set of questions, and the problem with the second approach was that the system might actually confuse the user with wrong deductions instead of satisfying user needs. Then the concept of user modeling was suggested and it was proposed that allowing the user to express his preferences gave more control towards meeting the user needs than making self adaptable interfaces, thus the adaptation process was made transparent. Two levels of adaptation were suggested when the system had to keep track of short-term and long-term changes. The low level adaptation kept track of short term changes in user needs and the high level adaptation adapted according to the low level changes.

Another user centered distributed system called "ATHENA" was developed in the late nineties to research adaptive interfaces for people with "motion ability impairments and conceptual difficulties" [SR99]. Motion ability impairments in this context means the difficulty in clicking an icon or a button with a mouse whereas

conceptual difficulties are problems faced in understanding computer user interfaces such as interacting with menus and buttons. The users were asked to fill in forms that explained their ability level to interface with the computer and then they were classified as beginners, intermediate and expert users. More adaptive techniques were deployed for beginners and less or none for experts, thus preventing the system to get in the way of expert users, this concept was called "Regression Adaptation" [SR99] principle. The execution agent presented information to the user whereas the perception agent processed the inputs from the user and it also interacted with the main system that the interface was attached to. The modeling agent had information about three general user categories (beginner, intermediate and expert) and an individual model for each user. The modeling agent updated the individual model for the user dynamically by applying production rules and thus the user category was determined. The adaptation agent adapted the interface, fixed any abnormal user inputs and provided training sessions if required.

2.2 Modeling user interfaces

The diversity among devices and within users demands user interfaces to be defined at an abstract level and leads to the need for user interface adaptation and modeling. The "map annotation assistant" [EVP01] has been developed as an effort to model user interfaces in a mobile environment. The modeling on this system is done considering various portable devices. However, this system emphasizes only the appearance of the graphics on devices with different screen sizes and does not provide solutions for different types of contents that could be delivered. Defining abstract user

interfaces is being researched with a focus to support different types of devices [PS02]. User interface adaptation is investigated in different contexts like information filtration such as Jester [GDNG99], a joke delivering system that considers the user's humor preferences and delivers only jokes that the user considers funny, and a customized news agent [BPC00] that delivers news on wireless devices based on the user's daily news preferences.

2.3 Inclusive design

Adaptive systems evolved as user interfaces were expected to accommodate the individual differences between users such as physical impairments, cognitive abilities, environment, user situations, culture, background, knowledge level, and age. Inclusive design is a terminology used by Keates et al. [KCHR00] [CC03] [KC03] implying designing a system for a wide range of users with different capabilities which may include people with disabilities and elderly. Considering the individual differences of end users, researchers were interested to develop customized adaptive interfaces [BM93] [BSHKM93]. The awareness and necessity of creating "inclusive designs and designing for all" [KCHR00] has led to the research of developing adaptive user interfaces for people with disabilities. "User Interfaces for All" [S01PG3] discusses the diversified use of computers that raises the need for adaptable user interfaces. "Intelligent Interfaces for All" [S01PG65] suggests multimodal input and output techniques to include a wide range of user groups and model-based decoupled user interfaces. Thinking beyond the desktop systems [S01PG81] considers the use of mobile devices such as phones and PDAs by users in different environmental conditions

and with accessibility levels. The software compatibility, the user interface compatibility and the data compatibility [S01PG91] are important factors to be considered when thinking beyond the desktop environment. The software compatibility is writing the application logic only once in spite of different deploying the application on different devices. The user interface compatibility is to have consistent look and feel of interfaces across the devices. The data compatibility is exchanging data between different devices by adapting to common protocols.

2.4 PDAs in health care

PDAs are extensively used in the medical industry both by service providers and patients. Some of the benefits of handheld computers in the field of medicine are storing educational medicine literature, maintaining drug information [EPOCRATES] [CSC04], patient tracking applications, business management, prescription management, pain management, treatment in the field of cardiology [FSMWL03], diabetes management [WLOKJMJ04] and many more. A study made about doctors' experience in using handhelds in clinical practice showed improved patient care and productivity; however the impression of the difficulty in using devices existed [BMJ]. The advantages of using handhelds by service providers includes reduced time in drug information retrieval, improved decision making, portability and bedside teaching [HOPKINS].

2.5 Applications benefiting people with disabilities

User interfaces coupled with individual applications are developed to assist people with disabilities, but they are targeted to benefit users with a specific type of

disability. Applications such as an alternative auditory hypertext navigation system [PMMOM97] [MPOM98], a customized search engine [LAB04] and non-visual interfaces [SS95] are designed for people with vision impairments and research efforts have been made to make web accessible to vision impaired users [ZP96] [GMM97] [BR01] [KPD96]. A "portable reading device for the blind" [D03] was developed by Dixit that used the technique of image capturing, interpreting text from the image and then converting the text to speech thus making the text understandable to visually impaired people, even when users are mobile.

An automatic highlighting based scanning technique was developed to receive quadriplegic[1] users' inputs [SC03]. Mouse and keyboard actions can be simulated by other means for quadriplegic users. An automatic scanning technique allows the user to view the highlighted area that changes at regular intervals of time and the user may express his selection with a single switch. An adaptive keyboard configuration [TP97] was tested using people with motor impairments. The idea of using handhelds to receive inputs from motor impaired users [MWYYNM02] has led to the following projects: PDAs are used instead of keyboards and cursors to assist people with muscular dystrophy and certain nervous system disorders in the Remote Commander project [REMCOM]. The ShortCutter project [SHOCUT] allows the user to customize input buttons on a PDA and thus improving the user's interaction with a personal computer. The previous projects provide solutions for users with motor impairments and elderly people, thus addressing a specific disability and the user interface is different for

[1] Complete paralysis of the body from the neck down.

different applications. Other examples of assistive technology applications are the visual recipe book [TMFMMP05] that converts the text information to pictures to assist people with aphasia[2] and a spell checker for assisting people with dyslexia[3] [SE97].

2.6 Service and device independent user interfaces

Research has been done as early as the late eighties to make applications independent of user interfaces in the context of distributed workstations environment, thus exploring ways to consider user preferences as well [BT86] [NMK91]. The ubiquitous web and information presentation has demanded adaptive systems not only adapt to user preferences and abilities, but also adapt to multiple devices. Splitting of a window into multiple windows based on presentation capabilities of a device is another device adaptation method [MP02] [YH04]. The demand for making the web available on portable devices has led to device adaptation techniques considering device profiles and content negotiation protocols [SH01] [W3CCPP]. Though there is research done to consider multiple devices and user profiles, they are oriented towards a particular service [M02A] [M02B].

With the proliferation of different varieties of devices such as phones and PDAs, service providers have been developing a huge series of applications for users. Various user interfaces have been designed for different applications and the user interface is usually tied to a specific device. For example, for a calendar application, considering the device capabilities, user interfaces are designed separately for a

[2] Partial or total loss of the ability to articulate ideas or comprehend spoken or written language, resulting from damage to the brain caused by injury or disease.
[3] A learning disorder marked by impairment of the ability to recognize and comprehend written words.

personal computer and a simple phone. The user is exposed to all kinds of interfaces while using different applications as well as different devices. Sometimes, the same service may be developed by different providers for different devices. This approach of redesigning user interfaces for different devices adds burden to application logic designers, which may be avoided. Sometimes, application designers may want to design the user interface as per their preferences and on the other hand, some service providers may not want to worry about user interfaces at all. Nylander [NB02A] discusses XWeb [OJNMF00] as an example for service providers leaving the user interface completely to the device and Hodes et al. [HKSR97] as another approach where a service maintains different user interfaces for different devices and uses the appropriate one according to the device characteristics. XWeb discusses problems of having different interfaces for different services, making the user learn more than what is needed.

For a given application, Nylander [NBW03] discusses two approaches to run a service on different devices and invents a better approach due to disadvantages of the first two approaches. The first approach is to have the same interface for all devices and its main disadvantage is that different device capabilities require different kinds (thin or thick) of user interfaces. For example, designing a user interface for a tiny mobile phone and for consistency, having the same user interface on a personal computer with a big monitor may actually prevent the user from enjoying a full fledged interface on his PC. The second approach is to create a new user interface for every device which overloads service providers by having them redesign interfaces for all kinds of targeted devices and it also raises an unpleasant user experience of interacting differently with different

devices. Device independence has been researched as a solution to ubiquitous and mobile computing as well as universal access. Nylander [NB02A] focuses on the ubiquitous part of the research by developing "The Ubiquitous Interactor". In Nylander's approach to device independence, an Interaction Specification Language is used for the service to specify the interaction acts with the user (such as making a selection from a given set of choices), interaction engines are used to interpret the interaction acts between services and the user, customization forms contain device and service specific details.

2.7 Multimodal applications

Multimodal applications may be defined as systems that can process more than one input mode and one output mode to the user. Providing alternative input and output modes may help users with disabilities as well as users in different situations such as a noisy environment or while driving a vehicle. Especially, the multimodal approach makes more sense with portable devices that may be equipped to provide different input and output modes. Oviatt [O02] describes different input modes which may be speech, pen, touch, manual gestures, gaze, and head and body movements and output modes which may be audio or visual. Oviatt points out that the future research is towards making conversational next generation interfaces. Robust interfaces require "active adaptation to the user, task, ongoing dialogue, and environmental context" [O02]. Oviatt's "QuickSet" [CJM97] application is an agent based multimodal interface that interacts with the user by getting both speech and pen based gesture inputs. "QuickSet" has been used by the "LeatherNet" system, "a distributed interactive training simulator

built for the US Marine Corps" [CJM97]. Using multimodal interfaces on a PDA, the user may hold a pen and say "red platoon" to create a new platoon and draw a line around using the pen and say "barbed wire" to put in a fence on the PDA. Oviatt describes ten common myths involved with multimodal systems [O99]; two of them are of our interest. The first myth describes that it is expected that users will use multimodal interactions in a multimodal system. The empirical results show that it is not always true. This concept applies to multimodal interfaces developed for people with disabilities since users with one type of disability mostly use one type of modality (visually impaired users using speech input instead of touch screen). Another myth states that "enhanced efficiency is the main advantage of multimodal systems" [O99]. This is not always true since a user's environment controls the modality preferences (using text output in a noisy environment or when security issues are involved) and also, a user with a particular disability may be able to interact only with a certain modality.

The multimodal approach was tried with an email system on mobile devices [L04] with users at random (not grouped according to their abilities). Users preferred multimodal over unimodal and the study showed that even though text and speech options were provided to the users, speech was preferred by most users and they wanted to use the graphical interface upon experiencing consecutive failures in speech recognition. Another multimodal email system [FH04] [F05] involved simulating voice expressions (human prosodies) for non linguistic elements in an email such as highlighted text and emoticons.

2.8 Background work by WWW on multimodal framework

The World Wide Web Consortium's Multimodal Interaction Activity group [W3MMI] is considering multimodal interfaces to be of interest to different industry sectors such as mobile phones, interfaces at work such as printers and fax machines, entertainment systems at home and automotive telematics. The framework to multimodal architecture explains two types of components; there are service components of the multimodal interface architecture such as the interaction manager, the data component, the runtime framework, the system and environment components; there are also modality components such as a text to speech component, and hand gesture recognizing component to interact with the user. The main goals of this framework are encapsulation, distribution, extensibility, recursiveness and modularity [W3MMIARCH]. Keeping the internal details of a component within that component itself and thus treating the component as a black box, is called encapsulation. Distributed implementation and integration of new components (extensibility) are required by the architecture. It is required to add new components dynamically without affecting existing components and the same concept goes with the deletion of existing components as well. Recursiveness is packaging several components and make it look like a single component from an external module's point of view. Modularity behavior requires the data, control and presentation to be well separated. The framework is a loosely coupled architecture suggesting the use of markup languages for interfaces between framework components. The concepts of distributed message based systems and model-view-controller (MVC) design are suggested by the framework. The data

component may be represented as the model, modality components may constitute the view part and a program called the interaction manager as the controller to interface with modality components. Each modality may be represented by an independently functioning modality component and an optional interaction manager to interface with these components. In the absence of the interaction manager, the runtime framework itself interfaces with modality components to interact with the user.

Multimodal systems are defined as "systems that support a user communicating with an application using different modalities such as voice (in human language), gesture, handwriting, typing, audio-visual speech, etc" [W3MMIREQ]. The multimodal interaction group defines the following terminologies related to the research. Supplementary modality is using the same modality (for example, text output in case of output modality) throughout a given user interaction. The supplementary use of multimodality is useful for a person wishing to use a speech output modality from 8 am to 9 am while driving to work and the text output modality for the rest of the day, thus allowing the user to set modality preferences based on the user's situation and individual needs. The delivery context may determine the modality. The delivery context is a term used to specify "a set of attributes that characterizes the capabilities of the access mechanism in terms of device profile, user profile (e.g. identify, preferences and usage patterns) and situation" [W3MMIREQ]. On the other hand, the complementary modality is using more than one type of modality to deliver a single piece of information. For example, a reminder may be delivered by a text modality on a mobile phone screen, prompting the user with a speech modality to push a response

choice for that reminder (for example, the system saying “Push Accept or Reject for your message”). The text and speech modality together delivers the intended message in this example. However, in case of complementary modalities, it may not be possible for the user to determine which modality to be used at a given instance; the decision may rather be made early during the content origination.

The multimodal interaction requirements [W3MMIREQ] explain a high level architecture of components involved in a multimodal architecture including consideration of users’ preferences. A dialog may be viewed as an interaction between the user and the application. The interaction manager that stands between the user and the application may be imagined as a series of dialogs. The interaction manager is the component delivering the content to the user and collecting user inputs. While delivering the content, the interaction manager may determine the modality by considering users’ preferences and the device information. The architecture describes the user’s focus and intent, external knowledge sources and the session context as factors determining the delivery of information. Interfacing with the application business logic may also be done by the interaction manager through sophisticated interfaces and the business logic may also be presented as distributed components of the system.

CHAPTER 3

IMPLEMENTATION

3.1 Motivation behind the thesis

People with disabilities are seeking assistance from computers and portable devices to enhance their life and there are many computer applications trying to fulfill their needs. However, most of those applications are difficult or even impossible to use by people with disabilities due to their physical limitations. Mastering many application interfaces requires sophisticated technical skills. The users find interfacing the most challenging task due to the lack of universal interface. Even if interfaces are tailored separately for different types of users (it is in any case very expensive to individually alter the application according to user needs), interfaces are not adapting to the user's preferences and priorities. This section presents various questions raised regarding the usability of interfaces designed for people with disabilities and explains how this work aims towards eliminating those drawbacks to improve the user experience.

3.1.1 Need for interfaces for people with different disabilities

There are various applications, such as email readers and news readers for people with disabilities. Do they provide simple and accessible interfaces for people with different disabilities?

As we have seen in the "Related work" section, there are various applications for users of portable computers and for people with disabilities. But in most of them, user interfaces are coupled together with the application, thus unable to provide a

general user interface solution. The applications are oriented towards helping users with a particular disability, for example, applications such as the auditory hypertext navigation [PMMOM97], a search engine for persons with visual impairments [LAB04], non-visual interfaces [SS95], a web accessibility application for visually impaired users [ZP96] [GMM97] [BR01] [KPD96], a portable reading device [D03] serve only people with vision impairments. Applications such as the highlighting based scanning [SC03] or the adaptive keyboard configuration [TP97] benefit only people with motor impairments. Applications using handhelds to get user inputs instead of using keyboards [MWYYNM02] [SHOCUT] [REMCOM] benefit only people with a certain nervous system disorder. A visual recipe book [TMFMMP05] helps people with language impairments and a spell checker [SE97] benefits people with a learning disorder. The drawbacks of most of the existing research towards developing applications for people with disabilities are that they provide customized solutions for people with a particular type of disability only (vision or motor or language impairments) and these systems also are focused on providing only one specific service to the user. Since previous applications are tied together with their user interfaces and do not provide a general solution for people with different disabilities, those applications need to be redesigned from scratch in order to accommodate altogether a different type of disability. This thesis proposes a better single multimodal user interface solution with which people with different disabilities may access different services.

3.1.2 Need for service independent user interfaces

Why do service providers have to develop a separate user interface for every service and for every user instead of just focusing on service functionality development?

Even if a common user interface system is used by service providers, is it usable by people with different types of disabilities?

Nylander's [NBW03] work is very close to what we are researching except that her work focuses on application and device independence in ubiquitous environment, and our work emphasizes service and device independent interfaces for universal access. With the rapid growth of portable devices, user interfaces are not only expected to be functional on mobile devices, but also to support people with disabilities. Nylander's approach in "device independence to mobile services" [NB02B] mainly focuses on presenting different services to a user by adapting to suitable graphical interfaces based on the device characteristics such as the screen size and the screen resolution. Though Nylander's research focuses on decoupling services from user interfaces, the user interface is not adapted to the user's ability. The user interface presented may look alike irrespective of the user's preferences. For example, the interface presented on a personal computer is aUI (Graphical User Interface) for general applications, and HTML [W3HTML] for web pages for all users without considering their abilities (some of them may have motor or vision impairments that may make the GUI navigation difficult or impossible).

