The WAMI Toolkit for Developing, Deploying, and Evaluating Web-Accessible Multimodal Interfaces

Alexander Gruenstein
MIT CSAIL
32 Vassar Street
Cambridge, MA 02139, USA
alexgru@csail.mit.edu

Ian McGraw
MIT CSAIL
32 Vassar Street
Cambridge, MA 02139, USA
imcgraw@csail.mit.edu

Ibrahim Badr
MIT CSAIL
32 Vassar Street
Cambridge, MA 02139, USA
iab02@csail.mit.edu

ABSTRACT

Many compelling multimodal prototypes have been developed which pair spoken input and output with a graphical user interface, yet it has often proved difficult to make them available to a large audience. This unfortunate reality limits the degree to which authentic user interactions with such systems can be collected and subsequently analyzed. We present the WAMI toolkit, which alleviates this difficulty by providing a framework for developing, deploying, and evaluating **W**eb-**A**ccessible **M**ultimodal **I**nterfaces in which users interact using speech, mouse, pen, and/or touch. The toolkit makes use of modern web-programming techniques, enabling the development of browser-based applications which rival the quality of traditional native interfaces, yet are available on a wide array of Internet-connected devices. We will showcase several sophisticated multimodal applications developed and deployed using the toolkit, which are available via desktop, laptop, and tablet PCs, as well as via several mobile devices. In addition, we will discuss resources provided by the toolkit for collecting, transcribing, and annotating usage data from multimodal user interactions.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*graphical user interfaces, natural language, voice I/O*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*language parsing and understanding, speech recognition and synthesis*

General Terms

Design, Experimentation

Keywords

multimodal interface, speech recognition, dialogue system, World Wide Web, Voice over IP

ICMI'08, October 20–22, 2008, Chania, Crete, Greece.

1. INTRODUCTION

A broad range of multimodal interfaces have been developed in which a spoken language understanding capability complements a graphical user interface (GUI). Many such interfaces provide for spoken natural language input in conjunction with drawing, clicking and/or other rich GUI interaction, for example: [6, 9, 12, 16]. Such interfaces offer compelling alternatives to GUI-only applications, particularly when used on devices such as tablet computers and mobile phones which offer relatively high-resolution displays and audio input/output capabilities, but limited keyboard input. Indeed, the prevalence of such portable but powerful devices offers an unparalleled opportunity to bring a variety multimodal interfaces to the mainstream.

Unfortunately, while researchers may develop many interesting multimodal applications, it is often challenging to make them available to a large number of users outside of the laboratory. The systems typically rely on a hodgepodge of components: speech and gesture recognition engines, natural language processing components, and homegrown graphical user interfaces. It can be challenging to package up these components so that they can be easily installed by novice users, particularly for an academic research lab with limited engineering resources. As such, system deployment is usually limited to hardware within the laboratory, limiting data collection to traditional in-lab user studies.

We believe, however, that breakthroughs in multimodal interfaces will come when they can more cheaply and easily be made available. Unimodal spoken language interfaces, for example, have benefited enormously from their accessibility via the telephone, which is a natural, easy, and cheap means of making them available to a large number of users. Interface quality benefits greatly from the iterative improvements which become possible by analyzing usage data gleaned from a large number of interactions with real users [14, 20].

In this paper, we present a toolkit which enables interface developers to cheaply develop, deploy, and evaluate multimodal interfaces. The toolkit, called WAMI, provides a basis for developing **W**eb-**A**ccessible **M**ultimodal **I**nterfaces, which are accessible to any user with a standard web browser and a network connection. Applications are easily accessed not only from desktop, laptop, and tablet computers, but from a variety of mobile devices as well. Moreover, because WAMI takes advantage of modern web-programming techniques revolving around AJAX (Asynchronous Javascript and XML), graphical user interfaces can be developed in the framework which rival the quality of traditional native interfaces.

Multimodal interfaces developed using web-standards have a number of key advantages beyond their easy accessibility to a large number of users. First, interfaces provided via the network can run computationally demanding processes such as speech recognition on fast servers, which is especially important for mobile devices. Second, web-based applications can make use of a growing array of powerful services available via the web. Third, collecting usage data for research is straightforward: users interact with the system via the comfort of their own devices, yet those interactions can be centrally logged.

The ability to analyze usage data collected via the web is critical for developers wishing to improve their multimodal interfaces; as such, the WAMI toolkit provides a streamlined interface for replaying, transcribing, and annotating usage data. System developers and annotators can replay a user's interaction, watching it unfold in a web browser as they transcribe utterances and/or annotate actions. In addition, a user-study management framework provides system designers with an easy way to gather data: users sign up for a study via the web and then are led through a series of tasks in their own browsers.

