ISCA Tutorial and Research Workshop - Multi-Modal Dialogue in Mobile Environments - June 17 - 19, 2002 - Kloster Irsee, Germany

Multi-modal Spoken Dialog with Wireless Devices[1]

Roberto Pieraccini, Bob Carpenter, Eric Woudenberg, Sasha Caskey, Stephen Springer, Jonathan Bloom, Michael Phillips
SpeechWorks International - 695 Atlantic Avenue, Boston MA 02111 - 17 State Street, New York, NY 10004

Abstract: We discuss the various issues related to the design and implementation of multi-modal spoken dialog systems with wireless client devices. In particular we discuss the design of a usable interface that exploits the complementary features of the audio and visual channels to enhance usability. We then describe two client-server architectures in which we implemented applications for mapping and navigating to points of interest.

Keywords: multi-modal, dialog, user interface, architecture.

1. INTRODUCTION

Speech-only dialog systems are a reality today and are being deployed in telephone-based commercial applications all over the world. As personal communication devices become more and more powerful, the distinction between personal telephones and hand-held computers is becoming fuzzier. In view of this evolution, combined with the convergence of the Internet and telephony, we are observing a growing interest in applications where speech cooperates with other communication modalities, such as visual display, pen, gestures, etc. Hence we are observing a growing interest both in research and industry toward what are generally referred to as multi-modal user interfaces.

[1] The study reported in this paper was funded by DARPA under contract N66001-00-1-8958 through SPAWAR.

Voice based user interfaces and graphical user interfaces have been successfully employed, independently, as the principal interaction modalities for many applications. However the integration of the two into a multi-modal interface still presents many unanswered questions. Considering hand-held devices, we must be aware that the same multi-modal application could be potentially used in a variety of different situations. For instance users may find themselves in situations where both hands are busy but their aural and visual senses are available, even intermittently, as in a car while driving. In other situations the users may not be allowed to freely use spoken input and output, as during a meeting, or the audio channel may be unreliable as in a noisy environment like an airport. Finally there might be situations where both hands and eyes are busy and the device is out of sight, but users can talk to the application through a remote wired or wireless microphone. A well-behaved user interface should be able to function in all of these situations.

Moreover, the design of user interfaces that integrate both audio and visual channels must take into account that the two modalities can be used co-operatively and take advantage of each other's capabilities, especially for coping with limitations that are intrinsic to each independently. Some of the limitations of the voice-only interface derive from the narrow information channel provided by speech. Users cannot absorb and retain large quantities of information such as long lists of items. If it is necessary to present a relatively long list, systems may rely on audio list browsing (e.g. "*Say next or previous"*), which is not particularly efficient. Information presented only through speech is volatile and systems must rely on the user's short-term memory. Thus it is often very hard to present additional information that could help the user keep the dialog on course, such as information about what the system knows at a certain point in dialog, or hints on what to say next. Finally, there is information, such as visual information like maps and pictures that cannot be conveyed through voice.

On the other hand, the limitations of graphical user interfaces on portable devices derive from the absence of convenient text input methods and the small size of the display. Anyone who has used a PDA for input of free form text knows how painfully slow the process is. The need for speech as a text input mechanism is evident. Moreover, as stated before, there are situations, such as in a car, where the user can readily take advantage of the visual output but cannot manually input data. Of course when the two modalities are brought together on the same device, the user interface should take advantage of their complementary capabilities. For instance, the display can include elements that suggest to the user what can be said and indicate how confident the recognizer is with the current interpretation of the utterance. At

the same time the system should offer non-speech input modalities, such as buttons and scrollable lists. Finally, spoken prompts and *earcons* (the audio equivalent of icons) can be used for reinforcing elements of the user interface.

System architecture is another important aspect of multi-modal applications. There are several approaches that can be taken within the limitations of today's technology. Those concern the choice of embedded or network based speech recognition and text-to-speech processing. An interesting hybrid solution is what is known today as distributed speech recognition (DSR), where part of the speech recognizer (typically the front end signal processing) resides on the client device. Of course the choice of architecture depends on the application and on the environment in which the application is used. Other architecture considerations are related to ease of development for multi-modal applications. Although good abstractions, generally based on the concept of call-flow, exist today for the design of voice-only dialog applications we do not have an equivalent abstraction for multi-modal applications. This complicates the design of multi-modal applications.