This thesis proposes an implementation of a service independent user interface component that adapts to people with different disabilities by interpreting their preferences. For example, preferences for the output modality may be a GUI interface and the input modality may be a touch screen for people with hearing impairments, whereas preferences for the output modality may be a speech interface and the input modality may be two big buttons on a wheel chair for people with limited head and hand movements.

In Nylander's work [NB02B], the service independence is achieved by designing an exclusive interaction engine for every device, for example, a GUI interaction engine for GUI rendering on a personal computer, and a cell phone interaction engine for displaying user interfaces on a phone. The services use interaction acts such as input, output, selection, start and stop to communicate with the interaction engine. However, it is not clear whether it is possible for multiple services to request the service of an interaction engine on the same device.

This thesis proposes and implements a system in which service programs use a well defined interface published by the user interface component to exchange the content between the service program and the user interface. By using the interface to the user component, requests to the user interface component may be made by several service programs simultaneously. The interface to the user component allows a service program to specify its priority. Based on the priority, the user interface component presents contents to the user and returns the user action to the service program, if applicable.

3.1.3 Need for user interfaces adapting to the individual user's preferences

There are many learning and adaptive interfaces developed for advanced users. However, is the adaptation simple enough for disabled users without interfering with the user's preference?

How can user interfaces adapt to users with different disabilities, as well as users in different environments and situations?

Some adaptive systems require the user to input data [SR99], some learn on the fly based on the user's interaction [BSHKM93], and some do both [K93]. Systems that collect user preferences may not always be usable by people with disabilities. Learning based on the user's interaction behavior may actually misinterpret the user's intention, when the user is making unexpected mistakes. This fact needs to be considered especially for people with disabilities, who are prone to enter wrong inputs due to their physical limitations. Sometimes, the interface may be used by a friend of a user with a disability, in which case the interaction behavior may be completely different from that of the user. Due to these factors, we have decided not to adapt our interfaces dynamically according to the user's interaction. We have developed a user profile generation system to collect users' preferences directly from users, irrespective of their disability type. The system interacts with the user based on default preferences that may be changed before the user starts using the system. However, service specific adaptation (this concept is not the same as user interfaces adaptation) may be done as future work, for example, a news service agent may be developed that may adapt dynamically according to the user's interest in news story fetching. Developing adaptive user

interfaces targeting a wide range of people based on their abilities and situations requires a good understanding of user parameters. Adaptive user interfaces may be developed by gathering and storing user preferences, which may be later used during the information presentation. The user preferences may consist of parameters such as input and output modality settings, preferences for the look and/or feel of the information being presented, and the measure of complexity of the information.

3.1.4 Need for separating user interfaces from services

How could user interfaces be made adaptable to user preferences, irrespective of the service being used?

Systems such as "the adaptive news access" [BPC00], and "a joke delivering system" [GDNG99] adapt to the user's preferences (such as the preferred news stories, preferred joke categories), but those preferences are service specific. There are service independent user preferences to be considered by an adaptive system such as the output modality, the input modality, how many times to repeat a given content when the user is not responding, delivery preferences for contents with different priorities (such as delivering a high priority message in red color and delivering a low priority news subject in green color) and so on. We call these delivery settings user preferences and they are taken into consideration by the user interface component. The user interface component operates based upon a user profile XML [W3XML] file all by itself, without any involvement from service components. By collecting the user's preferences, we have created a user profile for every user which may be stored in the user's device.

We have adopted the XML metadata format to store the user profile due to the flexibility that comes with using XML, rather than storing the user profile as a text file. XML is a very flexible and extensible language that can handle wide varieties of data. XML is a content oriented language allowing its tags to represent meanings. Adding or removing user parameters can be done without changing the user interface component (that delivers the content to the user) that uses the user profile. The computational overhead of using XML is not an issue in PPDs (Personal Portable Devices) that have 200 to 400 mega hertz processors.

Whenever a service requests the user interface component to present the content, the user interface component uses an XML parser to parse the user profile and then keeps user preferences in the program memory. The user interface component uses those parameters to deliver the content as per the user's preferences irrespective of the service used. An example is, if the user prefers content to be delivered on the screen, and if the desired font size is extra large, the content will be delivered using the extra large font, without differentiating the content type (for example, news or reminder). Another aspect of the user profile portability is that it may also be used on different devices. So the user may see the content in the extra large font size irrespective of if she is using her PPD or her personal computer.

3.2 Problem statement

The goal of this work is to provide a single user interface system that can service various groups of users (users with different disabilities such as vision, motor and hearing impairments, and people in different environments), present information of

different nature (such as news, reminder, and web reader), and support different types of devices (Pocket PC phone, personal computer, and smartphone).

3.3 The Connect project

3.3.1 Who may benefit?

Persons with a disability are those who have a "health problem or disability which prevents them from working or which limits the kind or amount of work they can do" [WCORNELL]. According to this definition, there are about 14 million [WCORNELL] people with disabilities in the United States. Until recently there were few possibilities to assist accommodating many of these disabilities; there were few jobs for a vision impaired person, and an individual with a brain injury was often shunned by society and most jobs were unavailable to her. In the years since personal computers became reasonably priced there have been great strides toward adapting these computers to individuals with disabilities. For example, there are screen readers for vision impaired people, Braille keyboards and displays, software that makes pointing devices such as a mouse possible to be operated by someone with motor/muscular difficulties, and adaptable menu and short cut keys for someone with cognitive issues.

Somewhat more recently, personal communications have become ubiquitous as well. At first pagers, and now cellular telephones have become inexpensive and services are provided at reasonable prices such that more than two thirds of all adults in the US have them. While cell phones may not often require special interfaces for people with disabilities, there have been several cell phone devices designed to make it easier to dial

and answer. There is a niche in the cell phone market to combine the power of fast CPUs, large memories, and multiple communication interfaces (radios) to create what has been dubbed “smart phones”. These devices have computer-like capabilities married to the communications possibilities of WiFi, cell-phones, Bluetooth, and other communications.

We have developed adaptive, multimodal user interfaces for people with disabilities in the Connect [ZKHL05] [ASSIST] project, a distributed messaging system that has been developed to provide an assistive technology solution. The goal of the Connect project is to use newly available smart phones and computer technologies to create synthetic environments to assist communities of people with disabilities. We present several scenarios to describe some of our services and then describe the design rationale and implementation of Connect.

Consider that Juan was in a car accident where he sustained traumatic brain injuries. He is on a regimen of various medicines that must be taken throughout the day. He is currently changing a pain-control medicine that had caused a serious side-effect and his health care provider wishes to remind Juan which medicines should be taken at what times, as well as monitoring Juan’s reactions to the new medicine. Connect will provide some support to help Juan remember his medicines and send feedback to Juan’s health-care provider. First, Juan is given a customized cell-phone; it may contain external buttons for easy input or it may simply be a smart phone with Connect software and a customized “profile” describing Juan’s characteristics (including his daily schedule, his difficulties to read small type on screens, his preferences for screen/type

color and other characteristics that help Juan interact with the Connect software and phone.) Next, Juan's health care provider registers herself with the Connect system through a simple web-page interface. She requests that Juan allow her to send him messages and other communication, and that request is sent to Juan. If he accepts, she may send him reminders and simple questionnaire-like messages that do not alert Juan or are not displayed until the appropriate time. For instance, she may send a reminder to Juan to take the small white pill at 10 AM, and the big pink pill at noon. These reminder messages are "delivered" to Juan at the time requested and if she wishes she may include as part of the reminder a response button, such as "I took the pill", "I am not feeling too good", "I can't find the pill", and other individualized responses. If Juan does not respond within a certain time the message is displayed again, and whether or not Juan responds, a response message is sent back to the web-page that Juan's health provider may look at. There are several additional possibilities. If the medicine is particularly critical, Connect may make several attempts to remind Juan, and then "escalate" to calling or reminding someone else that Juan is not responding. Since Juan is taking a new medicine, we might wish to send him a periodic message-questionnaire that asks him how he is feeling, how effective the medicine is, and whether he has felt any common (listed) side-effects. By employing Juan's profile, Connect will not wake him up during a nap, unless the message is urgent (or he wants to be woken up.) Since cell-phone (and most wireless communication) is not always reliable, typically messages are sent out hours or days before they need to be displayed and, where necessary sending these messages may be done many times, until the phone is reached.

Elvira is severely vision impaired; she can not see text on a screen. She wants to be active, and be part of her community. Particularly she has a group of friends in a social club that meet, go shopping, and join in other group activities. Connect provides "text-to-speech", reading messages to someone wishing that type of interface. The system provides a "news" service, which may provide customized news, sports, community activities, and other broadcast-like messages to Elvira.

Connect Personal Portable Devices (PPDs) are different types of smart phones that are customized to make them into "assistive" technology devices. Connect software has been implemented to operate on different types of phones, some with touch sensitive screens, some without, some with larger displays, and some with keyboards. These phones are assigned by caregivers and social workers to clients by considering potential adaptation to a person's impairments, so that people like Juan and Elvira can carry the PPD similar to carrying a cell phone. PPDs can also be mounted on a wheel chair if needed. A large screen PPD is provided to someone like Juan who might need some extra time to read a message, a PPD with good sound quality is given to someone like Elvira who has low-vision. External buttons are attached to PPDs for a person with muscular and motor impairments. Connect interprets the clients' external button press to be a response to a message or question from a health care provider or family member.

3.3.2 Overview of the Connect messaging system

The Connect messaging system involves Connect servers and mobile devices called Personal Portable Devices (PPDs). The Connect architectural design has a structural functionality that serves as a backbone, which is called the Connect

Infrastructure, and several additional plug-in utility functions called Services. Infrastructure modules handle the communication to the outside world, schedule tasks, and deliver messages (done by the delivery agent infrastructure) according to the user's delivery settings and disabilities. The infrastructure modules are: Call Handler Infrastructure (CHI), Message Gateway Infrastructure (MGI), Process Manager Infrastructure (PMI), Delivery Agent Infrastructure (DAI), Shell Processor Infrastructure (SPI) and Remote Update Infrastructure (RUI). On the other hand, Connect services such as the Reminder Service (RMS), the News Service (NWS), and the User Profile Service (UPS) provide specific functionality to clients such as delivering reminders and selected news, and letting the user edit her preferences. One of the benefits of this software organization is that new services may be written by anyone familiar with the simple interfaces and then "plugged in" to the PPD to provide new functionality without requiring any other changes on the PPD. Future services like providing transportation schedules, detecting and alerting the contact person that a client has fallen and can not get up may be provided using the existing Connect infrastructure.

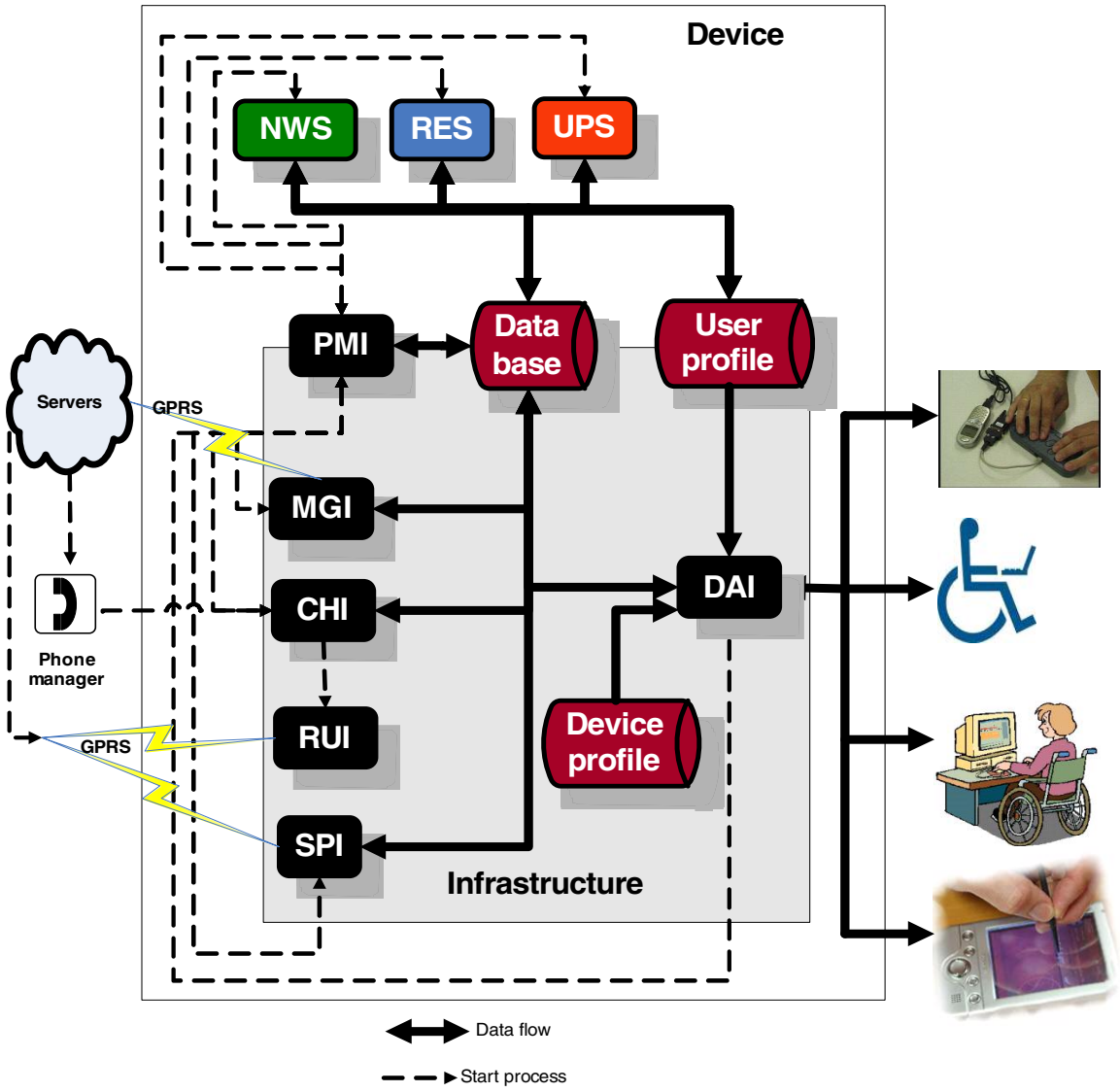


Figure 3.1. Connect messaging system with infrastructure and service modules.
This figure has blocks of Connect infrastructure and service modules residing in the device, abbreviated with three letters. There are MGI, CHI, RUI, SPI and DAI infrastructure modules marked in black color and service modules NWS, RES and UPS marked in green, blue and orange colors respectively. The "Database" and the "User Profile" are shown using data storage symbols. On the right hand side, the DAI block is connected to the outside world to a person using external buttons connected to a phone, a person with a disability using a laptop, a person on a wheel chair using a computer and a person using a PDA with a stylus pen. On the left hand side, outside the device, a cloud symbol represents Connect Servers, a phone is connected to servers, and GPRS connection symbols are shown from servers to MGI, RUI and SPI.

Figure 3.1 shows the Connect messaging infrastructure, different services and the user interface component. The health-care provider logs on to the Connect server and sends a reminder to one of her clients using a simple web page interface. The phone manager module running on the Connect server calls the corresponding client's PPD. The call handler infrastructure module running on the PPD verifies that the call is from one of the Connect servers and then invokes the message gateway agent in the PPD. The message gateway establishes a GPRS [GSM] connection from the PPD to the server and retrieves the message sent by the health-care provider. The status of the message is updated on server logs to update the health-care provider. The message gateway program identifies that the message has to be routed to the reminder service by looking at the header. The message gateway requests the process manager to start the reminder service. The information exchange between any two modules is through the device database. The Connect uses the SQLite [SQLITE] database to store messages in a PPD. When the reminder service is invoked by the process manager, the reminder service identifies the delivery time and other parameters (such as the message priority, what action to take on timeout, checking with user activities to figure out any possible priority clash between the message and the user activity). If the message can be delivered, the reminder service asks the process manager to invoke it at the delivery time. If the message has a past or current delivery time, then the reminder service asks the process manager to start the delivery agent immediately. If the message has a future delivery time, the process manager wakes up the reminder service, and then the reminder service requests the delivery agent's service.

The interaction between the reminder service and the delivery agent is through a well defined, content based XML interface, which is the implementation based on this thesis. The delivery agent understands the characteristics of the content to be delivered from this interface XML. Even though the delivery agent has the knowledge of which service program made the delivery request, it does not matter to the delivery agent if it is a news content or a reminder content. By this knowledge about the service program, the delivery agent returns user inputs to the service program, if there are any. Figure 3.1 also shows that it is the service independent delivery agent that has the knowledge of the user's preferences (through the user profile) to deliver messages accordingly. For example, a user with vision impairments may use an extension device attached to her PPD and use keys in the extension device. Some PPDs have well defined keys that may be usable by people with vision impairments. Another example is a person comfortable with using the stylus rather than keys. Since user interfaces are device independent, a person using a personal computer can access reminders in his computer.