In the remainder of this paper, we will describe the development framework and evaluation architecture provided by WAMI in greater detail. We will then discuss several applications built within the WAMI framework, showing off its flexibility and power. Next, we'll reflect on some of the usage data collected with those applications, noting some of the differences between web-based and in-lab user studies. Finally, we will describe our nascent efforts to make WAMI available to outside developers.

2. RELATED WORK

A notable early effort at making multimodal interfaces accessible via the web was WebGALAXY [11], which takes advantage of the easy accessibility of both the telephone and the Web. Users view a web site while speaking over the telephone to a spoken dialogue system which provides weather information. The system's responses to user queries are spoken over the phone, with corresponding information also displayed on the web page. The WAMI toolkit differs from WebGALAXY in two key respects. First, it does not require the use of the telephone, which is awkward in many situations. Second, while it does support interfaces like WebGALAXY, in which the graphical modality is primarily a means of displaying information also provided in the speech output, the WAMI toolkit is mainly intended to support much richer GUI interactivity.

More recently, efforts have been undertaken to develop standards for deploying multimodal interfaces to web browsers, typically entailing the use of VoIP or local speech recognition. The two major standards under development, Speech Application Language Tags (SALT) [18] and XHTML+Voice (X+V) [3] are both aimed at augmenting the existing HTML standard. The intent of these specifications is that, just as current web browsers render a GUI described via HTML, browsers of the future would also provide a speech interface based on this augmented HTML. As such, interfaces implemented using either of these specifications require the use of a special web browser supporting the standard.

SALT and X+V are designed mainly to enable the use of speech to fill in the field values of traditional HTML forms. X+V, for example, integrates the capabilities of VoiceXML into the browser, so that interactions typically consist of telephony-like mixed initiative dialogues to fill in field values. Related multimodal markup languages, such as those in [10] and [7], generally attempt to make designing these sorts of interactions more straightforward.

Unlike SALT and X+V, applications produced with the WAMI toolkit do not require the use of special web browsers; instead, they can be accessed from today's standard browsers. Furthermore, the toolkit is not intended to provide form-filling capabilities; though, it could potentially be used in this way. Though the formal structure of SALT and X+V provide the power to make authoring certain types of form-filling interfaces easier, this structure makes it difficult to design applications incorporating the rich and rapid multimodal interactivity in which we believe many researchers are interested.

Finally, several commercial multimodal applications have recently been released which are widely accessible via various mobile devices. These include Microsoft's Live Search For Mobile [1], TellMe's mobile download [17], and Yahoo's voice enabled oneSearch [19]. In these applications, users generally click on a text input box, speak a few query terms, view and correct the results, and then submit the query to a search engine. The WAMI toolkit could also be used to support such an interaction model, however it also supports much richer multimodal interactions. In addition, these are native applications, and as their download pages attest, they must be customized for each supported device—a time-consuming task for a researcher interested in exploring a novel multimodal interface.

3. SYSTEM DEVELOPMENT WITH *WAMI*

The WAMI toolkit supports two main threads of development. First, it can be used to connect a graphical user interface to "traditional" spoken dialogue systems built using the GALAXY architecture [15]. In this architecture, speech recognition using an n-gram language model typically serves as a first step before parsing, contextual integration, dialogue management, natural language generation, and speech synthesis. Interfaces built in this manner can be made to understand a wide range of natural language; yet they require a good deal of expertise to develop.

WAMI also supports a second, more lightweight development model meant to appeal to developers who don't have expertise in speech recognition, parsing, dialogue management, and natural language generation. WAMI's lightweight platform provides a standard structure with which developers can build highly interactive multimodal applications with modest natural language understanding capabilities. Experts also find the lightweight platform useful, as it provides an easy means to quickly prototype new applications.

3.1 Core Platform

At the center of both application development models is the core platform depicted in figure 1, which provides the basic infrastructure requisite for developing and deploying web-based multimodal applications. The platform consists of a small set of components split across a server and a client device running a standard web browser. Core components on the server provide speech recognition and (optionally) speech synthesis capabilities, a JavaEE (Java Enterprise Edition) web server, and a logger component which records user interactions. Applications must provide a lan-

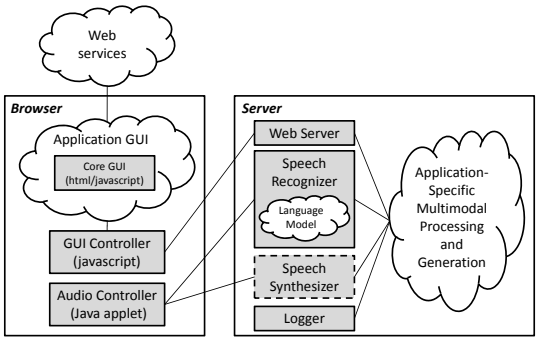


Figure 1: WAMI toolkit core platform. Shaded boxes show standard core components. White clouds indicate application-specific components.

guage model for the speech recognizer, and any application-specific processing modules for handling spoken input and generating appropriate verbal and graphical responses. On the client, a web browser renders an application-specific GUI, which exchanges messages with the server via an interface provided by the AJAX-powered GUI Controller component.