2. WHY MULTI-MODAL WIRELESS?

Mass adoption of wireless devices truly capable of supporting multi-modal applications has not yet occurred, mainly due to limitations in network infrastructure. One of the few exceptions is probably the WAP cellular telephone, which can count a significant number of users[2], especially in Europe. WAP telephones, which are characterized by a small text-based display, typically do not allow simultaneous transmission of voice and data. A multi-modal application designed for WAP telephones would require the user to switch between speech and text mode anytime that she needs to change input or output modality. However cumbersome that might seem, a WAP telephone multi-modal service would present several advantages over a speech-only or text-only application on the same device, at least for certain applications. For instance, let us consider a voice-only navigation application. The user would call a service's telephone number and be instructed to speak a starting address and a destination address. The system would compute the itinerary as a sequence of steps and recite them one at time, allowing the caller to navigate the list with simple commands such as *next*, *previous*, etc. For a long itinerary, a voice-only modality for providing the list of directions would prove quite impractical, since the user would have to keep

[2] However, field studies (http://www.useit.com/alertbox/20001210.html) indicate that at least 70% of owners of WAP telephones in UK said that they would not be using WAP in a year.

the call open for the entire journey, or take separate note of the steps, or call back every time the next step was required. Conversely, in a text-only version of the application, while the whole itinerary could be stored in the telephone and easily browsed by the user, entering a street address using the 12 keys of a telephone would prove tedious and impractical. This is certainly a case where even a multi-modal interaction that was strongly limited to one modality at a time would be useful. In fact the same navigation application, implemented on a WAP telephone and supported by an infrastructure for switching between voice and text modes, would allow the use of voice for entering an address yet store the resulting directions on the telephone as text. The user could browse them at leisure without any need for keeping the call open.

Besides WAP phones, we are witnessing increasing availability of more sophisticated wireless devices, such as PDAs (e.g. Palm, iPAQ) and smartphones, characterized by powerful processors and displays with higher resolution than those in cell phones. However sophisticated and powerful these devices might be, text and data entry will be constrained by their small size. Mobile devices are becoming smaller and smaller, preventing the use of a keyboard or even a touchpad. Text can be entered, albeit clumsily, with a stylus on a touch sensitive display using handwritten "graffiti" or through taps on a soft keyboard. Speech remains an obvious alternative choice for data entry for these devices. In spite of the fact that the number of owners of PDAs and smartphones is growing rapidly, they are not yet widespread due to their cost and size, and due to the paltry data rates (less than 14kbps), high fees, and spotty coverage of current wireless Internet.

We will probably see increased interest in wireless devices when 3G networks become available. 3G devices, characterized by the simultaneous flow of voice and data, and by being "continuously connected" to the network, will soon be widely available and at a reasonable cost. The continuing evolution of mobile devices toward larger memories, faster processors, and better displays, all in smaller and smaller packages, make the use of speech for data entry increasingly appropriate.

3. WALKING DIRECTIONS APPLICATION

The case study described in this paper refers to the development of an application supplying mapping and directional navigation to points of interest. The development of this application was funded by DARPA within the Communicator project [2] and carried out in collaboration with Compaq [3], Lobby7 [4] and MapQuest [5]. The application addresses the scenario of a user walking in an unknown city and trying to find her way to a destination

location. The target client device is a Compaq iPAQ 3765 PocketPC. The application provides functions that include selecting a city, a start and/or an end address, displaying a map, controlling the display of the map (zooming and moving), displaying directions on the map and displaying interactive directions in the form of a list of successive steps. The design of the user interface for this application assumes a mixed style of interaction, where the user will be able to either speak to the device, or enter information through a stylus, or both. We developed this application under two different architectures. The first assumes we have tight control of the widgets displayed on the client device. This would presumably require using a proprietary browser, or a least a browser that is capable of fine grain control of the GUI. The second implementation uses the standard Pocket Internet Explorer browser that comes with the PocketPC OS. Internet connectivity was established with a 802.11b wireless PC card connected to the iPAQ via its PC card sleeve.