The reminder service and the Connect infrastructure is an example showing the deployment of the adaptive, service independent user interfaces handled by the user interface component (delivery agent). When the delivery agent's service is requested, it checks with the user profile to infer the user's preferences and then delivers the message accordingly. If the user input (may be a selection choice) is required, the delivery agent collects the user input and gives it to the service program that requested the delivery agent's service. If applicable, the reminder service program returns the user selection to the server through the message gateway's mechanism for outgoing messages from the

device. The remote update program in the PPD is used to do a software upgrade over the air and the shell processor program is used to execute remote commands in the PPD. These Connect infrastructure modules and the reminder service is explained in detail in the next section.

3.3.3 The Connect infrastructure

3.3.3.1 Call handler infrastructure

The call handler and the message gateway infrastructure programs in a PPD work together to bring messages from a Connect Server to a client's PPD. The call handler program running in a PPD, which is an event based listener module, screens every incoming call to identify whether the call is from a Connect server and takes pre-defined actions. Based on the caller id, the call handler identifies the intention of the server's call to a PPD, and then registers with the process manager to invoke the message gateway program to retrieve messages or to invoke the remote update program to do an over-the-air software upgrade. If the call is not from a Connect server, the call handler either disconnects the call or notifies the client about the incoming call according to the client's preferences.

3.3.3.2 Message gateway infrastructure

All communications from a PPD to any of the Connect servers are done through the message gateway program running on that PPD using a secure communication protocol. The PPD's time has to be verified by the server periodically for the on time delivery of a message and the message gateway program synchronizes the PPD's time with the server time. The message gateway establishes a data connection with a Connect

server and then requests the server for any undelivered messages. If there are any undelivered messages for that PPD, they are transferred from the server to the PPD irrespective of their delivery time (even if the delivery time is far off in the future). Pushing the messages from the server to the PPD is done immediately as soon as the device can be contacted by the server. This avoids messages getting accumulated in the server for a PPD for the lack of network connection.

3.3.3.3 Process manager infrastructure and the database

There are different modules in a PPD that coexist with each other having well defined functionalities. Every module can be viewed as an agent in a multi agent system. Modules talk to each other through a common medium which is a database in the device and a scheduler program called the process manager infrastructure. For instance, if agent A wants to talk to agent B, agent A writes the data to agent B's table in the common database, then agent A registers with the process manager to initiate agent B so that agent A can request some services from agent B. The process manager initiates agent B as per agent A's request. Agent B comes up, finishes its services, informs agent A about the service completion by writing the completion information to agent A's table in the common database. Then agent B registers with the process manager to invoke agent A to process the information written into agent A's table.

3.3.3.4 Remote update infrastructure

The remote update infrastructure facilitates either updating the whole or a part of the Connect software. The new version is downloaded to a specific upgrade location and the device is soft reset by the remote update program. The installer program which

runs every time the device is soft reset, checks if any new over-the-air upgrade happened since the last time the device has been reset and installs the new software.

3.3.3.5 Shell processor infrastructure

The Connect servers may request any device to execute pre-defined commands by invoking the shell processor infrastructure in the device. The basic file commands such as copying a file from one location to another in the device, deleting a file and sending a file from a particular location in the device to the server are handled by SPI. Any Connect process may be remotely started by the server using the SPI. A device might be restarted using the SPI. The server delivers the message to the message gateway, which opens the envelope and recognizes that the SPI is the intended recipient of the message, thus invoking the SPI by utilizing the Connect infrastructure. The SPI implements all commands in the shell script and returns success if everything goes well; otherwise SPI stops processing commands immediately whenever a failure happens and indicates to the server which command has failed. The whole remote update process may be done as a shell script sent from the server to the device. The remote updating process may be done as separate file transfers from the server to the device using SPI. However, the database and the other Connect infrastructure modules have to be functional in order to remote update the software using SPI.

3.3.3.6 Design principles

All communications to clients' PPDs are initiated from the server. The client's PPD does not wake up at regular intervals to query the server for any new messages, since it leads to unnecessary battery consumption, and there can be some waiting

messages to be delivered on the server for a client when that client's device is in hibernating mode. The server initiates the communication by having a watchdog program that scans through undelivered messages in the server for every client, calls that client's PPD if there are any undelivered messages for that client, and wakes up the client's PPD.

Connect system modules are designed to do their well defined jobs completely and independently. Connect servers let health care providers input messages to their clients and the server sends those messages to clients' PPDs as soon as connections are made with those PPDs. The message gateway infrastructure inside a PPD serves as a postman, directing messages to appropriate services. After receiving a message, the service takes care of storing the message in the PPD, checking for priority clash with user activities, and waking up the PPD during the time of delivery. The service module decides if the message can be delivered at the intended delivery time. If the message can be delivered at the time of delivery, (the message cannot be delivered if the user has a high priority activity going on at the same time) the message is given to the delivery agent infrastructure. The delivery agent checks with the user's delivery settings and delivers messages according to user preferences. For example, a user with low vision might have the large fonts option; another user with vision impairments might have the text-to-speech option turned ON. The message is delivered a number of times with a beep or a vibration or both according to delivery settings. Response or timeout is returned to the service module registered for the response. The service module sends the response back to the message gateway which redirects the response to the server. The

health care provider, who is the originator of messages, views the responses to all her messages on Connect message logs on web pages.

3.3.4 The reminder service

The reminder service delivers reminder messages sent by a health-care provider to her patients and sends patients' responses back to the health-care provider. When being invoked by the process manager, the reminder service checks its own table to see if there are any current reminders to be delivered. For every reminder, the reminder service checks with the user profile to see if the reminder's priority is high enough to be delivered by interrupting the current user activity. If the reminder's priority is higher than the user activity, the reminder service gives the reminder to the delivery agent infrastructure module to deliver it to the user. The reminder service writes the reminder in the delivery agent table, registers with the process manger to invoke the delivery agent and starts the process manager. On the other hand, if the reminder's priority is lower than the current user activity's priority, the reminder service will not send the reminder to the delivery agent. The reminder service calculates the next available matching time by considering all user activities and reschedules the delivery time of the reminder. But if a reminder expires by the next available delivery time, then the reminder service sends the reminder back to the server through the message gateway, informing the health-care provider about the impossibility of delivering that reminder. These system generated information messages are sent as responses to the senders. For example, if a client is sleeping, and a low priority reminder such as "Do you want to go for a movie tonight?" is to be delivered immediately with a one hour expiration, the

reminder service decides that there is no point in delivering the message when the client wakes up in the morning and informs the sender about the non delivery of the message. However, if the sender wants to send an emergency message such as “Take your heart medicine”, the health-care provider will have an option of raising the priority of the reminder to a very high value so that it will be delivered to the client even if the client is doing a high priority activity at that time, such as sleeping.

The reminder service program (in fact any service module) interfaces with the message gateway, the process manager and the delivery agent infrastructure modules through the database. A service specific payload XML that can be interpreted by the reminder service is given by the message gateway to the reminder service. Command and parameters in the payload carries the information of what the reminder service is expected to do with the message. The reminders are archived after they are delivered to the client and responses are sent back to the message gateway (eventually to the server from the message gateway). The cleanup of archived reminders can be done by using a command called “delete archived reminders” to the reminder service. Another simple example for a command is a “new reminder” in which case the reminder service checks to see if the message can be delivered and then gives it to the delivery agent program. The reminders that are transmitted to the device but not yet delivered to the client may be stopped by the health-care provider from the web server interface.

3.3.5 The user interface language published by the user interface component

The user interface component is called the delivery agent in the Connect project. The delivery agent uses the appropriate modality method (such as the text output, the

text-to-speech output, and the wheel chair input interface) according to the user's preferences stored in the user profile. The delivery agent interface language allows a service program to specify the message output, the priority of the message, how many times to repeat the message before timing out due to the client's unavailability, different possible responses (user inputs) for the message if there are any, the action(s) to be taken for every user response, and the requirement of the exclusive delivery of a message in which case a message is delivered to the user without sharing the interface. Many messages may be displayed simultaneously by the delivery agent, if requesting services are not mutually exclusive.

Figure 3.2 shows an example of the user interface XML that is composed by the reminder service and given to the delivery agent, the user interface component. A simple reminder to take his heart medicine is sent by Andrea, the health care provider of Johnny. Andrea does not worry about the disability that Johnny may have. She does not have to construct the same message (for example, take your heart medicine) using different formats (text message, recorded voice and so on) for clients with different types of disabilities. Andrea has to log on to a simple web interface (Figure 3.4) to send a reminder to Johnny. The reminder messaging mechanism is handled by the Connect infrastructure and the reminder service together. The reminder service composes this XML message and keeps it in the reminder table in the device database. When it is the time to deliver the reminder, the reminder service writes this xml message into the delivery agent table in the device database and asks the process manager to service the

delivery agent as explained in the Connect messaging system section. The process manager invokes the delivery agent program.

```
<UI>
  <ID>REM_10001</ID>
  <Priority>100</Priority>
  <Requestor>RMS</Requestor>
  <State>
    <Variable> <Name> response </Name> <Value>""</Value> </Variable>
    <Variable> <Name> insideresponse </Name> <Value>""</Value> </Variable>
  </State>
  <Exclusive>
    <Repeats>
      <Count> 5 </Count>
      <OnDuration> 2 </OnDuration>
      <OffDuration> 3 </OffDuration>
    </Repeats>
    <Output> Reminder from Andrea "Take your heart medicine" </Output>
    <Input>
      <Button>
        <Output> OK Thanks</Output>
        <Press>
          <Variable>
            <Name> response </Name> <Value>"Johnny took his heart medicine"</Value>
          </Variable>
          <Command> returnstate </Command>
          <Command> break </Command>
        </Press>
      </Button>
    </Input>
    <Input>
      <Button>
        <Output> I  lost my pill!</Output>
        <Press>
          <Variable>
            <Name> response </Name>
            <Value>"Johnny lost his pill"</Value>
          </Variable>
          <Command> returnstate </Command>
          <Command> break </Command>
        </Press>
      </Button>
    </Input>
    <Timeout>
      <Expire>
        <Variable>
          <Name> response </Name>
          <Value>"Johnny is not responding"</Value>
        </Variable>
        <Command> returnstate </Command>
        <Command> break </Command>
      </Expire>
    </Timeout>
  </Exclusive>
</UI>
```

Figure 3.2. An example of a reminder message converted to the user interface XML language.

```
<Input>
            <Button>
     <Output> How does the pill look like?</Output>
      <Press>
        <Exclusive>
          <Output> The little, round blue and white pill </Output>
          <Input>
            <Button>
              <Output> OK, Thanks</Output>
              <Press>
                <Variable>
                  <Name> response </Name>
                  <Value>""ok”</Value>
                  <Command> returnstate </Command>
                  <Command> break </Command>
                </Variable>
              </Press>
            </Button>
          </Input>
        </Exclusive>
      </Press>
   </Button>
</Input>
```

Figure 3.3. An example of a nested reminder message converted to the user interface XML language.

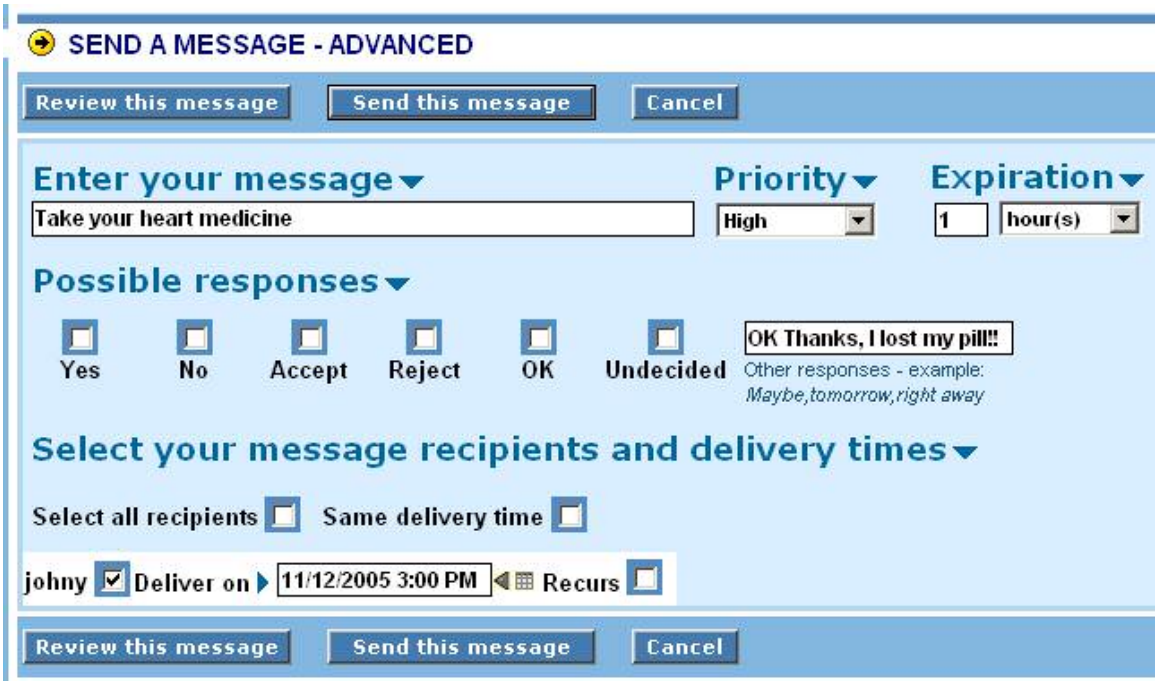


Figure 3.4. Simple "send reminder" interface on the server.

An example of Andrea, a health care provider using a simple web interface of the Connect project to send a reminder to Johnny, reminding him to take his heart medicine. The web interface has a box where Andrea has typed the message "Take your heart medicine". She has also provided custom response choices in a box as "OK Thanks, I lost my pill!!". There are also pre-defined response choices, "Yes", "No", "Accept", "Reject", "OK", and "Undecided". None of them are selected by Andrea. There are three buttons on the interface, a "Review this message" button, a "Send this message" button and a "Cancel" button.

The delivery agent may have several user interface requests by different services. It is explained in detail how the messages are delivered according to their priorities in the section 3.4. XML is used to describe the interface language due to its flexibility and extensibility. Tags can contain meaning assigned to them like in the example. "XML, used to design applications such as XHTML [W3XHTML], SMIL [W3SMIL], and SVG [W3SVG], provides no intrinsic guarantee of the accessibility of those applications" [W3XAG]. The user interface XML is designed in a device independent way, giving importance to the semantic representation. The XML includes

the content (belonging to any service) to be delivered and does not include the details of how the information is presented to the user (for example, font size, bold or italics, text or speech and so on).

The XML message in the example has the message ID, the priority of the message (1 being the highest) and the service that requests the delivery of that message (RMS here stands for the reminder service). The tag "State" encloses the variables whose values are to be returned by the delivery agent to the requesting service, upon "returnstate" command. The delivery agent composes an XML message with variables and values and writes it into the requesting service's table in the device database, and then requests the process manager to start the corresponding service program. The tag "Variable" denotes the variable name to be returned and the tag "Value" assigns an initial value to that variable, which may be changed later based on the user's action. For example, the variable "response" initially is defined as nothing in the state definition, which may change to "John took the heart medicine", if John pushes the "OK Thanks" choice and the value is returned to the reminder service in this example. The tag "Exclusive" tells the delivery component not to share the interface with any other messages (including messages from the same service or any other service). However, an emergency message (say, priority 0) is always displayed regardless of what is on the screen. The tag may also be "Non-exclusive" in which case the interface can be shared with other messages, if the other message is also okay with sharing the interface. The interface language contains information about how many times to repeat the message in the "Repeats" tag, how long (in seconds) to stay on the screen in the "OnDuration" tag,

what the sleep time is between repeats in the “OffDuration” tag. The message to the user is represented by the “Output” tag. The user may perform different actions, pressing a button in case of reminder is a user action and the action to be taken is defined in the “Input” tag. The “Timeout” tag explains the action to be taken when the user is not responding. If the user is not responding, the action to be taken in this example is defined in the “Expire” tag. In this example, the user pushing a button updates the variable “response” with a new value (based on what choice is selected), returns the state to the requestor (the reminder service) and breaks from the interface.

There is a semantic meaning to the list of items in <Press> and <Expire> where they are "executed" sequentially. However, <Exclusive>, <Input>, <Output>, and <Timeout> are "concurrent" which means they are displayed "at once" (they may still need to be divided since many buttons may not be displayed on one screen, and must be separated on several screens based on the device profile).

The interfaces may also be nested as in Figure 3.3. Nested messages develop an approach similar to conversational interfaces [O02] and in addition, these interfaces are service independent and can present content to people with disabilities. In this example, Johnny may want to be reminded of the pills’ appearance and then he acknowledges that he has identified and taken the pill. Note that these nested messages may be exclusive or non-exclusive and are respectively, delivered by sharing or not sharing the interface with the outer scope message and as well as any other separate message.