In addition, an Audio Controller component runs on the client to capture a user's speech and stream it to the speech recognizer, as well as to play synthesized speech generated on the server and streamed to the client. On devices which support Java, the Audio Controller is a Java Applet which is embedded directly in the web page, and which provides a button users click when they want to provide speech input. For other devices, native Audio Controllers must be developed, which are run externally from the web browser. For example, on each of the mobile devices we will discuss in this paper, the Audio Controller is a native application which is controlled by physical button presses. Once started, these native Audio Controllers then point the web browser on the device to the appropriate URL to bring up the GUI. Though native Audio Controllers are a break with our web-only philosophy, they need only be written once for a particular device, as they can be shared by any WAMI-powered interface. This means that GUI developers are freed from developing native code for a particular platform, and that users need only to install a single native application to gain access to any WAMI application.

The main value proposition of the core WAMI platform is that it provides developers with a standard mechanism for linking the client GUI and audio input/output to the server, insulating the developer from these chores. This is perhaps best understood by looking at the typical sequence of events involved in a user interaction. First, a user navigates to an application's web page, where the core WAMI components test the browser's compatibility and ensure sufficient capacity on the server exists for a new client. Next, the GUI and Audio Controllers connect to the server. Once both are connected, the web server notifies the GUI Controller, which passes the message along to the application-specific GUI so that it can initiate the interaction.

The user then presses the click-to-talk button (or holds a physical button on a mobile device) and begins speaking, causing the Audio Controller to stream the input audio to the speech recognizer. Concurrently, if the user interacts with the application's GUI—for example, by clicking or drawing—events encoded as XML are transmitted via the GUI Controller and Web Server to the application-specific module on the server. When the user finishes speaking, the recognizer's hypothesis is routed to the application-specific logic on the server, which formulates a response. It passes messages encoded as XML via the Web Server and GUI Controller to the GUI, which updates the display accordingly. It can also provide a string of text to be spoken, which is synthesized as speech and streamed to the Audio Controller.

3.2 Lightweight Platform

WAMI's core platform provides developers with a minimally sufficient set of components to build a web-based multimodal interface. A "lightweight" development platform which provides structure on top of the core components is designed to ease the development of multimodal interfaces with modest natural language understanding needs. Developers provide speech recognition language models for their applications in the form of grammars, written using the JSGF (Java Speech Grammar Format) standard [8]. Moreover, the grammars are constructed in a standard way, so that associated with each recognition hypothesis is a set of slots filled in with values. Figure 2 shows an example grammar for a hypothetical lightweight WAMI interface, as well as the slot/value pairs which are associated with an example utterance.

Because the lightweight platform relies on small constrained language models which output sequences of slot/value pairs, speech recognition and natural language understanding processing often occur rapidly, in near real time. We have taken advantage of this low overhead to add a perhaps surprising feature to the lightweight platform: incremental speech recognition and natural language understanding. The speech recognizer has been augmented to emit incremental recognition hypotheses as an utterance is processed. Because the grammar-based language models used provide strong constraints, the incremental hypotheses tend to be accurate, especially if the "bleeding edge" is ignored—so long as the user's utterance remains within the confines of the grammar. This means that while a user is speaking, slots and their values can be emitted as soon as the partial utterance is unambiguous. The grammar in figure 2 is written in just such a way: slots and values are assigned as soon as possible. This means that the slot/value pairs shown with the example utterance in that figure are emitted as soon as the user finishes speaking the word shown above each pair. An Incremental Understanding Aggregator on the server collects these slots and values as they are emitted.