4. SPEECH TECHNOLOGY FOR MULTI-MODAL WIRELESS

Speech processing for multi-modal wireless application centers on two technologies: ASR (automatic speech recognition) for speech input and TTS (text to speech) for speech output. One question that arises when considering a wireless multi-modal application accessing network based content concerns the configuration of ASR and TTS engines. In general there are two possible choices for each engine: a network server or an embedded configuration. In the network server configuration, the engine (either the ASR or the TTS, or both) runs as a server at a remote location with speech being transported to/from the client via wireless connection. In the embedded configuration the engine runs instead on the client. The choice between a network or embedded configuration depends upon several factors. For ASR, it depends on the availability of a small memory footprint speech recognizer that supports the required vocabulary with the speed and accuracy needed for the application. For TTS, the choice depends on the quality requirements for the speech output. Generally small memory-footprint embedded TTS has a lower speech quality than network TTS engines. One can think also of configurations where network and embedded engines are used at different times during an application. An application might use a network speech recognizer for the data entry phase of its operation, for instance to enter the starting and ending address in a mapping application. The same application might sensibly use an embedded recognizer for navigating the downloaded information, such as the list of direction steps (e.g. *next, previous, etc.*) or maps (e.g. *zoom in, zoom out, move left, etc.*). In this case, network connectivity is re-

quired only during a limited portion of the transaction, but speech can continue to be used as an input modality even when the network is not available.

Another consideration for the choice between network and embedded speech resources is with respect to the required bandwidth. It is obvious that network-based speech processing requires a larger bandwidth to the device than would embedded processing. For instance, in the case of an embedded speech recognizer, one would need to send to the application server only the transcription (or *n*-best transcriptions) obtained by the embedded ASR engine. Similarly, in the case of embedded TTS, only the text version of a prompt needs to be sent to the client through the wireless connection. Alternatively, when network resources are used, the whole speech signal (possibly in a compressed format) needs to be exchanged between the wireless client and the application server.

A solution that offers the benefits of network based speech recognition and the limited bandwidth required by an embedded engine is what is commonly known as *distributed speech recognition* (DSR). With DSR the speech recognition front end (i.e. the part of the speech recognizer that collects speech and transforms it into a sequence of feature vectors) resides on the client. Only the compressed feature vectors (generally one feature vector for every 10msec of speech) are encoded and transmitted over the wireless digital network. The speech recognition search engine resides on the application server and processes the feature vectors to produce the recognition results. The obvious advantage of DSR is the reduced bandwidth required: recognition performance equivalent to that on toll quality speech can be achieved at a meager 4.8 to 9.6 kbps data rate (vs. 64kbps for conventional toll quality telephony). Latencies can also be lowered due to the fact that end pointing can be implemented on the client. Moreover, the computational power required on the client for the implementation of DSR is small when compared to the network-based recognizer. We can also envision solutions where the embedded and the network recognizers share the same front end, which remains on the client. ETSI, the European Telecommunication Standards Institute, created the Aurora Working Group [1] with the task of defining the feature extraction algorithms and standardizing the DSR protocol.

5. USER INTERFACE ISSUES

One can think of a multi-modal interface from two perspectives: a visual-centric or a speech-centric one. In the visual-centric approach the design starting point is an existing Graphical User Interface (e.g. from a web or a standalone application), and the multi-modality is achieved by adding the

option of speech input, for instance allowing the use of speech for operations such as clicking links, pushing buttons and entering text in fields. Conversely, the speech-centric approach starts with an existing Voice User Interface (VUI) and adds GUI elements in order to enhance and improve the dialog experience. While we do believe that a multi-modal application should be designed as *multi-modal* from the beginning, we feel that the speech-centric approach is useful for understanding how the GUI and VUI can complement each other synergistically, rather than simply attempting to *speech enable* an existing GUI application.

For a speech only application, several elements would greatly benefit from the addition of visual information. Many forms of information are not easily and effectively conveyed by speech alone. Certainly images, like maps or pictures, cannot be presented in a speech-only application, but even medium to long lists are difficult without the aid of a display. Speech-only navigation of long lists can be tedious and cumbersome to the user, while this is relatively easy with the help of a scrollable display. Another element that makes the design of speech-only interfaces hard is the difficulty of making the user aware of the state of the dialog. Users can get lost in a large application, not knowing where they are in the call flow, and not having in front of them the choices they have made during previous turns. Displaying the state of the dialog, both by showing the previous choices and by persistently identifying the application state would certainly help users. Finally, letting the user know what was understood and what to say next in a visual manner would improve the usability of the application.