3.4 User interfaces handling service priorities

The Connect infrastructure prioritizes delivery of messages based on two types of priorities. Every service has a priority associated with it which is called the global priority of a message (for example, reminder may have a high priority whereas news may have a medium priority). The local priority of a message is based on the nature of the delivery of a message (for example, a reminder called "Confirm your appointment with Dr. Carter" may have a higher priority than the reminder "Brush your teeth") which is determined during the message creation by a health-care provider. It is the service program that considers the local priority of a message and decides whether a message can be delivered with the current user activity priority and then requests the delivery agent's service for the delivery. Many services may hand over messages concurrently to the delivery agent. The priority of the requesting service (global) and the priority of the message (local) are forwarded to the delivery agent by the requestor in the delivery agent XML. The service priority followed by the message priority is taken into account by the delivery agent for delivering messages according to their importance. For example, if the reminder service has a higher priority than the news service, all pending reminders are delivered first, followed by all news broadcasts. To deliver reminders, the priority of reminders are considered, higher priority ones being delivered first, followed by medium and then low priority. The low priority message delivered to the client (irrespective of if it is a reminder or news) may be interrupted by the delivery agent if the delivery agent is requested to service a higher priority message by some service module. When an emergency reminder is sent by a health-care

provider, a client may be reading a really long news story or editing her user profile. The client needs to be informed immediately about the emergency reminder. So the delivery agent stops the delivery of the current lower priority message, displays the emergency message, collects the user response and sends it back to the requestor module, and then continues with the delivery of the lower previous priority message.

3.5 Service independent user interfaces

The XML interface language published by the delivery agent may be used by any of the Connect service programs wanting to deliver messages on a client's PPD. The communication between a service module and the delivery agent is achieved using the delivery agent interface. Since the delivery agent takes care of the client's delivery preferences and is the sole component interfacing with the client, service programs do not have to worry about the client's delivery profile and delivering messages to the client. The generic delivery agent interface language allows a service program to specify the content to be delivered rather than determining delivery styles such as the forms, the font size, and the position of the message on the screen. The delivery style settings are derived from the delivery profile by the delivery agent during the delivery of a message. To request the delivery agent service, a service module composes the delivery agent XML and writes the XML into the delivery agent table, and registers with the process manager requesting the service of the delivery agent. Concurrent requests may be made to the delivery agent in which case the delivery agent determines the order of the delivery based on the priority of the service and the priority of a message within a service. The delivery agent publishes a set of classes to define the

interface XML and an XML Composer to construct the interface XML from the set of published objects. A service program may itself compose the delivery agent XML adhering to the language rules. Alternatively, a service program may populate the delivery agent interface object and use the interface composer to create the interface XML. XML, due to its extensibility, facilitates the delivery agent interface to define many levels of messages and as many user inputs as needed by the service.

The delivery agent delivers messages according to the user's preferences as specified in the user profile. The user profile consists of sensitive information such as the user's personal profile (name, contact information), the user's activities throughout the day and so on. Keeping the user profile getting exposed only to the delivery agent ensures that the user's private data is kept within the device. This mechanism makes the user profile available only to the required user interface program and not to any other service specific programs. Hence, service providers do not have access to clients' data that includes details about their disabilities.

The service program also specifies a set of variables to be returned back after the delivery process is over. This set of variables is maintained in the state of the delivery agent language. The state may be modified based on the user response and the state changing behavior is predefined by the service program and given to the delivery agent through the interface. The actions to be taken on timeout is explained in the interface similar to the actions to be taken for a given user response. Some messages may require the client's response(s) and some do not. The interface allows many messages to be delivered to the client one after another based on the client's response.

For example, a message such as “Do you want to go for a movie today?” may have responses such as “Yes, what time?”, “Maybe tomorrow”, and “Not interested at all”. The user responding to “Yes, what time?” may proceed towards displaying another message such as “Select your convenient time from the following” with responses as “6 pm”, “7 pm”, and “10 pm”. The user responses are returned to the service program as requested. In the previous example, the service program may request only the final response or all user responses. The service program may define different variables and values for different responses and appends them to the state. Finally the delivery agent returns the state to the service program and invokes the service program using the Connect messaging mechanism. The service program may send the response back to the server or take an action as required. The timeout action behavior is defined by the service as part of the interface XML.

3.6 User interfaces adaptation

3.6.1 Adaptation to the user

The user interface adapts to the user’s delivery preferences by gathering preferences from the user and storing it in the user profile, which is later parsed during the message delivery. The delivery profile stores the client’s delivery preferences such as how to interface with the client to deliver a message and to receive responses and other user commands from the client. There is a default delivery profile to begin with, which can be changed later at any time by the client. The delivery profile consists of information such as preferred output methods, input methods, whether to allow switching over from one input mode to another during the operation (sequential

modality), font size and type, background and foreground color, number of responses that a client can handle at the same time and whether to turn on the text to speech feature. For example, for a client with vision impairments, the text-to-speech may be turned on in the delivery settings and the input mode for that client may be hard buttons or external buttons instead of touch screen buttons. The color contrast (foreground and background) may be adjusted by a client having a color recognition problem. Differentiating different priority messages may be done by setting a unique delivery style for every priority. The client may understand the importance of the message from the delivery style. For example, a low priority message delivery style may be shown in white background with no beep and vibration, whereas a high priority message may be delivered in red background with beep and vibration to grab the client's immediate attention. The device profile stores the information about a client's PPD such as if the PPD has a screen where the message can be displayed as text, the number of buttons that the PPD can support and some other characteristics of the device. The delivery profile and the device profile have their own unique XML schema. These profiles are stored in the PPD as well as in the server, and they are synchronized automatically, similar to the user profile synchronization.

People with certain disabilities may take a longer time to understand the delivery content and select a response. The user profile allows the user to set the time duration that he might need to understand the message, and to select a choice. The user may also set the message to be repeated a number of times before the delivery agent

decides that the user is not available and sends a timed out message to the requesting service.

The delivery agent might have more than one message to be delivered upon its invocation, in which case messages are delivered based on their priorities. However, the delivery profile and the device profile are parsed and stored in the memory only once during the delivery agent startup. Both profiles are taken into consideration while delivering messages and appropriate delivery decisions are made. For example, the delivery profile set by the user may inform the delivery agent that the user prefers to see a maximum of ten response choices on the screen, whereas the device may support only two response choices (due to the presence of only two physical buttons). In the previous example, the device profile overrides the user preference. Another scenario may be the device being able to support the text to speech feature whereas the user may not want that feature; in that case, the user preference is taken into account.

3.6.2 User profile schema

The user profile schema is given in Figures 3.5, 3.6, 3.7, and 3.8. The parsing and storing of the user profile as an object is done by user profile utilities. The user profile XML may be modified to add more user parameters at any time without changing the user interface component. The “version” tag in the user profile schema is for storing the user profile version information. The user profile may be modified by the user at any time either on the device or on the server using simple web pages. The user profile is synchronized periodically between the server and the device and the version information is required for the synchronization. The user profile consists of the account

profile, the personal profile, the activity profile and the delivery profile. The account profile consists of the user's account information such as the login name, the unique device identification and the account type (whether a client or a care giver). The user's personal information consists of the name and the contact information. The login name is the unique way of identifying a client in a service provider's list. A device is assigned to the user and the device has a unique id associated with it.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="unqualified" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="UserProfile">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Version"/>
        <xs:element name="AccountProfile" type="AccountType"/>
        <xs:element name="PersonalProfile" minOccurs="0">
        <xs:complexType>
        <xs:sequence>
          <xs:element name="Name" type="NameType" minOccurs="0"/>
          <xs:element name="Age" type="xs:anySimpleType" minOccurs="0"/>
          <xs:element name="ContactInfo" type="ContactInfoType" minOccurs="0"/>
        </xs:sequence>
        </xs:complexType>
        </xs:element>

        <xs:element name="ActivityProfile">
        <xs:complexType>
        <xs:sequence>
        <xs:element name="DefaultActivity">
        <xs:complexType>
        <xs:sequence>
          <xs:element name="Priority"/>
          <xs:element name="DeviceID" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        </xs:complexType>
        </xs:element>
        <xs:element name="Activity" type="Epoch" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        </xs:complexType>
        </xs:element>

        <xs:element name="DeliveryProfile">
        <xs:complexType>
        <xs:sequence>
        <xs:element name="DefaultDelivery" type="DeliveryPreferences"/>
        <xs:element name="DeliveryEntry" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Priority" type="xs:integer"/>
            <xs:element name="Style" type="DeliveryPreferences" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
        </xs:element>
        </xs:sequence>
        </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="SensationType">
  <xs:sequence>
    <xs:element name="Repeats" type="Repetition" minOccurs="0"/>
    <xs:sequence minOccurs="0">
      <xs:element name="Vibration" minOccurs="0"/>
      <xs:element name="Shock" minOccurs="0"/>
    </xs:sequence>
  </xs:sequence>
  </xs:complexType>
```

Image credit: http://assist.uta.edu/Connect/Design.html - October 2005

Figure 3.5. User profile XML schema – part 1.

```
<xs:complexType name="VideoType">
<xs:sequence>
<xs:element name="Repeats" type="Repetition" minOccurs="0"/>
<xs:choice minOccurs="0">
<xs:element name="Window" minOccurs="0">
<xs:complexType>
  <xs:sequence>
    <xs:element name="WindowStyle" type="WindowStyle" minOccurs="0"/>
    <xs:element name="Message" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:sequence>
</xs:complexType>

<xs:complexType name="AudioType">
<xs:sequence>
<xs:element name="Repeats" type="Repetition" minOccurs="0"/>
<xs:choice minOccurs="0">
  <xs:element name="AudioFileName" minOccurs="0"/>
  <xs:element name="Beep" minOccurs="0"/>
  <xs:element name="TextToSpeech" minOccurs="0"/>
</xs:choice>
<xs:element name="Volume" minOccurs="0"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="AlarmType">
  <xs:sequence>
    <xs:element name="Visual" type="VideoType" minOccurs="0"/>
    <xs:element name="Audio" type="AudioType" minOccurs="0"/>
    <xs:element name="Sensation" type="SensationType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Epoch">
  <xs:sequence>
    <xs:element name="Priority"/>
    <xs:element name="Time"/>
    <xs:element name="Device" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Location"/>
    <xs:element name="Task"/>
  </xs:sequence>
</xs:complexType>
```

Image credit: http://assist.uta.edu/Connect/Design.html - October 2005

Figure 3.6. User profile XML schema – part 2.

```
<xs:complexType name="TextStyle">
  <xs:sequence>
    <xs:element name="Font" type="xs:string" minOccurs="0"/>
    <xs:element name="Size" type="xs:integer" minOccurs="0"/>
    <xs:element name="ForegroundColor" minOccurs="0">
      <xs:simpleType>
        <xs:restriction base="xs:hexBinary">
          <xs:length value="6"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="BackgroundColor" minOccurs="0">
      <xs:simpleType>
        <xs:restriction base="xs:hexBinary">
          <xs:length value="6"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Style" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="DeliveryEntry">
  <xs:sequence>
    <xs:element name="Priority"/>
    <xs:element name="TextStyle"/>
    <xs:element name="NotificationStyle"/>
    <xs:element name="Overrideable"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Repetition">
  <xs:sequence>
    <xs:element name="Repeat" type="xs:integer"/>
    <xs:element name="OnDuration" type="xs:integer"/>
    <xs:element name="OffDuration" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="AccountType">
  <xs:sequence>
    <xs:element name="LoginName"/>
    <xs:element name="Nickname" minOccurs="0"/>
    <xs:element name="AccountType" minOccurs="0"/>
    <xs:element name="DeviceID" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="NameType">
  <xs:sequence>
    <xs:element name="Prefix" type="xs:string" minOccurs="0"/>
    <xs:element name="FName" type="xs:anySimpleType" minOccurs="0"/>
    <xs:element name="MName" type="xs:anySimpleType" minOccurs="0"/>
    <xs:element name="LName" type="xs:anySimpleType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Image credit: http://assist.uta.edu/Connect/Design.html - October 2005

Figure 3.7. User profile XML schema – part 3.

```
  <xs:complexType name="AddressType">
    <xs:sequence>
      <xs:element name="Number" type="xs:anySimpleType"/>
      <xs:element name="StreetName" type="xs:anySimpleType"/>
      <xs:element name="Apt" type="xs:anySimpleType"/>
      <xs:element name="City" type="xs:anySimpleType"/>
      <xs:element name="State"/>
      <xs:element name="ZipCode" type="xs:anySimpleType"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ContactInfoType">
    <xs:sequence>
      <xs:element name="Email" type="xs:anySimpleType"/>
      <xs:element name="PhoneNumber" type="xs:anySimpleType"/>
      <xs:element name="FaxNumber" type="xs:anySimpleType"/>
      <xs:element name="PreferredMethodofContact" type="xs:anySimpleType"/>
      <xs:element name="Address" type="AddressType"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Point">
    <xs:sequence>
      <xs:element name="xCoordinate"/>
      <xs:element name="yCoordinate"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="WindowStyle">
    <xs:sequence>
      <xs:element name="MaxNumberofButtons" type="xs:integer"/>
      <xs:element name="Frame" type="Rectangle"/>
      <xs:element name="Button" type="Rectangle" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="TextStyle" type="TextStyle"/>
      <xs:element name="Color" type="xs:hexBinary" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Rectangle">
    <xs:sequence>
      <xs:element name="UpperLeft" type="Point"/>
      <xs:element name="LowerRight" type="Point"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="DeliveryPreferences">
    <xs:sequence>
      <xs:element name="AlarmType" type="AlarmType"/>
      <xs:element name="DeliveryStyle">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="VisualDelivery" type="VideoType"/>
            <xs:element name="AudioDelivery" type="AudioType"/>
            <xs:element name="SensationDelivery" type="SensationType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Image credit: http://assist.uta.edu/Connect/Design.html - October 2005

Figure 3.8. User profile XML schema – part 4.

The “ActivityProfile” tag is a complex entity that describes the user’s activities and priorities associated with activities. A message may be delivered only if the priority of the message is higher than that of the user activity during the delivery time. For example, while at work a user may want only high priority messages such as “Take your medicine” to be delivered and may want all other low priority messages to be delivered later. Every user activity is represented as a cron and there can be many user activities spread throughout the day. The cron concept is traditionally used to schedule tasks in the Unix operating system. There is a cron associated with every activity of the client and the cron is composed during the creation or the alteration of an activity. A client may edit her activities either in the server or in the device. User profile utilities collaborate with cron utilities to determine the possible delivery time of a message if a priority clash arises because of a user activity. The cron utilities’ functionality includes the parsing and the composing of a cron structure, determining the previous and the next occurrences of an activity, validating the cron and so on. For example, the user profile editor program takes all required inputs (by requesting the delivery agent to interface with the user) for an activity and uses one of the cron functions to compose a cron structure from the given set of activity parameters. An example is the reminder service finding out that the user is in the middle of a higher priority activity than the priority of the message to be delivered, and determining the right time to deliver the message by using user profile utilities and cron utilities. User profile utilities take into account that the user might finish the current high priority activity and another high

priority activity might start before the current high priority activity ends; in that case the message is to be delivered only after both high priority activities are completed.

Figure 3.9. Editing delivery profile on the server.

The "DeliveryProfile" tag defines the user's delivery preferences which are mainly determined based on their abilities. The screenshot of editing the delivery profile on the server is given in Figure 3.9. There is a default delivery profile defined in the tag "DefaultDelivery", which may be changed by the user on the server. Different messages have different priorities and the user may prefer a unique delivery profile for every priority message. For example, the user may prefer high priority messages to be delivered with large font and red background and may set an alarm with long beep. The user may set small fonts in white background for a message with a lower priority. If the user has not defined a delivery profile for a given priority, then the message will be

delivered according to the default delivery profile settings. The delivery profile for a given priority is defined as a "DeliveryEntry" in the user profile schema with a priority associated in the "Priority" tag and the delivery style in the "Style" tag. The delivery style is composed of the alarm settings and the actual delivery settings of a message. The user may define an optional alarm to accompany a message. The alarm and the delivery style can be aural or visual or sensational (such as a vibration or a shock). The "Audio" tag in the alarm type defines the audio alarm settings that include the number of times to repeat the audio alarm as in the "Repeats" tag, the number of times to beep, the wav file to be played and whether to turn on the text-to-speech option. An aural alarm may be used to play a wav file and to beep a number of times to notify the message arrival. The "Visual" tag in the alarm type defines visual alarm settings where the "Repeats" tag defines the number of times to repeat the visual alarm with a window style and an optional text message. An example of a visual alarm is to flash the screen ten times with red color background. The "Sensational" tag in the alarm type defines vibration and shock settings. An example of sensational alarm is to vibrate the phone; this feature may especially be useful to a person with hearing impairments. The delivery style is also classified as aural, visual and sensational to satisfy the needs of persons with different abilities. Aural delivery style converts the text-to-speech (if the option is turned on in the aural delivery style settings) and delivers the message to the user. An optional audio file may be played along with the message such as "You have a new message". The beep type, the volume level and the number of times to repeat the

message can be set by the user. The visual and sensational delivery style parameters' meanings are similar to that of visual alarm settings.

3.6.3 Examples of delivery profile settings

Figure 3.10 shows the user profile settings of Jane, a person with low vision impairments. Jane has specified the large font (interpreted as font 40 in the metadata) option in her delivery profile. Upon a high priority (priority 1) message arrival, Jane has set an audio alarm to beep for ten seconds and a sensational alarm to vibrate for ten seconds as well. She has set the number of buttons that she can handle at a time as three. She sets the message to be displayed for ten minutes (on-duration is 600 seconds) to give adequate time to read. The interval between message repeats is set as two minutes (off-duration is 120 seconds) and the number of times to repeat a message is set to five times. A message with three response choices is displayed on the screen with all three responses displayed simultaneously and the message is displayed in large font. The background color selected is yellow in this case. The screenshot of the message delivered on the device is given in Figure 3.12.

John with cognitive impairments changes his user profile to view only two buttons at a time and to repeat the message ten times before the system times out. As in John's user profile in Figure 3.11, John is comfortable with small fonts itself (font 10). He has set five minutes on-duration (300 seconds) and two minutes off-duration (120 seconds). He has set a long beep as an audio alarm upon a high priority (priority 1) message arrival. He has selected the small font for message display and the green color to be the background. John has also opted the audio delivery setting preferences in

addition to the visual delivery settings. The message is delivered as a text and also as a speech simultaneously on John's device. Due to the limitation that John can handle only two buttons at a time, the previous message with three responses is displayed in two screens with two responses in each screen as in Figure 3.13.