Incremental speech recognition hypotheses can be used to provide incremental language understanding and immediate graphical feedback, which has been shown to improve the user experience and the system accuracy [2]. For instance, an application making use of the grammar in figure 2 could highlight the set of appropriate shapes as each attribute is spoken. Indeed, careful readers will note that the grammar is constructed so as to support false starts and repetitions, since visual feedback may prompt users to correct utterances as they speak. For instance, an in-grammar utterance with a false start would be: *select the small . . . the large red rectangle.* Such an utterance might arise because as soon as the user sees graphical confirmation of "small", he might realize he actually meant to say "large" and correct himself

```
<command>    = <select> | <drop> | <make> ;
<select>     = (select | choose) {[command=select]} <shape_desc>+ ;
<shape_desc> = [the] <shape_attr>+ ;
<shape_attr> = <size> | <color> | <shape_type> ;
<size>       = <size_adj> [sized] ;
<size_adj>   = (small | tiny) {[size=small]} | medium {[size=medium]} | (large | big)  {[size=large]} ;
<color>      = red {[color=red]} | green {[color=green]} | blue {[color=blue]} ;
<shape_type> = (rectangle | bar) {[shape_type=rectangle]} | square   {[shape_type=square]}
             | circle            {[shape_type=circle]}    | triangle {[shape_type=triangle]} ;
...
```

select	*the*	*small*	*red*	*rectangle*
[command=select]		[size=small]	[color=red]	[shape=rectangle]

Figure 2: Sample JSGF grammar snippet for a lightweight application and an example utterance with associated slot/value output.

mid-utterance. The vocabulary game applications described below in section 4.1 explore the utility of such incremental feedback.

3.3 Traditional Dialogue System Platform

The core WAMI toolkit also provides a platform for integrating web-based graphical user interfaces with "traditional" spoken dialogue systems built using the GALAXY architecture. Such systems typically provide an n-gram language model, which may be dynamically updated as a user interacts with an application. Speech recognition hypotheses are passed to the dialogue manager architecture for processing. The dialogue system can update the graphical user interface in one of two ways. First, as in the lightweight case, messages encoded as XML can be sent to the application-specific GUI, where calls to javascript then update the display. Alternatively, HTML may be output, in which case the GUI is refreshed using this HTML. The first method is usually preferred, as it can be used to create a dynamic, native-like interface. However, generating HTML via the dialogue system itself can be useful as well, as it can be generated, for instance, using traditional language generation techniques. It is an easy way to develop an application which is primarily a speech interface, but which benefits by showing content to the user in response to spoken input.

In addition, the GUI can send messages to notify the dialogue system of events such as clicks and gestures. These may be encoded directly in javascript in the native message passing format of the dialogue system architecture. This allows events from the GUI to be easily integrated into the normal processing chain.

3.4 Supported Devices

The WAMI toolkit can be used to produce applications which run on any network connected device with a microphone for audio input and a modern web browser. On desktop, laptop, and tablet computers, it supports Windows, GNU/Linux, and OS X running recent versions of Firefox, Opera, Safari, or Internet Explorer. Audio input and output on these platforms is handled via a Java applet embedded directly in the browser.

The toolkit also supports mobile devices with standards-compliant browsers; however, generally a native Audio Controller must be written. Currently, the toolkit has been tested on two mobile platforms:

- Several smartphone and PDA devices running Windows Mobile 5, with the Opera Mobile browser,

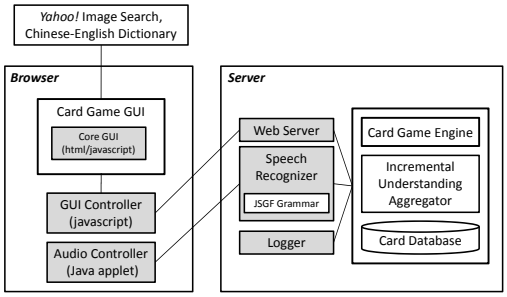


Figure 3: *Chinese Cards* Architecture; WAMI toolkit modules shaded.

- The Nokia N810 internet tablet, using a native audio controller and the pre-installed Mozilla-derived browser.

We are exploring an Audio Controller for the iPhone.

4. EXAMPLE APPLICATIONS

In this section, we will briefly showcase several applications which highlight the utility and flexibility of the WAMI toolkit. We focus on applications in which both the speech and graphical modalities play important roles.

4.1 *Chinese Cards*

The lightweight platform has served as the basis for a novel group of multimodal games designed to help non-native speakers practice Mandarin Chinese. The games share a common *Chinese Cards* architecture depicted in figure 3. Students and teachers first login to a web application and build a personalized deck of image-based flash cards, a task which is made easier because Yahoo Image Search and an online Chinese-English dictionary[1] are integrated directly into the application. The personalized cards are stored in a back-end database, where they can then be used in any of the card games. JSGF grammars, personalized to include the vocabulary from each user's deck of flash cards, are used as language models for each card game.