5.1 Situationalization

One of the problems with designing the UI for an application based on a multi-modal mobile client is that there is a wide variety of situations in which the user could be interacting with the system. For instance, a user could potentially be in any of the following situations:

- Sitting at a desk,
- Getting out of a cab, building, subway and preparing to walk somewhere,
- Walking somewhere with hands free,
- Walking somewhere carrying things,
- Driving somewhere in heavy traffic,
- Driving somewhere in light traffic,
- Being the passenger in a car and not having to watch traffic,
- Being in a high noise environment such as a station, airport, or restaurant

All of these situations have a different ideal balance of visual and audio interaction. A single invocation of a multi-modal application might persist through several situations such as those above as the user moves through the real world. And it is probably unrealistic to assume that the multi-modal device could detect when the user is looking or whether their hands are available or not.

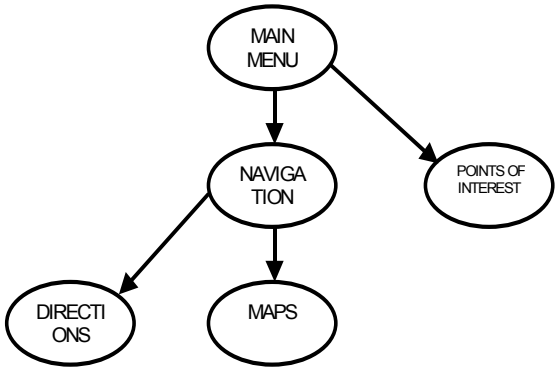

Figure 1: High level application description

One could possibly reduce all the different balances of visual and audio to 4 complementary modalities. Two of them are the single modes (visual only and speech only) and two multi-modes that we can identify as *visual-dominant* and *speech-dominant*. The choice of the modality would affect the design of the prompts and of the GUI. However, from a practical point of view, it is too expensive to consider even four modalities, since this would result in designing four different interfaces. Another problem would be for the user to learn how to select the right modality. A reasonable approach involves designing only one mixed-mode user interface, and allowing for two sets of prompts: short prompts for the mixed-mode or visual-dominant use, and long and more informative prompts for voice-only or voice-dominant use.

5.2 Description of the application

As in a spoken dialog system we can represent an application as a graph (often called a call-flow in spoken dialog system) where each node represents a distinctive state. The graph shown in Figure 1 is a high-level description of the considered application. For a multi-modal application it is convenient to associate each distinct page or form that will appear on the visual display with a node in the graph representation. Thus a node can represent the collection of more than one piece of information. For instance the MAIN MENU node would collect, from the user, which branch of the application should be activated (i.e. NAVIGATION or POINTS OF INTEREST), but

also, for instance, the information about which city/state is the object for the following requests.

Moreover each node would include other universal commands, such as a *main menu* - that would force the application to return to the main menu page, *back* - that would cause the return to the previously executed node, and a *help* button that would display a help page or play a helpful prompt. Additionally the page might have tabbed panes for moving to other branches and options to change previously selected features. An example of the display layout for a certain node of the application is shown in Figure 2.

Figure 3 shows an example of the low-level interaction description within one node of the application (commands such as back and help are excluded for simplicity). The transitions are ordered and have conditions. For instance, if a NUMBER is not available (condition !NUMBER), the node called NUMBER is invoked, which would set the focus on the number field and select the ASR grammar for numbers. At the start node an ASR grammar is activated that allows for compound sentences that include all the fields, as for example "*twenty three mountain boulevard"* or "*the number is twenty three and the street is mountain boulevard"*.

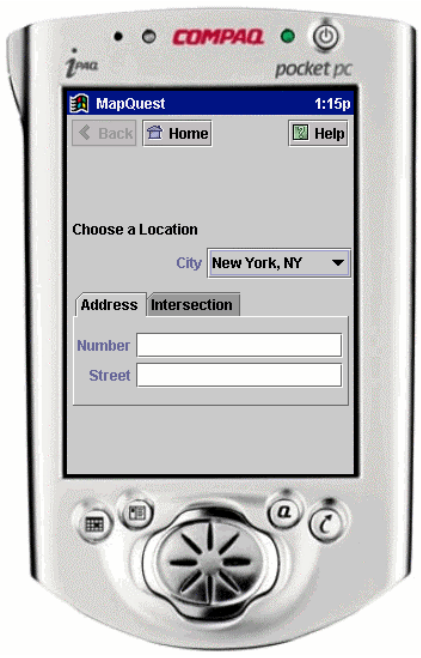

Figure 2: Fields and commands on the multi-modal device

5.3 Speech Collection Protocol

Unlike a telephone-based speech application, where the speaker is prompted and the dialog turns alternate synchronously between the user and the machine, our multi-modal applications are inherently asynchronous. Users are not compelled to speak or input visual data within a certain period after the prompt. Moreover, a system based on a handheld client cannot run in an *always listening* mode due to issues of battery consumption and because spurious speech and noise can inadvertently be taken as input. In our implementation we adopted a push-to-talk paradigm, which we believe is appropriate