```
<UserProfile>
  <DeliveryProfile>
    <DeliveryEntry>
      <Priority>1</Priority>
      <Style>
        <AlarmType>
          <Audio>
            <Repeats>
              <Repeat>0</Repeat>
              <OnDuration>10</OnDuration>
              <OffDuration>0</OffDuration>
            </Repeats>
            <Beeptype>Short</Beeptype>
            <Volume>7</Volume>
          </Audio>
          <Sensation>
            <Repeats>
              <Repeat>0</Repeat>
              <OnDuration>10</OnDuration>
              <OffDuration>0</OffDuration>
            </Repeats>
            <Vibration></Vibration>
          </Sensation>
        </AlarmType>
        <DeliveryStyle>
          <Visual>
            <Repeats>
              <Repeat>5</Repeat>
              <OnDuration>600</OnDuration>
              <OffDuration>120</OffDuration>
            </Repeats>
            <Window>
            <WindowStyle>
              <NumberofButtons>3</NumberofButtons>
              <TextStyle>
              <Font></Font>
              <Size>40</Size>
              <ForegroundColor></ForegroundColor>
              <BackgroundColor>Yellow</BackgroundColor>
              <Style></Style>
              </TextStyle>
            </WindowStyle>
            <Message></Message>
            </Window>
          </Visual>
        </DeliveryStyle>
      </Style>
    </DeliveryEntry>
  </DeliveryProfile>
</UserProfile>
```

Figure 3.10. User profile XML of a person with low vision.