One of the card games which can be played with the personalized deck of flash cards is *Word War*, a picture matching race [13]. A single player can race against the clock, or two players can race against each other from their respective web browsers, each using his or her own deck of

[1]http://www.xuezhongwen.net

Figure 4: The *Word War* flash card game for language learners.

cards. Figure 4 shows a snapshot of two players competing on a five-column game grid. The goal of *Word War* is to use spoken commands to move images from the bottom two rows of the grid into a set of numbered slots in the second row, so that they match the images along the top. While a simple task in a person's native language, it can be a challenging and fun way to practice vocabulary in a foreign language. Each player uses his or her own flash cards—so the two players may be practicing different vocabulary words, or even be speaking in different languages—but they compete over shared access to the numbered slots. When an image is matched, the slot is captured and the matching image appears on *both* players' game grids. Notice that in figure 4, Player 1 has captured the third and fourth slots, while Player 2 has only captured the first slot. The winner is the first player to fill in a majority of the slots.

The language model used for the game is conceptually similar to the example snippet shown in figure 2. For example, Player 1 in figure 4 can utter the Mandarin equivalent of *Take the hand and put it in slot number five* causing this action to be carried out on the GUI. Since the grammars are tailored to each player's vocabulary, they remain relatively small, aiding robust recognition of non-native speech.

Word War demonstrates well some of the advantages offered by the incremental speech recognition and the Incremental Understanding Aggregator. Most importantly, the incremental understanding is used to provide visual feedback *while* the student is speaking. For instance, as soon as Player 1 has said *Take the*, all of the available items which can be selected have been highlighted. Likewise, once Player 2 has uttered *Select the sheep and drop it* the sheep has already been highlighted, as have the available slots in which it can be dropped.

In a more conventional multimodal interface, the user would have to finish speaking a complete command before receiving visual confirmation of the system's understanding. In this case, incremental feedback provides immediate feedback to users as they speak. This means that students of a foreign language can become aware of errors immediately and correct them mid-utterance. They can also string together a sequence of commands in a single utterance, allowing them to rapidly practice speaking fluently.

4.2 Multimodal Dialogue Systems

The WAMI toolkit has also served as a platform for developing several multimodal interfaces which make use of more traditional spoken dialogue system components. We

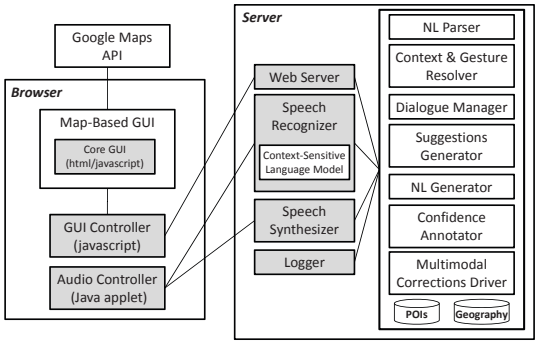


Figure 5: *City Browser* architecture. WAMI modules shaded.

will briefly describe two of the applications here with the intent of emphasizing that their interfaces behave with a responsiveness and sophistication typically associated with native multimodal interfaces, rather than traditional web interfaces.

City Browser is a map-based WAMI application which provides users access to urban information via an interface which leverages the Google Maps API [5]. Users can speak addresses to locate them on the map, obtain driving directions, get public transportation information, and search for Points of Interest (POIs) such as restaurants, museums, hotels, and landmarks. Search results are shown on the screen, and users can use a mouse, pen, stylus, or finger—depending on the device being used—to select a result, circle a set of results, outline a region of interest, or indicate a path along which to search.

Figure 5 depicts *City Browser*'s architecture, which is typical of a multimodal dialogue system built on top of the WAMI core. The map-based GUI, shown on a mobile device in figure 6(a), is built around the WAMI skeleton. The GUI passes messages to the natural language understanding components on the server via the GUI Controller and Web Server. The natural language parser, context and gesture resolution component, dialogue manager, and natural language generator are previously developed generic components which have been used in the service of many dialogue systems. Application-specific logic for *City Browser* is encoded by the grammars, dialogue control scripts, generation templates, and so forth used by these components.

We have also leveraged a similar set of dialogue system components in conjunction with the WAMI toolkit to de-

(a) *City Browser* on a Nokia N810 in its Mozilla-based browser

(b) Home entertainment multimodal interface on a Windows Mobile smartphone running the Opera browser and displayed on a television via Firefox on OS X

Figure 6: Web-based multimodal interfaces built using the WAMI toolkit, running on various devices.

velop and deploy a multimodal interface to a living-room multimedia server containing a music library and digital video recorder [4]. The interface, shown in figure 6(b), is split across a mobile device and a television screen. The microphone and speakers on the mobile device are used for speech input and synthesized speech output, and its GUI provides access to the music and video libraries, television listings, and recording settings. Users can speak and use a stylus or the arrow keys to browse television listings and schedule recordings on the mobile device from anywhere with a network connection. Then, while at home, a television display enables the playback of music and recordings.