for this kind of application. The button on the left-hand side of the client device is depressed to start speech collection, and kept depressed until the end of the utterance. This allows the user to operate the application with only one hand. An alternative would be to touch the screen with the stylus to start speech collection. Although this solution would also allow the focus to be set on the displayed page (allowing the appropriate recognition grammar to be simultaneously chosen when a field or text box is selected), we discarded it because it requires the use of both hands, which can be impractical in many situations.

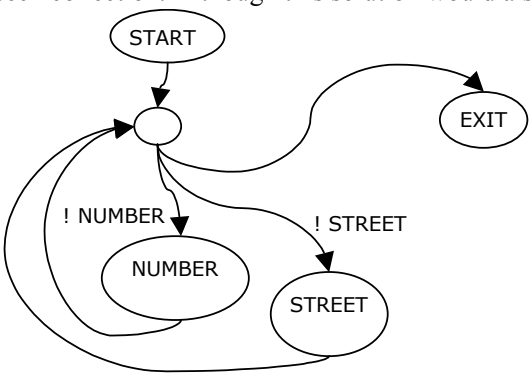

Figure 3: Low level interaction description.

Another issue is that in a push-to-talk paradigm speakers tend to speak simultaneously with, or even slightly before, pushing the collection button. The problem becomes more serious when any delay exists between the time the collection button is depressed and the time recording actually begins. In an informal test we noticed that up to 30% of the utterances collected in a push-to-talk modality were cut at the beginning. We solved this problem by playing a short tone subsequent to the button press as soon as the system actually starts recording speech. The tone helps users learn when to speak. As a result almost none of the utterances that were collected after the start-collection tone was introduced were cut at the beginning. An analogous problem is that speakers tend to release the button right at or just before the end of the utterance. However this problem can be corrected by allowing the speech collection to continue for a short time after the button is released.

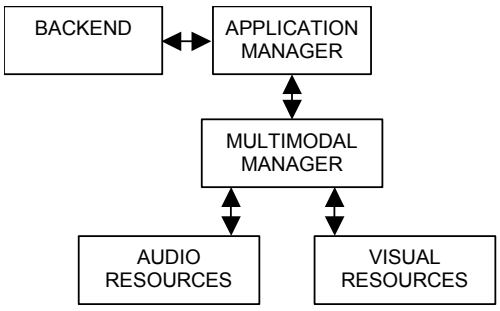

Figure 4: Abstract multi-modal architecture

6. MULTI-MODAL ARCHITECTURAL ISSUES

The state of a multi-modal application can be decomposed into two sub-states. The first is the state of the application's internal logic, identified by each node of Figure 1; the second is the state of the application's user interface. Both the visual and the audio interfaces go through different states, and a transition in one, caused by external events originating from the user or the backend, can precipitate a transition in the other. This is shown by example in Table 1, which describes the state of the audio and visual interfaces during input to a *Form* having two input fields (F1 and F2) and a submit button.

Figure 4 shows an abstract architecture of a multi-modal system. The application manager is responsible for the application state while the multi-modal manager communicates with the audio and visual resources and is

User interaction events	Visual State	Audio State
	Display form	
	Set focus on field F1	
		Start playing prompt for field F1
		Start recognizer with grammar F1
User selects field F2		
	Set focus on field F2	
		Halt recognizer
		Start recognizer with grammar F2
User speaks item for F2		
		Recognizer returns result for F2
	Result displayed in F2	
	Set focus on field F1	
		Start playing prompt for field F1
		Start recognizer with grammar F1
User speaks item for F1		
		Recognizer returns results for F1
	Result displayed in F1	
	Set focus on submit	

Table 1: Visual and audio states of a data input Form.

responsible for the interface state. The distinction between the blocks of Fig. 4 may not be obvious or they may be connected differently depending on the implementation of the particular multi-modal system, although their functionality still remains.