```
<UserProfile>
  <DeliveryProfile>
    <DeliveryEntry>
      <Priority>1</Priority>
      <Style>
        <AlarmType>
          <Audio>
            <Repeats>
              <Repeat>0</Repeat>
              <OnDuration>10</OnDuration>
              <OffDuration>0</OffDuration>
            </Repeats>
            <Beeptype>Long</Beeptype>
            <Volume>7</Volume>
          </Audio>
        </AlarmType>
        <DeliveryStyle>
          <Visual>
            <Repeats>
              <Repeat>5</Repeat>
              <OnDuration>300</OnDuration>
              <OffDuration>120</OffDuration>
            </Repeats>
            <Window>
            <WindowStyle>
              <NumberofButtons>2</NumberofButtons>
              <TextStyle>
              <Font></Font>
              <Size>10</Size>
              <ForegroundColor></ForegroundColor>
              <BackgroundColor>Green</BackgroundColor>
              <Style></Style>
              </TextStyle>
            </WindowStyle>
            <Message></Message>
            </Window>
          </Visual>
          <Audio>
            <Repeats>
              <Repeat>10</Repeat>
              <OnDuration>300</OnDuration>
              <OffDuration>120</OffDuration>
            </Repeats>
            <AudioFileName></AudioFileName>
            <Volume>7</Volume>
          </Audio>
        </DeliveryStyle>
      </Style>
    </DeliveryEntry>
  </DeliveryProfile>
</UserProfile>
```

Figure 3.11. User profile XML of a person with cognitive impairments.

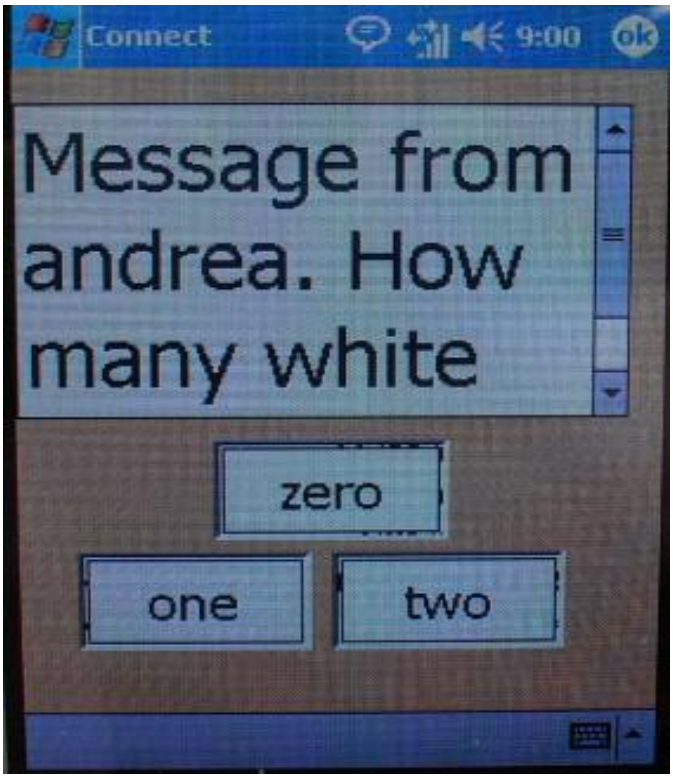
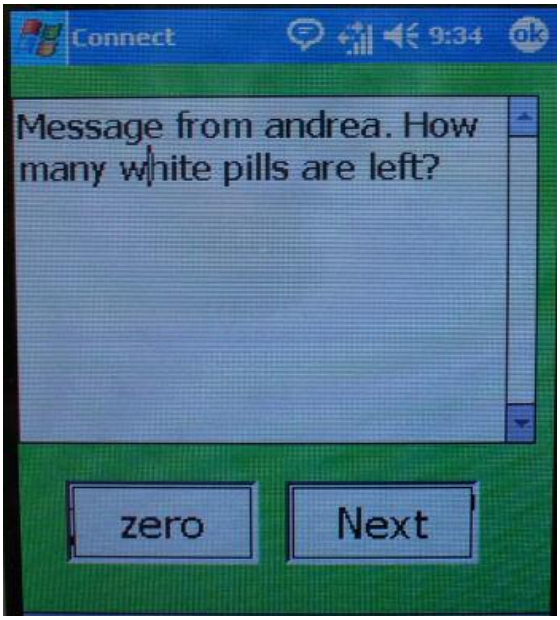


Figure 3.12. Message delivery for a person with low vision.



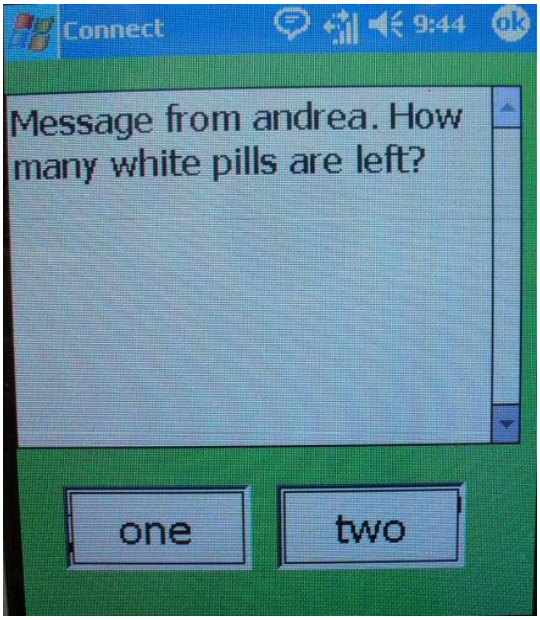


Figure 3.13. Message delivery
for a person with cognitive impairments.

3.6.4 Adaptation to the device

The user interface component uses the device profile to determine device capabilities in terms of the screen size, the resolution, the number of input and output modes and so on. The device profile is stored as an XML file in the device and more device parameters can be easily added in the future due to the flexibility and extensibility that comes with the XML. Both the device profile and the user profile are consulted during the delivery of a message. The user interface component makes decisions after verifying that both the user and the device are capable of handling the interface.

To understand the concept of user interfaces adapting to various devices, consider that Johnny's device is a PDA with a touch screen, his preferred output is the text mode and his preferred input is the touch screen mode. A message with four user choices is sent to Johnny by his health-care provider. For example, according to his user profile, Johnny is capable of understanding and interfacing with all four choices at once and make a selection. But his PPD is able to handle only two buttons at a time due to the device capabilities. After considering both Johnny's user profile and the device profile, the user interface component decides to present the message to Johnny giving him only two choices at a time. The second choice may essentially be a "Next" button that navigates the user to the actual second choice of the message and so on.

Figure 3.14 shows touch screen supported Pocket PC interfaces for a message with four response choices, but the device is capable of displaying only two at a time. This situation may also arise, if the user prefers to handle only two buttons at a time.

For example, as in Figure 3.14, a message such as “When do you need the refill?” with possible user responses as “Today”, “Monday”, “Friday”, and “Sunday”, may be presented with the first choice, “Today”, and a “Next” choice. Selecting the “Next” choice may present the message with the second choice, “Monday” and a “Next” choice and so on. But, if Johnny’s device is a personal computer which is capable of displaying four user choices at once, then Johnny may be presented with the message and all four choices together (since Johnny’s user profile tells that he is capable of handling four choices at the same time).

Figure 3.15 shows another user scenario, where the same message is delivered on a smartphone that has no touch screen display. Smartphone interfaces for a message with four responses are shown, but the device is capable of displaying only two at a time. The hard keys on the phone right below response choices correspond to the response displayed on the top. The user may respond using those hard keys. In the smartphone model, even if the user of that device may handle more than two choices, he can be presented with only two at a time, due to the limited number of buttons on the device. Note that the interface adapts to the device screen size as well.

Figure 3.16 shows an advanced user using her PC to receive reminders, and she can handle all four response choices at once and make a selection. The interface presents all four choices simultaneously, since both the user and the device are capable of handling all four choices at the same time.

Figure 3.14. A four response choices message presented on a Pocket PC.
The figure shows three touch screen supported Pocket PC interfaces for a message with four response choices. The first interface shows a reminder message displayed in a touch screen, which reads “Message from andrea. When do you need the refill?” and has two touch screen response choices, “Today” and “Next”. The second interface shows the same reminder message displayed in a touch screen and has two touch screen response choices, “Monday” and “Next”. The third interface shows the same reminder message displayed in a touch screen and has two touch screen response choices, “Friday” and “Sunday”. The titles of all screens are “Connect”.

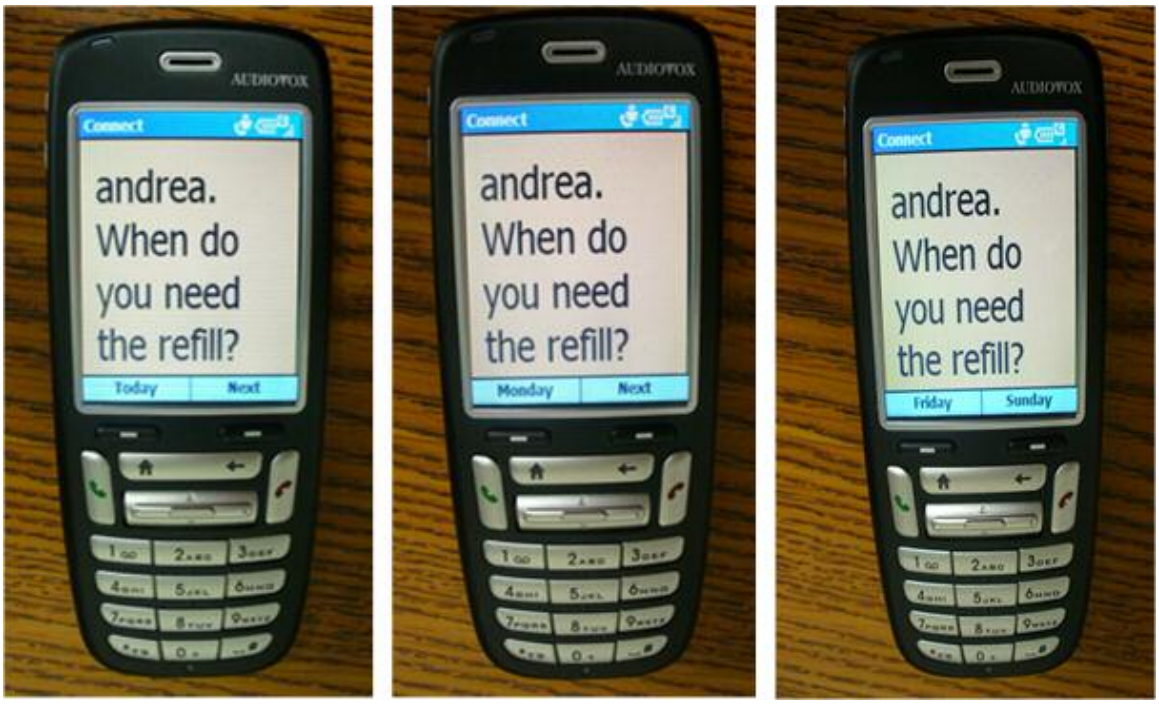


Figure 3.15. A four response choices message presented on a Smartphone.
The figure shows three Smartphone interfaces for a message with four response choices. The device shown here is capable of displaying only two at a time since there are only two hard buttons. The interface is not a touch screen. The user may respond using the hard keys on the phone right below the response choices. The first interface shows a reminder message displayed in a touch screen, which reads “Message from andrea. When do you need the refill?” and has two touch screen response choices, “Today” and “Next”. The second interface shows the same reminder message displayed in a touch screen and has two touch screen response choices, “Monday” and “Next”. The third interface shows the same reminder message displayed in a touch screen and has two touch screen response choices, “Friday” and “Sunday”. The titles of all screens are “Connect”.

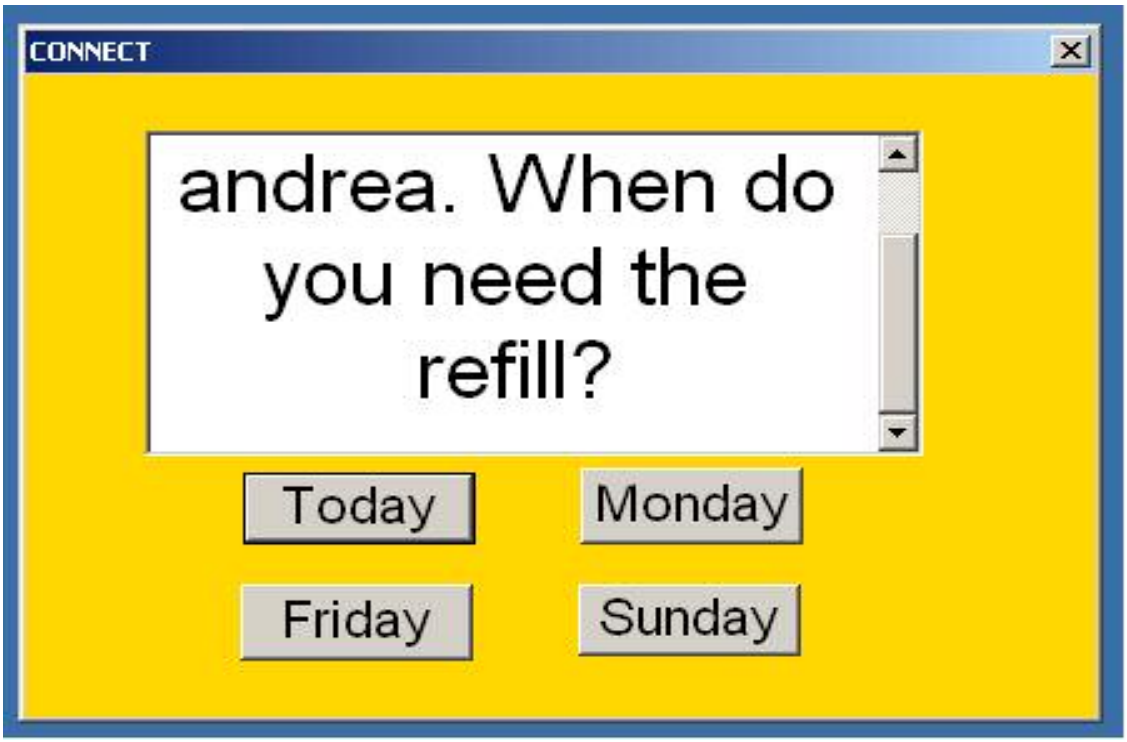


Figure 3.16. A four response choices message presented on a Personal Computer. The figure shows one PC based interface displaying a message with four response choices. The interface shows a reminder message displayed in a relatively larger size when compared to phone interfaces shown previously. The reminder reads “Message from andrea. When do you need the refill?” and has four response choices that are clickable by a mouse or navigated using the tab key of a keyboard. Response choices are “Today”, “Monday”, “Friday” and “Sunday”. The title of the interface displays “CONNECT”.

Software engineering benefits are also motivating factors to develop applications independent of user interfaces. For example, the application logic (in our case, the reminder service) has to be developed only once irrespective of the device that it resides in. We had to write reminder service code only once for the Pocket PC, smartphone, and personal computer, but compile and deploy them separately for different devices. The main advantage of this device independence from the application’s view is easy maintenance. A bug found in the reminder service needs to be

fixed only in one place and it is automatically taken care of in all platforms. Decoupling the service from user interfaces of different devices makes it possible to develop user interfaces completely independent of the service.

3.6.5 Multimodality

Multimodality is adapted by the delivery agent to benefit clients belonging to different disabilities. If a user's delivery profile allows multiple input modes, for a series of messages, the user can respond using touch buttons on the screen for some messages, whereas for some other messages, the user can respond using buttons mounted on her wheel chair. Sometimes, allowing multiple input modes may not be preferred by a client, such as a client who is on a wheel chair might not want any accidental PPD button pushes to be mistaken as her response; instead she might want responses to be interpreted only from her intentional wheel chair button pushes. Currently, the supported output modes are display on the PPD's screen and voice output. The display can be adjusted to different font sizes (for example, large font for people with low vision impairments). The supported input modes are touch buttons on the screen, hard buttons mounted on the PPD, and the big buttons mounted on the wheel chair in convenient positions to the user. Voice input may be supported as the next step using voice recognizing component and gesture recognition may be developed subsequently.

CHAPTER 4

USE CASES AND RESULTS

4.1 Use cases

This section consists of use cases to receive a reminder for people with different disabilities using different systems. Systems taken into consideration are mobile devices such as Pocket PC phones, personal computers, and personal portable devices with our adaptive interfaces software. A personal portable device may be a computer or one of the supported mobile phones.

The aim of use cases in Tables 4.1, 4.2, and 4.3 is to compare the user experience (of a person with low vision) with adaptive interfaces against entirely different systems such as a personal computer without adaptive interfaces. Use cases in Tables 4.1, 4.2, and 4.3 explain different scenarios for a person with low vision receiving a reminder using a personal computer, a Pocket PC, and a PPD without our adaptive interfaces software respectively. The user may use auxiliary software such as a screen reader and a screen magnifier while receiving a reminder on her PC. The user may use a friend's help or a magnifying lens while reading a reminder on her Pocket PC. There is no external help required while using our adaptive interfaces on her PPD. Use cases in Tables 4.4 and 4.5 compare the user experience of a person with hearing impairments receiving a reminder on a Pocket PC that does not have adaptive interfaces with a Pocket PC having adaptive interfaces. The use case in Table 4.6 explains that the user with cognitive impairments is not able to use our adaptive interfaces system with

the default delivery settings. The use case in Table 4.7 explains how customizing the delivery settings may benefit the same user with cognitive disabilities Use cases in Tables 4.8 and 4.9 compare the user experience of a user with motor impairments using our adaptive interfaces without delivery settings customization and with the customization.

Use case title	*A person with low vision receiving a reminder on her computer using a screen reader and a screen magnifier.*
Description	This use case describes a scenario in which a user with low vision uses a screen reader software to receive a reminder about her dentist appointment on her computer and responds to that reminder. The user may use a mouse or a keypad to interact with interfaces.
Actors	User receiving the reminder, PC
Possible devices	PC
Assumptions	• The user is comfortable with using the screen reader and the screen magnifier software. • Either the user is capable of using a mouse to point and click on the screen or the user is capable of using the keyboard to navigate to the desired object and make a selection.
Limitations	• The reminder in this case is similar to an alarm application and not a communication mechanism, in which someone other than the user can create the reminder and receive the user's response. • The scope of this use case is limited to the user interface part of the reminder delivery. • The reminder might be created only locally on the device by the user or a friend by using the calendar feature of an email application installed on the PC.
Events	1. When it is time to remind the user, the PC gives an audio alarm to the user. This audio alarm cannot be customized by the user. 2. The message appears on the PC's screen with the currently selected system font and predefined buttons "Disable" and "Snooze". 3. The user reads the message using the screen reader software. 4. With the help of the screen magnifier software, the user identifies buttons and their captions. 5. The user pushes the "Disable" button. 6. The reminder alarm is deactivated.
Alternate scenario 1	The user pushes the "Snooze" button instead of the "Disable" button. Steps 1, 2, 3, 4 are same as in the "Events" section. 5. The user pushes the "Snooze" button. 6. The reminder appears again on the screen after predefined amount of time. 7. All of the above steps from step 1 in the "Events" section repeat again.
Alternate scenario 2	The user is not present near her PC. Steps 1, 2, are same as in the "Events" section. 3. The message stays on the screen until the user gets to see it.

Table 4.1. Use case for a person with low vision using a PC to receive a reminder.

Use case title	*A person with low vision receiving a reminder on her Pocket PC.*
Description	This use case describes a scenario in which a user with low vision receives a reminder about her dentist appointment on her Pocket PC device and responds to that reminder. The user is provided with only the stylus pen interface modality.
Actors	User receiving the reminder, Pocket PC
Possible devices	Pocket PC
Assumptions	• The user is capable of using a stylus pen to point and click on the Pocket PC screen.
Limitations	• The reminder in this case is similar to an alarm application and not a communication mechanism, in which someone other than the user can create the reminder and receive the user's response. • The scope of this use case is limited to the user interface part of the reminder delivery. • The reminder might be created only locally on the device by the user or a friend by using the "appointments" application in the device.
Events	1. When it is time to remind the user, the Pocket PC gives an audio alarm to the user. This audio alarm cannot be customized by the user. 2. The message appears on the device's screen with a font that cannot be customized by the user and the message has predefined buttons "Dismiss" and "Snooze". 3. The user uses a magnifying lens to read the message and the text on buttons. 4. The user pushes "Dismiss" using a stylus pen. 5. The reminder alarm is deactivated.
Alternate scenario 1	The user does not have a magnifying lens. Steps 1, 2 are same as in the "Events" section. 3. The user gets help from a friend to read the message. Steps 4, 5 are same as in the "Events" section.
Alternate scenario 2	The user pushes the "Snooze" button instead of the "Dismiss" button. Steps 1, 2, 3 are same as in the "Events" section. 4. The user selects the snooze interval to be five minutes. 5. The user pushes the "Snooze" button 6. The reminder appears again on the screen after the snooze interval. 7. All of the above steps from step 1 in the "Events" section repeat again.
Alternate scenario 3	The user does not have the Pocket PC with her. Steps 1, 2, are same as in the "Events" section. 5. The message stays on the Pocket PC's screen until the user gets to see it. Note that this drains the Pocket PC's battery continuously.

Table 4.2. Use case for a person with low vision using a Pocket PC to receive a reminder.

Use case title	*A person with low vision receiving a reminder on her PPD that has adaptive interfaces.*
Description	This use case describes a scenario in which a user with low vision receives a reminder about her dentist appointment on her PPD and responds to that reminder. The user is provided with different mechanisms to interact with the system such as a stylus pen, touch-screen buttons and hard buttons mounted on the device. The reminder might be created locally by the user, or remotely by a service provider or a family member who wants to send reminders to the user. The reminder has different responses that the user may select and the response is later sent back to the requested service.
Actors	User receiving the reminder, PPD, the reminder service program on the PPD
Possible devices	Pocket PC, Smartphone, PC
Assumptions	• The user prefers the large font display and hard keys instead of touch buttons. • The user prefers the message to be timed out after repeating five times. • The user prefers to handle only two choices at the same time. • All these user preferences are set by the user on the device in advance to the reminder delivery. • The device characteristics such as the screen size, the number of supported buttons are available in the device (device profile).
Limitations	• The scope of this use case is limited to the user interface part of the reminder delivery and thus does not cover how the reminder is created by the user or a service provider.
Events	1. The reminder appears on the PPD at the previously set time, with large font and custom responses set by the user. 2. The user pushes one of the choices for the message using hard keys on the PPD, which is her preferred input mode (the user does not need any external assistance since the message is displayed adapting to the user's preferences). 3. The user response is sent back to the reminder service program.
Alternate scenario 1	The user does not have the PPD with her. Step 1 is same as in the "Events" section. 2. The message is repeated five times with the interval as preferred by the user. 3. The interface times out and the "timed out" system response is sent to the reminder service program. 4. The screen turns off and hence the device battery is not drained continuously.

Table 4.3. Use case for a person with low vision using adaptive interfaces to receive a reminder.

Use case title	*A person with hearing impairments receiving a reminder on her Pocket PC.*
Description	This use case describes a scenario in which a user with hearing impairments receives a reminder about her dentist appointment on her Pocket PC device and responds to that reminder. The user is provided with only the stylus pen interface modality.
Actors	User receiving the reminder, Pocket PC
Possible devices	Pocket PC
Assumptions	• The user is having the Pocket PC in her pocket during the message delivery. • The user is capable of using a stylus pen to point and click on the Pocket PC screen.
Limitations	• The reminder in this case is similar to an alarm application and not a communication mechanism, in which someone other than the user can create the reminder and receive the user's response. • The scope of this use case is limited to the user interface part of the reminder delivery. • The reminder might be created only locally on the device by the user or a friend by using the "appointments" application in the device.
Events	1. When it is time to remind the user, the Pocket PC gives an audio alarm to the user. The user is not visually seeing the Pocket PC at this time. The user cannot hear this audio alarm due to her abilities and is not able to customize the alarm to some kind of sensational alarm (vibration or shock alarm). 2. The message stays on the Pocket PC's screen until the user gets to see it. Note that this drains the Pocket PC's battery continuously. This scenario is similar to that of the user not having the Pocket PC with her. The reminder is useless to the user in this case.
Alternate scenario1	The user does not have the Pocket PC with her. Same as the "Events" section.

Table 4.4. Use case for a person with hearing impairments using a Pocket PC to receive a reminder.

Use case title	*A person with hearing impairments receiving a reminder on her PPD that has adaptive interfaces.*
Description	This use case describes a scenario in which a user with hearing impairments receives a reminder about her dentist appointment on her PPD and responds to that reminder. The user is provided with different mechanisms to interact with the system such as stylus pen, touch-screen buttons and hard buttons mounted on the device. The reminder might be created locally by the user, or remotely by a service provider or a family member who wants to send reminders to the user. The reminder has different responses that the user may select and the response is later sent back to the requested service.
Actors	User receiving the reminder, PPD, the reminder service program on the PPD
Possible devices	Pocket PC, Smartphone, PC
Assumptions	• The user is having the PPD in her pocket during the message delivery. • The user prefers to be notified by a long vibration alarm whenever a message is ready for the presentation. • The user prefers the large font display and hard keys instead of touch buttons. • The user prefers the message to be timed out after repeating five times. • The user prefers to handle only two choices at the same time. • All these user preferences are set by the user on the device in advance to the reminder delivery. • The device characteristics such as the screen size, the number of supported buttons are available in the device (device profile).