We conclude this section by emphasizing that the screenshots in figures 4 and 6 show WAMI-based interfaces running on a variety of devices, using various web browsers: two mobile devices, a windows laptop, and a Mac attached to a television screen. As these applications demonstrate, high quality web-based multimodal interfaces can cheaply and easily be deployed to users in various environments.

5. DATA COLLECTION AND EVALUATION ARCHITECTURE

A fundamental step in the development process for any multimodal interface involves evaluating prototypes based on authentic user interactions. Because applications built using the WAMI toolkit are web-based, it's easy to make them available to any user with a web browser. However, because these users are remote, their interactions can't simply be observed and/or videotaped. In this section, we describe resources provided by the WAMI toolkit which enable developers to run studies to collect such interactions, as well tools for transcribing and annotating the collected usage data.

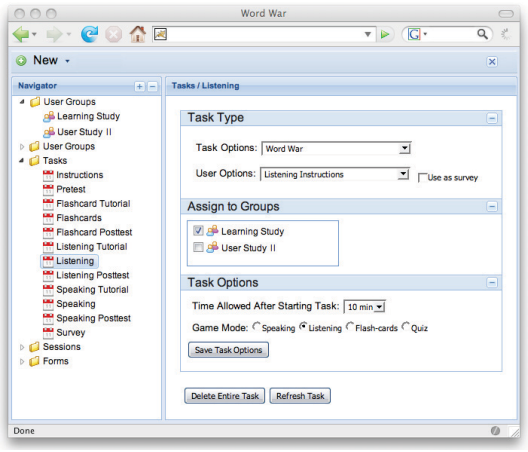


Figure 7: Web-based user study management.

5.1 User Study Management

One method of data collection for a web-based application is to simply deploy the application, publicize its URL, and analyze whatever user interactions occur. However, it is often useful for researchers to have a greater amount of control over the types of users recruited and tasks performed. The WAMI toolkit provides a web-based user study management interface that gives researchers tools to manage studies involving remote users. Several user studies have been conducted with increasingly sophisticated versions of the tools.

To begin a new user study, a set of tasks must first be defined using the management interface, as shown in figure 7. Typical tasks might include: reading instructions, completing a warmup exercise, solving a problem, and filling out a survey. Defining common tasks, such as surveys, is particularly easy, as all WAMI applications inherit the ability to display user-defined web forms and record the results in a database. Application-specific tasks can also be defined, in which case the application must display appropriate content and monitor a user's progress.

Once the tasks are defined, one or more user groups are created, and each group is assigned a sequence of tasks which must be completed. User accounts can be created by the study administrator, or users can sign up online via an advertised URL. When a user logs into a WAMI application in user-study mode, he or she is presented with a sequence of tasks to complete. For example, in a recent *City Browser* study, which usually required 30-60 minutes, subjects were first presented with instructions on how to use the interface and then given a simple warmup task. Next, they were led through ten scenarios in which they were asked to, for example, get directions between two locations or find a restaurant with certain attributes. Then, they were given "free play" time to interact as they wished. Finally, they completed a survey. A less complex study performed with *Word War* simply required each user to complete as many games as possible in 10 minutes.

5.2 Transcription, Annotation, and Playback

When designing any application which makes use of a speech recognizer, it is critical to transcribe user utterances: transcripts are necessary to measure error rates, and improve the language and acoustic models used by the speech recog-

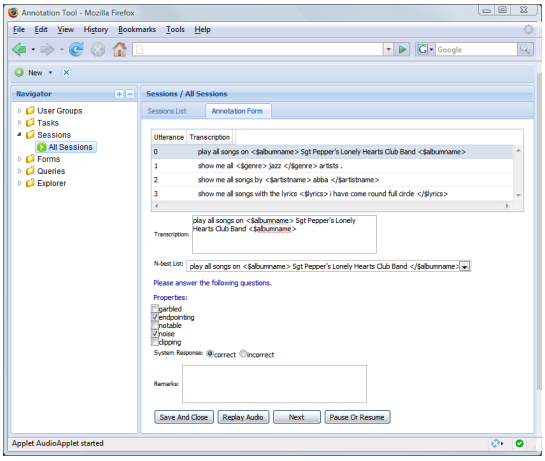


Figure 8: Web-based Session Playback.

nizer. Beyond transcripts, interface designers are typically interested in understanding how users interacted with an application. Such knowledge allows them to identify system malfunctions and identify the strengths and weaknesses of an interface. For speech-only interfaces, this sort of knowledge can generally be gleaned from transcripts, speech recognition hypotheses, and system responses. In developing multimodal interfaces, however, transcripts do not suffice: it is immensely helpful to also understand what the GUI looked like at the time of interaction, and what sort of non-speech interaction occurred (clicking, circling, *etc*).