In a multi-modal wireless application the visual and the audio resources can be entirely or partially on the client. For instance, in the case of DSR, the speech recognizer is partially on the client (i.e. the front end) while the rest (i.e. the search engine) is on the server. If an embedded solution is adopted, the speech recognizer and the TTS are entirely on the client. The separation of the audio and visual resources between the client and the server also determines the degree of interaction possible between them.

6.1 Tight multi-modal control

Figure 5 shows the architecture that we implemented to allow tight multi-modal control of the audio and visual resources. The wireless device includes a thin audio client that is able to collect speech and send its digital representation to the server. Similarly the audio client is able to receive prompts and play them. The GUI client was implemented in Java[3]. It receives control frames from the server and is able to create and destroy widgets or modify their properties (e.g. change the color or shape of a button, or fill a text field). The GUI proxy maintains a symbolic representation of the visual state. Anytime the visual state changes because of a user event (e.g. the user clicking on a button) the GUI proxy is notified. Similarly, any-

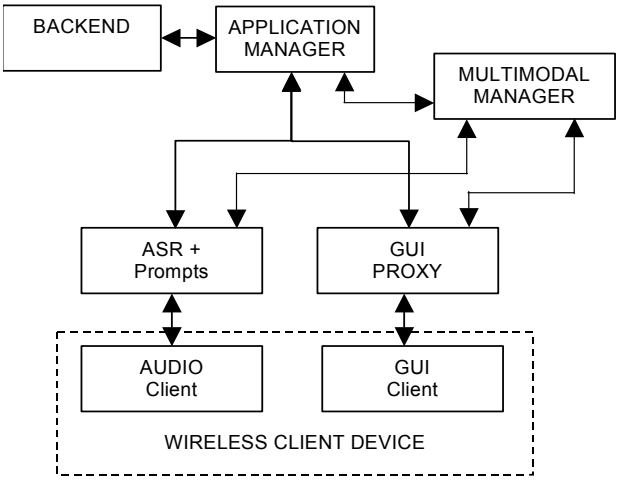

Figure 5: Tight multi-modal control architecture

[3] At the time of the implementation of this architecture Java's Swing interface package was not available for Pocket PC.

time the multi-modal manager requests a change on the display (e.g. filling a field with the text resulting from recognition of a speaker utterance), the state of the GUI proxy is changed accordingly, and the GUI proxy propagates the state change to the GUI client, which effects the change.

The multi-modal manager is implemented as a set of event handlers. Significant events are the return of an ASR result or a change in the visual state. Event handlers act on the ASR process, the prompt output, and the visual state. For instance, when a visual event is detected (e.g. a button is clicked), the ASR process is halted and a playing prompt is interrupted (i.e. a *visual* barge-in). Similarly when the ASR returns a result, an event handler changes the visual state accordingly.

The application manager is implemented by ETUDE, a dialog manager based on a recursive transition network representation of dialog [7]. The ETUDE engine offers the possibility of describing the dialog with a graph similar to the ones shown in Figures 1 and 3 where the arcs are ordered and have logical conditions that are verified on the application state. The application state is stored as a frame (i.e. a hierarchical structure of key/value pairs). The ETUDE framework offers, as an additional feature, the possibility of having most of the universal commands (e.g. *backup*, *main-menu*, *start-over*, *go to* <another branch of the application>) handled directly by the engine. For instance, the designer can specify whether a node is a *backup anchor* and whether a node allows the user to perform a backup action. When the user chooses *backup*, if the current node allows it the engine will return to the most recently visited backup anchor node and restore the application state (i.e. the engine will perform an *undo* operation). This set of universal commands is quite useful in multi-modal applications, and including support logic for them in the dialog engine simplifies adding them to any application.

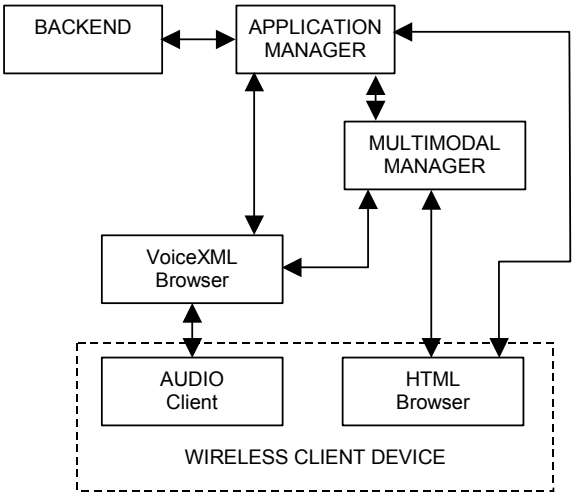

Figure 6: Loose multi-modal control architecture

6.2 Loose multi-modal control

Although the architecture described in the previous section allows for arbitrarily tight control and interaction between the visual and audio state, there are several issues that concern its practical implementation in commercial applications. Among those:

- A proprietary browser that runs on the mobile client has to be implemented for each possible device. Some devices may not support such an implementation. For instance, at the time the demonstrator was built, we could not find a Java virtual machine for PocketPC that implemented Java's Swing graphics package.
- In addition, having to download and configure a proprietary browser results in an additional burden for potential users.
- Depending on how tightly coupled the audio and the visual modalities are, the amount of communication between the server and the client can make the application slow and unusable on a network with low bandwidth or high latency.
- Proprietary communication and presentation protocols, as opposed to standard ones, such as HTML and VoiceXML, can make it difficult to port applications across platforms.

In order to alleviate these problems, multi-modal platforms and application makers favor solutions that involve standard markup languages. In the absence, as yet, of a multi-modal standard[4], developers tend to rely on existing standards for voice (i.e. VoiceXML) and visual browsers (i.e. HTML).

We developed a fully functional prototype of the discussed application in collaboration with Lobby7 [4], using Lobby7 technology that leverages the VoiceXML and HTML standards. The prototype follows the architecture described in Figure 6. A thin client collects speech from the device microphone and sends it, through a wireless Internet connection, to a voice browser (SpeechWorks' Open Speech Browser). Additionally, the client plays prompts, monitors barge-in, and navigates *Pocket IE* (the PocketPC HTML browser) to HTML pages generated by the multi-modal manager.

The function of the multi-modal manager is that of updating the VoiceXML and HTML documents and causing their respective browsers to load them after each user input (input being speech or visual-based).

[4] The World Wide Web Consortium (W3C - http://www.w3.com) formed a Multi-modal Interaction Activity working group (http://www.w3/2002/mmi). The group was formed in February 2002 with the charter of creating the specification for a multi-modal standard by February 2004.

The application manager consists of a set of Java Server Pages managed by the Tomcat servlet container on an Apache web server. The backend data (maps and directions) is provided by the MapQuest enterprise [5] database server (Figure 7).

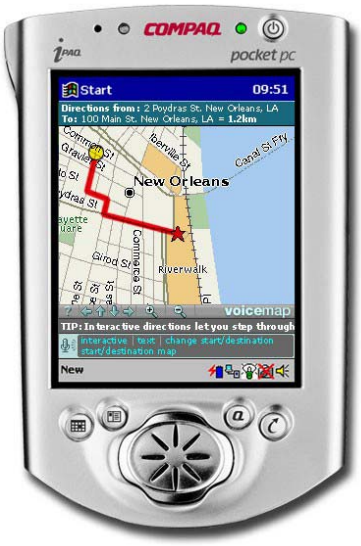

Fig. 7: Map and direction rendering.

7. CONCLUSIONS

We discussed issues related to the implementation of a multi-modal application on a wireless client device. The user interface issues are important since they determine the usability of the application. Exploiting the complementary features of the visual and audio channels we can build applications that will provide usability superior to GUI-only and speech-only interfaces. However, the adoption of multi-modal wireless applications depends not only on the quality and usability of the applications themselves, but also on other elements such as the availability and reliability of wireless digital networks and the existence of standards supporting them.

8. ACKNOWLEDGMENTS

The authors wish to thank the entire Lobby7 [4] team for their support in building the VoiceMap application and Sandeep Sibal from Kirusa [6] for many discussions on multi-modal architecture.

9. REFERENCES

[1] Distributed Speech Recognition - AURORA: http://www.etsi.org/frameset/home.htm?/technicalactiv/DSR/dsr.htm
[2] DARPA Communicator Project - http://www.darpa.mil/ipto/research/com/index.html
[3] Compaq home page - http://www.compaq.com
[4] Lobby7 home page - http://www.lobby7.com
[5] MapQuest home page - http://www.mapquest.com
[6] Kirusa home page - http://www.kirusa.com
[7] Pieraccini, R., Caskey, S., Dayanidhi, K., Carpenter, B., Phillips, M. "ETUDE, a Recursive Dialog Manager with Embedded User Interface Patterns," Proc. of ASRU01 – IEEE Workshop, Madonna di Campiglio, Italy, Dec. 2001