Limitations	• The scope of this use case is limited to the user interface part of the reminder delivery and thus does not cover how the reminder is created by the user or a service provider.
Events	1. When it is time for the reminder delivery, the PPD vibrates and gets the user attention before presenting the message. 2. The reminder appears on the PPD with large font and custom responses set by the user. 3. The user picks up the PPD from her pocket because of the vibration notification. 4. The user pushes one of the choices for the message using hard keys on the PPD, which is her preferred input mode (the user does not need any external assistance since the message is displayed adapting to the user's preferences). 5. The user response is sent back to the reminder service program.
Alternate scenario1	The user does not have the PPD with her. Steps 1, 2 are same as in the "Events" section. 3. The message is repeated five times with the intervals as preferred by the user. 4. The interface times out and the "timed out" system response is sent to the reminder service program.

Table 4.5. Use case for a person with hearing impairments using adaptive interfaces to receive a reminder.

Use case title	*A person with cognitive impairments receiving a reminder on her PPD without customizing her delivery profile.*
Description	This use case describes a scenario in which a user with cognitive impairments receives a reminder about her dentist appointment on her PPD and responds to that reminder. The user is provided with different mechanisms to interact with the system such as stylus pen, touch-screen buttons and hard buttons mounted on the device. The reminder might be created locally by the user, or remotely by a service provider or a family member who wants to send reminders to the user. The reminder has different responses that the user may select and the response is later sent back to the requested service.
Actors	User receiving the reminder, PPD, the reminder service program on the PPD
Possible devices	Pocket PC, Smartphone, PC
Assumptions	• The user has not customized her delivery profile, so the default delivery profile is present in the device. • The device characteristics such as the screen size, the number of buttons supported by the device are ten (device profile). • The reminder message has ten response choices and the user may select one of them.
Limitations	• The scope of this use case is limited to the user interface part of the reminder delivery and thus does not cover how the reminder is created by the user or a service provider.
Events	1. The reminder appears on the PPD at previously set time with medium font, no text-to-speech and all ten choices appear on the screen at the same time. 2. Due to her cognitive limitations, the user is not able to understand the message and respond to it. 3. The system times out after three minutes (default time settings). 4. The "timed out" system response is sent to the reminder service program.
Alternate scenario 1	The user does not have the PPD with her. Step 1 is same as in the "Events" section. 2. The message is repeated three times (default repeat settings) with the default interval timings. 3. The interface times out and the "timed out" system response is sent to the reminder service program.

Table 4.6. Use case for a person with cognitive impairments using the non-customized default delivery profile to receive a reminder.

Use case title	*A person with cognitive impairments receiving a reminder on her PPD and she has a customized delivery profile in her PPD.*
Description	This use case describes a scenario in which a user with cognitive impairments receives a reminder about her dentist appointment on her PPD and responds to that reminder. The user has different mechanisms to interact with the system such as stylus pen, touch-screen buttons and hard buttons mounted on the device. The reminder might be created locally by the user, or remotely by a service provider or a family member who wants to send reminders to the user. The reminder has different responses that the user may select and the response is later sent back to the requested service.
Actors	User receiving the reminder, PPD, the reminder service program on the PPD
Possible devices	Pocket PC, Smartphone, PC
Assumptions	• The user has customized her delivery profile to display messages in medium font, but to display only two choices at a time. She has also turned ON the text-to-speech option in the delivery profile to understand the message by hearing it along with her reading the message. • The device characteristics such as the screen size, the number of buttons supported by the device are ten (device profile). • The reminder message has ten response choices and the user may select one of them.
Limitations	• The scope of this use case is limited to the user interface part of the reminder delivery and thus does not cover how the reminder is created by the user or a service provider.
Events	1. The reminder appears on the PPD at previously set time with medium font. 2. The message is spoken out by using the text-to-speech component on the device. 3. Only two choices appear on the screen at the same time, one being the actual choice and the other being a "Next" button to traverse through choices. 4. The user either selects a choice by pushing the corresponding choice button on the screen or the user navigates to the next choice by pushing the "Next" button, and then selects the desired response. 5. The user response is sent back to the reminder service program.
Alternate scenario 1	The user does not have the PPD with her. Steps 1, 2, 3 are same as in the "Events" section. 2. The message is repeated as many times as preferred by the user (delivery profile settings). 3. The interface times out and the "timed out" system response is sent to the reminder service program.

Table 4.7. Use case for a person with cognitive impairments using a customized delivery profile to receive a reminder.

Use case title	*A person with motor impairments receiving a reminder on her PPD without customizing her delivery profile.*
Description	This use case describes a scenario in which a user with motor impairments receives a reminder about her dentist appointment on her PPD and responds to that reminder. The user is provided with different mechanisms to interact with the system such as stylus pen, touch-screen buttons and hard buttons mounted on the device. The reminder might be created locally by the user, or remotely by a service provider or a family member who wants to send reminders to the user. The reminder has different responses that the user may select and the response is later sent back to the requested service.
Actors	User receiving the reminder, PPD, the reminder service program on the PPD
Possible devices	Pocket PC, Smartphone, PC
Assumptions	• The user has not customized her delivery profile, so the default delivery profile is present in the device. • The device characteristics such as the screen size, the number of buttons supported by the device are two (device profile). • The reminder message has two response choices and the user may select one of them.
Limitations	• The scope of this use case is limited to the user interface part of the reminder delivery and thus does not cover how the reminder is created by the user or a service provider.
Events	1. The reminder appears on the PPD at previously set time with medium font, no text to-speech and all two choices appear on the screen at the same time. 2. Due to the motor limitations, the user is not able to push the button on the screen or on the device. The user is also not able to hold the stylus pen. 3. The system times out after three minutes (default time settings). 4. The "timed out" system response is sent to the reminder service program.
Alternate scenario 1	The user does not have the PPD with her. Step 1 is same as in the "Events" section. 2. The message is repeated three times (default repeat settings) with the default interval timings. 3. The interface times out and the "timed out" system response is sent to the reminder service program.

Table 4.8. Use case for a person with motor impairments using the non-customized default delivery profile to receive a reminder.

Use case title	*A person with motor impairments receiving a reminder on her PPD and she has a customized delivery profile in her PPD.*
Description	This use case describes a scenario in which a user with motor impairments receives a reminder about her dentist appointment on her PPD and responds to that reminder. The user has different mechanisms to interact with the system such as stylus pen, touch-screen buttons and hard buttons mounted on the device. The reminder might be created locally by the user, or remotely by a service provider or a family member who wants to send reminders to the user. The reminder has different responses that the user may select and the response is later sent back to the requested service.
Actors	User receiving the reminder, PPD, the reminder service program on the PPD
Possible devices	Pocket PC, Smartphone, PC
Assumptions	• The user has customized her delivery profile to deliver messages on the screen with large font and selected the input mechanism to be external buttons positioned to be reachable by the user. • The user has turned ON the text-to-speech in her delivery profile. • The device characteristics such as the screen size, the number of buttons supported by the device are two (device profile). • The reminder message has two response choices and the user may select one of them.
Limitations	• The scope of this use case is limited to the user interface part of the reminder delivery and thus does not cover how the reminder is created by the user or a service provider.
Events	1. The reminder appears on the PPD at previously set time with large font and all two choices appear on the screen at the same time. 2. The message and responses are converted to speech and spoken out on the device. 3. The user hears the message and response choices without changing her position, thus causing no stress to her head and neck. 4. The user pushes the external button corresponding to her selected response. 5. Adaptive interfaces match the external button push event with the actual response. 6. The user response is sent back to the reminder service program.
Alternate scenario 1	The user does not have the PPD with her. Steps 1,2 are same as in the “Events” section. 3. The message is repeated as many times as preferred by the user (delivery profile settings). 4. The interface times out and the “timed out” system response is sent to the reminder service program.

Table 4.9. Use case for a person with motor impairments using a customized delivery profile to receive a reminder.

4.2 Connect observations

As a proof of concept, we have developed the reminder service in the Connect project, which uses the common adaptive user interface component to present messages. The reminder service may be used by people with vision impairments, people with motor impairments, people with hearing impairments and users in wheel chairs. Reminders may be delivered on different devices; our test devices include PPDs such as Pocket PCs, smartphones, and personal computers. The analysis of the reminder service of the Connect reveals that typical users sent about three messages per day whereas expert users sent about 10 to 30 messages per day [ZKHL05].

4.3 The experiment

4.3.1 User groups

The results are based on the observation of a small group of seven users with various disabilities and two users with no disabilities, interacting with their PPDs to receive and respond to reminders having different number of response choices. The user group consisted of one user having hearing impairments (user 1), two users with no disabilities (user 2 and user 3), three users having vision impairments (user 4, user 5 and user 6), and three users having motor impairments (user 7, user 8 and user 9). Each user was sent at least three reminders with two, three and four possible response choices.

4.3.2 User settings

As we have seen previously, the user profile settings allow the user to specify the number of times to repeat a message, the duration in seconds for a message to stay on the PPD screen (on-duration), and the duration in seconds between message repeats (off-duration). The time to respond to a message may differ for users with different abilities. The response time may be very high for a user with difficulty in moving her hands and the same concept is true for a visually impaired user. The other settings such as the font size and color preferences may also be selected by the user, which may benefit people with low vision color blindness impairments. The user may either opt to use touch screen buttons or hard buttons mounted on the PPD. The phone may be set to vibrate or beep or both, upon a message arrival. In this experiment, users had their own settings as per their preferences, which were not recorded.

4.3.3 Device types

Except for people with vision impairments, Pocket PCs were used as test devices for all other users. Smartphones were used for people with vision problems since smartphones do not have a touch screen, thus the risk of user accidentally pushing any keys on the display was avoided. Pocket PCs were able to accommodate two buttons on the touch screen, which was mapped to two external hard buttons mounted on the top of the device. Smartphones were accommodating two hard buttons located at the bottom of the screen. Due to the limitation of two buttons, any reminder with more than two response choices was split and displayed in many interfaces, each interface

having one actual response choice and a “Next” choice for traversing to the next interface. The final interface had two actual response choices and no “Next” choice.

4.3.4 Message types

The reminder may be an announcement type of message that does not require a user response, or it may provide many choices to the user to select a response. For example, a simple message with no response may be “Brush your teeth” and a message with many responses may be “Please confirm your appointment on Wednesday at 8 am” with response choices as “yes” and “no”, in which the user may select either of them. A more complex message may be having more than two responses such as “When do you want to do the refill for your heart medicine?” with possible user responses as “Monday”, “Wednesday”, “Friday” and “Not now, remind me next week”.

For a given message, the message complexity and the number of user selectable responses had an impact on the response time of the users. The reminder with two response choices required the user to select one of them in the first interface itself, since there was only one interface, thus having only one user interaction. The reminder with three response choices gave options to the user to either select a response from the first interface, which required only one user interaction, or to traverse to the next interface by selecting the “Next” choice, and then selecting the desired response, which required two user interactions. Similarly, the reminder with four response choices required a minimum of one and a maximum of three user interactions.

4.4 Observations

It was observed that a person with hearing disabilities preferred to have the vibration settings turned ON so that the reminder arrival can be notified to the user.

User 1 (with hearing impairments) used the PPD's touch screen button interface to select the response using a stylus pen. The response time was between 2 seconds for a reminder with two possible choices, 2 seconds for a reminder with three possible choices and 9 seconds for a reminder with four possible choices. It was noted that the response time was about the same as that of people with no disabilities.

On average, the response time for persons with no disabilities (User 2 and User 3) was 2 seconds for a reminder with two choices, 4 seconds for reminder with three choices and 8 seconds for a reminder with four choices.

For persons with visual impairments (users 4, 5 and 6), the response time was anywhere between 5 seconds to timing out (system timed out in 180 seconds before one of the users could respond) based on the number of responses they had to navigate to arrive at the ultimate response of their choice. User 4 took 60 seconds for the first message with two choices, 5 seconds for the second message with two choices, 30 seconds for the third message with three choices, and 30 seconds for the last message with four choices. User 5 took 60 seconds for the first message with two choices, 15 seconds for the second message with two choices, 25 seconds for the third message with three choices, 68 seconds for the last message with four choices. User 6 took 52 seconds for the first message with two choices, and then the system timed out after 180

seconds for messages with two, three and four choices. User 6 was not able to push any response for the second, third and fourth messages due to the lack of time.

People with motor difficulties (users 7, 8, and 9), especially, all of them having difficulty in moving their hands, used hard buttons on top of the device. None of them preferred to use buttons on the touch screen. For persons with motor impairments, the response time was between 5 seconds to timing out (the system timed out in 300 seconds before the user could respond). User 7 took 10 seconds for the first reminder with two choices, could not respond and the interface timed out in 300 seconds for the second reminder with two choices, 5 seconds for the third message with three responses, and 50 seconds for the fourth message with four responses. User 8 took 5 seconds for a message with two responses and 10 seconds for another message with three responses, and 20 seconds for messages with four responses. User 9 took 5 seconds for the first message with two responses, again 2 seconds for the second message with two responses, 30 seconds for the last message with three and four responses.

The results are represented in graphical format in Figure 4.1 and in tabular format in Table 4.10. Reminder messages with different number of responses were presented to nine users and the time to respond to a reminder was recorded. Some users had more than one message with the same number of response choices. In that case, the longest duration recorded is taken into consideration while drawing the graph. However, the graph does not capture the fact that a user could not push a button for the entire on-duration time and the system time out (refer user 6 and user 7). From the

graph, it is inferred that users with no disabilities (users 2 and 3) and the user with hearing disabilities (user 1) found it easy to use the interface (easiness measured in terms of the user's response time), when compared to users with vision and motor impairments. It may also be observed that people with vision impairments took relatively longer time to respond to their very first message. The response time for people with motor impairments was more or less the same, irrespective of if it was the first, the second or the last message.

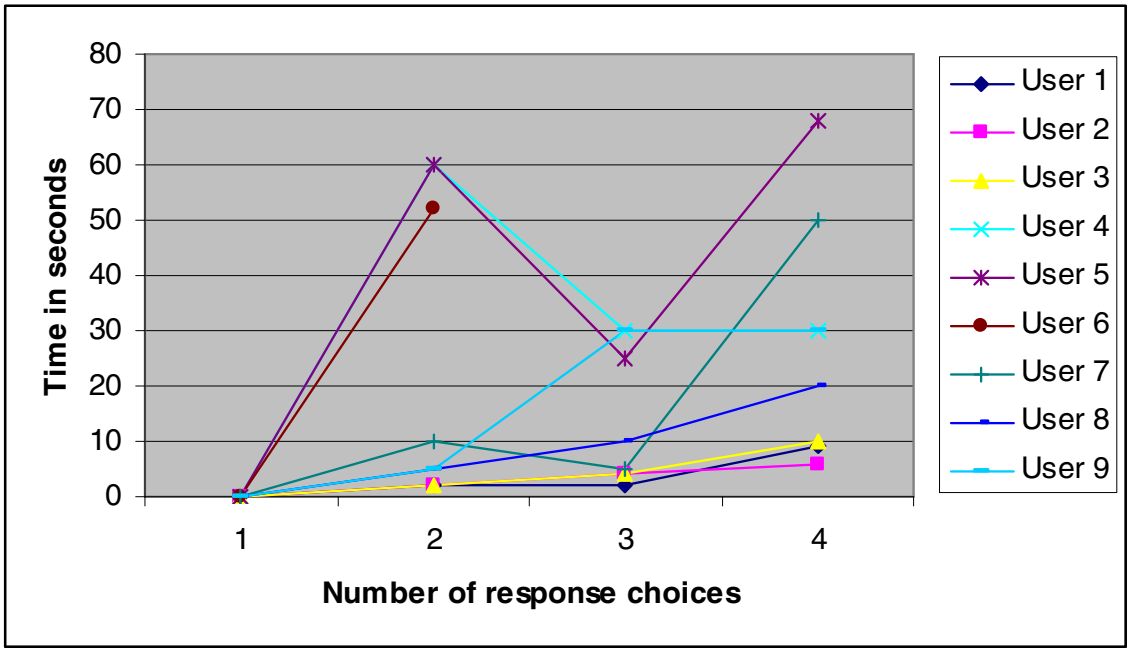


Figure 4.1. Graphical response time analysis of people with different disabilities.
X axis is the number of choices that starts from one and ends at four.
Y axis is time in seconds that goes up to 80 seconds in the graph.
9 curves are drawn based on the time taken for users with different abilities to respond to reminder messages with variable number of responses.
User 1 belonged to people with hearing impairments group.
Users 2 and 3 had no disabilities.
Users 4, 5, and 6 were people with visual impairments.
Users 7, 8 and 9 were people with motor impairments (problem with hands, in specific).
The fact that Users 6 and 7 were not able to respond at all is not shown in this graph.
Refer to the previous text for the explanation.

	Time taken to select a response choice (in seconds)				
User	Reminder with two choices	Reminder with two choices	Reminder with three choices	Reminder with four choices	Disability type
1	No data	2	2	9	Lack of Hearing
2	No data	2	4	6	None
3	No data	2	4	10	None
4	60	5	30	30	Lack of vision
5	60	15	25	68	Lack of vision
6	52	System timed out after 180 seconds	System timed out after 180 seconds	System timed out after 180 seconds	Lack of vision
7	10	System timed out after 300 seconds	5	50	Difficulty in hand movements
8	No data	5	10	20	Difficulty in hand movements
9	5	2	30	30	Difficulty in hand movements

Table 4.10. Response time analysis of people with different disabilities.

4.5 Conclusion

Persons with hearing impairments preferred to have the vibration setting turned ON whenever a reminder was delivered, and they had a similar user interface experience as that of people with no disabilities.

The data shows that people with visual impairments (partial and complete) took more time for the first message since they had to feel buttons positions, and took