When performing studies in the laboratory, experimenters can observe and/or videotape a user's interaction; however, this is simply not possible for web-based systems accessed remotely. Thus, another key capability of the WAMI toolkit is the ability to log user interactions and then "replay" the interactions later. We have found that this is often the only feasible manner in which an annotator can, for example, accurately judge whether or not an application's response to a user's utterance was appropriate. Playback and annotation tools can also be useful in evaluating *human* performance. For example, teachers are able to judge a student's speaking ability by replaying student interactions with an application like *Word War*.

The WAMI toolkit includes a logging component as part of the core architecture, as shown in figure 1. The logger records the user utterance waveform, recognition hypotheses, and the system's natural language responses. In addition, any events sent from the client GUI to the application specific logic are logged, so that notifications of clicks, gestures, the scrolling of windows, and so forth are recorded. On the flip side, any messages sent from the application-specific logic residing on the server to the client GUI are also logged.

To play back these logs, the toolkit provides a web-based Transcription and Annotation tool as part of the management interface. Using this interface, depicted in figure 8, developers or annotators can control a server-based Log Player, which loads a log of one user's interaction and replays the time-stamped events at the appropriate pace. The web-based tool loads the application's GUI, which plays the events as it receives them from the Log Player. For instance, recorded user utterances are played back, while any GUI interactions such as gestures are shown on the GUI. As the log is replayed, the annotator can watch the interaction unfold, while simultaneously transcribing the user utterances and making any appropriate annotations.

5.3 Usage Data Analysis

We have used the tools described in this section to collect, transcribe, and annotate usage data for the *Word War* and *City Browser* applications described above. For very early prototypes, we have relied on traditional laboratory studies, in which users come to the lab and are assigned a set of tasks to perform. As prototypes advance, we move to remote studies, in which subjects are recruited via public announcements and login from their own computers. In our experience, this allows us to collect data much more rapidly with much less time invested on our part.

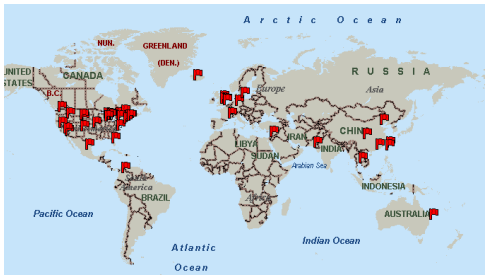
Moreover, our impressions after transcribing and annotating thousands of utterances is that users act more naturally when they are in the comfort of their home or office. In particular, they experiment more with the capabilities of the system and push the bounds of what's possible. For the researcher, watching these interactions in playback mode can be a simultaneously thrilling and sobering experience. While replaying a *Word War* interaction captured in single player practice mode, for example, it became apparent that a father and daughter were practicing Chinese together, at times sounding out words simultaneously! Similarly, we have observed interactions in which *City Browser* is used by two or more people at once in a social setting. A more subtle difference is that our impression has been that in-lab users of both applications tend to stick more to the provided sample sentence structures, while remote users are willing to experiment more.

Since individual users rely on their own hardware resources to perform the tasks in a remote user study, the audio quality also varies greatly. Though clear instructions and tutorial tasks can prevent many problems, users with poor microphones or mis-configured audio settings are difficult to avoid entirely. While in some ways this is a disadvantage, by deploying a WAMI application to the web and performing an error analysis on recognition results, the researcher can gain insight into the bottlenecks with regards to robust recognition in a realistic environment. In [13], *Word War* recognition performance was analyzed along four dimensions, one of which was audio quality. While its effects were significant, it turned out that the "non-nativeness" of a student's speech was a larger impediment to accurate recognition.

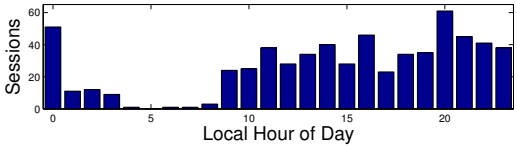
Finally, another clear advantage of deploying web-based interfaces is that users can be recruited from all over the world, and can use the interface whenever is most convenient. Figure 9 shows the geographical distribution of users who have interacted with *Word War* and *City Browser* and a histogram of the user's local time of day during which these interactions took place. Subjects logged in from all over the world, and there was heavy usage outside of normal business hours, when user studies are typically conducted.

6. CONCLUSIONS AND FUTURE WORK

We have presented the WAMI toolkit which allows the development, deployment, and evaluation of highly interactive web-based multimodal interfaces on a wide range of network-connected devices. We have demonstrated the feasibility of this approach by describing several rich interfaces using the



(a) Geographical distribution of study subjects.



(b) Local time of day of interactions.

Figure 9: *City Browser* and *Word War* usage.

toolkit, and by showing how the provided web-based study management system makes it straightforward to gather and annotate user interactions with such systems. Our hope is that research in the field will benefit by making it relatively cheap and easy to deploy multimodal interfaces to a large number of users.

To that end, we are currently in the process of making the toolkit publicly available. In particular, we are developing a portal at `http://web.sls.csail.mit.edu/wami/` which will provide speech recognition services to developers. Any developer will be able to create a multimodal application using the lightweight model described in section 3.2 without having to configure a speech recognizer; instead, the developer will upload a JSGF grammar and connect to a recognizer maintained on our servers. We hope that by significantly lowering the barrier to entry, we will enable developers with fresh ideas to create novel, compelling multimodal interfaces.

7. ACKNOWLEDGMENTS

Thanks to Stephanie Seneff for feedback on drafts of this paper. This research is sponsored by the T-Party Project, a joint research program between MIT and Quanta Computer, Inc., and by the Industrial Technology Research Institute.

8. REFERENCES

[1] A. Acero, N. Bernstein, R. Chambers, Y. C. Jui, X. Li, J. Odell, P. Nguyen, O. Scholz, and G. Zweig. Live search for mobile: Web services by voice on the cellphone. In *Proc. of ICASSP*, 2008.

[2] G. Aist, J. Allen, E. Campana, C. G. Gallo, S. Stoness, M. Swift, and M. K. Tanenhaus. Incremental understanding in human-computer dialogue and experimental evidence for advantages over nonincremental methods. In R. Artstein and L. Vieu, editors, *Proc. of the 11th Workshop on the Semantics and Pragmatics of Dialogue*, pages 149–154, 2007.

[3] J. Axelsson, C. Cross, J. Ferrans, G. McCobb, T. V. Raman, and L. Wilson. Mobile X+V 1.2. Technical report, 2005. http://www.voicexml.org/specs/multimodal/x+v/mobile/12/.

[4] A. Gruenstein, B.-J. P. Hsu, J. Glass, S. Seneff, L. Hetherington, S. Cyphers, I. Badr, C. Wang, and S. Liu. A multimodal home entertainment interface via a mobile device. In *Proc. of the ACL Workshop on Mobile Language Processing*, 2008.

[5] A. Gruenstein, S. Seneff, and C. Wang. Scalable and portable web-based multimodal dialogue interaction with geographical databases. In *Proc. of INTERSPEECH*, 2006.

[6] A. Hjalmarsson. Evaluating AdApt, a multi-modal conversational dialogue system using PARADISE. Master's thesis, KTH, Stockhom, Sweden, 2002.

[7] M. Honkala and M. Pohja. Multimodal interaction with XForms. In *Proc. of the 6th International Conference on Web Engineering*, pages 201–208, 2006.

[8] Java speech grammar format. http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/.

[9] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor. MATCH: An architecture for multimodal dialogue systems. In *Proc. of ACL*, 2002.

[10] K. Katsurada, Y. Nakamura, H. Yamada, and T. Nitta. XISL: a language for describing multimodal interaction scenarios. In *Proc. of the 5th International Conference on Multimodal Interfaces*, 2003.

[11] R. Lau, G. Flammia, C. Pao, and V. Zue. WebGALAXY: beyond point and click – a conversational interface to a browser. *Computer Networks and ISDN Systems*, 29:1385–1393, 1997.

[12] O. Lemon and A. Gruenstein. Multithreaded context for robust conversational interfaces: context-sensitive speech recognition and interpretation of corrective fragments. *ACM Transactions on Computer-Human Interaction*, 11(3):241–267, 2004.

[13] I. McGraw and S. Seneff. Speech-enabled card games for language learners. In *Proc. of the 23rd AAAI Conference on Artificial Intelligence*, 2008.

[14] A. Raux, D. Bohus, B. Langner, A. Black, and M. Eskenazi. Doing research on a deployed spoken dialogue system: One year of Let's Go! experience. In *Proc. of INTERSPEECH-ICSLP*, 2006.

[15] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue. Galaxy-II: A reference architecture for conversational system development. In *Proc. ICSLP*, 1998.

[16] A. Stent, J. Dowding, J. M. Gawron, E. O. Bratt, and R. Moore. The CommandTalk spoken dialogue system. In *Proc. of ACL*, 1999.

[17] Tellme mobile. http://m.tellme.com.

[18] K. Wang. SALT: A spoken language interface for web-based multimodal dialog systems. In *Proc. of the 7th International Conference on Spoken Language Processing*, 2002.

[19] Yahoo onesearch. http://mobile.yahoo.com.

[20] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington. JUPITER: A telephone-based conversational interface for weather information. *IEEE Transactions on Speech and Audio Processing*, 8(1), January 2000.