relatively less time for consecutive messages. They had to spend some time in the beginning to identify the left and the right buttons, and once they had identified those buttons and get a feel for their positions, responding to latter reminders was relatively easy for them. However, one of the users (User 6) was not able to select any response for three of the four reminders. The user had difficulty in locating response buttons, since the device moved a little bit from its original position. This might be avoided if the user settings had more than 180 seconds as the message display time (on-duration). However, in this scenario, the time required to make a selection completely depends upon the user's capabilities and chances of locating the buttons. The user interface may be designed to have regular intervals of audio signal output from the device, which may aid the user to move towards the direction of the device. Another observation was that people with vision impairments were comfortable with using phone key buttons interfaces and there were no questions asked about speech recognition or any other alternative input modes.

People with motor impairments identified the left and the right buttons at once by looking at them, but for every message, they had to spend an additional effort to move their hands close to the response button thus taking relatively more time to respond to every message, which is in contradiction to the experience of people with vision impairments, who took extra time only for the first message and less time for subsequent messages. The difficulty of these users was identified as the small size of response buttons that were mounted on top of the PPD. Increasing the size of buttons by

attaching external input mechanisms may help people with motor impairments to respond easily.

User 7 in the group of persons with motor impairments responded quickly for the first message (5 seconds), and could not respond for 5 minutes for the next message. It is worth noting that the same user was able to respond in 5 seconds for the third message and 50 seconds for the fourth message. The continuous usage of hands may be a factor for the user not responding for 5 minutes for the second message. After a significant rest, the user was able to have free hand movement. The problem was that the user was able to go pretty close to the response button, but could not touch the button and after some time, even if the button was touched by the user, there was not enough pressure applied on the button to have an effect. Providing very sensitive buttons with a relatively bigger size may benefit this user.

CHAPTER 5

FUTURE WORK

We have implemented the reminder service to demonstrate the user interface capabilities in this thesis. The user interface component and the Connect infrastructure allow services to be added dynamically (for example, searching the web and displaying results to people with disabilities, a monitoring program to check if a wheel chair user is comfortable and is not falling down). The Connect messaging infrastructure supports the addition or deletion of a service component during runtime while other service components are still running on the device. However, deletion of a particular service makes that service unavailable for the future. The general guidelines to add a service to use the existing infrastructure and the user component are as follows:

The developers of a new service component don't have to worry about other existing service modules (for example reminder, news), instead only the knowledge of interfaces to infrastructure modules is required. The new service module does not communicate with the user directly since it is not aware of the user's delivery preferences and the device capabilities, instead the service module requests the delivery agent service. Similarly, to establish communication to Connect servers, the service module has to request the message gateway since only the message gateway has the knowledge about different possible ways to connect to the outside world (GPRS, Bluetooth). The new service module has to request the service of the process manager to schedule any other programs. The developer of the service module has to be familiar

with all Connect utilities which enhances the reusability as well as reduces the likeliness of similar errors. The new service module has to adhere to the general table schema so that the communication between any infrastructure module and the service module is consistent. If required to deliver contents to the user, the new service module interfaces with the delivery agent, by composing a delivery agent XML according to the guidelines of the delivery agent interface language.

Initially, when the user interface component was not defined, the user profile editor service (UPS) and the reminder editor service (RES) were developed with their own interfaces interacting to the user. They can be made adaptable to the user by interfacing with the delivery agent, the service independent user interface component.

As future work, the XML user interface language can be extended to accommodate different types of contents such as images, voice, movies, music and so on. Standard Music Development Language (SMDL) [SMDL], voice xml [VXML] and similar content representation may be explored to represent wider variety of data. An image may have to be presented with an explanation (similar to a write up of images in accessible websites, to benefit people using screen readers for web surfing), so that the delivery agent may represent the content to people with vision impairments. Similarly, an audio content may have to have text notes attached to it (similar to closed caption), so that it can be delivered to people with hearing impairments. Note that the service program does not actually know the user's ability; the service program provides all necessary modes of the content to the user interface component, when the content is not

a simple text. However, if the content is a simple text, we have implemented text-to-speech modality for people with vision and cognitive impairments.

Since our main focus of this research is to develop a semantic user interface language with emphasis on service independence and the user's preferences, we have not tried many multimodal techniques (such as gestures, eye movement, external buttons, speech recognition and so on). Many researchers have tried these multimodal techniques for their specific applications [O02] [CJM97] [FH04] [F05]. In the future, those multimodality components can be developed and integrated with our user interface component to achieve a wide variety of modality choices creating more "inclusive" [KCHR00] user interfaces.

Users' experiences need to be studied further by considering a greater number of diversified users to accommodate different groups of disabilities. More research needs to be done to collect requirements for finding services needed by users to improve their every day life. Our research focused on the usability of interfaces with the simple reminder service as an example. The usability of interfaces presenting complex contents of other services (such as the news service, the user profile editor service) needs to be studied.

REFERENCES

[ACM1] http://www.acm.org/ - October 2005.

[ACM2] http://www.acm.org/constitution/code.html#sect1 - October 2005.

[ACM3] http://www.acm.org/serving/se/code.htm - October 2005.

[ASSIST] http://assist.uta.edu/ - October 2005.

[BFM92] de Baar, D.J.M.J., Foley, J., Mullet, K.E. (1992). Coupling Application Design and User Interface Design. CHI'92 Conference Proceedings, Reading, Addison Wesley, 259-266.

[BM93] Benyon, D., and Murray, D. Developing adaptive systems to fit individual aptitudes. In Proceedings of the 1993 International Workshop on Intelligent User Interfaces (Orlando, Fla., 1993).

[BMJ] http://bmj.bmjjournals.com/cgi/content/full/bmj;328/7449/1162 - October 2005.

[BPC00] Billsus, D., Pazzani, M. J., and Chen, J. (2000). A learning agent for wirelessnews access. In Proceedings of the Srh International Conference on Intelligent User Interfaces, (IUI '00), USA, pg 33-36.

[BR01] Brown, S. S., & Robinson, P. A World Wide Web mediator for users with low vision. Paper presented at the CHI'2001 Conference on Human Factors in Computing Systems Workshop No. 14. Seattle, WA.

[BSHKM93] Benyon, D. R. (1993b) Accommodating Individual Differences through an Adaptive User Interface. In Schneider-Hufschmidt, M., Kühme, T. and

Malinowski, U. (eds.) Adaptive User Interfaces - Results and Prospects, Elsvier Science Publications, North-Holland, Amsterdam. 1993.

[BT86] Michael A. Bauer , Henry K. Ting, Developing application independent interfaces for workstations in a distributed environment, Proceedings of the 1986 ACM SIGSMALL/PC symposium on Small systems, p.7-15, December 1986, San Francisco, California, United States.

[CC03] Clarkson, J., R. Coleman, et al. (2003). Inclusive Design: Design for the whole population. London, Springer.

[CEN1] http://www.census.gov/prod/2005pubs/p23-208.pdf - Figure 1 - October 2005.

[CEN2] http://www.census.gov/prod/2005pubs/p23-208.pdf - Table D - October 2005.

[CEN3] http://www.census.gov/prod/2005pubs/censr-23.pdf - October 2005.

[CEN4] http://dsc.ucsf.edu/pdf/report13.pdf - Table A - October 2005.

[CEN5] http://www.ncddr.org/du/researchexchange/v06n01/survey2000.html - figures 2 and 3 - October 2005.

[CJM97] Cohen, P., Johnston, M., McGee, D., et al. Quickset: Multimodal interaction for distributed applications. In Proceedings of the Fifth ACM International Multimedia Conference (New York, NY) ACM Press, NY, 1997, 31–40.

[CSC04] Clauson KA, Seamon MJ, Clauson AS, Van TB. Evaluation of drug information databases for personal digital assistants. Am J Health Syst Pharm. 2004 May 15;61 (10) 10115-24.

[D03] Dixit, A., (2003) Portable device for the blind, MS thesis, University of Texas at Arlington.

[EPOCRATES] http://www2.epocrates.com/index.html - October 2005.

[EVP01] Einsenstein, J., Vanderdonckt, J., and Puerta, A., Applying Model-Based Techniques to the Development of UIs for Mobile Computers, in Proceedings of ACM Conference on Intelligent User Interfaces IUI'2001, ACM Press, New York, 2001, pp. 69-76.

[F05] P Froehlich., Non-speech sound and paralinguistic parameters in mobile speech applications. Conference on Human Factors in Computing Systems, 2005.

[FH04] Fröhlich P. and Hammer F., Expressive Text-to-Speech: A user-centred approach to sound design in voiceenabled mobile applications. Proc. Second Symposium on Sound Design (2004).

[FSMWL03] Fischer S., Stewart T., Mehta S., Wax R., Lapinsky S. Handheld computing in medicine. J Am Med Inform Assoc 2003.

[GDNG99] D. Gupta, M. Digiovanni, H. Narita and K. Goldberg, "Jestor 2.0: Collaborative Filtering to Retrieve Jokes", Proceedings on the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Berkeley, California, USA, August 1999.

[GMM97] Gori, M., Maggini, M., and Martinelli, E. Web-Browser Access Through VoiceInput and Page Interest Prediction. Proceedings of the Sixth International Conference on User Modeling. Wien, New York, Springer: 1997, 17-19.

[GN01] Gregor, P. and Newell, A.F., Designing for Dynamic Diversity - Making accessible interfaces for older people, Workshop on Universal Accessibility of Ubiquitous Computing, 2001.

[GSM] http://www.gsmworld.com/technology/gprs/intro.shtml - October 2005.

[H00] Hanson, V. L., Web access for elderly citizens, in Proceedings of the Workshop on Universal on Accessibility of Ubiquitous Computing, WUAUC'01 (Alccer do Sal, Portugal, May, 2000), ACM Press, 14-18.

[HKSR97] Hodes, T.D., Katz, R., H., Servan-Schreiber, E. and Rowe, L., Composable Ad-hoc Mobile Services for Universal Interaction. in MobiCom 1997, (1997).

[HOPKINS] http://mobileathopkins.jhu.edu/content/handouts/vandec.pdf - October 2005.

[K93] Kuhme, T. A User-Centered Approach to Adaptive Interfaces. Proceedings of the 1993 International Workshop on Intelligent User Interfaces. pp 243-246. Orlando, FL. New York: ACM Press 1993.

[KC03] Keates, S., and Clarkson, J., Countering design exclusion – An introduction to inclusive design. Springer-Verlag London Limited, 2003.

[KCHR00] Simeon Keates , P. John Clarkson , Lee-Anne Harrison , Peter Robinson, Towards a practical inclusive design approach, Proceedings on the 2000 conference on Universal Usability, p.45-52, November 16-17, 2000, Arlington, Virginia, United States.

[KPD96] Kennel, A., Perrochon, L., and Darvishi, A. WAB: World-Wide Web Access for Blind And Visually Impaired Computer Users. New Technologies in the Education of the Visually Handicapped. ACM SIGCAPH Bulletin, June 1996.

[L04] Lai, J.C. Facilitating Mobile Communication with Multimodal Access to email Messages on a Cell Phone. In Proceedings of CHI 2004.

[LAB04] Leporini B., Andronico P., Buzzi M. Designing Search Engine User Interfaces for the visually impaired. In The Proceedings of the ACM W4A International Cross-Disciplinary Workshop on Web Accessibility 2004, at the Thirteenth International World Wide Web Conference, 18th May 2004, New York (NY).

[M02A] G. Menkhaus, "An Architecture for Supporting Multi-Device, Client-Adaptive Services," Annals of SoftwareEngineering, vol. 13, June 2002.

[M02B] G. Menkhaus, (2002), Adaptive User Interface Generation in a Mobile Computing Environment, Ph.D. dissertation, University of Salzburg, Austria.

[MP02] Guido Menkhaus, Wolfgang Pree: A Hybrid Approach to Adaptive User Interface Generation. In: Journal of Computing and Information Technology (CIT), 10(3), 2002.

[MPOM98] S. Morley, H. Petrie, A. O'Neill, and P. McNally, "Auditory navigation in hyperspace: Design and evaluation of a nonvisual hypermedia system for blind users," in Proce. 1998 3rd Int. ACM Conf. Assistive Technol., ASSETS'98, Marina del Ray, CA, Apr. 1998, pp. 100–107.

[MWYYNM02] Brad A. Myers, Jacob O. Wobbrock, Sunny Yang, Brian Yeung, Jeffrey Nichols, and Robert Miller. "Using a Handheld to Help People with

Motor Impairments". Fifth International ACM SIGCAPH Conference on Assistive Technologies; ASSETS 2002. July 15-17, 2002. Edinburgh, Scotland. pp. 89-96.

[NB02A] Nylander, S. and Bylund, M. (2002) Device Independent Services, SICS Technical Report T2002-02, Swedish Institute of Computer Science.

[NB02B] Nylander, S. and Bylund, M., Providing Device Independence to Mobile Service. in 7th ERCIM Workshop User Interfaces for All, (2002).

[NBW03] The Ubiquitous Interactor – Mobile Services with Multiple User Interfaces. Stina Nylander, Markus Bylund, Annika Waern. Published as a SICS Technical Report, no. T2003-17. Shorter version submitted to the conference Computer-Aided Design of User Interfaces.

[NG97] Newell, A.F. & Gregor, P., (1997). Human computer interfaces for people with disabilities, in Handbook of Human-Computer Interaction, Helander, M., Landauer, T.K. and Prabhu, P. (eds), Elsevier Science BV, (ISBN 0 44481862 6) pp 813-824.

[NMK91] Nakatsuyama, M., Murata, M., Kusumoto, K.: A new framework for separating user interfaces from application programs. SIGCHI Bulletin 23(1):88 - 91, 1991.

[O02] Sharon Oviatt, Chapter in Handbook of Human-Computer Interaction, (ed. by J. Jacko & A. Sears), Lawrence Erlbaum: New Jersey, 2002.

[O99] Oviatt, S.L. Ten myths of multimodal interaction. Communications of the ACM 42, 11, (Nov. 1999), 74–81.

[OJNMF00] Olsen, D.J., Jefferies, S., Nielsen, T., Moyes, W. and Fredrickson, P., Cross-modal Interaction using XWeb. in UIST 2000, (2000).

[PMMOM97] H. Petrie, S. Morley, P. McNally, A. O'Neill, and D. Majoe, "Initial design and evaluation of an interface to hypermedia systems for blind users," in Proc. ACM Conf. Hypertext, New York, Apr. 1997, pp. 48–56.

[PS02] Paternò F. and Santoro C., One Model, Many Interfaces, in Proceedings of CADUI 2002, the 4th International Conferenceon Computer-Aided Design of User Interfaces (Valenciennes, France, May 2002), 143-154.

[REMCOM] http://www.pebbles.hcii.cmu.edu/software/remotecmd/index.php - October 2005.

[S01] Stephanidis, Constantine, User Interfaces for All: Concepts, Methods, and Tools, Mahwah, N.J. Lawrence Erlbaum Associates, Inc., 2001. ISBN 0805829679.

[S01PG7] Stephanidis, C., User Interfaces for All: Concepts, Methods, and Tools, Mahwah, N.J. Lawrence Erlbaum Associates, Inc., 2001. ISBN 0805829679, pg 7.

[S01PG3] Stephanidis, Constantine, User Interfaces for All: Concepts, Methods, and Tools, Mahwah, N.J. Lawrence Erlbaum Associates, Inc., 2001. ISBN 0805829679, pg 3-15.

[S01PG65] Stephanidis, Constantine, User Interfaces for All: Concepts, Methods, and Tools, Mahwah, N.J. Lawrence Erlbaum Associates, Inc., 2001. ISBN 0805829679, pg 65-78.

[S01PG81] Stephanidis, Constantine, User Interfaces for All: Concepts, Methods, and Tools, Mahwah, N.J. Lawrence Erlbaum Associates, Inc., 2001. ISBN 0805829679, pg 81-82.

[S01PG91] Stephanidis, Constantine, User Interfaces for All: Concepts, Methods, and Tools, Mahwah, N.J. Lawrence Erlbaum Associates, Inc., 2001. ISBN 0805829679, pg 91-92.

[SC03] Steriadis, C.E. and Constantinou, P. Designing Human-Computer Interfaces for Quadriplegic People. ACM Transactions of Computer-Human Interaction. Volume 10, Number 2. June 2003.

[SE97] SPOONER, R. I. W., and EDWARDS, A. D. N. User Modelling for Error Recovery: A Spelling Checker for Dyslexic Users. Proceedings of the Sixth International Conference on User Modeling. Wien, New York: Springer, 1997, 147-157.

[SH01] Lalitha Suryanarayana, Johan Hjelm, "CC/PP for content negotiation and contextualization", Second International Conference on Mobile Data Management, MDM 2001. Also in Lecture Notes in Computer Science, vol. 1987, Springer Verlag 2001.

[SHOCUT] http://www.pebbles.hcii.cmu.edu/software/shortcutter/index.php - October 2005.

[SMDL] http://xml.coverpages.org/xmlMusic.html#smdl - October 2005.

[SQLITE] http://www.sqlite.org/ - October 2005.

[SR99] Siebra S. de A. and Ramalho G. L. Athena: An user-centered adaptive interface. In H.-J. Bullinger and J. Ziegler, editors, Human-Computer Interaction, volume 1, pages 346–350. Lawrence Erlbaum Associates, 1999.

[SS95] Savidis, A., and Stephanidis, C. Developing Dual User Interfaces for Integrating Blind and Sighted Users: the HOMER UIMS. Proceedings of the CHI'95 Conference on Human Factors in Computing Systems, Denver, CO, 1995, 106-113.

[TP97] TREWIN, S., and PAIN, H. Dynamic Modelling of Keyboard Skills: Supporting Users With Motor Disabilities. Proceedings of the Sixth International Conference on User Modeling. Wien, New York: Springer, 1997, 135-146.

[TMFMMP05] Kimberly Tee, Karyn Moffatt, Leah Findlater, Eve MacGregor, Joanna McGrenere, Barbara Purves, Sidney S. Fels, A visual recipe book for persons with language impairments, Proceedings of the SIGCHI conference on Human factors in computing systems, April 02-07, 2005, Portland, Oregon, USA.

[VXML] http://www.voicexml.org/ - October 2005.

[W3CCPP] http://www.w3.org/TR/CCPP-struct-vocab/ - October 2005.

[W3DI] http://www.w3.org/2001/di/ - October 2005.

[W3HTML] http://www.w3.org/MarkUp/ - October 2005.

[W3MMI] http://www.w3.org/2002/mmi/- October 2005.

[W3MMIARCH] http://www.w3.org/TR/mmi-arch/ - October 2005.

[W3MMIREQ] http://www.w3.org/TR/mmi-reqs/ - October 2005.

[W3SMIL] http://www.w3.org/TR/SMIL-access/ - October 2005.

[W3SVG] http://www.w3.org/TR/SVG-access/ - October 2005.

[W3WAI] http://www.w3.org/WAI/ - October 2005.

[W3XAG] http://www.w3.org/TR/xag - October 2005.

[W3XHTML] http://www.w3.org/WAI/References/HTML4-access October 2005.

[W3XML] http://www.w3.org/XML/ - October 2005.

[WCORNELL] http://www.ilr.cornell.edu/ped/disabilitystatistics/cps.cfm?n=2&submit=true&statistic=prevalence - October 2005.

[WLOKJMJ04] Jim Warren, Maria Lundström, David Osborne, Monica Kempster, Sara Jones, Chunlan Ma, Mark Jasiunas. A Multi-Interface, Multi-Profiling System for Chronic Disease Management Learning. Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 6 - Volume 6.

[YH04] Jing-Hua Ye and John Herbert: Framework for User Interface Adaptation. In: 8th ERCIM Workshop “User Interfaces for All” 2004.

[ZKHL05] Záruba G., Kamangar F., Huber M., and Levine D. CONNECT: A Personal Remote Messaging and Monitoring System to Aid People with Disabilities, IEEE Communications Magazine, September 2005.

[ZP96] Zajicek, M. and Powell, C. 1996, Building a conceptual model of the World Wide Web for visually impaired users. Proceedings of Ergonomics `96.

BIOGRAPHICAL INFORMATION

The author received her Masters of Science in Computer Science and Engineering in December 2005 from The University of Texas at Arlington. She received her Bachelor of Engineering in Electrical Engineering from the University of Madras, India, in November 1997. She worked for Infosys technologies, Chennai, India for two years as a systems analyst and worked as a software consultant for three years for 3M in Austin, Texas in their telecom systems division, designing and developing software for test and measurement handheld devices used by today's leading telecom service providers in the industry.

She got interested in the assistive technology in her early stages of her Master's program. She started working as the technical lead for the Connect project at UTA, while pursuing her thesis in the related area. She has received competitive departmental research assistantships and internships throughout her studies at UTA with the recommendation from her professors.

The author has served as the vice president (2004-2005) for the Tau Beta Pi (Texas Eta chapter), and is a member of the Upsilon Pi Epsilon (Texas Gamma chapter).